der Bundeswehr
Universität München

# Side-Channel and Fault Attacks in Modern Lattice-Based Cryptography

Julius Johannes Hermelink

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

angenommenen Dissertation.

Gutachter/Gutachterin:

1. Prof. Dr. Gabi Dreo Rodosek
2. Prof. Dr. Mark Manulis

Die Dissertation wurde am 06.12.2023 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 06.03.2024 angenommen. Die mündliche Prüfung fand am 15.03.2024 statt.

der Bundeswehr
Universität München

# Side-Channel and Fault Attacks in Modern Lattice-Based Cryptography

Julius Johannes Hermelink

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

angenommenen Dissertation.

Gutachter/Gutachterin:

1. Prof. Dr. Gabi Dreo Rodosek
2. Prof. Dr. Mark Manulis

Die Dissertation wurde am 06.12.2023 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 06.03.2024 angenommen. Die mündliche Prüfung fand am 15.03.2024 statt.

# Acknowledgments

First and foremost, I want to thank Dr. Thomas Pöppelmann for his support, encouragement, and helpful feedback over more than five years. You raised my enthusiasm for cryptography while I was a working student at Infineon, enabled me to learn in these two years, and then made me aware of the position I ended up applying to, which lead to this thesis. Thank you, Thomas, I would not have found my way into cryptography without you!

Prof. Dr. Gabi Dreo deserves huge and sincere thanks. You provided constant and helpful advice, insights, and constructive feedback while I was your PhD student for more than three years. Not only did you enable this thesis, improved my work, and showed me how to enhance my skills and critical thinking, but you managed to create a work environment that brings out the best in everyone. I owe you, Gabi, a lot of knowledge and skills, and this thesis would not have been possible without you!

I am indebted to Dr. Simona Samardjiska and Dr. Peter Pessl: Your incredible knowledge, willingness to pass it on, and the selfless help you offered me whenever I needed it did not go unnoticed. In the more than two years I was lucky to know you, you both offered me a lot of your time, which was very helpful and is greatly appreciated. Moreover, you provided advice in every regard needed and connected me to the people I now work with.

I want to thank my second supervisor, Prof. Dr. Mark Manulis, for his helpful comments and advice in the last year, in regard to the thesis and to academia as well as on how to continue from hereon. Introducing me to your team is something else I am grateful for!

The team of Prof. Dr. Gabi Dreo's chair played a large role for writing this thesis: Thank you, Daniel Heinz, for the always insightful and interesting discussions on cryptography and on our theses. I would like to especially thank Nils Rodday for always finding times to discuss problems and provide feedback. Apart from that your cheerful personality was one of the reasons going to work was something I always looked forward to. This is true for Klement Hagenhoff as well, whom I also need to thank for feedback and the technical and philosophical debates we had. Tobias Fritz, you are not only a colleague, but have also been a good friend for many years, thank you for the support and the discussions we have had for so many years.

It should also be noted that several other people at UniBwM and Infineon helped me before and during my thesis. Most notably, this includes PD Dr. Corinna Schmidt, Dr. Markus Gail, and Dr. Florian Mendel. Working with all of you has always been a pleasure, and I am grateful for the opportunities you provided me with. I also want to thank Silvan Streit and Emanuele Strieder of Fraunhofer AISEC for the many helpful discussions and the work we did together!

Far beyond this thesis, I am incredible lucky with my two supportive parents, Dr. Kerstin Hermelink and Jan Hermelink. I could not have wished for better parents; thank you for everything you did for me, including your support during this thesis. I have to thank my siblings, Kai Hermelink and Ada Hermelink – I could not imagine life without you!

Most importantly, I want to thank the love of my life Daniela Regler. You are the reason for the path in life I chose in the first place and where I am today.

# Abstract

The prospect of large-scale quantum computers threatens currently used asymmetric cryptography – Shor's algorithm could render commonly used cryptography unsafe. In 2022, the National Institute for Standards and Technology (NIST) announced several quantum-secure cryptographic algorithms for key exchanges and digital signatures for standardization, and three out of four are based on hard lattice problems. The main candidate for key exchanges is a key encapsulation mechanism called Kyber, which bases its security on the Module Learning with Errors problem. In contrast to quantum cryptography, these post-quantum algorithms run on classical computers and currently available hardware.

Data that is encrypted now often needs to be secured for a long time. In addition, potentially vulnerable cryptography is present in many products, protocols, and use cases; therefore, deploying different algorithms is a difficult task that could take a substantial amount of time and effort. Clearly, migrating to quantum-safe cryptography cannot wait until large-scale quantum computers are available. Thus, and due to the imminent standardization, lattice-based cryptography is likely to run on a wide variety of devices in the near future.

In this work, we present attack strategies against lattice-based cryptography. Our attack strategies combine chosen-ciphertext attacks with side-channel analysis as well as with fault attacks and target two major components of the decapsulation that process the secret key. Moreover, we present techniques to recover the secret key from the obtained information and show how to combine those methods with algebraic approaches.

We first present an attack strategy on the number theoretic transform: A chosen ciphertext is pre-computed such that it cancels out values during the inverse number theoretic transform that processes the secret key. The reduced entropy allows for an improved subsequent side-channel analysis. Using our technique, the number theoretic transform can be targeted with highly increased noise tolerance. We take countermeasures into account and state techniques that enable the adaptation of belief-propagation-based attacks to protected settings.

Then, we target the error correction and the Fujiskai-Okamoto transform. We show how a fault may turn the Fujisaki-Okamoto transform into a decryption failure oracle; this enables a chosen-ciphertext attack targeting the error correction. Our fault-enabled chosen-ciphertext attack requires an adversary to target only public data, may be administered in various locations, and allows the use of an unreliable fault. In addition, we show how belief propagation can be employed to recover the secret key from decryption failure information. We then propose using a combination of countermeasures to mitigate our attack strategy.

Finally, we explain how to recover the secret key from decryption failure information in an error-resistant way that also allows for security estimates. We modify the belief-propagation-based approach to achieve error-resistance and explain how belief propagation may be combined with lattice reduction. Our method outperforms previous algorithms in terms of required information, is error-tolerant, and allows for security estimates in cases where the secret key cannot be fully recovered. These security estimates enable the design and evaluation of countermeasures and provide an assessment of the impact of several attacks that exploit decryption failures.

# Kurzfassung

Die Entwicklung von Quantencomputern bedroht die aktuell verwendete asymmetrische Kryptographie – Shors Algoritmus könnte weit verbreitete kryptographische Verfahren brechen. Im Jahr 2022 hat das National Institute for Standards and Technology (NIST) mehrere Kandidation für quantensichere Verschlüsselung als Standardisierungskandidaten vorgestellt und drei von vier davon basieren auf Gitterproblemen. Der Hauptkandidat für Schlüsselaustausche ist ein Key Encapsulation Mechanismus namens Kyber, dessen Sicherheit auf dem Module Learning with Errors Problem basiert. Im Gegensatz zur Quantenkryptographie laufen diese Postquantenverfahren auf aktuell verfügbarer, klassischer Hardware.

Daten die zurzeit verschlüsselt werden, müssen oft noch für lange Zeit ausreichend sicher bleiben. Zusätzlich ist potentiell vulnerable Kryptographie weite verbreitet und kommt in vielen Produkten, Protokollen und Anwendungsfällen vor; deshalb könnte der Wechsel zu quantensicheren Verfahren substanziell viel Zeit und Aufwand kosten. Ganz eindeutig kann der Wechsel zu quantensichereren Verfahren also nicht erst passieren, nachdem ausreichend leistungsstarke Quantencomputer bereits verfügbar sind. Deshalb, und aufgrund der anstehenden Standardisierung, werden gitterbasierete Algorithmen vermutlich in Kürze weit verbreitet sein und auf vielen Geräten laufen.

In dieser Arbeit stellen wir mehrere Strategien vor, mit denen gitterbasiere Verfahren angegriffen werden können. Unsere Angriffsstrategien kombinieren Chosen-Ciphertext Angriffe mit Seitenkanalanalyse und Fehlerangriffen und zielen auf zwei Hauptkomponenten der Decapsulation-Routine ab. Außerdem präsentieren wir Techniken, mit denen der geheime Schlüssel aus den in den Angriffen gewonnen Informationen gewonnen werden kann und zeigen wie diese Methoden mit algebraischen Ansätzen kombiniert werden können.

Zunächst erklären wir eine Angriffsstrategie auf die Number Theoretic Transform: Ein Ciphertext wird vorberechnet, sodass er Werte innerhalb der inverse Number Theoretic Transform, die den geheim Schlüssel transformiert, auslöscht Die dadurch reduzierte Entropie erlaubt anschließend eine verbesserte Seitenkanalanalyse. Durch die Verwendung unserer Methode kann die Number Theoretic Transform mit stark erhöhter Rauschresistenz angegriffen werden. Wir untersuchen zudem Gegenmaßnahmen und geben Techniken an, mit denen Belief Propagation an diese angepasst werden kann.

Anschließend zielen wir auf die Fehlerkorrektur und die Fujisaki-Okamoto Transformation ab. Wir zeigen, wie ein Fehlerangriff die Fujisaki-Okamoto Transformation zu einem Entschlüsselungsfehlerorakel werden lassen kann; dies passiert durch einen Chosen-Ciphertext Angriff, der auf die Fehlerkorrektur abzielt. Unser durch einen Fehlerangriff ermöglichter Chosen-Ciphertext Angriff kann von einem Angreifer durchgeführt werden, der lediglich öffentliche Daten manipulieren kann, kann mehreren Zeitpunkten durchgeführt werden und benötigt lediglich einen unzuverlässigen Fehlerangriff. Zusätzlich zeigen wir, wie Belief Propagation verwendet werden kann, um den geheimen Schlüssel aus Entschlüsselungsfehlerinformation zu gewinnen. Anschließend geben wir dann mögliche Gegenmaßnahmen an, die unseren Angriff abschwächen können.

Schließlich erklären wir, wie der geheime Schlüssel aus Enschlüsselungsfehlerinformation in fehlertorleranter Weise mit einem Verfahren, das Sicherheitsabschätzungen zulässt, gewonnen werden kann. Wir modifizieren den Belief Propagation basierten Ansatz, um Fehlertoleranz zu erreichen und erklären, wie Belief Propagation mit Gitterreduktion verbunden werden kann. Unsere Methode liefert bessere Ergebnisse als frühere Algorithmen in Bezug auf benötigte Information, ist fehlertolerant, und gibt Sicherheitsabschätzungen in allen Fällen, in denen der geheime Schlüssel nicht direkt vollständig zurückgewonnen werden kann. Diese Sicherheitsabschätzungen ermöglichen das Design und die Evaluation von Gegenmaßnahmen und erlauben eine Einschätzung der Auswirkungen mehrere Angriffe, die auf Entschlüsselungsfehlern basieren.

# Contents

Chapter 1

# Introduction

The need to secure and authenticate communication has existed for millennia. From simple substitution of letters and handwritten signatures to complex machinery – humanity has always searched for algorithms verifying authenticity and keeping information secret. Accessing encrypted information has been equally crucial, especially during wartime. For example, breaking the "Enigma" machine using a device co-developed by the mathematician Alan Turing – who also laid the foundation for modern computers – led to enemy communications being accessible by allied troops. While thus even historically the significance of cryptography cannot be overstated, it affects almost every aspect of everyone's life in the modern connected world.

Computers communicating over the Internet send their information over various hops until it reaches their intended destination. Without encryption and authentication, everyone could access and manipulate all messages passing through their servers. But not only devices connected to the Internet require secured cryptography. Devices such as smartcards, digital locks, and even modern car keys use cryptographic schemes to keep unauthorized users from accessing the corresponding service. If current cryptography was to be broken, our digital world would be entirely defenseless, and the majority of applications would not be appropriately secure anymore and soon cease to exist. From private social media accounts to online banking and smartcards opening the doors to secured areas to military communications – every connected digital service could be accessed by those being able to break current schemes. The digital world fully relies on modern cryptography being appropriately secure.

Fortunately, current cryptographic schemes are widely considered appropriately secure even against state-level attackers. Asymmetric cryptography is commonly used to authenticate and establish a shared secret between two parties willing to communicate, and symmetric cryptography provides the confidentiality and integrity of further exchanged information. A simple example to illustrate the difference between and applications of asymmetric and symmetric cryptography is depicted in Figure 1.1. Asymmetric cryptography relies on one-way functions, which are efficient to compute but computationally very expensive to invert. It is, for example, efficient to multiply large prime numbers but no fast algorithm for factorization running on classical computers is known. Cryptographic schemes that base their security on these kind of hard problems might be flawed in how they are implemented, but the algorithms themselves are unlikely to be broken without significant technological advances.

The advent of quantum computers in the early twenty-first century gave rise to concerns regarding the long-term security warranted by established problems. A large-scale quantum computer could break hard problems such as integer factorization or discrete logarithm problems using Shor's algorithm [Sho94, Sho97]. This would allow anyone in possession of such a device to break commonly used asymmetric cryptography. Grover's search [Gro96], another quantum

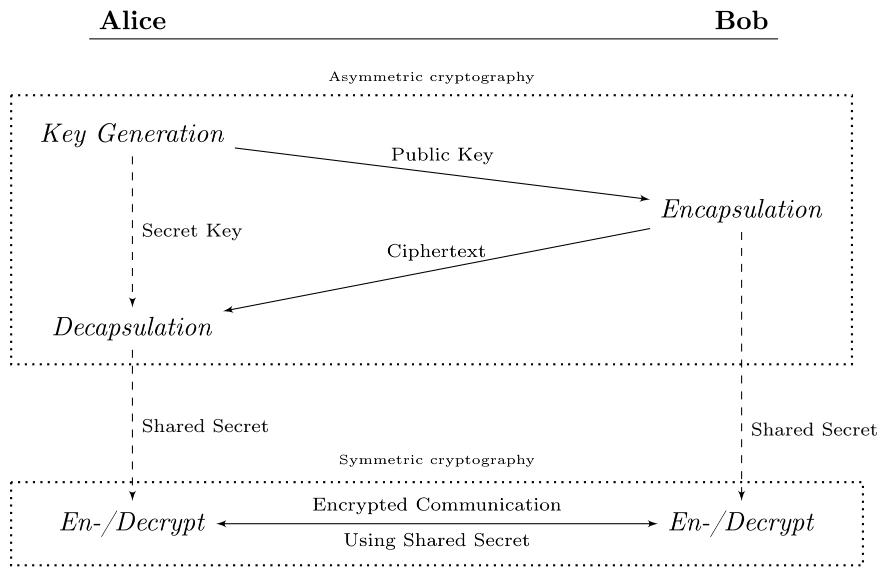


Figure 1.1: Simplified usage of symmetric and asymmetric cryptography. A key encapsulation mechanism (asymmetric) is used to establish a shared secret from which a common but secret key is derived. This enables Alice and Bob to communicate using symmetric cryptography.

algorithm, could lower the security of symmetric schemes by about half of their current bit security. While the vulnerability of symmetric cryptography to quantum-computing can be mitigated by using longer keys, asymmetric cryptography relying on many currently used problems is likely to be completely broken (see e.g., [BL17]). Considering the importance of cryptography, this calls for schemes that rely on different, “quantum-secure” problems.

Current quantum computers do not possess the computational capacities to break cryptographic schemes in use [NistPqc, BsiQu, BDH+21a]. It is unclear how fast current quantum technology will improve and when a major breakthrough that accelerates development will happen. But even though such developments might take several decades, the migration to post-quantum cryptography has to happen sooner because many products and currently processed information need to stay secure beyond this time frame. Sensitive information recorded today and being decrypted many decades later could still have serious implications. Additionally, protocols and products that are in development today might still be in use several decades later – to replace them for security reasons could be a major problem in many cases. Therefore, achieving post-quantum security is a pressing matter that cannot be postponed.

In 2016, the National Institute of Standards and Technology (NIST) initiated a standardization process for post-quantum cryptography aiming to develop quantum-safe standards for the future of secure communications [NistCfp]. The following years brought many new cryptographic algorithms based on several hard problems falling into different categories, each with its own merits and caveats. The process is currently in the fourth round [NistCfpv4], and a supplementary call [NistSig] for newly submitted signatures has started.

After several schemes have been ruled out, many of them either broken or with severe disadvantages, NIST selected several schemes for standardization after the third round of the process [NistR3]. These include one single primary candidate as *key encapsulation mechanism*, used to exchange keys for, e.g., use with symmetric cryptography, and three signature schemes used to verify the authenticity of a message. Other “alternate candidates” might also be standardized to allow for a comprehensive choice depending on the individual use case [NistR3].

Several candidates in the NIST process were based on hard lattice problems [NistR1]. Lattices are algebraic structures from which several computational problems arise for which no classical or quantum algorithm for efficient solving is known. From these problems, for example from variants of the *(unique) shortest vector problem* and *closest vector problem*, cryptographic schemes have been derived for several decades. An early example is the NTRU scheme [HPS98] of which several variants have been submitted to the NIST process [NistR1]. An example of a 2-dimensional lattice and the shortest and the closest vector problem is depicted in Figure 1.2.

Schemes based on the *learning with errors* problem are among the most efficient candidates in the NIST process in terms of performance and key sizes. The learning with errors problem was introduced by Regev in 2005 [Reg05, Reg09] and can be reduced (randomized) to lattice problems that are assumed to be hard even for a quantum computer [Reg05, Reg09, Pei09a, APS15]. Especially the *ring-* [LPR13] and the *module learning with errors* problems [BGV14, LS15], which are more structured variants of learning with errors, lead to efficient algorithms. The scheme selected for standardization by the NIST, Kyber [BDK+18, ABD+21b], is based on the module learning with errors problems and therefore lattice-based. This makes lattice-based cryptography and especially learning with errors based protocols particularly interesting for embedded devices, which are usually more constrained in terms of performance and key sizes.

While the process is running, some lattice-based post-quantum schemes have been tested and are already used in practice. For example, NewHope [ADPS16b] has been used in a test run in the Google Chrome browser in combination with Transport Layer Security (TLS) to secure key exchanges in an experiment conducted in 2016 [Bra16]. NewHope is a lattice-based scheme that is similar, but based on a slightly different problem, to the now selected algorithm Kyber. Kyber itself is available for certain Amazon web services and has been evaluated in this context for hybrid TLS [Jar22], a recent release of libsignal (powering the Signal messenger) implements Kyber [Sig23], and the latest version of the Chrome browser uses hybrid key exchanges for TLS that combine elliptic curve cryptography and Kyber [OBr23].

It has been shown that schemes based on these problems are suitable in practical use cases with full protocols even on constraint devices, e.g., in [HPS+20], and several – secured and unsecured – implementations of learning-with-error-based schemes exist, see for example [OG17, BGR+21, FBR+22, KSSW22, HKL+22, BBC+23]. But also in settings such as TLS, the performance of lattice-based cryptography has been evaluated [Pei14, BCNS15, SSW20, GW22] and is already used in practice [Jar22, OBr23]. The currently primarily selected scheme for key encapsulation and a selected scheme for signatures are module learning with errors schemes, additionally drawing attention to learning with errors and especially to module learning with errors.

## 1.1 Problem Description

As learning with errors based schemes rely on different hard problems compared to currently used cryptography, they come with different properties and are constructed from distinct kinds of building blocks. For example, in learning with errors schemes, fast multiplication algorithms are often realized using a *number theoretic transform*. In terms of scheme design, key encapsulation mechanisms based on the learning with errors problem are often constructed from a *public key encryption* scheme using a *Fujisaki-Okamoto transform* [FO99, FO13, TU16, HHK17] to obtain a *key encapsulation mechanism* and achieve security against chosen-ciphertext attacks. Another notable property is the usage of an *error correction* to remove noise from a message. The error correction may in many schemes fail with very low probability even without manipulation.

The theoretic security of lattice-based schemes has been thoroughly analyzed and is debated

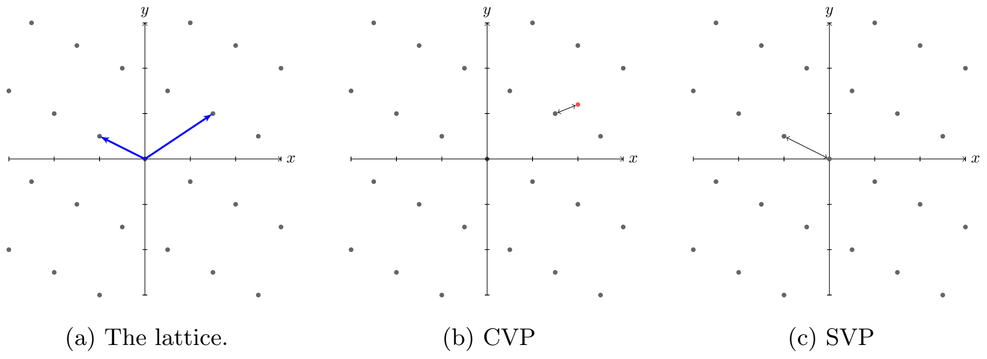


Figure 1.2: Lattice problems in two dimensions. Figure 1.2a shows a two dimension lattice. The closest vector problem (Figure 1.2b) asks to find the closest lattice element to a given point. The shortest vector problem (Figure 1.2c) is posed as finding the shortest non-zero vector in a lattice.

extensively (see, e.g., [BDH+21a, BsiPqc, NistR3, PqcFo]). But whenever cryptographic schemes are used in practice, side-channel and fault attacks are an attack vector that has to be considered. The threat posed by physical attacks on embedded devices has been a major concern since the seminal work by Kocher [Koc96] on timing attacks and by Kocher et al. [KJJ99] on differential power analysis in 1999. For example, an application may leak through a side-channel such as timing, electromagnetic radiation, or power consumption, or an adversary may manipulate execution by introducing a fault. Implementation security is especially relevant when considering embedded or mobile devices as these kinds of use cases often allow an attacker to – at least temporarily – gain access to a device running a cryptographic algorithm that contains secrets that the adversary wants to extract. As lattice-based cryptography utilizes entirely different building blocks and deviates from classic algorithms in many regards, migrating to learning with errors schemes opens up new implementation vulnerabilities (as already shown e.g., in [PPM17, GJN20, RRCB20, RBRC20]) and requires analysis before post-quantum cryptography can be deployed on a large scale.

### 1.1.1 Research Objective

Securing key exchange schemes in the near future is particularly important: An adversary may record current encrypted data for later decryption by a large-scale quantum computer. In addition, post-quantum key Key Encapsulation Mechanisms (KEMs) have been proven to be practical in terms of performance in several use cases, e.g., [Bra16, SSW20, HPS+20, Jar22, GW22]. Moreover, the standardization process is near its end and NIST has already selected the KEM Kyber for standardization [NistR3].

The secret key of the KEM, which is the long-term secret in non-ephemeral key establishment protocols, is a particularly interesting target as it may allow an attacker, depending on the protocol, to obtain not only the current but also past and future session keys. The long-term secret may be targeted during the key generation and the decapsulation routine. The key generation is only called once per key pair and may be performed offline or while the device is not under the control of an attacker. Therefore, the building blocks of the decapsulation are especially valuable targets. Evaluating their security, finding conceptual attack strategies, and designing countermeasures to these threats is fundamental to the migration and widespread adoption of post-quantum cryptography. For these reasons, this thesis focuses on the decapsulation routine in Learning with Errors (LWE) based key exchange schemes with a focus on the following areas.

### The Number Theoretic Transform

The *number theoretic transform* is an algorithm enabling fast multiplication. This routine is used in several lattice based schemes, e.g., [ADPS16b, DKL+18, BDK+18, LS19], including Kyber. The usage of a Number Theoretic Transform (NTT) has also been proposed for schemes where the underlying algebraic structure does not directly allow for the usage of an NTT [CHK+21, ACC+21]. In particular, the inverse NTT processes the secret key during the decapsulation; the input depends on the ciphertext, which allows an attacker to influence its execution. Vulnerability of the number theoretic transform against side-channel attacks has been shown in [PPM17, PP19]. These works show that the NTT may be targeted during key generation, encapsulation, and decapsulation and require only a single trace after having created a template. The vulnerability of the NTT is not a purely theoretical matter, and attacks have been verified on physical devices. But in current attacks the noise tolerance is reduced when targeting the long-term secret or a masked[1] implementation (compared to targeting the session key in an unprotected setting). This means, in a realistic setting, the adversary needs to be able to take precise measurements of intermediate variables[2] to recover the long-term secret. It is unclear whether the long-term secret can be extracted in masked settings with increased measurement noise.

To understand the degree of vulnerability of the NTT, is necessary to quantify the capabilities required by an adversary to launch an attack. Therefore, finding and – in a second step – mitigating attack strategies allowing for attacks with increased noise-tolerance on the NTT is of high relevance to the security of affected lattice-based schemes. Moreover, shuffling countermeasures, e.g., presented in [RPBC20], could prevent these attacks, and adaptations for these types of attacks are not yet known. The extent of vulnerability of the number theoretic transform is therefore a major open question with regard to implementation security. In our first research question, we aim to give an answer to this issue.

### Error Correction and Fujisaki-Okamoto Transform

Current lattice-based schemes make use of an error correction to recover the message (from which, e.g., the session key is derived) from a noisy polynomial. The error correction is used during the decryption, which is called in the decapsulation routine. Even though the error correction recovers the session key, the input to the error correction directly depends on the long-term secret. With very high probability, the noise is small enough to allow for correct recovery. In rare cases, a *decryption failure* may occur, which allows an observer to obtain information about the secret key; this is taken into account by security estimates. Decryption failures may also be caused by the usage of a manipulated ciphertext; if an adversary can potentially cause and then observe whether a decryption failure happens, they obtain information about the secret key. Previous schemes have been attacked, and in some cases even broken without the usage of a side-channel, using this attack strategy [JJ00, HNP+03, Flu16, BBLP18, GJY19, BGRR19]. A *Fujisaki-Okamoto Transform* [FO99, FO13, TU16, HHK17] prevents chosen-ciphertext attacks in the IND-CCA2 model and thereby, in particular, prevents the observation of decryption failures caused by a chosen-ciphertext. Invasive attacks target the comparison of the Fujisaki-Okamoto (FO)-transform and thereby remove the protection the FO-transform offers – the need to protect the comparison and similar operations against faults is therefore well-known [VOGR18, OSPG18, BGRR19, XIU+21]. Another attack path has been the exploitation of either side-channel leakage [BGRR19, GJN20, RRCB20, BDH+21b, DHP+22] during the error correction. Note that countermeasures mitigating these attacks have already been provided in the respective work. Nevertheless, an attack similar to [GJN20, BDH+21b, DHP+22] applies whenever side-channel leakage allows

[1]Masking is a countermeasure, see Section 2.3.3.
[2]For a high-level comparison of required noise levels see Table 1.1.

observing the outcome of the decryption subroutine during the decapsulation. Moreover, in these attacks, the ciphertext only differs by a single bit from a valid ciphertext and is therefore hard to detect. The attack of [PP21] applies a fault – but only a fault and no chosen-ciphertext – and can fully recover the secret key in practice. However, it is focused on a particular implementation of the decoder and requires a reliable fault on a single unprotected location.

Thus, regarding the area of invasive attacks, it is unclear whether the error correction may be targeted with an unreliable fault that can be administered in various locations, i.e., allows for a large attack surface, and is independent of the exact implementation. Therefore, it is mostly an open question under which conditions and to what extent implementation attacks allow to exploit decryption failures in the presence of an FO-transform. The second research question focuses on attacks that exploit decryption failures and target the FO-transform.

**Recovery Methods for Decryption Failure Information**

Side-channel and fault attacks commonly make use of statistical information obtained during the attack. Veyrat-Charvillon et al. [VGS14] proposed the usage of a coding theoretic approach, and since then, many attacks rely on belief propagation, e.g., [PPM17, GRO18, PP19, KPP20]. Using belief propagation, an adversary may recover the secret key if a sufficient amount of information has been obtained, and belief propagation has also previously been combined with an algebraic step that, for example, exploits the structure of an NTT [PPM17]. In many cases, however, there is currently no technique to make use of the underlying lattice problem, which is posed by the public key equation and targeted by classical attacks on learning with errors schemes. In particular, the recovery of the secret key from decryption failure information that occurs in attacks such as, e.g., [BDH+21b, PP21, HPP21, DHP+22][3] yet misses a comprehensive *recovery method* which combines the advantages of different approaches. The side-channel information framework of Dachman-Soled et al. [DDGR20, DGHK22] considers different kinds decryption failure information and offers estimates for information as occurring in the attacks of [BDH+21b, PP21, HPP21, DHP+22]. However, full key recovery in these latter cases is computationally expensive, and estimates indicated that it requires more information than current statistical algorithms (see [DHP+22, Section 3.1]).

Previous statistical methods [PP21, HPP21, Del22] can practically recover the secret key from sufficient information. However, those methods either do not allow for error tolerant recovery or require more information than necessary; in addition, those approaches do not offer security estimates in case of partial attacks. Error tolerance enables attacks in which information about decryption failures cannot be classified with certainty – a property often required in real-world scenarios. Estimates on the remaining security after an attack on an instance of a scheme are particularly important to understand the impact of attacks and design appropriate countermeasures. Unfortunately, current methods allowing for error-tolerant key recovery do not achieve the performance in terms of required information that other methods offer, and practical methods do not offer security estimates at all. This has several implications: The key can only be recovered if no incorrect information is obtained from the attack – often an unrealistic assumption –, or the adversary needs to obtain more information than other methods would require. Further, in case an adversary retrieves insufficient information to recover the secret key, the current state of the art does not allow to estimate the remaining security.

The lack of security estimates leads to uncertainty about the impact of attacks exploiting decryption failures, e.g., [BDH+21b, PP21, DHP+22, Del22], and could result in the deployment of insufficient countermeasures that do not achieve the required security level. The issue of security estimates for decryption failure information is covered in the third research question.

[3]Note that these attacks obtain different information than failure boosting attacks (e.g., [DVV19]).

### 1.1.2 Research Questions

From the problems described in the previous section, we derive the following research questions:

**Research Question 1: To what extent is the number theoretic transform vulnerable to side-channel analysis?**

Fast multiplication in several lattice-based schemes, e.g., [ADPS16b, DKL+18, BDK+18, LS19], including Kyber, is realized through the usage of a so-called NTT. This procedure is used throughout the whole scheme and therefore offers multiple points of attack. In particular, the inverse NTT of the product of the long-term secret and a component of the ciphertext is computed during every key exchange, making it a valuable target. The vulnerability of the NTT to side-channel analysis has previously been shown [PPM17, PP19]. These attacks are practical, have been carried out on physical devices, and, after a template has been recorded, only require a single-trace. Current attacks, however, do not yet target the long-term secret or masked settings under high-noise conditions. Moreover, shuffling countermeasures are not considered and prevent these attacks. A general attack strategy targeting the long-term secret that considers high noise levels and an assessment of vulnerabilities of the NTT, especially in the presence of countermeasures, is yet missing. Understanding the security of the NTT, in regard to the required noise level as well as countermeasures, is of high relevance as the long-term secret is processed over a long execution time during the decryption routine.

**Research Question 2: How can implementation attacks target the error correction and exploit decryption failure leakage in the presence of an FO-Transform?**

In LWE-based schemes, during the decryption step, the message has to be recovered from noisy polynomial coefficients – small errors have to be corrected. The error correction and the FO-transform have previously been identified as a target (see, e.g., [VOGR18, OSPG18, BGRR19, GJN20, XIU+21, BDH+21b, PP21, DHP+22]). An attack strategy for side-channel analysis already exists and has been exploited in [GJN20, BDH+21b, DHP+22]. However, current invasive attacks require an implementation to be vulnerable or unprotected in a specific location, have a small attack surface, or they require a reliable fault. In addition, these attacks target operations that are already considered sensitive (e.g., the comparison operation of the FO-transform or the error correction of the session key), and no attack strategies targeting only public data are known. It is yet unclear if decryption failure leakage can be utilized for more general fault attacks and to what extent the FO-transform in lattice-based cryptography is vulnerable. The error correction processes the secret key and may be targeted using a chosen ciphertext. Therefore, understanding the impact of decryption failures in regard to side-channel and fault attacks is of high importance.

**Research Question 3: Which techniques allow for key recovery from partially leaked decryption failure information?**

Side-channel and fault attacks often require a key recovery step because recorded information does not directly reveal the key but, instead, allows for the key to be retrieved from it. In particular, solving for the secret key from decryption failure leakage requires sophisticated methods as the information occurs in terms of inequalities in a high-dimensional space. This type of key recovery from decryption failure leakage is of high relevance for security of lattice-based cryptography, as such information occurs in a variety of attacks such as [PP21, BDH+21b, HPP21, Del22]. Several techniques exist [DDGR20, PP21, HPP21, Del22, DGHK22], each with their own merits and caveats, using either statistical or algebraic methods but not both. Current recovery methods are either not error-tolerant, i.e., they fail in the presence of incorrect information, need substantially[4]

[4]This is quantified in Table 1.3 as well as Section 7.2.2.

more information than required by other algorithms, or do not allow for security estimates. In addition, security estimates are not available for any practically used method in this area, and other means overestimate the remaining security, i.e., underestimate the loss of security caused by these attacks. The lack of a practical approach that allows for security estimates leads to uncertainty about the impact of attacks that exploit decryption failures. Therefore, it is unclear what countermeasures are needed to achieve the required level of security. As a consequence, attacks could be underestimated, and deployed countermeasures might not achieve the targeted level of security.

## 1.2 Contributions

This thesis builds upon several previously published works the author contributed to. The author of this thesis was a main contributor to each of the following work. The authors of [HHP+21] and [HSST23] are mentioned in alphabetic order[5]; in [HHP+21] the role of main contributor is shared with several other researchers, mainly Silvan Streit, and in [HSST23], it is shared with Silvan Streit. The first research questions is addressed in [HHP+21] and [HSST23]. The second research question is mainly addressed in [HPP21] with some improvements in [HMS+23], and the third research question is addressed in [HMS+23].

In [HHP+21], we show how to exploit algebraic properties of the NTT in order to enable noise-resistant side-channel analysis. A chosen-ciphertext, which is crafted using either lattice reduction or a less costly but also less effective algorithm, is first sent to the device under attack. This ciphertext is created such that decompression and subsequent transformation into NTT domain result in a sparse polynomial. The pointwise multiplication with the secret then reduces the entropy inside the inverse NTT, allowing for improved side-channel analysis. A subsequent belief propagation that is similar to the belief propagation used in [PP19] but works in the setting of attacking the decapsulation and thereby targeting the secret key, can then recover parts of the key. Using the algebraic structure of the NTT – essentially a computation of the Chinese remainder theorem – additionally allows for recovering even more coefficients of the key. This can be achieved by using lattice reduction or, again, by using a less expensive but also less effective algorithm. We evaluate our proposed attack strategy and give results in terms of measurement-noise tolerance. The noise tolerance in terms of standard deviation of the measurement error when targeting the long-term secret is increased by a factor of up to greater than five. We thereby give an answer to the first research question by stating an attack on the NTT in lattice based schemes, giving a strategy to reduce the entropy during the inverse NTT using a chosen-ciphertext which increases the noise tolerance, and explaining how to recover the secret key in such attacks. Further, we provide an evaluation of the vulnerability of the inverse NTT during the decryption routine which processes the long-term secret.

Following up with the previous analysis of susceptibility of the NTT to side-channel analysis, we examine the impact of hiding countermeasures in [HSST23]. A standard masking countermeasure, e.g., the one used in [RRVV15, RRC+16, OSPG18], affects the attack of [HHP+21] only to a low degree. This leads to the question of the impact of standard hiding countermeasures on the attack proposed in [HHP+21]. Several countermeasures have already been proposed by Ravi et al. in [RPBC20]; these include three hiding countermeasures which are realized by shuffling the computations inside the NTT. We propose several ways of adapting to these countermeasures. The first one is a computationally inexpensive method that is realized by mixing priors and counters the lowest degree of protection proposed by [RPBC20]. A more sophisticated method, a

[5]This is common in the areas of cryptography and mathematics, see https://www.ams.org/profession/leaders/CultureStatement04.pdf.

so-called shuffle node, targets the same countermeasure and allows for higher noise resistance. Shuffle nodes are adaptive nodes that modify the belief propagation algorithm based on the processed information. Finally, we adapt the attacker model and develop a matching algorithm that allows adapting to higher levels of protection but is computationally expensive and has higher requirements on the attacker. Thereby, we complement our answer to the first research question by taking countermeasures into account, providing an analysis of their impact, and explaining general strategies to adapt such attacks to countermeasures. A high-level comparison to previous work is given in Table 1.1.

Table 1.1: Comparison of [HHP+21] to previous work (targeting Kyber or [LPR13]) in terms of noise tolerance, target of the attack, and the consideration of countermeasures. The noise tolerance is given as standard deviation of the error distribution. Table as depicted in [Her23b].

| Work | Noise Tolerance | Long-Term Secret | Countermeasures |
|---|---|---|---|
| Primas, Pessl, and Mangard [PPM17] | $\sigma \leqslant 0.6$ | Yes | Masking |
| Pessl and Primas [PP19] | $\sigma \leqslant 2.0$ | No | Masking |
| This Thesis [HHP+21, HSST23] | $\sigma \leqslant 1.7$ (3.1)[6] | Yes | Masking and Hiding |

In [HPP21], we show how decryption failures can be exploited using the combination of a chosen ciphertext and a fault attack. A chosen ciphertext contains an additional term that potentially – depending on the coefficients of the secret key – causes a decryption failure in the decryption routine. The FO-transform prevents an attacker to observe such failures and thus, without an implementation attack, no information can be obtained (in the appropriate model). Whenever an implementation attack allows observing such decryption failures, an adversary may retrieve inequalities over the secret key by using the chosen ciphertext also used in this work. Previous work [GJN20, BDH+21b] exploited a faulty comparison operation using a similar technique or used a fault to directly cause a decryption failure [PP21]. In this work, we show that a fault may be used to turn the FO-transform – in place to prevent chosen-ciphertext attacks – into a decryption failure oracle leaking inequalities that hold secret information. As noted in [DHP+22], our approach is comparable to the concept of safe-error attack [YJ00] in symmetric cryptography. Subsequently, the secret key has to be recovered from those inequalities. For this purpose, we present a new technique using belief propagation which improves upon the previous technique of [PP21]. It allows working with a very unreliable fault, requires to target only public data, and may be administered at a wide variety of locations. We thus answer the second question by explaining how the FO-transform may be turned into a decryption failure oracle using a fault, which is a more generally applicable attack strategy. A high-level comparison to previous work is given in Table 1.2.

As decryption failure leakage occurs in a variety of attacks, for example in [PP21, BDH+21b, HPP21, Del22], several recovery methods, such as [DDGR20, PP21, HPP21, Del22, DGHK22], exist. Every single recovery method comes with advantages and caveats. Until now, no recovery method offered a minimum of required inequalities, error tolerance, security estimates, and practicability on commonly available hardware. In [HMS+23], we show that a combination of statistical and algebraic tools offers all those properties in the case of the most commonly occurring types of decryption failures. By first using an improved belief propagation that achieves error tolerance, we can recover the secret key in all cases where previous algorithms could. If the given information was not sufficient for immediate recovery, we explain how belief propagation

[6]The number in brackets applies only to the highest Kyber security level.
[7]The work of [Del22] is a follow-up up attack to our work further enlarging the attack surface.
[8]A fault that may fail or targets incorrect locations; our attack may still make use of such a fault.

Table 1.2: Comparison of [HPP21] to previous work (targeting Kyber or FrodoKEM) in terms of type of attack (side-channel analysis (SCA) or fault), point of attack, and requirements, and robustness. Grayed out entries were published after our work. Table adapted from [Her23b].

| Work | Type | Attack Vector | Requirement/Robustness |
|---|---|---|---|
| Guo et al. [GJN20] | SCA | Comparison | Timing Leakage |
| Bhasin et al. [BDH+21b] | SCA | Comparison | Faulty Comparison |
| Pessl and Prokop [PP21] | Fault | Decoding | Reliable Fault |
| D'Anvers et al. [DHP+22] | SCA | Comparison | Max. First Order Protection |
| Delvaux [Del22][7] | Fault | Multiple | Unreliable Fault |
| Fahr et al. [FKK+22] | Failure Boosting | Key Generation | Rowhammer on Key Generation |
| This Thesis [HPP21] | Fault | Multiple | Unreliable Fault[8] |

output may be integrated into lattice problems. Thereby, we can further lower the number of inequalities and, in addition, give estimates on the remaining security for partially successful attacks. The work of [HMS+23] thereby answers the third research question of how to work with partially leaked information in regard to decryption failure leakage, and, in addition, it provides a more general applicable technique for attacks on lattice-based cryptography utilizing belief propagation. A high-level comparison to previous work is given in Table 1.3; by being applicable, we denote that a method has been used in practice for end-to-end key recovery in an attack on Kyber retrieving inequalities such as, e.g., [BDH+21b, PP21, HPP21, DHP+22].

Table 1.3: Comparison of [HMS+23] to previous work in terms of required inequalities in Kyber512, applicability to decryption failure information as arising in the aforementioned attacks, error tolerance, and availability of security estimates. Table as depicted in [Her23b].

| Work | Inequalities | Applicable | Error Tolerant | Estimates |
|---|---|---|---|---|
| Pessl and Prokop [PP21] | 7500 | Yes | No | No |
| Delvaux [Del22] | 8500 | Yes | Yes | No |
| Dachman-Soled et al. [DDGR20] | $\geqslant 10000$[9] | No | No | Yes |
| Dachman-Soled et al. [DGHK22] | n.a. | No | No | Yes |
| This Thesis [HMS+23] | 5500 | Yes | Yes | Yes |

**Approaches used to answer the research questions.** To exploit algebraic properties of the scheme under attack, an adversary often needs to interact with the device in an active, as opposed to a purely measuring-observing, way. Therefore, the techniques to answer the first and the second research question include *chosen-ciphertexts*, i.e., a dishonestly generated ciphertext crafted to manipulate the device into certain computations that expose information. Lattice problems do not only appear in the security proofs of learning with errors but are also useful to recover the secret key in a variety of scenarios when information was obtained from implementation attacks. Thus, the technique of *lattice reduction* – used to find a short vector in a lattice – appears in several chapters. Furthermore, throughout this work, the *belief propagation* algorithm is used as a statistical tool allowing efficient processing of probability information. In fact, we show that belief propagation may be used to recover the secret key from decryption failure information. Further, we provide several techniques related to belief propagation in the context of Soft Analytical Side-Channel Attack (SASCA) [VGS14] and solving inequalities; these methods

[9]As estimated in [BDH+21b] using obtained approximate equations.

allow to circumvent countermeasures, to deal with incorrect information, or to combine belief propagation with lattice reduction. As Kyber is the primary KEM selected for standardization by the NIST, we evaluate our findings on the example of Kyber.

## 1.3 Thesis Organization

This thesis is structured as follows: In Chapter 2 we introduce concepts such as cryptographic schemes in general, in Section 2.1 we give an overview of lattice-based cryptography in Section 2.2, and explain implementation attacks in Section 2.3. The chapter contains a brief overview of the necessary background required to state our results, and it introduces notation.

In Chapter 3, we provide an overview of the current state of the art regarding our research questions. The chapter describes directly related work; in contrast to Chapter 2, it goes into more detail where this is required to understand our work and the current state of the art.

Chapter 4 addresses the first research question by proposing and analyzing an attack strategy that targets the NTT using a chosen ciphertext. We answer the first research question by proposing an attack scheme on the NTT which improves upon previous work by targeting the long-term secret while maintaining a high noise tolerance.

Chapter 5 addresses the second research question; we provide a fault-enabled chosen-ciphertext attack on Kyber which is an instantiation of a more general class of implementation attacks. We explain how a fault may be used to turn the FO-transform into a decryption failure oracle – this allows for a variety of concrete instantiations of fault attacks and thereby answers the second research question.

Chapter 6 addresses the third research question by providing a new recovery method for decryption failure leakage, which, in particular, explains how to integrate belief propagation output into a lattice problem. In addition, we provide security estimates and state a more generally applicable way to integrate belief propagation output into lattice problems occurring in LWE schemes. Such kinds of leakage – both decryption failure and general belief propagation output – occur in a variety of attacks; therefore, improving the recovery method impacts the implementation security of LWE schemes.

We provide evaluation of our attacks and methods in Chapter 7, and we compare them against the previous state of the art. Finally, Chapter 8 concludes the thesis and discusses future work. The outline of the thesis is depicted in Figure 1.3.

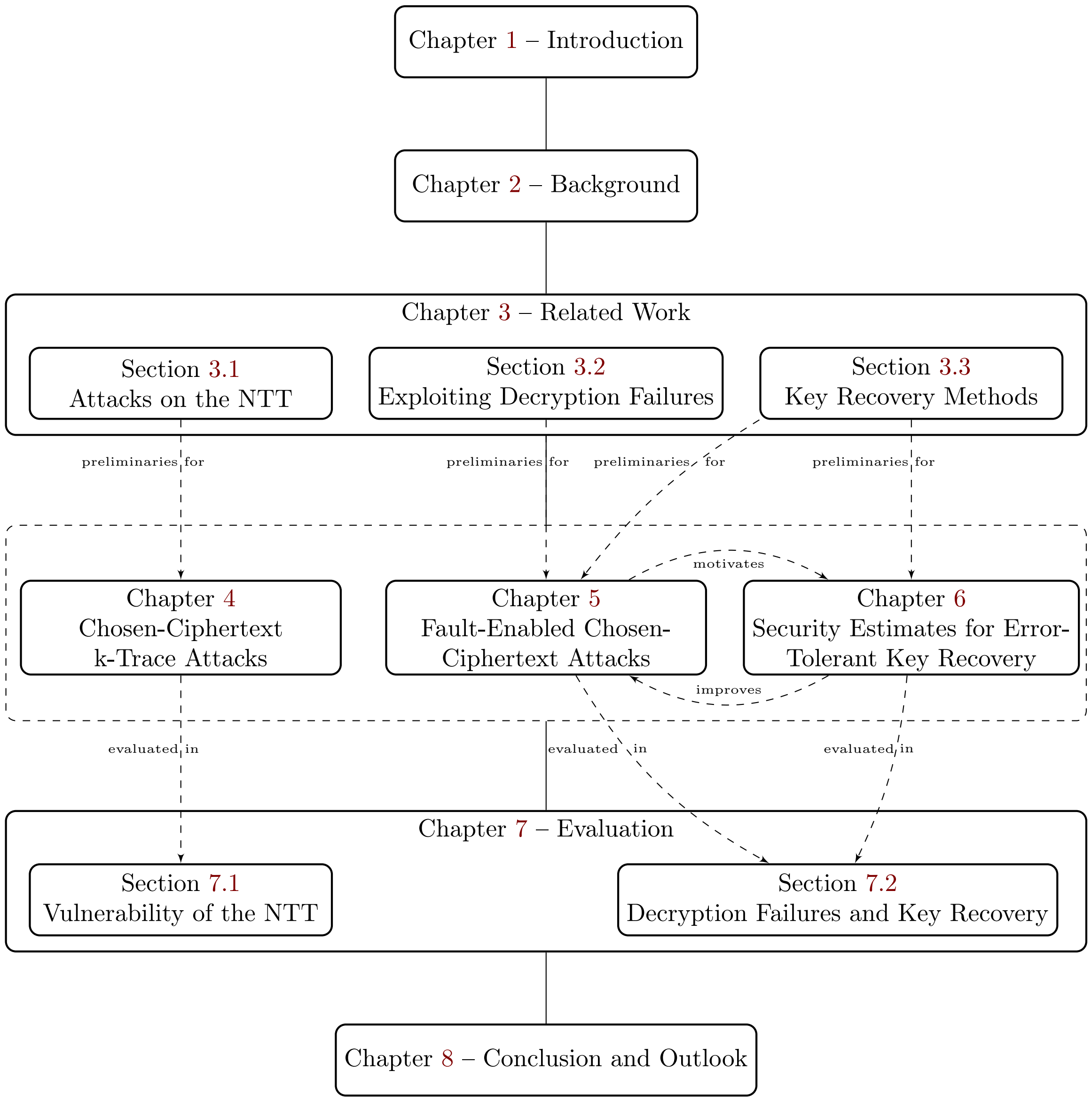

Chapter 1 – Introduction
Chapter 2 – Background
Chapter 3 – Related Work
Section 3.1 Attacks on the NTT
Section 3.2 Exploiting Decryption Failures
Section 3.3 Key Recovery Methods
preliminaries for
preliminaries for
preliminaries for
preliminaries for
motivates
Chapter 4 Chosen-Ciphertext k-Trace Attacks
Chapter 5 Fault-Enabled Chosen-Ciphertext Attacks
Chapter 6 Security Estimates for Error-Tolerant Key Recovery
improves
evaluated in
evaluated in
evaluated in
Chapter 7 – Evaluation
Section 7.1 Vulnerability of the NTT
Section 7.2 Decryption Failures and Key Recovery
Chapter 8 – Conclusion and Outlook

Figure 1.3: Outline: The left column of the figure provides an answer to the first research question, the middle column to the second research question, and the right column to the third research question. However, the evaluation of the second and third research question make use of similar methodologies and are both presented in Section 7.2

.

# Chapter 2

# Background

The National Institute of Standards and Technology (NIST) contest [NistCfp] aims to find post-quantum replacements for current asymmetric schemes with a focus on signature and key establishment schemes. In this work, we state attacks on lattice-based key encapsulation algorithms with a focus on the to-be standardized scheme Kyber [BDK+18, ABD+21b]. In this chapter, we establish the necessary background and give commonly used definitions and notations.

This includes fundamentals of cryptographic key exchanges, focusing on Public-Key Encryption (PKE) and Key Encapsulation Mechanism (KEM), and common security notions regarding chosen-ciphertext attacks. Symmetric encryption is not repeated in this work; for this, and in general, for a more extensive introduction, we refer to [PP10].

After having established the necessary basics of cryptographic schemes, we turn to lattice-based cryptography. This kind of cryptography is very relevant in the context of Post-Quantum Cryptography (PQC), and we reiterate a main problem, the Learning with Errors (LWE) problem [Reg05]. To give an introduction to the LWE problem, we first require some mathematical background. Then, LWE and the sub-problems Ring Learning with Errors (RLWE) [LPR13], and Module Learning with Errors (MLWE) [LS15] are reiterated. We explain how common schemes, including the NIST standardization contest winner Kyber, are defined. If such algorithms are running on embedded devices, an important factor is implementation security. Therefore, we summarize implementation attacks such as Side Channel Attacks (SCAs) as well as fault attacks.

For more comprehensive introductions, we refer to [PP10] for cryptographic schemes in general, to [BL17] and [BDH+21a] for an overview over post-quantum cryptography, to [Pei16, BsiPqc, BDH+21a] for lattice-based cryptography, and to [MOP07] and [JT12] for side-channel and fault attacks.

**Notation.** In this thesis, the natural numbers contain 0, that is $0 \in \mathbb{N} = \mathbb{N}_0$. We generally count beginning with 0, and indices start at 0. Finite fields are denoted by $\mathbb{F}_q$ where $q$ is a prime power; in the case of $q$ being prime, this means that $\mathbb{F}_q = \mathbb{Z}/q\mathbb{Z}$. Polynomial rings over a ring $R$ are denoted as $R[x]$ where $x$ is the variable (and might also be replaced by $y$ or any other letter). Depending on the context, there usually is a base ring we work over, in the case of Kyber this is $\mathbb{F}_q[x]/(f)$ with $f = x^n + 1$ where $n = 256$.

We denote vectors and matrices over the corresponding base ring in bold letters, for example, $\mathbf{e}$ for a vector over $R$ for some ring $k$, i.e., $\mathbf{e} \in R^k$ for some $k \in \mathbb{N}$ with $k > 1$. The coefficients of a vector or matrix are denoted by subscripts (ignoring co- and contravariance) and are not bold. For a vector $\mathbf{e} \in R^k$, we write $e_0$ for the first component, $e_1$ for the second component, and so on. For a polynomial, we denote the coefficient in the same way, unless the polynomial is denoted as a component of a vector of polynomials. This means, let $R = \mathbb{F}_q[x]$, $v \in R$, and $\mathbf{b} \in R^k$ for some

$k \in \mathbb{N}$ with $k > 1$, then the first coefficient of $v$ is denoted as $v_0$, but the second coefficient of the third component of $\mathbf{b}$ is denoted by $(b_3)_2$. To illustrate this, let $k = 2$,

$$v = 1 + 2x + 3x^2 \tag{2.1}$$

and

$$\mathbf{b} = \begin{pmatrix} 4 + 5x + 6x^2 \\ 7 + 8x \end{pmatrix} \tag{2.2}$$

then $v_1 = 2$ and $(b_0)_2 = 6$. Using this notation, it should always be clear what kind of mathematical object a letter with subscripts denotes.

Vectors are generally denoted as row vectors; for a vector $\mathbf{b} \in R^k$, we denote the corresponding column vector by $\mathbf{b}^\top$. The scalar product of two vectors is denoted by matrix multiplication, i.e., for $\mathbf{a}, \mathbf{b} \in R^k$, we write $\mathbf{a}\mathbf{b}^\top$ to denote

$$\mathbf{a}\mathbf{b}^\top = \sum_{i=0}^{k} a_i b_i. \tag{2.3}$$

This is regardless of $R$, for example, $R$ may be a polynomial ring $R = \mathbb{F}_q[x]$, in which case

$$\mathbf{a}\mathbf{b}^\top = \sum_{i=0}^{k} \sum_{j=0}^{\deg(a)+\deg(b)} \sum_{l=0}^{j} (a_i)_l (b_i)_{j-l} x^j \tag{2.4}$$

where $\deg(\cdot)$ denotes the degree of a polynomial. To illustrate this further, let

$$\mathbf{a} = \begin{pmatrix} 0 + 1x + 2x^2 \\ 2 + 4x \end{pmatrix} \tag{2.5}$$

and

$$\mathbf{b} = \begin{pmatrix} 6 \\ 7 + 8x \end{pmatrix} \tag{2.6}$$

then

$$\mathbf{a}\mathbf{b}^\top = 6(0 + x + 2x^2) + (2 + 4x)(7 + 8x) = 14 + 50x + 44x^2 \tag{2.7}$$

Note that the base ring will often be of the form $\mathbb{F}_q[x]/(f)$, and then, elements can be seen as $n$-dimensional vectors, where $n = \deg(f)$, with an additional multiplicative structure. In this case, where a Number Theoretic Transform (NTT) exists (c.f. Section 2.2.3), we denote the vector corresponding to $\mathbf{a}$ in the NTT domain by $\hat{\mathbf{a}}$. Vectors in NTT domain are multiplied as polynomials of a lower degree or pointwise; we will not indicate this by a special notation.

We often denote equivalence classes by a representative of that equivalence class without further indication if it is clear from the context which ring the object is from. Addition, multiplication, and other operations will take place in the specified ring without denoting it by, e.g., a "mod". For example, let $a, b \in \mathbb{F}_5$ with $a = 2$ and $b = 3$, then $ab = 1$ and what we mean is that $a = 2 + 5\mathbb{Z}$, $b = 3 + 5\mathbb{Z}$, and thus $ab = 6 + 5\mathbb{Z} = 1 + 5\mathbb{Z}$. If we need $a, b$ to be considered as elements of $\mathbb{Z}$, then we will explicitly state it.

If $a, b \in \mathbb{Z}$ and we desire to multiply them modulo a natural number $l$, we denote this by $ab \bmod l \in \mathbb{Z}$, where the result is an integer again. We thereby differentiate between the operation of reducing modulo $l$, which results in an integer in $\{0, \ldots, l-1\}$ and the map to $\mathbb{Z}/l\mathbb{Z}$, which is denoted as $c = ab + l\mathbb{Z} \in \mathbb{Z}/l\mathbb{Z}$. A similar notation is applied in general and not only for $\mathbb{Z}$ and integer ideals.

Reducing $a \in \mathbb{Z}$ modulo $l \in \mathbb{Z}$ such that the residue $x$ fulfills $-l/2 < x \leqslant l/2$ is denoted by $a \bmod^{+} l \in \mathbb{Z}$. Rounding $a \in \mathbb{Z}$ to the closest smaller integer is denoted by $\lfloor a \rfloor$; rounding to the closest greater integer is denoted by $\lceil a \rceil$; and rounding to the closest integer is denoted by $\lceil a \rfloor$, where $a + 1/2$ is rounded to $a + 1$.

We denote functions in a mathematical context as, e.g., Decode. Ciphertext, secret key, public key, and message/plaintext are denoted by `ct`, `sk`, `pk`, `m`, respectively. We denote the Gaussian distribution with mean $\mu$ and standard deviation $\sigma$ by $\mathcal{N}(\sigma, \mu)$ and sampling from it when the exact algorithm does not matter by $e \leftarrow \mathcal{N}(\sigma, \mu)$. By a binomial distribution, we denote a *centered* binomial distribution (as only centered binomial distributions occur in this work); such a distribution has the parameter $\eta$ that defines the number of trials occurring with probability $\frac{1}{2}$. Sampling in other contexts will be denoted as described in the text.

## 2.1 Cryptographic Schemes

Cryptographic schemes consist of algorithms that together achieve a cryptographic goal such as, e.g., encrypting a message. Asymmetric cryptography, often called public key cryptography, is the set of schemes working with key pairs consisting of a *public* and a *private key*. The former may be published and enables a certain operation, for example, encryption or verifying a signature, while the latter has to be kept private and allows a related operation for example decryption or creation of a signature. Commonly, asymmetric cryptographic schemes define a *key generation*, an *encryption*, *encapsulation*, or *verify* function, and a *decryption*, *decapsulate*, or *sign* function, respectively. On the other hand, in symmetric cryptography, both operations are enabled by the same key or no key is required.

A common example of the usage of several asymmetric and symmetric schemes is an authenticated key exchange using public key cryptography and subsequent communication using symmetric cryptography: First, a *shared secret* is exchanged and, in the process, one or both parties authenticate themselves. Then, this shared secret serves as a basis for *derivation* of a common key. Finally, further communication is secured using symmetric cryptography which is usually more performant allowing for encryption of large amounts of data. An example of such a scheme is depicted in Figure 2.1 – this specific example is as in Hermelink et al. [HPS+20]; we, in [HPS+20], instantiate a scheme proposed by Guilhem, Smart, and Warinschi [SSW17] with post-quantum algorithms.

Quantum computers could break currently used asymmetric cryptography and could halve the bit security of symmetric schemes using Shor's [Sho94, Sho97] and Grover's [Gro96] algorithm, respectively. The later can be mitigated by doubling key lengths of currently used symmetric algorithms (see, e.g., [BL17]). Therefore, the NIST contest [NistCfp], which aims to find and standardize replacement candidates, focuses on public-key cryptography, in particular key establishment schemes and digital signatures. The main candidate for standardization in the area of key establishment is Kyber – internally defining a public key encryption scheme (KyberPKE) and a key encapsulation mechanism (KyberKEM). In the following, we give an overview of PKEs, KEMs, and the class of attacks called Chosen-Ciphertext Attacks (CCAs) as well as corresponding security notions.

**Public key encryption schemes** A PKE defines the following operations:

- **Key generation**: *Input*: Randomness. *Output*: Public key `pk`, secret key `sk`.
- **Encryption**: *Input*: Randomness, message `m`, public key `pk`. *Output*: Ciphertext `ct`.
- **Decryption**: *Input*: Ciphertext `ct`, secret key `sk`. *Output*: Message $\mathtt{m}'$ or $\perp$.

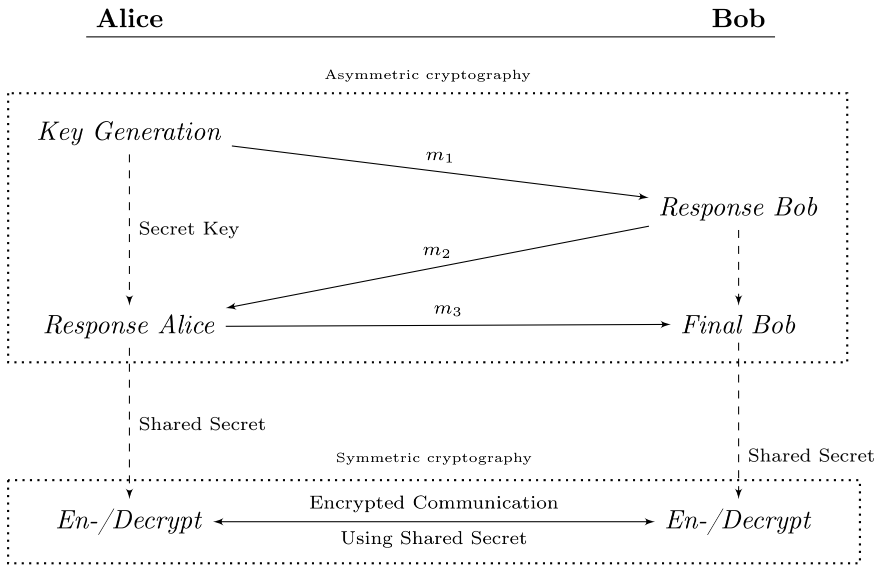


Figure 2.1: Example of an authenticated key exchange with subsequent symmetrically secured communication as, e.g., proposed by Guilhem, Smart, and Warinschi [SSW17] and instantiated with post-quantum schemes in an automotive context by Hermelink et al. [HPS+20]. Compared to a pure key exchange, this example features a third message, and Alice commits to a value in her first message; this can later be checked by Bob.

Note that the randomness is either passed as a seed or only an implicit parameter and realized through a Random Number Generator (RNG) call. The PKE is correct if `m′` is equal to `m` given the correct secret key `sk` belonging to the `pk` used for ciphertext creation. In the PQC domain, many schemes, most notably lattice-based schemes, are only correct with a high probability but may fail (without any manipulation). The message `m` is also called plaintext.

**Key Exchange Mechanisms** A KEM defines the following operations:

- **Key generation**: *Input*: Randomness. *Output*: Public key `pk`, secret key `sk`.
- **Encapsulation**: *Input*: Randomness, public key `pk`. *Output*: Ciphertext `ct`, shared secret `K`.
- **Decapsulation**: *Input*: Ciphertext `ct`, secret key `sk`. *Output*: Shared secret `K`’ or $\perp$.

The comments about the randomness, definitions, and failures of PKEs in the previous paragraph apply to for KEM as well. Often, the KEM is built from a PKE and internally samples a message `m`, which is encrypted and from which the shared secret is derived (c.f. Section 2.1.2). In this case, the message `m` is commonly called the plaintext (of the underlying PKE). In a hybrid encryption scheme, i.e., key establishment using public key cryptography and subsequent symmetrically secured communication, we will sometimes call the exchanged key the session key, and the secret key of the asymmetric scheme the long-term secret. Note that the key pair of the asymmetric scheme may also be *ephemeral* – re-generated after every or a certain number of key exchanges. While ephemeral schemes do prevent attacks requiring multiple observed key exchanges, they are also less efficient due to the repeated key generation.

### 2.1.1 Chosen-Ciphertext Attacks

Chosen-Ciphertext Attacks (CCAs) were first formalized by Naor and Yung [NY90] with various notions and relations in-between those notions introduced by Bellare et al. [BDPR98]. One notable representative of this class of attacks are the Bleichenberger attack [Ble98] and its derivates, for example [JSS12] and [NSS+17]. We give an introduction based on the work of Bellare, Desai, Pointcheval, and Rogaway [BDPR98].

Informally, Chosen-Plaintext Attacks (CPAs) are a class of attacks where the attacker chooses a plaintext to be encrypted causing the decryption routine to leak information through some mechanism. CCAs, a stronger class of attacks, utilize a chosen ciphertext instead of a plaintext – the ciphertext does not need to come from a valid plaintext. In the non-adaptive version of a CCA, the attacker may not modify their ciphertext depending on the information learned from the routing under attack, whereas the adaptive version allows this.

**Formal definitions.** A more formal description, following [BDPR98], works as follows: Let $(K, E, D) = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ be an instance of a PKE or a KEM with keypair $(\mathtt{pk}, \mathtt{sk})$ where $K, E, D$ run in polynomial time. The adversary is defined by two deterministic algorithms $A = (A_1, A_2)$ where $A_1$ models the first stage and $A_2$ the second stage of the attack. In the first stage, described by $A_1$, the attacker is given access to the public key $\mathtt{pk}$ and may query certain functions depending on the scenario. In the second stage, described by $A_2$, the attacker is issued a challenge ciphertext $\mathtt{ct}$ and asked to solve the challenge given arbitrary state information previously returned by $A_1$.

In the CPA scenario, $A_1$ is given access to the public key allowing them to compute the ciphertext for any plaintext. The non-adaptive CCA (CCA1) calls $A_1$ with the public key as a parameter and gives access to a decryption oracle – in the first phase of the attack, any ciphertext can be decrypted. $A_2$ on the other hand does not have access to the decryption oracle but only to the public key. In the adaptive CCA (CCA2), the attacker additionally has access to the decryption error oracle but merely may not query it on $\mathtt{ct}$ itself.

A CCA2 attack may model a CCA1 attack which again is allowed to take every action a CPA attack is permitted to perform. Thus, the order in which the models are presented here is in order of growing strength.

**Security against chosen-plaintext/ciphertext attacks.** Three main security notions in regard to CPA and CCA attacks exist; we summarize them, again following [BDPR98]. All three notions are based on *indistinguishably of encryption* [GM84] and the definition of *negligible* function; a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if it converges faster against zero than any polynomial function, i.e., if for every $c \in \mathbb{R}$ there is $n_c \in \mathbb{N}$ such that $f(n) \leqslant k^{-c}$ for $n \geqslant n_c$. Let again $\Pi = (K, E, D)$ be an encryption scheme, and $A = (A_1, A_2)$ an attacker. $A_1$ now outputs a triple $(m_0, m_1, s)$ where $m_i$ are valid plaintexts and $s$ is a state passed to $A_2$. $A_2$ is passed the triple as well as a ciphertext $\mathtt{ct}$ which either encrypts $m_0$ or $m_1$; $A_2$ is asked to decide whether $m_0$ or $m_1$ were used to generate $\mathtt{ct}$. For $\mathtt{attack} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$ and the respective oracle accesses in $A_1, A_2$, the advantage of the attacker is defined as

$$\begin{aligned}
\mathrm{Adv}_{A,\Pi}^{\texttt{attack}} &= 2P( && (2.8)\\
(\texttt{pk}, \texttt{sk}) &\leftarrow K; && (2.9)\\
(m_0, m_1, s) &\leftarrow A_1; && (2.10)\\
b &\leftarrow \{0,1\}\,; && (2.11)\\
\texttt{ct} &\leftarrow E(m_b); && (2.12)\\
A_2(m_0, m_1, s, \texttt{ct}) &= b && (2.13)\\
&) - 1. && (2.14)
\end{aligned}$$

A scheme is called IND-`attack` (secure), for $\texttt{attack} \in \{CPA, CCA1, CCA2\}$ if the advantage $\mathrm{Adv}_{A,\Pi}^{\texttt{attack}}$ is negligible.

### 2.1.2 Fujisaki-Okamoto Transform

Public key encryption schemes, in particular in the post-quantum domain, often fulfill IND-CPA but not IND-CCA security (e.g., [LP11, LPR13]). A Fujisaki-Okamoto (FO)-transform [FO99, FO13] allows turning an IND-CPA secure PKE into a IND-CCA2 secure KEM; note that many differing variants exist (see, e.g., [HHK17]). The variant of the FO-transform used in Kyber is similar to the transform presented in [TU16].

Let in the following $(K_{\text{pke}}, E_{\text{pke}}, D_{\text{pke}})$ again be a PKE and let $(K_{\text{kem}}, E_{\text{kem}}, D_{\text{kem}})$ be the KEM given by the FO-transform. Informally, the encapsulation routine, $E_{\text{kem}}$, commonly consists of sampling a random message `m`, deriving randomness from `m`, and deterministically (depending on the randomness) encrypting the message `m` using $E_{\text{pke}}$ to a ciphertext `ct`. The decapsulation routine now not only decrypts `ct` to `m` using $E_{\text{pke}}$ but also re-encrypts `m` to $\texttt{ct}'$, using the same randomness derivation from `m`, and comparing `ct` to $\texttt{ct}'$. If the ciphertexts do not match, no shared secret is returned. Thereby, the decapsulation routine essentially checks whether `ct` was honestly generated by re-performing the relevant steps. In the case that the ciphertext is manipulated, the decapsulation routine outputs a rejection from which an attacker does not learn anything. An exemplary, and high-level, depiction is given in Figure 2.2, and a depiction of the decapsulation routine is shown in Figure 2.3. We do not give a fully formal treatment of the topic as the highly complex nuances are not relevant to this work because we only work with the FO-transform used in Kyber; instead, we refer to [HHK17].

## 2.2 Lattice-Based Cryptography

Several schemes competing in the NIST competition were and are lattice based [NistCfp, NistR1, NistR2, NistR3]. The main candidate for key establishment is Kyber, which is based on the MLWE problem [LS15] and therefore lattice-based. In this section, we provide a short overview over lattice-based cryptography from a practical viewpoint with focus on LWE. For a more thorough introduction, we refer to [NS13] for the general mathematical background, to [MG02] for background on lattices in particular, and to [MR09] for LWE-based schemes.

### 2.2.1 Mathematical Background

We give a very short overview over basic definitions in regard to lattice we require in the following chapters following [NS13]: A lattice $\mathcal{L}$ is a discrete subgroup of the $\mathbb{R}^n$ which is isomorphic to $\mathbb{Z}^k$.

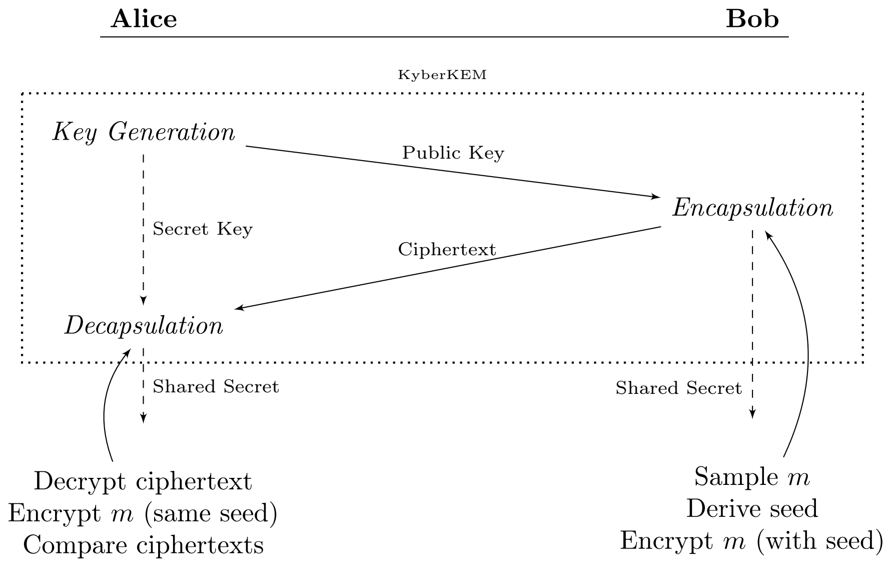


Figure 2.2: Exemplary high-level depiction of an FO-transform as used in Kyber (which is similar to the transform presented in [TU16]). The routines defined by the KEM rely on the method defined by the PKE. Figure adapted from [Her23b].

This is equivalent (see, e.g., [NS13, Proposition 4.2]) to $\mathcal{L}$ being the linear $\mathbb{Z}$-span of $k$ linearly independent vectors $b_i \in \mathbb{R}^n$, i.e.

$$\mathcal{L} = \langle b_0, \dots b_{k-1} \rangle_{\mathbb{Z}} \tag{2.15}$$

with all $b_i \in \mathbb{R}^n$ being linearly independent; the set

$$B = \{b_0, b_1, \dots, b_{k-1}\} \tag{2.16}$$

is then called a basis of $\mathcal{L}$, and $k = |B|$ is the dimension. For any basis $B$, any element of $v \in \mathcal{L}$ may thus be written as

$$v = \sum_{i=0}^{k-1} a_i b_i \tag{2.17}$$

for some $a_i \in \mathbb{Z}$.

In a sense, a lattice may be thought of as a discrete analogue of a (sub-) vector space; but note that many properties of vector spaces do not translate to lattices. For example, lattice $L_0, L_1$ generated by bases $B_0, B_1$, respectively, with $|B_0| = |B_1|$ do not necessarily generate the same lattice. Let

$$B_0 = \{(1,0),(0,1)\} \tag{2.18}$$

and

$$B_1 = \{(2,0),(0,1)\} \tag{2.19}$$

then clearly

$$\langle B_0 \rangle_{\mathbb{R}} = \langle B_1 \rangle_{\mathbb{R}} \tag{2.20}$$

by basic linear algebra (as $2^{-1} \in \mathbb{R}$). But

$$\mathbb{Z}^2 = \mathcal{L}_0 = \langle B_0 \rangle_{\mathbb{Z}} \neq \langle B_1 \rangle_{\mathbb{Z}} = \mathcal{L}_1 \tag{2.21}$$

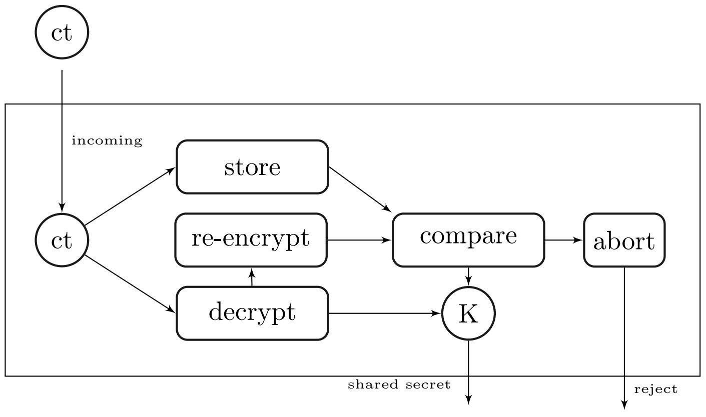


Figure 2.3: Exemplary depiction of the decapsulation routine of an FO-transform as used in Kyber (which is similar to the transform presented in [TU16]). An incoming ciphertext is stored, decrypted, encrypted again, and the result is compared against the stored ciphertext. If the submitted and re-computed ciphertext match, a shared secret (based on the message) is returned. Figure first presented in [Her23a]

as the system of equation

$$\begin{pmatrix} x_0 & x_1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} \tag{2.22}$$

clearly has no solution over $Z$ (as $2^{-1} \notin \mathbb{Z}$); instead, only $L_1 \subset L_0$ holds. The lattices $L_0 = \mathbb{Z}$ and $L_1$ are depicted in Figure 2.4, and a third lattice with basis vectors arising from a rotation and scaling of $\mathbb{Z}^2$ is shown in Figure 2.5a.

The determinant of a lattice $\mathcal{L}$ with basis $B = \{b_0, b_1, \ldots, b_{k-1}\}$ is defined as

$$\det(\mathcal{L}) = \sqrt{BB^\top} = \sqrt{\sum_{i=0}^{k-1} \sum_{j=0}^{n-1} b_{ij} b_{ji}}. \tag{2.23}$$

The determinant is the volume of a fundamental mesh of the lattice which is defined as

$$F_{\mathcal{L}} = \left\{ \sum_{i=0}^{k-1} a_i b_i \mid a_i \in \mathbb{R}, 0 \leqslant a_i < 1 \right\} \tag{2.24}$$

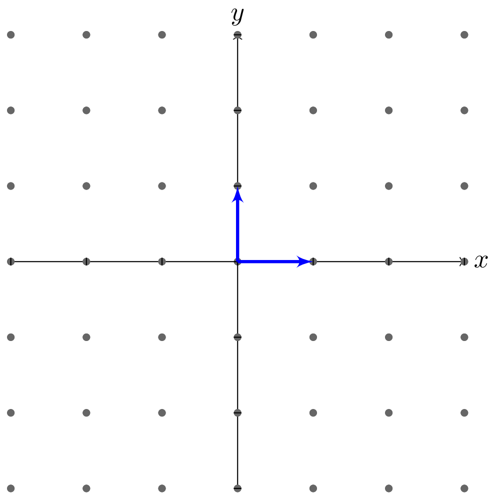
and can be thought of the “base” mesh spanned by the basis vectors – $x \sim y \Leftrightarrow x - y \in \mathcal{L}$, which defines an equivalence relation on $\mathbb{R}^n$ with the fundamental mesh forming a representation system. The fundamental mesh is exemplarily depicted in Figure 2.5b. The successive minima $\lambda_i(\mathcal{L})$ for $i \in \{1, \ldots, n\}$, first defined by Minkowski to prove that every compact subset of the $\mathbb{R}^n$ of volume greater or equal $2^n$ contains a lattice point, are the smallest radii such that $i$ linearly independent lattice vectors of norm $\lambda_i$ exist. Clearly, the relation

$$\lambda_1(\mathcal{L}) = \min_{0 \neq v'}(v') \tag{2.25}$$

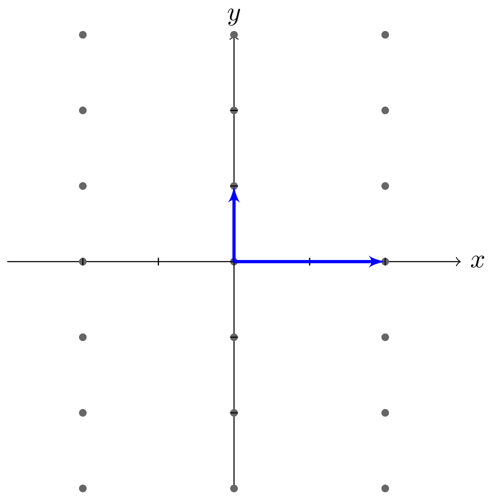
holds. $\lambda_1(\mathcal{L})$ may be approximated using the Gaussian heuristic

$$\lambda_1(\mathcal{L}) \approx \frac{\Gamma(1 + \frac{n}{2})^{\frac{1}{n}}}{\sqrt{\pi}} \det(\lambda)^{\frac{1}{n}} \tag{2.26}$$

where $\Gamma$ is the $\Gamma$-function, relating determinant, dimension, and shortest vector length.



(a) $L_0$, equal to $\mathbb{Z}^2$.



(b) $L_1$, not equal to $\mathbb{Z}^2$.

Figure 2.4: Two lattices generated by a basis of the same length which are not equal. Basis of $B_0$ and $B_1$ are depicted as vectors.

## NP-Hard Lattice Problems

Several NP-hard (for randomized reduction) problems in relation to lattices are known; we here focus on the shortest and closest vector problem following [MG02]. The classical NP-hardness of the Closest Vector Problem (CVP) problem and of the Shortest Vector Problem (SVP) in infinity norm has been shown in [Emd81]. Ajtai [Ajt96, Ajt98] showed NP-hardness in Euclidean norm for randomized reductions, and both problems are conjectured to be quantum-hard.

The SVP is defined as follows: Given a lattice $\mathcal{L} \subset \mathbb{R}^n$, find $v \in \mathcal{L}$ such that

$$||v|| = \lambda_1(\mathcal{L}) = \min_{v'}(||v'||) \tag{2.27}$$

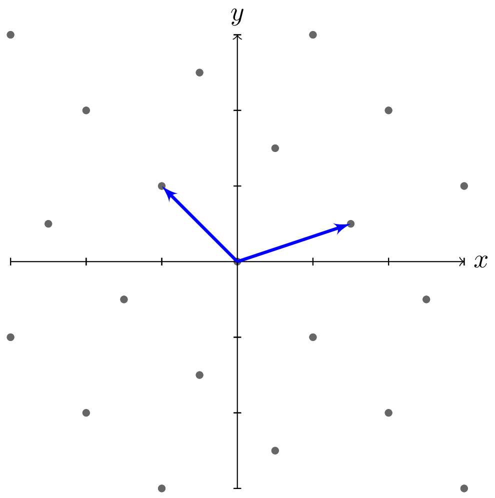
where $||\cdot||$ denotes the Euclidean norm. A more general problem is the CVP: Given a lattice $\mathcal{L} \subset \mathbb{R}^n$, a vector $x \in \mathbb{R}^n$, find $0 \neq v \in \mathcal{L}$ such that

$$||v - x|| = \min_{v' \in \mathcal{L}}(||v' - x||). \tag{2.28}$$

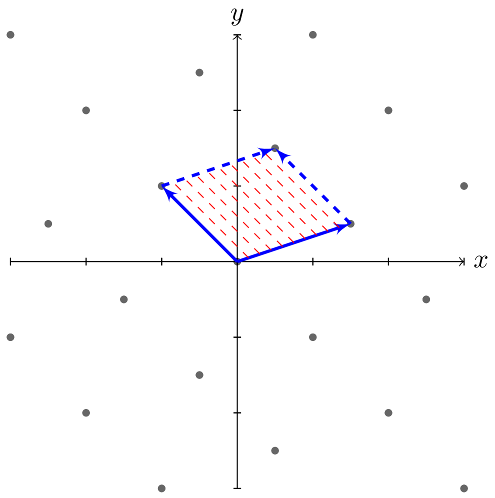
The problems are related, and the SVP may be reduced to the CVP [GMSS99]. These problems also occur in their Gap$_\gamma$ variants where an algorithm is asked to output if the shortest/closest lattice vector is within distance smaller than $\gamma \geqslant 1$. The unique Shortest Vector Problem (uSVP) with parameter $\gamma \geqslant$ asks to find a shortest non-zero vector if $\lambda_2(\mathcal{L}) \geqslant \gamma\lambda_1(\mathcal{L})$; clearly, for $\gamma = 1$ this directly gives the SVP.

Another closely related problem is the Bounded Distance Decoding (BDD) problem with parameter $\gamma$: Given a lattice $\mathcal{L} \subset \mathbb{R}^n$, a vector $x \in \mathbb{R}^n$ with the closest vector $v \in \mathcal{L}$ having distance smaller than $\frac{\lambda(\mathcal{L})}{2\gamma}$, find $v$. The BDD may be reduced to the SVP using Kannan's embedding [Kan87]: Let $B$ be a basis of $\mathcal{L}$, then we define the lattice $\mathcal{L}_{\text{svp}}$ with dimension $\dim(\mathcal{L}) + 1$ as generated by the basis

$$\begin{pmatrix} \mathbf{B}_{\text{svp}} & 0 \\ \mathbf{x} & c \end{pmatrix} \tag{2.29}$$



(a) Another lattice.



(b) Fundamental mesh of the lattice.

Figure 2.5: The lattice generated by $\{(1.5, -1), (1, 0.5)\}$, which is also two-dimensional but neither equal to $L_0$ nor $L_1$ and its fundamental mesh.

for some $c \in \mathbb{R}$ (which may be set to an optimal value or, as often done in practice in the LWE setting, to 1). Finding a shortest vector $0 \neq v_{\text{svp}} \in \mathcal{L}_{\text{svp}}$ gives a vector

$$\mathbf{v} = \mathbf{v}_{\text{svp}} - \mathbf{x} \tag{2.30}$$

which directly gives an element of $\mathcal{L}$ that is a closest vector to $\mathbf{x}$.

### Lattice Reduction

Lattice reduction algorithms aim to find a *reduced* basis given any basis $B$ of a lattice $\mathcal{L}$. Several notions of *reduced* exist; they depend on the algorithm and define properties to control the size of the resulting basis vectors. Thereby, in particular, (Gap-)SVP may be solved if the reduction is of sufficiently high quality. A commonly used algorithm was proposed by Lenstra Jr., Lenstra, and Lovász [LLL82] – the Lenstra, Lenstra, and Lovasz (LLL) algorithm. This led to numerous other lattice reduction algorithms, for example [Sch87, SE94, SH95, GN08a, HPS11, MW16], improving upon LLL. The current state of the art in practical attacks are Blockwise Korkine-Zolotarev (BKZ) variants, such as [CN11], derived from the original BKZ algorithm [SE94]. BKZ and its derivates feature a *block-size*, commonly denoted $\beta$. The block size determines the dimension of a sub-lattice in which an exhaustive search (often leaving out some nodes) is performed. Therefore, the block-size is a trade-off between runtime and quality of the reduced basis: A block size of 2 is similar to LLL whereas a block-size of $\dim(L)$ is similar to an exhaustive search. The runtime of LLL on a lattice $L$ with basis $B$ and largest basis vector $b_{\max}$ is $O(k^5 n log_3 ||b_{\max}||)$ (see, e.g., [NS09, Fig. 1]), i.e., in polynomial time, whereas BKZ was experimentally observed to run in exponential time in practice [GN08b]. To solve a uSVP instance on a lattice $\mathcal{L}$ with $\lambda_1(\mathcal{L}) = v$, the required $\beta$ can be estimated using the formula given by [ADPS16b, AGVW17]:

$$||v||\sqrt{\beta/\dim(\mathcal{L})} \leqslant \delta_\beta^{2\beta-\dim(\mathcal{L})-1} \det(\mathcal{L})^{1/\dim(\mathcal{L})} \tag{2.31}$$

where $\delta_\beta$ is the smallest integer satisfying

$$\sqrt{\beta} \leqslant \delta_\beta^{2\beta-\dim(\mathcal{L})-1} \det(\mathcal{L})^{1/\dim(\mathcal{L})}. \tag{2.32}$$

In this thesis, we think of BKZ as a black box with the aforementioned properties and use the implementation of [CN11] given in FPLLL [Fplll].

### 2.2.2 Learning with Errors Problems

The Learning with Errors (LWE) problem, introduced by Regev in [Reg05], serves as a basis for several cryptographic schemes. It may be reduced to (a variant of) Gap-CVP and the Shortest Independent Vector Problem (SIVP) [Reg05, Pei09a, BLP+13] (under certain conditions we omit here) and allows for the construction of cryptographic schemes. We give a short overview and refer to [MR09] for a more thorough treatment of the subject. In addition, we refer to [Reg10] for a concise and less technical overview.

#### Learning with Errors

The LWE problem is commonly (for example in [Reg10, LP11]) defined as follows: For the modulus $q$, the *error distribution* $\chi$ over $\mathbb{Z}$, and a secret vector $\mathbf{s} \in \mathbb{Z}$ let $A_{\mathbf{s},\chi}$ be the distribution on $\mathbb{Z}^n \times \mathbb{Z}$ that outputs a tuple

$$(\mathbf{a}, \mathbf{a}\mathbf{s}^\top + e \bmod q) \in \mathbb{Z}^n \times \mathbb{Z} \tag{2.33}$$

where $e \leftarrow \chi$ and $\mathbf{a}$ is uniformly random. The LWE problem asks to find $\mathbf{s}$ given access to samples of $A_{\mathbf{s},\chi}$. For a fixed amount of samples, the problem can be thought of as solving a distorted system of equations

$$\mathbf{A}\mathbf{S}^\top + \mathbf{E} = \mathbf{B} \in \mathbb{Z}^{m\times n} \tag{2.34}$$

where $\mathbf{A} \in \mathbb{Z}^{n\times m}$ is made up of the samples $\mathbf{a}$, $\mathbf{S} \in \mathbb{Z}^{n\times m}$ are multiple secrets (in the language of the definition), and $\mathbf{E} \in \mathbb{Z}^m$ are the errors. In cryptographic practice, $\mathbf{S}$ is often sampled from $\chi$ as well, $\chi$ is commonly chosen as discrete Gaussian or binomial distribution with small support, and $n = m$ (see, e.g., [LP11, LPR13, BCD+16, ADPS16b, BDK+18]). Note that several small variations and generalizations exist.

A closely related problem is the Decision Learning with Errors (D-LWE) problem which asks to differentiate between an LWE sample and a uniformly randomly sampled vector. D-LWE may be reduced to LWE as shown by Peikert in [Reg09]; we shortly repeat the basic argument and refer to [Reg09] for details. Given access to an algorithm that allows differentiating between a uniformly sampled vector and a LWE sample, the secret $\mathbf{s}$ may be found as follows: For $(\mathbf{a}, b)$ and a guess $g \in \mathbb{Z}/q\mathbb{Z}$, the first component of $\mathbf{s}$ may be found by deciding whether

$$(\mathbf{a} + (l, 0, \ldots, 0), b + lg) \tag{2.35}$$

is a LWE sample (for uniformly random $l \in \mathbb{Z}/q\mathbb{Z}$). For $l \in \mathbb{Z}/q\mathbb{Z}$, we have that for $g = s_0$, the distribution of Equation (2.35) follows $A_{\mathbf{s},\chi}$ and is otherwise uniformly random. Therefore, coefficients of $s$ may be found by iterating through all possible values (for a single coefficient at a time) and creating sufficiently many samples to differentiate between a LWE sample and uniform randomness.

### Ring Learning with Errors

The Ring Learning with Errors (RLWE) problem, first introduced in [LPR13], is a variant of the LWE problem. Instead of working with vectors, RLWE utilizes polynomials over a ring $R = \mathbb{F}_q[x]/(f)$ for some polynomial $f$. We may think of polynomials in $R$ as vectors of length $n = \deg(f)$ with a multiplicative structure. The RLWE problem is then posed as finding $s$ given outputs of $A_{\mathrm{rlwe},s,\chi}$, where $s \in R$, $\chi$ is a distribution over $\mathbb{Z}$, and $A_{\mathrm{rlwe},s,\chi}$ outputs samples

$$(a, as + e) \in R \times R \tag{2.36}$$

for coefficient-wise uniformly samples $a$ and coefficient-wise $\chi$-sample $e$.

A single RLWE equation, i.e., an equation of the form

$$as + e = b \in R \tag{2.37}$$

with known $a, b$ and secret $s, e$ can be seen as $n$ LWE samples. This is because multiplication with $a$ may be written as matrix multiplication with an $n \times n$ matrix over $\mathbb{F}_q$ and two polynomials are equal if and only if all their coefficients are equal. In contrast to an LWE sample according to the LWE problem, these samples follow a structure determined by $f$. This structure allows for particularly efficient schemes. The potential security implications have previously been discussed (see, e.g., [Ber21, NistSt21]), but currently no attacks which may be carried out in practice against candidates in the standardization contest are known. Potential security risks concerning NTTs (the existence of an NTT is a consequence of the structure RLWE introduces) have been discussed in [BBPS19, DGKS20].

### Module Learning with Errors

The Module Learning with Errors (MLWE) problem, introduced in [BGV14, LS15], is a variant of the LWE problem and a generalization of the RLWE problem. Using the definitions from the previous section on RLWE and introducing a parameter $k$, the problem is defined as finding $\mathbf{s}$ given outputs of $A_{\mathrm{mlwe},\mathbf{s},\chi}$, where $\mathbf{s} \in R^k$, $\chi$ is a distribution over $\mathbb{Z}$, and $A_{\mathrm{mlwe},\mathbf{s},\chi}$ outputs samples

$$(\mathbf{a}, \mathbf{as} + \mathbf{e}) \in R^k \times R^k \tag{2.38}$$

for coefficient-wise uniformly samples $\mathbf{a}$ and coefficient-wise $\chi$-sample $\mathbf{e}$.

For $k = 1$ we directly obtain the RLWE problem; for $k > 1$, $n = 1$ and monic $f$, we obtain the standard LWE problem. For $k > 1$ and $n > 1$, we may see MLWE as a generalization offering a trade-off in terms of introduced structure between LWE and RLWE. Therefore, the discussions about security implications referenced in the previous sections apply as well.

## 2.2.3 Learning with Errors Schemes

The different variants of the LWE problem may be used to construct cryptographic schemes. Several of the schemes were considered in the NIST contest on post-quantum cryptography [NistR1]. In this section, we give a short overview over the final candidate for key exchanges, the KEM Kyber [BDK+18, ABD+21b], as well as an RLWE scheme introduced in [LPR13] and the LWE scheme of [LP11]. For a more comprehensive overview on Kyber, we refer to the official specification [ABD+21b]. In the context of the NIST contest, the status reports [NistR1, NistR2, NistR3] provide an overview over all considered schemes. In addition, the German Bundesamt für Sicherheit in der Informationstechnik (BSI) provides an overview and evaluations in [BsiPqc].

**Number Theoretic Transform**

Common techniques used in the context of lattice-based cryptography for fast multiplication of polynomials include Karatsuba's algorithm [KO62, WP06], the Toom-Cook algorithm [Too63, CA69], and the number theoretic transform [Pol71, AB75, Nus81, Win96, BJL08, GG13]. Several schemes based on the RLWE or MLWE problem make use of an Number Theoretic Transform (NTT), e.g., [ADPS16b, DKL+18, BDK+18], and the usage of an NTT has been proposed [CHK+21, ACC+21] for schemes using constructions which do not directly allow for the usage of an NTT, e.g., for NTRU [HPS98], NTRUPrime [BCLV17], and Saber [DKRV18]. The NTT may be seen as the algebraic equivalent to a Fast-Fourier Transformation (FFT) and allows for fast multiplication of polynomials. For a ring $R = \mathbb{F}_q[x]/(f)$ with $f$ factoring into $f = \prod_i f_i$, it provides an efficient way to compute the Chinese Remainder Theorem (CRT), i.e., the isomorphism

$$R \to \prod_i \mathbb{F}_q[x]/(f_i), \tag{2.39}$$

and its inverse; the latter is called inverse NTT or INTT.

**Efficient computation.** For $R = \mathbb{F}_q[x]/(x^n + 1)$ where $q$ is (a power of) a prime, an NTT mapping to $\mathbb{F}_q^n$ exists if and only if $\mathbb{F}_q$ contains a primitive $2n$-th root of unity $\zeta$ as, in this case, $x^n + 1$ factors into $n$ polynomials of the form $f_i = x - \zeta^i$. The NTT of a polynomial $\mathbf{a} = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_q[x]$ may then be computed as $\hat{\mathbf{a}} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1}) \in \mathbb{F}_q^n$ with

$$\hat{a}_i = \mathbf{a}(\zeta^i) = \sum_{j=0}^{n-1} a_j \zeta^{ij} \tag{2.40}$$

and the inverse NTT as

$$a_i = \frac{1}{n} \sum_{j=0}^{n-1} a_j \zeta^{-ij}. \tag{2.41}$$

These equations may be computed in $O(n \log(n))$ using Cooley-Tukey [CT65] or Gentleman-Sande [GS66] butterfly operations. Cooley-Tukey [CT65] employ the following divide and conquer strategy: The computation of $\hat{a}_i$, $i < n/2$ may be split up in sums of even, respective odd, terms, i.e.

$$\hat{a}_i = \sum_{j=0}^{n-1} a_j \zeta^{ij} = \sum_{k=0}^{(n-1)/2} a_{2k} \zeta^{2ik} + \zeta^i \sum_{k=0}^{(n-1)/2} a_{2k+1} \zeta^{2ik} \tag{2.42}$$

and an easy computation shows

$$\hat{a}_{i+n/2} = \sum_{j=0}^{n-1} a_j \zeta^{(i+n/2)j} = \sum_{k=0}^{(n-1)/2} a_{2k} \zeta^{2ik} - \zeta^i \sum_{k=0}^{(n-1)/2} a_{2k+1} \zeta^{2ik}. \tag{2.43}$$

The terms

$$S_{\text{even}} = \sum_{k=0}^{(n-1)/2} a_{2k} \zeta^{2ik} \tag{2.44}$$

and

$$S_{\text{odd}} = \sum_{k=0}^{(n-1)/2} a_{2k+1} \zeta^{2ik} \tag{2.45}$$

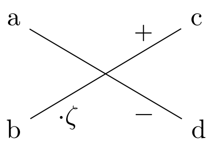


Figure 2.6: A butterfly operation as used in an NTT following Cooley-Tukey [CT65] computing $c = a + \omega b$ and $d = a - \omega b$.

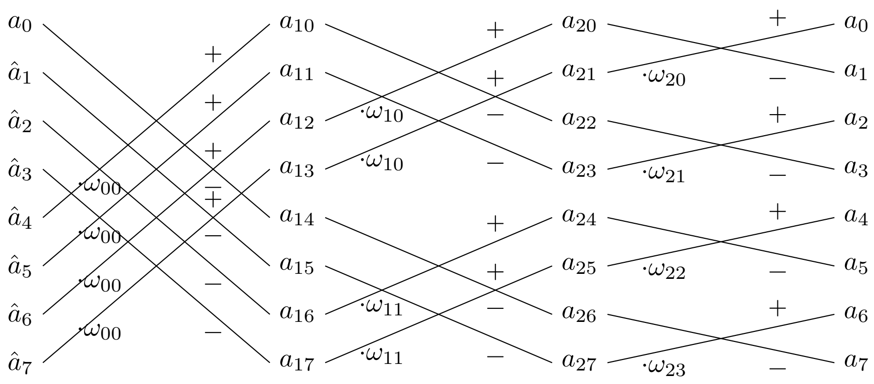


Figure 2.7: Computational graph of an inverse NTT with intermediate values in Cooley-Tukey decimation. Note that $\omega_i$ does not denote $\zeta^i$.

occur in both computations and are itself NTTs with input length $n/2$. We have

$$\hat{a}_i = S_{\text{even}} + \zeta^i S_{\text{odd}} \tag{2.46}$$

and

$$\hat{a}_{i+n/2} = S_{\text{even}} - \zeta^i S_{\text{odd}}. \tag{2.47}$$

Splitting up the smaller NTTs $\log_2(n)$ times leads to the computation of $n \cdot \log_2(n)$ additions and subtractions. This may be thought of as performing $n/2$ *butterflies* (Equation (2.46) and Equation (2.47)) with *twiddle factors* $\omega = \zeta^i$ in $\log_2(n)$ *layers*. We will denote primitive roots of unity by $\zeta$ and roots of unity which are not necessarily primitive by $\omega$. Note that $\omega_i$ is not necessarily denoting $\zeta^i$, often the indices merely express belonging to a block in the NTT. As a computational graph, this may be depicted as in Figure 2.7; a butterfly is shown in Figure 2.6. The computation of the inverse NTT is performed in a similar manner.

**As a linear function.** The NTT and the inverse NTT are an isomorphism,

$$R \to \prod_i \mathbb{F}_q[x]/(f_i), \tag{2.48}$$

of vector spaces over $\mathbb{F}_q$. Thus, as a linear function, they may be expressed as matrices $\mathbf{N}$ and $\mathbf{N}^{-1}$ over $\mathbb{F}_q$. The matrix $\mathbf{N}$ can be obtained by computing the number theoretic transform on the basis vectors of $\mathbb{R}$. As the number theoretic transform consists of substitution of $2n$-th roots of unity $\zeta^i$ for the indeterminate $x$, we obtain a matrix of the form

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 & \\ 1 & \zeta & \zeta^2 & \dots & \zeta^{n-2} & \zeta^{n-1} \\ 1 & \zeta^2 & \zeta^4 & \dots & -\zeta^{n-2} & -\zeta^{n-2} \\ 1 & \zeta^3 & \zeta^6 & \dots & \zeta^{n-6} & \zeta^{n-3} \\ \vdots & & & & & \\ 1 & \zeta^{n-1} & \zeta^{n-2} & \dots & \zeta^{(n-1)(n-2)} & \zeta^{(n-1)^2} \end{pmatrix}. \tag{2.49}$$

We will often interpret $\mathbf{N}$ and $\mathbf{N}^{-1}$ as matrices over $\mathbb{F}_q$ which when applied to a vector give the not-reduced NTT- (or normal-) domain representation.

**Smaller groups of roots of unity.** For a smaller group of unities over $R$, the number of layers is reduced and the degree of the $f_i$ is greater or equal than 0. In terms of a splitting up into butterflies and layers, this means the number of layers is reduced. This is the case in Kyber and described at the end of the next section. For a smaller group of unities, the NTT matrix $\mathbf{N}$ has different entries and is of smaller dimension. It may be obtained in the same way by inserting basis vectors into the NTT.

### Error Correction

To function correctly, LWE-based schemes require an error correction. This is because both parties only agree on a noisy version of a shared secret or message. For a not only approximate but exact version, an error correction or a reconciliation mechanism is required.

For example, in Kyber or NewHope, during encryption, a message (in bits) is mapped to a vector (or vector of polynomials) over a finite field $\mathbb{F}_q$ or ring $\mathbb{Z}/l\mathbb{Z}$ for prime-power $q$ or $l \in \mathbb{N}$. During decryption, the other party obtains a noisy version of the vector and has to coefficient-wise recover the original message from these noisy coefficients. This is called *error correction* or *decoding* depending on the scheme. In Kyber, the *compression* routine is used to describe the error correction step as well. Depending on the context, we will use both decoding and error correction to describe this method; note that in Kyber *Decode* denotes a different function, but the error correction is still sometimes called *decoding*. In this work we mainly reiterate the error correction of Kyber in Section 2.2.3. In addition, we give a short overview over existing reconciliation methods based on the summary of [ADPS16a].

**Encryption-based approaches.** Using the terminology of [ADPS16a], encryption-based approaches encode a secret message `m` into the high bits of a vector or a polynomial. The message is then recovered by ignoring low-bits of the noisy message. In case of a low enough noise, which does not affect the high bits, the message can be recovered without error. This approach has been mentioned in the presentations of [Pei09b] and [Gab10], described in [LP11] and [LPR12, LPR13], used in a NewHope variant presented in [ADPS16a], and is also used in Kyber. For the case of Kyber, a description in more detail is given in Section 2.2.3.

**Reconciliation-based approaches.** The approach called reconciliation-based by [ADPS16a] works similar to fuzzy extractors [DORS08] (see also [BGS+08, HKM+12]). Instead of encoding the information into the high-bits of a message, Ding [Din12][1], proposes sending an addition reconciliation vector. This vector allows for recovery of a shared secret and can reduce bandwidth

[1]In later versions: Ding and Lin [Jin12a] and Ding, Lin, and Lin [Jin12b]

requirements (for an analysis see, e.g., [Pei14, Section 3]). In its original version, this approach led to a biased secret; this was noted and fixed in [Pei14, Section 3]. Ding, Xie, and Lin [Jin12b] proposed a similar technique to avoid this bias as well. The approach is also used in NewHope as presented in [ADPS16b]. A version of NewHope without reconciliation was presented in [ADPS16a].

### The LP Encryption Scheme

The public key encryption scheme of Lindner and Peikert [LP11] improves upon previous LWE encryption schemes [PVW08, GPV08] derived from the scheme of [Reg05]. We state a simplified variant with the notation chosen similar to those used in later schemes such as Kyber. Note that the defined PKE per se does not fulfill the IND-CCA property; later schemes based upon [LP11] commonly use a variant of the FO-transform to define an IND-CCA2 secure KEM.

The scheme is parameterized by the triple $(n,\ q,\ \sigma)$, where $n$ is the dimension[2], $q$ is the modulus, and $\sigma$ is the standard deviation of the error distribution $D_\sigma$. The key pair is given by $(\mathbf{B}, \mathbf{S}) \in \mathbb{F}_q^{n\times n} \times \mathbb{F}_q^{n\times n}$ where $\mathbf{B} = \mathbf{S}\mathbf{A}^\top + \mathbf{E}$ with $\mathbf{A}$ being uniformly random and either a public parameter or part of the public key. The scheme additionally requires an encoding and a decoding function which maps bits to coefficients of a vector over $\mathbb{F}_q$ and vice-versa. Gaussian noise is obtained from the error distribution $D_\sigma$; for vectors this is done coefficient-wise. Applying first encoding and then decoding is the identity; and, additionally, encoding a bit, adding a small additive noise term, and decoding it has to result in the original bit. The key generation is given in Algorithm 1, the encryption of a message `m` in Algorithm 2, and the decryption in Algorithm 3.

**Algorithm 1** Key generation as defined in the scheme of [LP11] in a notation similar to [ABD+21b].

```
Require: Uniformly sampled $\mathbf{a} \in R$
Ensure: Secret key sk, public key pk
1: $\mathbf{E}, \mathbf{S} \in \mathbb{F}_q^{n\times n} \leftarrow D_\sigma$
2: $\mathbf{T} \leftarrow \mathbf{E} - \mathbf{A}\mathbf{S}$
3: return $(\mathtt{pk} = \mathbf{T}, \mathtt{sk} = \mathbf{S})$
```

**Algorithm 2** Encryption as defined in the scheme of [LP11] in a notation similar to [ABD+21b].

```
Require: Public key $\mathbf{T} = \mathtt{pk}$, Message m as bit string
Ensure: $\mathtt{ct} = (c_1, c_2)$
1: $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow D_\sigma$
2: $\mathbf{m} \leftarrow \mathrm{Encode}(\mathtt{m})$
3: $\mathbf{u} \leftarrow \mathbf{r}\mathbf{A}^\top + \mathbf{e}_1$
4: $\mathbf{v} \leftarrow \mathbf{r}\mathbf{T}^\top + \mathbf{e}_2 + \mathbf{m}$
5: return $\mathtt{ct} = (\mathbf{u}, \mathbf{v})$
```

### The LPR Encryption Scheme

In [LPR13], Lyubashevsky, Peikert, and Regev introduce the RLWE problem and note how a PKE may be defined based on it. The scheme is, again, not IND-CCA secure (but only IND-CPA) and later schemes (e.g., [ADPS16b]) building upon the work of [LPR13] often use

[2]Note that the scheme as described in [LP11] defines 3 instead of a single dimension.

**Algorithm 3** Decryption as defined in the scheme of [LP11] in a notation similar to [ABD+21b].

```
Require: Secret key $\mathbf{r}_2 = \texttt{sk}$, ciphertext $\texttt{ct} = (\mathbf{u}, \mathbf{v})$
Ensure: Message $\texttt{m}'$
1: $\mathbf{m}' \leftarrow \mathbf{u}\mathbf{s}^\top + \mathbf{v}$
2: $\texttt{m}' \leftarrow \text{Decode}(\mathbf{m}')$
3: return $\texttt{m}'$
```

an FO-transform. First works on the security of RLWE-based schemes proposed attacks on a scheme designed in this way, which is why we reiterate it here. Among those attacks is the work of Primas, Pessl, and Mangard [PPM17], which is summarized in Section 3.1.2.

The scheme is parameterized by the triple $(n, q, \sigma)$ with $n$ being the dimension, $q$ the prime modulus, and $\sigma$ the standard deviation of the error distribution $D_\sigma$. It works over $R = \mathbb{F}_q[x]/(x^n + 1)$, and the key pair is given by a $s \in R$ and $(a, b) \in R \times R$ with $b = as + e \in R$. An encoding function, Encode, maps a bit string to polynomial coefficients, and a decoding function, Decode, maps coefficients back to bit; the concatenation of encoding and decoding is the identity, and a sufficiently small additive noise term on previously encoded coefficients must not affect the results of decoding[3]. Gaussian noise is obtained from the error distribution $D_\sigma$, coefficient-wise for polynomials. The key generation is given in Algorithm 4, the encryption of a message $\texttt{m}$ in Algorithm 5, and the decryption in Algorithm 6.

For appropriately chosen parameters, the scheme allows for the usage of an NTT which enables fast multiplication. To multiply two polynomials, they are mapped to the NTT-domain, pointwise multiplication is performed, and the result is mapped back to the normal domain. Therefore, if using an NTT, Algorithm 6 features, an inverse NTT call on the term $c_1 r_2$ (and $r_2$ is the secret key).

**Algorithm 4** Key generation as defined in the scheme of [LPR13] in a notation similar to [ABD+21b].

```
Require: Uniformly sampled $\mathbf{a} \in R$
Ensure: Secret key $\texttt{sk}$, public key $\texttt{pk}$
1: $e, s \in R \leftarrow D_\sigma$
2: $t \leftarrow e - as$
3: return $(\texttt{pk} = t, \texttt{sk} = s)$
```

**Algorithm 5** Encryption as defined in the scheme of [LPR13] in a notation similar to [ABD+21b].

```
Require: Public key $p = \texttt{pk}$, Message $\texttt{m}$ as bit string
Ensure: $\texttt{ct} = (c_1, c_2)$
1: $r, e_1, e_2 \leftarrow D_\sigma$
2: $\texttt{m} \leftarrow \text{Encode}(m)$
3: $u \leftarrow ar + e_1$
4: $v \leftarrow tr + e_2 + m$
5: return $\texttt{ct} = (u, v)$
```

[3]Note that the terminology in other schemes is the other way around: Encode maps from polynomial to bit string and Decode from bits to polynomial.

**Algorithm 6** Decryption as defined in the scheme of [LPR13] in a notation similar to [ABD+21b].

```
Require: Secret key $r_2 = \mathtt{sk}$, ciphertext $\mathtt{ct} = (u, v)$
Ensure: Message $\mathtt{m}'$
1: $m' \leftarrow us + v$
2: $m' \leftarrow \mathrm{Decode}(m')$
3: return $\mathtt{m}'$
```

### Crystals-Kyber

Kyber [BDK+18, ABD+21b] is a finalist in the NIST contest [NistCfp] and was selected for standardization after the third round [NistR3]. It was first introduced in [BDK+18], and its current version is specified in [ABD+21b][4].

Kyber works over the ring $R = \mathbb{F}_q[x]/(x^n + 1)$ where $q = 3329$ and $n = 256$; the parameters $k \in \{2, 3, 4\}$, $\eta_1 \in \{2, 3\}$, $d_u \in \{10, 11\}$, and $d_v \in \{4, 5\}$ depend on the security level, $\eta_2$ is set to 2. The parameters per security level are stated in Table 2.1. As an MLWE scheme, the key pair is given by $\mathbf{s} \in R^k$ and $(\mathbf{A}, \mathbf{b}) \in R^{k \times k} \times R^k$ where $\mathbf{b} = \mathbf{s}\mathbf{A}^\top + \mathbf{e}$.

| Security Level | $n$ | $k$ | $q$ | $\eta_1$ | $\eta_2$ | $d_u$ | $d_v$ |
|---|---|---|---|---|---|---|---|
| Kyber512 | 256 | 2 | 3329 | 3 | 2 | 10 | 4 |
| Kyber768 | 256 | 3 | 3329 | 2 | 2 | 10 | 4 |
| Kyber1024 | 256 | 4 | 3329 | 2 | 2 | 11 | 5 |

Table 2.1: Kyber parameters as stated in [ABD+21b].

Kyber uses an FO-transform to define an IND-CCA2 secure KEM from an IND-CPA secure PKE. The key generation of KyberPKE is given in Algorithm 7, the encryption of a message $\mathtt{m}$ in Algorithm 8, and the decryption in Algorithm 9. The method $\mathrm{Sample}_{\mathrm{uniform}}$ samples deterministically (depending on $\rho$) samples from a uniform distribution over $\mathbb{F}_q$. The method $\mathrm{Sample}_{\mathrm{binom}}$ samples deterministically (depending on $\sigma$) from a binomial distribution centered around 0. Note that in our high-level description of the algorithm, the sampling methods have an internal state reset after exiting the parent method; a counter and an extendable output function or a pseudo-random function are used. Sampling of polynomials or vectors of polynomials is understood to be component- and coefficient-wise. The compression and encoding functions are explained in the following paragraphs.

The KyberKEM, also called Kyber, algorithms are given in Algorithm 10, Algorithm 11, Algorithm 12. The functions H and G are (distinct) hash functions, and KDF is a Key Derivation Function (KDF) (realized as SHA-256 in all current variants).

**Compression.** The compression and decompression with compression parameter $d$ map a coefficient in $\mathbb{F}_q$ to a d-bit integer. Applied component- and coefficient-wise to vectors of polynomials, the size of the ciphertext is reduced. Moreover, compression and decompression add noise and thereby provide additional security [ABD+21b, Section 4.4]. In the case of attacks utilizing decryption failure leakage (see Section 3.2 and Chapter 5) decreases the information that may be obtained as described in [BDH+21b] and reiterated in Section 3.2.

The compression and decompression functions are defined as

$$\mathrm{Compress}(x, d) = \left\lceil x \frac{2^d}{q} \right\rfloor \bmod 2^d \tag{2.50}$$

[4]The official website may be found under `https://pq-crystals.org/kyber/`.

**Algorithm 7** Key generation of KyberPKE as specified in [ABD+21b].

```
Ensure: Secret key sk, public key pk
1: $d \leftarrow U^{32}$
2: $(\rho, \sigma) \leftarrow G(d)$
3: $\hat{\mathbf{A}} \in R^{k \times k} \leftarrow \text{Sample}_{\text{uniform}}(\rho)$
4: $\mathbf{s} \in R^k \leftarrow \text{Sample}_{\text{binomial},\eta_1}(\sigma)$
5: $\mathbf{e} \in R^k \leftarrow \text{Sample}_{\text{binomial},\eta_2}(\sigma)$
6: $\hat{\mathbf{s}}, \hat{\mathbf{e}} \leftarrow \text{NTT}(\mathbf{s}), \text{NTT}(\mathbf{e})$
7: $\hat{\mathbf{t}}\ \hat{\mathbf{s}}^\top \leftarrow \hat{\mathbf{A}} + \hat{\mathbf{e}}$
8: return $(\texttt{pk} = \text{Encode}_{12}(\hat{\mathbf{t}} || \rho), \texttt{sk} = \text{Encode}_{12}(\hat{\mathbf{s}}))$
```

**Algorithm 8** Encryption of KyberPKE as specified in [ABD+21b].

```
Require: Public key pk, Message $m \in \{0,1\}^{32}$, Seed $r \in \{0,1\}^{32}$
Ensure: $\texttt{ct} = (c_1, c_2)$
1: $\hat{\mathbf{t}}, \rho \leftarrow \text{Decode}_{12}(pk)$
2: $\hat{\mathbf{A}} \in R^{k \times k} \leftarrow \text{Sample}_{\text{uniform}}(\rho)$
3: $\mathbf{r} \leftarrow \text{Sample}_{\text{binom},\eta_1}(r)$
4: $\mathbf{e}_1 \leftarrow \text{Sample}_{\text{binom},\eta_2}(r)$
5: $\mathbf{e}_2 \leftarrow \text{Sample}_{\text{binom},\eta_2}(r)$
6: $\hat{\mathbf{r}} \leftarrow \text{NTT}(\mathbf{r})$
7: $u \leftarrow \text{NTT}^{-1}(\hat{\mathbf{r}}\hat{\mathbf{A}}) + e_1$
8: $v \leftarrow \text{NTT}^{-1}(\hat{\mathbf{r}}\hat{\mathbf{t}}^\top) + e_2 + \text{Decompress}(\text{Decode}_1(\texttt{m}), 1)$
9: $c_1 \leftarrow \text{Encode}_{d_u}(\text{Compress}_q(\mathbf{u}, d_u))$
10: $c_2 \leftarrow \text{Encode}_{d_v}(\text{Compress}_q(v, d_v))$
11: return $\texttt{ct} = (c_1, c_2)$
```

and

$$\text{Decompress}(x, d) = \left\lceil x \frac{q}{2^d} \right\rfloor. \tag{2.51}$$

Compression maps to the index of the closest multiple of $\frac{q}{2^d}$, i.e., to $l \in \{0, \ldots, 2^d - 1\}$ where $l$ is the integer such that $x$ is closest to $\frac{q}{2^d}$ for all $l' \frac{q}{2^d}$ with $l' \in \{0, \ldots, 2^d - 1\}$. Decompression maps, and integer $l$ (back) to $l\frac{q}{2^d}$. Visually, we may imagine the field $\mathbb{F}_q$ as discrete points on a circle with circumference $q$ on which $2^d$ evenly spaced compression points are selected starting at 0. Compression then maps any value to the index of the closest compression point, and decompression maps back to a compression point. This visualization is depicted in Figure 2.8.

**Algorithm 9** Decryption of KyberPKE as specified in [ABD+21b].

```
Require: Secret key sk, ciphertext $\texttt{ct} = (c_1, c_2)$
Ensure: Message $\texttt{m}'$
1: $\mathbf{u} \leftarrow \text{Decompress}(\text{Decode}_{d_u}(c_1), d_u)$
2: $v \leftarrow \text{Decompress}(\text{Decode}_{d_v}(c_2), d_v)$
3: $\hat{\mathbf{s}} \leftarrow \text{Decode}_{12}(sk)$
4: $\texttt{m}' \leftarrow \text{Encode}_1(\text{Compress}(v - \text{NTT}^{-1}(\hat{\mathbf{s}}\,\text{NTT}(\mathbf{u})^\top), 1))$
5: return $\texttt{m}'$
```

**Algorithm 10** Key generation of Kyber as specified in [ABD+21b].

```
Ensure: Secret key sk, public key pk
1: $z \in \{0,1\}^{32} \leftarrow \text{Sample}_{\text{uniform}}()$
2: $(\texttt{pk}, \texttt{sk}_{\text{pke}}) \leftarrow \text{KeyGen}_{\text{pke}}()$
3: return $(\texttt{pk}, \texttt{sk} = (\texttt{sk}_{\text{pke}}, \texttt{pk}, \text{H}(\texttt{pk}), z))$
```

**Algorithm 11** Encapsulation of Kyber as specified in [ABD+21b].

```
Require: Public key pk,
Ensure: $\texttt{ct} = (c_1, c_2)$, shared secret $K$
1: $\texttt{m} \in \{0,1\}^{32} \leftarrow \text{H}(\text{Sample}_{\text{uniform}}())$
2: $(\bar{K}, r) \leftarrow \text{G}(m || \text{H}(\texttt{pk}))$
3: $\texttt{ct} \leftarrow \text{Encrypt}(\texttt{pk}, \texttt{m}, r)$
4: $K \leftarrow \text{KDF}(\bar{K} || \text{H}(\texttt{ct}))$
5: return $(\texttt{ct}, K)$
```

**Encoding.** The function $\text{Decode}_l$ maps $32l$ byte arrays to (vectors of) polynomials and the function $\text{Encoding}_l$ is its inverse. Encoding of vectors of polynomials is done per component of the vector and concatenation. Decoding is defined to interpret the byte array as concatenated $l$-bit integers where each integer represents a polynomial coefficient. Polynomial coefficients in Kyber are elements of $\{0, \ldots, q-1\}$ and may be represented by $l$-bit integers with $l = \lceil \log_2(q) \rceil = \lceil \log_2(3329) \rceil = 12$. The encode function with $l = 1$ represents interpreting an element in $\{0, 1\}$ as bit.

**Error correction.** During the encryption the message is decoded to a polynomial as described in the previous paragraph. The other party, during the decryption, retrieves a noisy version of this polynomial with an additional, additive noise term on every coefficient. The noise term is given by

$$\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta\mathbf{u})^\top + e_2 + \Delta v \tag{2.52}$$

where the $\Delta$-terms arise from compression; the polynomial is coefficient-wise small with very high probability. Recovering the original bit from the noisy message is often called decoding, but this terminology is not used in Kyber. In Kyber, the error correction is described by using the compression function with compression parameter 1. Compression with parameter 1 maps a value $x$ in $\mathbb{F}_q$ to either 0 or 1 depending on whether $x$ is closer to 0 or to $\lceil \frac{q}{2} \rfloor$. The noise removal

**Algorithm 12** Decapsulation of Kyber as specified in [ABD+21b].

```
Require: Secret key $\texttt{sk} = (\texttt{sk}_{\text{pke}}, \texttt{pk}, \text{H}(\texttt{pk}), z)$, ciphertext ct
Ensure: Shared secret $K$
1: $\texttt{m}' \leftarrow \text{Decrypt}(\texttt{sk}_{\text{pke}}, \texttt{ct})$
2: $(\bar{K}', r') \leftarrow \text{G}(\texttt{m}' || \text{H}(\texttt{pk}))$
3: $\texttt{ct}' \leftarrow \text{Encrypt}(\texttt{pk}, \texttt{m}', r')$
4: if $\texttt{ct} = \texttt{ct}'$ then
5:     return $K \leftarrow \text{KDF}(\bar{K} || \text{H}(\texttt{ct}))$
6: else
7:     return $K \leftarrow \text{KDF}(z || \text{H}(\texttt{ct}))$
8: end if
```

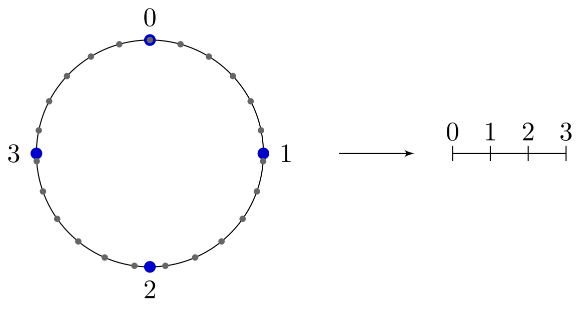


Figure 2.8: Compression visualized with $d = 2$ and $q = 23$. Points are mapped to the index of the closest compression point. The compression points are at $0, \frac{q}{4}, \frac{q}{2}$, and $\frac{3q}{4}$.

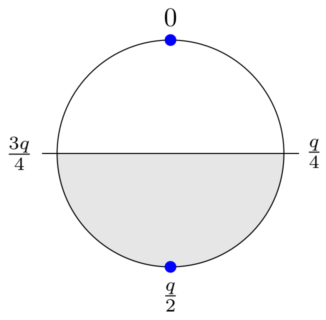


Figure 2.9: Removing noise realized as compression with $d = 1$. The upper half of the circle is closer to 0 and mapped to a 0-bit; the lower half is closer to $\frac{q}{2}$ and therefore mapped to a 1-bit.

is depicted in Figure 2.9.

**Number Theoretic Transform.** The ring $R = \mathbb{F}_q[x]/(x^n + 1)$ with $q = 3329$ and $n = 256$, as used in Kyber, does not contain a 512-root of unity, which is equivalent to not containing the nth root of $-1$. Instead, the root of unity used in Kyber is a $256^{\text{th}}$ root-of-unity, i.e., a 128-th root of $-1$, namely $\zeta = 17$. Therefore, $R$ is not isomorphic to a product of dimension-0 rings, and we only have

$$R = \mathbb{F}_q[x]/(x^n + 1) \cong \prod_{i=0}^{128} \mathbb{F}_q[x]/(x^2 - a_i) \tag{2.53}$$

for some factors $a_i$. In other words: $R$ splits into $\mathbb{F}_{q^2}^{\frac{n}{2}}$ instead of $\mathbb{F}_q^n$. The computational graph of the NTT therefore consists of $\log_2(n) - 1 = 7$ instead of 8 layers – the last layer is left out. The inverse NTT consists of 7 layers as well and the first layer does not exist. Multiplication in NTT domain is not pointwise (degree 0 multiplication) but the multiplication of polynomials of degree 1.

### Attacks

Common attacks used for security estimates in schemes such as Kyber, FrodoKEM, Saber, or NewHope are the primal and the dual attack (see specifications [ABD+21b, ABD+21a, BMJ+21, AAB+19] as well as [ACD+18]). These attacks allow to obtain a SVP from the given

LWE instance. The lattice problem may then be solved using lattice reduction, for example, by employing BKZ 2.0 [CN11] (see Section 2.2.1). Note that these attacks depend on many properties of the concretely posed scheme such as the modulus $q$, the number of samples $\mathtt{m}$, the quality of the basis reduction, the error distributions, etc.; for detailed analysis and proofs, we refer to [Reg09], [Pei14], and [Pei16] as well as the security analysis of, e.g., NewHope [ADPS16b] and Kyber [BDK+18] and the algorithm specifications which include security analyses (see, e.g., [ABD+21b]).

**The primal attack.** A LWE instance occurring in LWE-based schemes is typically given by a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, vectors $\mathbf{b}, \mathbf{e} \in \mathbb{Z}_q^m, \mathbf{s} \in \mathbb{Z}_q^n$ where $\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b}$. The vectors $\mathbf{s}$ and $\mathbf{e}$ are unknown to the attacker and are to be found. To achieve this, the LWE (which gives a BDD instance) can be embedded into a CVP instance: The secret vector $(\mathbf{e}, \mathbf{s})$ can be found by looking for an element of the lattice generated by the rows of

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} \\ \mathbf{A}^\top & \mathbf{I}_n \end{pmatrix} \tag{2.54}$$

close to the target vector $(\mathbf{b}, \mathbf{0})$. To solve the CVP instance, the attacker may use Kannan's embedding [Kan87] to obtain a uSVP instance given by the matrix

$$\mathbf{B}_{\text{svp}} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & 0 \\ \mathbf{A}^\top & \mathbf{I}_n & 0 \\ \mathbf{b} & \mathbf{0} & c \end{pmatrix} \tag{2.55}$$

where $c$ is commonly chosen to be 1. An optimal choice for $c$ is to set it to the standard deviation of the secret distribution (see, e.g., [DDGR20]). The secrets $(\mathbf{e}, \mathbf{s})$ may be recovered by finding a short vector in lattice generated by the rows of $\mathbf{B}_{\text{svp}}$. A detailed introduction to the primal attack and other attacks on LWE may be found in [APS15].

**The dual attack.** The dual attack, see [MR09, RS10, LP11], works by finding a short vector in the dual lattice (to the lattice $\mathcal{L}$ generated by $\mathbf{A}$) which is given by

$$\mathcal{L}^\top = \left\{(\mathbf{v}, \mathbf{w}) \in \mathbb{Z}^n \times \mathbb{Z}^m \mid \mathbf{A}^\top\mathbf{v}^\top \equiv \mathbf{w} \mod q\right\}. \tag{2.56}$$

For a small vector $(\mathbf{v}, \mathbf{w})$ in $\mathcal{L}^\top$, we have that for

$$z(\mathbf{x}) = \mathbf{x}\mathbf{v}^\top \tag{2.57}$$

$z(\mathbf{b})$ is equal equivalent to

$$z(\mathbf{b}) \equiv \mathbf{s}\mathbf{A}^\top\mathbf{v}^\top + \mathbf{e}\mathbf{v}^\top \equiv \mathbf{w}\mathbf{v}^\top + \mathbf{e}\mathbf{v}^\top \mod q \tag{2.58}$$

modulo $q$ and is therefore small. For uniformly random vectors $\mathbf{x}$, $z(\mathbf{x})$ is uniformly random. Using $z$, we may therefore differentiate between an LWE sample and a uniformly random vector, i.e., solve the D-LWE and in turn, depending on the concrete situation, the posed LWE problem.

**Security estimates.** The runtime of BKZ depends mostly on the block-size $\beta$. The required BKZ-$\beta$ can be estimated using the formula given in [ADPS16b, AGVW17] (refined in [DDGR20]): Denoting the lattice generated by the rows of $\mathbf{B}_{\text{svp}}$ by $\mathcal{L}$ and the root Hermite factor by $\delta_\beta$, the BKZ-$\beta$ needed to solve the uSVP instance can be estimated as the smallest $\beta \in \mathbb{N}$ satisfying

$$||s||\sqrt{\beta/\dim(\Lambda)} \leqslant \delta_\beta^{2\beta-\dim(\Lambda)-1}\operatorname{Vol}(\Lambda)^{1/\dim(\Lambda)}. \tag{2.59}$$

The volume in the case of Equation (2.55) is given by $q^{n-r}$ and the root Hermite factor may be estimated using the following equation (see [CN11])

$$\delta_\beta = ((\pi\beta)^{\frac{1}{\beta}} \frac{\beta}{2\pi e})^{\frac{1}{2\beta-2}}. \tag{2.60}$$

Note that there is no clear consensus on the most fitting model of estimating the security of LWE schemes. In [ADPS16b], it was proposed to use the Core-SVP hardness which is the computational effort of a single SVP-oracle call as arising during BKZ. It is noted in [ABD+21b, Section 5.2] that while this leads to conservative estimates, it does, for example, not take lattice sieving into account, which has seen considerable improvements since the introduction of Core-SVP [BDGL16, MLB17, Duc18, ADH+19]. As the details on the exact classical hardness of lattice-based schemes is of lesser relevance to our attacks, we do not expand on this topic. We do require the estimate of BKZ-$\beta$ for our attacks and treat BKZ as a black box fulfilling Equation (2.59).

## 2.3 Implementation Attacks

Implementation attacks target an implementation or device instead of (purely) the mathematical description or model of an algorithm. The publications of Kocher [Koc96] on timing and Kocher, Jaffe, and Jun [KJJ99] on differential power analysis showed that implementations attacks can be a threat to devices running cryptographic algorithms. In this section, we give a brief introduction to side-channel analysis and fault attacks and the models we evaluate our work in. Our work does not focus on the practical, physical aspects of SCA but on the properties of lattice-based cryptography which allow for implementation attacks, and on strategies to exploit them. Therefore, we limit ourselves to giving a brief overview of simple and differential power analysis, template attacks including the appropriate model for evaluation, Soft Analytical Side-Channel Attack (SASCA), recent developments in regard to neural networks and deep learning, and the results of carrying out fault attacks. For a comprehensive introduction, we refer to [MOP07] for side-channel analysis and to [JT12] for fault attacks.

### 2.3.1 Side-Channel Analysis

Side Channel Attack (SCA) makes use of information leaked during the execution of a cryptographic algorithm. Common types of leakage are for example timing, power consumption, or electromagnetic radiation. Recording such leakage may result in information that allows an adversary to obtain secret values, for example, to recover the secret key. Figure 2.10 shows an exemplary setup for recording power traces, i.e., measurements of power consumption, using a ChipWhisperer [OC14], and Figure 2.11 shows and example of a power trace as obtained from such a setup.

#### Timing Attacks

Timing attacks, published in [Koc96], make use of leakage in the time domain. If the execution time of a cryptographic algorithm depends on secret values, an attacker may infer information. These attacks are especially threatening as they may in some cases be executed remotely, i.e., without the device under attack being in control of the adversary. This has been shown, for example, by Brumley and Boneh [BB05] in an attack against the OpenSSL library. For this reason, cryptographic implementations should always be implemented such that their execution time does not depend on (secret) values, i.e., without conditional branching. This property is called being *constant time*. Related types of attacks are cache-timing and cache attacks, presented

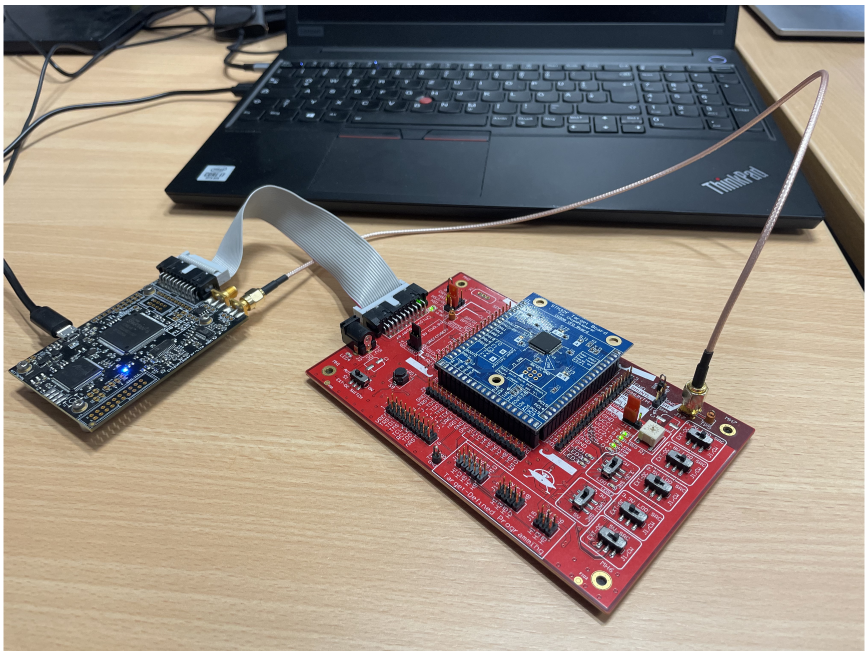


Figure 2.10: Recording power traces using a ChipWhisperer [OC14].

in [Ber05] and [OST06], respectively. There have been several attacks on lattice-based schemes exploiting or being improved by leakage in time, for example [SW07, PPM17, DTVV19, GJN20].

### Simple and Differential Power Analysis

Power analysis makes use of measurements of the power consumption of a device. Such attacks were published by Kocher, Jaffe, and Jun [KJJ99] in 1999 and are still a highly relevant threat to modern cryptography. It has been shown in [MD99] that power consumption correlates with the Hamming weight of processed values. This also led to the more general Hamming distance model and Correlation Power Analysis (CPA) [BCO04]. These types of attacks have been shown to be relevant for lattice-based cryptography as well, see, e.g., [WZW13, PPM17, PP19, ACLZ20, RRCB20, NDGJ21, RBRC22, XPR+22, UXT+22].

We here give an overview over Simple Power Analysis (SPA) and Differential Power Analysis (DPA). Other closely related techniques, summarized under the term DPA by, e.g., [SIH+23], are Mutual Information Analysis (MIA) [GBTP08], Linear Regression (LR) [SLP05], Collision Attacks (CA) [SLFP04], and Template Attacks [CRR02]; the latter is summarized in the next section. Note that the classification differs slightly, and, for example, template attacks are also sometimes classified as SPA (e.g., in [MOP07]).

**Simple Power Analysis.** SPA as described in [KJJ99] involves the inference of information from a single power trace, i.e., a measurement of the power consumption while the targeted algorithm is running on the device under attack. For example, a conditional branch depending on the secret key may be visible in a measurement of power consumption while the operation is carried out. This then leads to leaked information about the value the branching depends on. In [MOP07, Chapter 5], SPA is described as an attack in which the adversary “tries to derive the

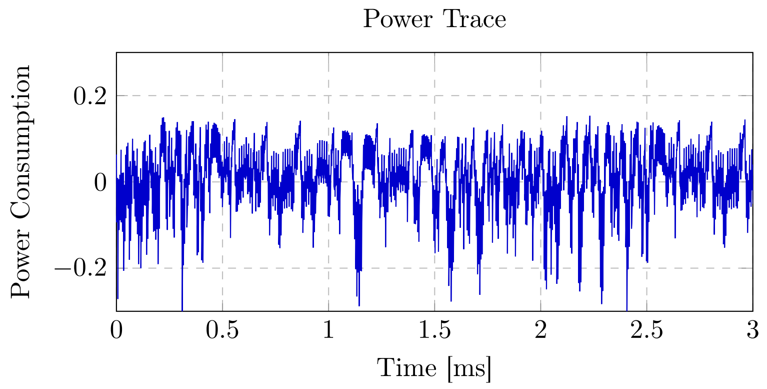


Figure 2.11: A power trace recorded using a ChipWhisperer [OC14].

key more or less directly from a single trace". The authors note that SPA often requires precise knowledge of the algorithm and the device under attack.

**Differential Power Analysis.** DPA as presented in [KJJ99] and [KJJR11] employs a divide-and-conquer approach by testing for values of sub-keys. In contrast to SPA, DPA does not require as detailed knowledge on the exact implementation of the algorithm and the internals of the device under attack. The adversary first chooses a selection function depending on a ciphertext and a sub-key mapping to $\{0, 1\}$. In the online phase, the adversary then first records a larger amount of traces of the encryption and stores the corresponding ciphertexts[5]. In the offline phase, the adversary then takes guesses for the sub-key. The recorded traces are grouped by the selection function, and the average of traces corresponding to a zero bit and of traces corresponding to a one bit are computed. If the sub-key guess is incorrect, the output of the selection function will be distributed uniformly random, and therefore the power trace will be uncorrelated with the averaged traces. For a correct guess, the output of the selection function will be constant and therefore correlated with the averaged traces. Therefore, the difference of the averaged traces (one corresponding to a zero bit and the other corresponding to a one bit) will be close to constant zero for an incorrect guess and differ to a larger extent for a correct guess (if a good selection function had been chosen).

In [MOP07, Chapter 7], DPA attacks are described as attacks depending on a large number of traces recorded of the device under attack (e.g., in contrast to template attacks which record a large number of traces of a *similar* device). According to the authors, the advantage is the reduction in required knowledge about the algorithm and device under attack.

### Template Attacks

Template attacks, published by Chari, Rao, and Rohatgi [CRR02], are a type of so-called profiling attacks. An adversary first records a large amount of traces of a device to create a template, i.e., a "profile" of a cryptographic algorithm running on a device in terms of power consumption per targeted intermediate value. A trace which is recorded while secret data is processed can then be matched against the template to obtain probability distributions of the intermediate values. An introduction in more detail is given in [MOP07, Chapter 5.3].

[5]Note that depending on the attack the plaintext needs to be known and stored.

**Noisy Hamming weight model.** Template attacks require a leakage model upon which the model creation is based. A common model is the Noisy Hamming weight model (compare with the results of [MD99, BCO04]). This model assumes that measurements of a variable with true value $a$ are of the form

$$\mathrm{HW}(a) + \mathcal{N}(0, \sigma) \tag{2.61}$$

where $\mathrm{HW}(a)$ is the Hamming weight of $a$ and $\mathcal{N}(0, \sigma)$ denotes the normal distribution with mean 0 and standard deviation $\sigma$.

**Template building.** To create the template, the adversary first records a large amount of traces using a similar (or in evaluation labs often the same) device. These will be required to create an accurate model of the computations running on the device. Then, points of interest corresponding to targeted operations are selected. Every point of interest will later result in a distribution for a processed value. Assuming the model, the adversary then computes the distribution giving Hamming weights per power consumption.

**Template matching.** By recording a single trace (or more with subsequent averaging) and matching the points of interest against the template, i.e., obtaining the distribution of Hamming weight for the recorded values at the points of interest, the adversary may retrieve a probability distribution for every value at every point of interest.

**Model and simulation.** Using the assumed model, e.g., the noisy Hamming weight model, a template attack may be simulated. First, the true value $a$ of an intermediate is obtained from the simulation, and its Hamming weight is computed. Then, an error sampled from a normal distribution with standard deviation $\sigma$ is added to the Hamming weight. This means we obtain

$$\tilde{a} = \mathrm{HW}(a) + e \tag{2.62}$$

for

$$e \leftarrow \mathcal{N}(0, \sigma). \tag{2.63}$$

We then simulate the obtained distribution from the template matching as a Gaussian distribution around $\tilde{a}$, i.e., the probability distribution (on Hamming weights) we obtain for the intermediate is

$$\mathcal{N}(\tilde{a}, \sigma). \tag{2.64}$$

The recovery methods making use of these distributions for the intermediates usually require values in the integers or in a finite field and not Hamming weights. This is because the key and therefore intermediates have to be recovered as integers, and mapping to Hamming weights in almost all cases loses a lot of information. If we, for example, record intermediate values during an (inverse) NTT in Kyber in a masked setting (see Section 2.3.3), values from a range of length (at least, depending on the implementation) $q = 3329$ are mapped to the set ranging from 0 to bit length required to store $q$, i.e., to $\{0, 1, \dots, \lceil \log_2(q) \rceil\} = \{0, 1, \dots, 12\}$. To obtain distributions on integer values in a larger range instead of Hamming weights, the Gaussian distributions around $\tilde{a}$ will often be mapped to distributions on integers in a certain range.

### Soft-Analytical Side-Channel Analysis

Data obtained in side-channel attacks, in particular from template attacks, often does not immediately yield the secret key. Veyrat-Charvillon, Gèrard, and Standaert [VGS14] propose a coding-theoretic approach to side-channel attacks. In practice, this is realized by performing a template attack and then using the obtained probability information as priors for belief propagation. Soft Analytical Side-Channel Attack (SASCA) aims to improve upon the noise tolerance and complexity of other approaches such as Algebraic Side-Channel Attack (ASCA) [RS09] (see also [RSV09] and [CFGR12]), which rely on building an algebraic model and then using e.g., a SAT solver to obtain the secret key (as well as on Tolerant Algebraic Side-Channel Attack (TASCA) [OKPW10, ORSW12] which employs an optimizer instead of a solver). Comparisons between ASCA, TASCA, and SASCA have been conducted in, e.g., [GS15] or [SIH+23].

Our work in Chapter 4 is an instantiation of SASCA, and we additionally show that belief propagation can be used more generally in implementation attacks. Belief propagation is summarized in Section 2.4.

### Neural Networks and Deep Learning

Recent advances in side-channel analysis in combination with neural networks were able to break several implementations including some which were higher-order protected. Several works target (protected) Advanced Encryption Standard (AES) [DR98, NistAes] implementations and show how countermeasures may be defeated using neural networks and deep learning, e.g., [MZ13, GHO15, MDM16, MPP16, CDP17, KPH+19, Tim19, PHJ+19, MS23].

While many works (see above) target AES, lattice-based cryptography has recently seen an increased focus: Ngo et al. [NDGJ21] target the message in the implementation of Saber in [BDK+21b]. The works of Ngo et al. [NWDP22] shows how higher-order masked implementations of lattice-based schemes may be targeted using a neural network. A message recovery attack on Kyber and Saber using deep learning has been proposed by Wang et al. [WND22]. The work of Dubrova et al. [DNG22] improves upon previous work using a technique the authors call recursive learning and targets Kyber.

Note that these techniques are being actively developed with very recent, conceptual improvements; in particular regarding threat and leakage models: In 2023, Masure et al. [MCLS23] answer the question of how to make use of knowledge on a masking schemes without relying on data usually not available to an adversary in practical machine learning-based attacks; they propose the *scheme-aware* threat model. A recent work, also from 2023, by Wu et al. [WAR+23], introduces a deep learning technique that does not depend on leakage models such as the Hamming weight or Hamming distance model, but that instead learns on independent bits.

The combination of belief propagation and neural networks has also already been proposed; some examples include the following works: Nachmani, Be'ery, and Burshtein [NBB16] introduce a belief propagation variant in which edges are weighted with weights that can be learned. Knobelreiter et al. [KSS+20] propose using belief propagation as a (trainable) layer in a neural network. Kuck et al. [KCT+20] propose belief propagation neural networks, which is a generalization of belief propagation using neural networks. Satorras and Welling [SW21] combine belief propagation with graph neural networks [SGT+09] and show that their method can improve upon the decoding of Low-Density Parity Check (LDPC)-codes [Gal62] under certain conditions.

### Side-Channel Analysis and Chosen-Ciphertexts

Several attacks on lattice-based cryptography have previously been enabled or improved by using a chosen ciphertext. This includes attacks that cause decryption failures, e.g., [BGRR19, GJN20, BDH+21b, DHP+22] (see Section 3.2). Another example of side-channel analysis improved by a chosen ciphertext is presented by Park and Han [PH16]; their work targets modular addition in the RLWE scheme presented in [LPR13] aided by a ciphertext in which every coefficient is set to one. Ravi et al. [RRCB20] use a chosen ciphertext and side-channel analysis to establish a plaintext checking oracle in Kyber and several other lattice-based schemes. Ngo et al. [NDGJ21] use a chosen ciphertext to launch an attack on the message in Saber enabled by deep learning techniques (c.f. Section 2.3.1). A side-channel attack that targets the message in several lattice based schemes (including Kyber) and is assisted by a side-channel was presented by Ravi et al. [RBRC22]. The work of Xu et al. [XPR+22] targets the secret key during the inverse NTT as well as the message in Kyber (see Section 3.1.4). Xu et al. [XPOZ22] target NTRU variants with a chosen ciphertext that enables side-channel analysis using Electro-Magnetic Radiation (EM) leakage. Sim, Park, and Han [SPH22] target the Barret reduction in Kyber to recover the secret key with six to eight traces. Ravi et al. [REB+22] state several attacks on variants of NTRU that make use of plaintext-checking oracles, decryption-failure oracle, and full-decryption oracles.

### 2.3.2 Fault Attacks

A fault attack is an invasive type of attack that manipulates data or execution of a device during its runtime. This includes, for example, voltage or clock glitching (see, e.g., [AK96, AK97]), manipulating the execution of instructions, or the usage of a laser to manipulate bits of data stored in memory [SA02, WWM11]. Another notable example is the Rowhammer attack [KDK+14], which allows manipulating DRAM memory by using malicious memory access patterns. The threat of fault attacks to cryptographic applications has been known for considerable time; among the first published works in this area is the “Bellcore Attack“ by Boneh, DeMillo, and Lipton [BDL97] who present attacks in the context of attacks on cryptographic schemes including the Rivest–Shamir–Adleman (RSA) scheme [RSA78, RSA83] and Fiat-Shamir schemes [FFS87].

The physical properties and the exact manner of fault attacks are of lesser relevance to our work. We treat the application of faults as a black box and describe the outcome an adversary needs to be able to achieve to carry out our attack presented in Chapter 5. Therefore, we do not give further details in this section and refer to [JT12] instead.

### 2.3.3 Countermeasures

Countermeasures against implementation attacks have already been proposed in the first papers on timing attacks and differential power analysis by Kocher [Koc96] and Kocher, Jaffe, and Jun [KJJ99], respectively. In general, countermeasures come in the form of hardware countermeasures and in form of software countermeasures. Important classes of software countermeasures relevant to our work in this thesis are constant time implementations, hiding countermeasures, and masking countermeasures.

#### Constant Time

To avoid dependency on secret data, cryptographic algorithms should be *constant time* as already proposed in [Koc96]. That means no conditional branching depending on secret data should take place. Note that it is often difficult to determine which parts of a cryptographic algorithm may potentially leak secret data if not implemented in constant time. For example, Guo et al. [GJN20] show that in a lattice-based setting an FO-transform, which is not implemented in constant time may leak information about the secret key even though no secret data is being processed directly. This is because a manipulated ciphertext may lead to the input to the FO-transform depending on the secret key. Thereby, the data processed in the FO-transform does indirectly depend on the secret key, and timing differences may leak information to an adversary.

### Hiding

Hiding countermeasures, already proposed in [KJJ99], randomize the execution of an algorithm, in time or amplitude dimension, to prevent an attacker from correlating a measurement with an intermediate variable. If an adversary cannot map a measurement to an intermediate value because they have been randomized, attacks are often mitigated to a certain extent. The countermeasures and known adaptations relevant to our work are summarized in Section 3.1.7; for a broader overview we refer to [MOP07, Chapter 7].

### Masking

Masking countermeasures, introduced in [GP99, CJRR99] (see also [Mes00, CG00, ISW03]), randomize the values processed by a cryptographic algorithm and thereby prevent an attacker from correlating the measurement with the true value of an intermediate. Common masking techniques include Boolean and arithmetic masking. This means a value to be randomized is either xor'ed with, added to, or multiplied with a random *mask*, i.e., a random value. The computation then takes place on the masked value which may be removed later by carrying out the operation or a similar operation on the mask as well. For example, to additively mask a linear function $f$ with input $x$, one could choose $n$ values $m_0, \dots, m_{n-1}$ and compute

$$f(x + m_0 + m_1 + \dots + m_{n-1}) \tag{2.65}$$

as well as all

$$f(m_i) \tag{2.66}$$

for $i \in \{0, \dots, n-1\}$. Then, $f(x)$ is computed as

$$f(x) = f(x + m_0 + m_1 + \dots + m_{n-1}) - f(m_0) - f(m_1) - \dots - f(m_{n-1}). \tag{2.67}$$

Thereby, $f(x)$ may be computed without having the computation of $f$ depending on $x$ alone. The number $n$ is called masking order; note that higher-order masking may be attacked by, e.g., higher order DPA. Masking is explained in more detail in [MOP07, Chapter 9]

## 2.4 Belief Propagation

Belief propagation is a message-passing algorithm first introduced in [Gal62] to decode LDPC codes and is also known as sum-product algorithm. Given probability distributions on several random variables with constraints/relationships, belief propagation computes the marginals of the joint probability distribution. While originally proposed for usage in coding theory, in particular, to decode LDPC codes, belief propagation has also proven to be a valuable tool for side-channel analysis [VGS14] (c.f. Section 2.3.1). Several works have identified use cases for belief propagation in the area of implementation attacks, e.g., targeting AES [VGS14, GRO18], Keccak [KPP20], or lattice-based cryptography [PPM17, PP19]. Theoretic analysis in regard to SASCA using belief propagation is provided by Guo et al. [GGSB20]. Note that belief propagation is only guaranteed to converge on graphs without loops, which is not the case in the aforementioned works, but *loopy belief propagation* has been known to provide useful results as well (see also [FM97]). In this work, we use belief propagation for side-channel analysis of the NTT as well as for recovery of information in case of side-channel or fault attacks. This section gives a short introduction based on [Mac03].

**Notation.** We denote elements $\mathbf{u}$ which are in NTT-domain by $\hat{\mathbf{u}}$; the same notation is used for FFTs. The inverse NTT is either simply called "inverse NTT" or INTT. Primitive roots of unity are denoted by $\zeta$ while potentially non-primitive roots of unity are denoted $\omega$; an index does not indicate a certain power of a primitive root of unity. Messages in belief propagation instantiations from node $i$ to node $j$ at time $t$ are denoted by $\mu_{i,j,t}$.

### 2.4.1 Factor Graph

Belief propagation works by passing messages along edges of a bipartite graph, called Tanner or *factor graph*. The factor graph consists of *variable nodes* and *factor nodes*: Variable nodes represent (potentially unknown) variables, and factor nodes model constraints on the variables. In other words: Belief propagation models the relationships between variables using variable and factor nodes. Variable nodes and factor nodes are connected if and only if the factor node models a constraint of the variable node. Nodes only connect to the respective other type of node.

More formally, the belief propagation graph is defined as follows: Given random variables $\mathbf{X} = (X_0, \dots, X_{n-1})$ with joint mass function $p(\mathbf{X})$ with

$$p(\mathbf{X}) = \prod_{i=0}^{n'} f_i(I_i) \tag{2.68}$$

for $n' < n$, $I_i \subseteq \{X_0, \dots, X_{n-1}\}$, and functions

$$f_i : I_i \to [0, 1] \subseteq \mathbb{R}, \tag{2.69}$$

the factor graph consists of variable nodes for the $X_j$ and factor nodes representing $f_i$. Edges from variable node $j$ to factor node $i$ express the relationship $X_j \in I_i$, i.e., that $f_i$ depends on $X_j$. Using this factor graph, belief propagation now aims at computing the marginal distributions,

$$p(X_0), \dots, p(X_{n-1}) \tag{2.70}$$

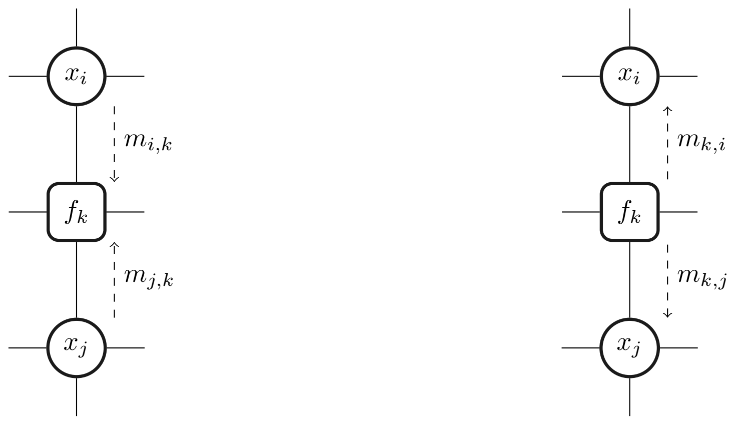
of the joint mass function $p(\mathbf{X})$.

### 2.4.2 Message Passing

Belief propagation works by passing messages from variable to factor nodes and vice-versa. Messages represent beliefs in the value of a random variable and are proportional to probability distributions. Nodes, variable nodes as well as factor nodes, update messages based on the messages received from their neighbors.

Many different proposals for scheduling methods for message-passing exist (see, e.g., [EMK06, SLG07, GK08, SM12]). In the area of side-channel attack, for example, Pessl and Primas [PP19] propose a different scheduling to improve upon a previous belief propagation instantiation.

In this work, we rely on a straightforward approach in which variable nodes hold a *prior distribution* which is being sent to factor nodes in the first step. This prior distribution can be thought of as an initial belief and comes from measurements or known values. The prior can also be realized as a separate factor node connected to a single variable node sending a constant message (as in, e.g., [PPM17, PP19]). In the second iteration, the factor nodes send out their updated messages to the variable nodes. We call an update process from variable to factor nodes and vice-versa, as shown in Figure 2.12, a *full iteration*.



(a) Variable to factor nodes. (b) Factor to variable nodes.

Figure 2.12: Belief propagation iteration phases as shown in [HMS+23]. The sub-graph consists of a factor node $f_k$ and two variable nodes $x_i$ and $x_j$.

### 2.4.3 Update Functions

Each node updates its beliefs based on the incoming messages (representing beliefs) from neighboring nodes (which all are of the respective other type of node). Denoting messages from node $i$ to node $j$ at iteration $t$ as $\mu_{i,j,t}$, the update rules are given as follows:

- From variables to factors: Variable nodes (here with index $i$) compute messages $\widehat{\mu}'_{i,j}$, which are sent to factor nodes with index $j$, as

  $$\mu_{i,j,t+1}(x) = \prod_{j' \neq j} \mu_{j',i,t}(x). \tag{2.71}$$

  The message represents the belief of the variable not taking into account the $j$-th node itself and is computed from all other adjacent factor nodes.

- From factors to variables: Factor nodes (here with index $j$) compute messages $\mu_{j,i,t+1}(x)$, which are sent to the $i$-th variable node, as

  $$\mu_{j,i,t+1}(x) = \sum_{\mathbf{x}, x_i = x} f_j(\mathbf{x}) \prod_{i' \neq i} \mu_{i',j,t}(x). \tag{2.72}$$

  The message is computed from all received messages, i.e., from all neighboring variable nodes, by marginalizing over all beliefs except for the $i$-th one.

- Belief Update: After the two passes, variable nodes hold updated beliefs given by

  $$b_i(x) = \prod_{j} \mu_{j,i}(x). \tag{2.73}$$

  This means that the beliefs at a variable node are given by the product of the beliefs of all adjacent nodes. Note that the belief $b_i$ does not need to be computed in every iteration but only whenever results for the $i$-th variable are to be obtained.

### 2.4.4 Marginal Distributions

Upon convergence or after an abort criterion has been reached (see Section 7.1.1), the computed (approximate for the) marginal distribution at a variable node with index $i$ is given by the normalization of $b_i$. Note that convergence is not guaranteed for cyclic belief propagation graphs. Nodes can be merged; merging all nodes corresponds to an exact computation, and merging factor nodes means computing this sub-function in an exact manner. From this point of view, belief propagation offers a trade-off between exactness and computational complexity. This principle is being made use of in SASCA – an exact computation would yield the same or better results but is, in many cases, computationally infeasible.

Chapter 3

# Related Work

Implementation vulnerabilities in post-quantum cryptography, especially Learning with Errors (LWE)-based schemes, are an important and pressing matter. In particular, the vulnerabilities of major building blocks such as the Number Theoretic Transform (NTT), the Fujisaki-Okamoto (FO)-transform, and the error correction (which may leak information due to decryption failures) require profound understanding before being implemented and used in a wide variety of use cases. Some analysis of the vulnerability of major building blocks of LWE based schemes has already been provided by previous work, but the questions posed in Section 1.1.2 are yet unanswered. In this chapter, we give an overview of related work and summarize the state of the art concerning vulnerabilities of the NTT, the utilization of decryption for implementation attacks, and the recovery of secret information from decryption failure information.

We evaluate related work with regard to our research questions stated in Section 1.1.2. The relevant properties are thus the targeted secret value, the requirements on the capabilities of the attacker, whether straight-forward or known countermeasures prevent the attack, and if security estimates are available in cases where the attacker cannot fully recover the secret (c.f. Table 1.1, Table 1.2, and Table 1.3). In this chapter, we first and foremost reiterate previous work that fulfills particularly many of these desired properties as well as works that are necessary to understand our methods. We also give a short overview of other approaches, e.g., with different targets, that are less relevant regarding our research questions. An overview of the related work is provided in Section 3.4 and summarized in Table 3.4.

We first summarize two previous attacks [PPM17, PP19] targeting the NTT: Primas, Pessl, and Mangard [PPM17] present a side-channel attack on the inverse NTT during the decryption process of an Ring Learning with Errors (RLWE) scheme as proposed by [LPR13]. Pessl and Primas [PP19] target the NTT during encryption and improve the belief propagation of [PPM17]; they thereby achieve a higher noise tolerance but, in most cases, cannot recover the long-term secret. In addition, Xu et al. [XPRO20] propose an Simple Power Analysis (SPA)-like approach to target the inverse NTT using a chosen ciphertext but cannot target protected implementations. We then reiterate the countermeasures of Ravi et al. [RPBC20], which were proposed in response to previous attacks on the NTT. Finally, we discuss how countermeasures have previously been circumvented in different settings.

In regard to targeting the error correction and utilizing decryption failures for implementation attacks, several attacks have previously been published. In this section, we reiterate the following works that target Kyber and use a chosen ciphertext that differs from a valid ciphertext by a single bit; these chosen-ciphertext attacks cannot be prevented by a straight-forward statistical countermeasure. The works of Bhasin et al. [BDH+21b] and D'Anvers et al. [DHP+22] use a side-channel to observe the outcome of the FO-transform after a chosen-ciphertext attack; similar strategies have been discussed in [BGRR19] and used in [GJN20]. Pessl and Prokop [PP21] use a fault in the decoding routine to potentially cause a decryption failure in Kyber. Delvaux [Del22] improves upon an attack presented in this thesis in Chapter 5. We give a short overview of fault attacks with different targets and their limitations in Section 3.2.5, but these works are out-of-scope in regard to our research questions.

Several approaches to solving for the secret key from decryption failure information exist. The framework of Dachman-Soled et al. [DDGR20] allows dealing with side-channel information in a widely applicable way. Bhasin et al. [BDH+21b] show that it can be used to estimate the required information in the case of decryption failures in Kyber as arising in the attacks of [PP21, BDH+21b, HPP21, DHP+22], but do not perform an end-to-end key recovery, which is computationally expensive (see, e.g., [DHP+22, Section 3.1]). An extension to the framework of [DDGR20] published by Dachman-Soled et al. [DGHK22] allows solving the information obtained in the attack of Fahr et al. [FKK+22]; due to the different nature of inequalities in [FKK+22] and the aforementioned attacks on Kyber, an end-to-end key recovery – again – proves to be computationally expensive. The work of Pessl and Prokop [PP21] uses a statistical approach and requires less information compared to the amount estimated to be required with [DDGR20]. Delvaux [Del22] improves the methods presented in [HPP21] as part of this thesis and [PP19] to include error tolerance and reduce the runtime[1].

This chapter first summarizes the state of the art on attacks on the (inverse) NTT in Section 3.1; this section serves as a basis for Chapter 4. The second section, Section 3.2, states current implementation attacks utilizing decryption failures, relating to Chapter 5. The last section, Section 3.3, gives current recovery methods to obtain the secret key from decryption failure information; our method is presented in Chapter 6.

## 3.1 Attacks on Number Theoretic Transforms

Multiplication of polynomials in several RLWE and Module Learning with Errors (MLWE) schemes, including Kyber and NewHope, is performed using an NTT. As a main building block realizing multiplication, the NTT is a particularly valuable target. To multiply two polynomials, they are transferred to the NTT-domain using the NTT (in $O(n \log n)$). In NTT-domain, the polynomials may be multiplied point wise or almost point wise. The inverse NTT on the result of the multiplication in NTT domain gives the product of both polynomials in normal domain.

The attacks presented in [PPM17] and [PP19] are single-trace attacks. Another example of a single trace attack targeting lattice-based cryptography is stated by Amiet et al. [ACLZ20]; they target the message in an unprotected NewHope implementation. Note that similar attacks could apply to different multiplication schemes as well. The work of Mujdei et al. [MBB+22] targets the polynomial multiplication in several lattice-based schemes using Correlation Power Analysis (CPA). Their analysis includes Saber [DKRV18], NTRU [HPS98], and Kyber in an older version [ABD+19a].

### 3.1.1 The NTT as Target

In MLWE-based schemes an incoming ciphertext commonly contains two components, namely $(v, \mathbf{u})$. During the decryption the polynomial

$$\tilde{m} = v - \mathbf{u}\mathbf{s}^{\top} \tag{3.1}$$

[1]The method in [HPP21] did not include error tolerance; the improved method [HMS+23], presented in this work, does.

has to be computed where $\mathbf{s}$ is the secret polynomial, and $\tilde{m}$ is the noisy message. If an NTT is used, this computation involves an inverse NTT (INTT) and is commonly carried out as

$$\tilde{m} = v - \mathrm{INTT}(\hat{\mathbf{u}}\hat{\mathbf{s}}^\top) \tag{3.2}$$

where $\hat{\mathbf{u}}, \hat{\mathbf{s}}$ are the respective NTT-images of $\mathbf{u}$ and $\mathbf{s}$. Therefore, the decryption step of an MLWE based scheme calls an inverse NTT with secret input. Obtaining side-channel information may therefore reveal the secret key, and this makes the inverse NTT during decryption a valuable target. In addition, the NTT is used during the encryption to compute

$$\hat{\mathbf{r}} = \mathrm{NTT}(\mathbf{r}), \tag{3.3}$$

which is used to compute $(v, \mathbf{u})$ in the first place as

$$\mathbf{u} = \mathrm{NTT}^{-1}(\hat{\mathbf{r}}\hat{\mathbf{A}}) + \mathbf{e}_1 \tag{3.4}$$

and

$$v = \mathrm{NTT}^{-1}(\mathrm{NTT}(\mathbf{t})\hat{\mathbf{r}}^\top) + e_2 + \mathrm{Decompress}_1(\mathrm{Decode}_1(m)), \tag{3.5}$$

where the variables $\mathbf{t}$, $e_2$ are as defined in Section 2.2.3, and $\mathrm{Decompress}_1(\mathrm{Decode}_1(m))$ is the message mapped to a polynomial. Thus, an attack targeting the NTT during encryption may recover the message (which could, e.g., be used to derive the session key).

The NTT does not allow for a direct, full key recovery as the secret key is only loaded or stored at the very beginning or end. But, during the NTT, information about the secret key is processed over a comparably long execution time, and therefore the NTT offers many different, uncorrelated points of measurement. In addition, as noted in [PPM17], the intermediate values are connected arithmetically by relatively simple operations. Information on intermediate values may be obtained by using a template attack (c.f. Section 2.3.1). The information arises in the form of probability distribution for each intermediate value; those probability distributions are correlated with the secret key but usually do not allow for direct recovery due to measurement noise. Instead, the distributions are processed using belief propagation, which allows recovering secret key in some cases and requires an additional lattice reduction step in others.

Belief propagation is a statistical, message-passing algorithm that allows computing the marginal distributions of a joint probability function (see Section 2.4). In this case, an attacker retrieves probability distributions for intermediate values from the template attack, which are used as priors in the belief propagation. Belief propagation then computes the marginal for the joint probability function, i.e., the probability distribution of each coefficient given the distributions of all coefficients. To achieve this, a factor graph representing the computation graph of the NTT and all targeted intermediate values is initiated with the measured priors. Message passing and Bayesian updating then result in the marginals. Depending on the attack, the key can then either be directly re-constructed or fed into a lattice reduction algorithm.

### 3.1.2 The Attack of Primas, Pessl, and Mangard

The first attack on the (inverse) NTT in the fashion described above was published by Primas, Pessl, and Mangard [PPM17] in 2017. For their practical evaluations on a real device, the authors of [PPM17] target a Cortex-M4F using Electro-Magnetic Radiation (EM) leakage. They recover the secret key from a single observation of the decryption routine of an RLWE scheme as presented in [LPR13] (c.f. Section 2.2.3) featuring an NTT with parameters $n = 256$, $q = 7681$, $\sigma = 4.51$. Similar parameters are used in a variety of implementations, e.g., [CRVV15, GFS+12, PG13, LSR+15, POG15, RRVV15]. Note that in the case of an RLWE scheme, $k = 1$ and here, the subtraction of Equation (3.2) is turned into an addition due to a slightly differently constructed scheme. This means that the targeted point of computation is the calculation of

$$\tilde{m} = v + \mathrm{INTT}(\hat{u}\hat{s}). \tag{3.6}$$

The attack works by first employing a template attack on the NTT computation in Equation (3.2), running a belief propagation, and then a lattice reduction step. The lattice reduction step is needed as, to optimize the belief propagation run-time, the NTT graph is reduced in size and splits into two separate graphs. While this reduces computational effort, it prevents an immediate recovery without an additional lattice reduction step.

## Template Attack on the NTT

The work in [PPM17] targets the modular arithmetic in the butterflies of the inverse NTT in the implementation of [CRVV15] on a Cortex M4. The modular arithmetic in this implementation is realized by using the internal multiplication and division instructions, and, to reduce modulo $q$, a conditional statement potentially causes a subtraction/addition. The authors of [PPM17] are able to detect the chosen branch (of this conditional statement) in a template attack with high accuracy; this is featured in their attacker model.

For each twiddle factor, i.e., for each block of butterflies per layer and each operand, they record 100 traces leading to a total number of about 100 million traces. The first trace is used to distinguish the division operations (by timing), which depends on the bit size of the operand and allows classifying values by Hamming weight in a first step. The remaining traces are used to create templates for the multiplications. The authors report that in the metric of [SMY09], the average entropy per key coefficient lies at around 7 bit, but depends on the twiddle factor; the entropy without further information is $\log_2(q)$ which is about 12.9 bit.

**Hamming weight leakage model and simulation.** In addition to their attack on a physical device, a simulated attack is provided, which makes their work reproducible. To simulate the attack, the Hamming weight leakage model (see Section 2.3.1) is used. The simulations are carried out with different noise levels given by a standard deviation $\sigma > 0$. For a targeted value $a$ an attacker obtains a normal distribution with expected value

$$a + e \text{ where } e \leftarrow \mathcal{N}_\sigma(0). \tag{3.7}$$

This means for every butterfly with twiddle factor $\omega$, the attacker obtains two distributions,

$$\mathcal{N}_\sigma(a + e_0) \text{ and } \mathcal{N}_\sigma((a\omega \mod q) + e_1), \tag{3.8}$$

where

$$e_0, e_1 \leftarrow \mathcal{N}_\sigma(0). \tag{3.9}$$

## Belief Propagation

From the template attack, the authors obtain distributions for intermediate values in the inverse NTT. Those are now modeled as variable nodes in a belief propagation graph. The factor nodes are multiplication, addition, and subtraction nodes; those represent the arithmetic operations carried out during the NTT as well as whether a reduction in the respective computation happened. The addition and subtraction are operations of degree 3, whereas “mul” is multiplication with the (constant) twiddle factor and therefore of degree 1. Using these nodes, the computational graph of the NTT is represented as a belief propagation graph. Figure 3.1 shows a butterfly modeled

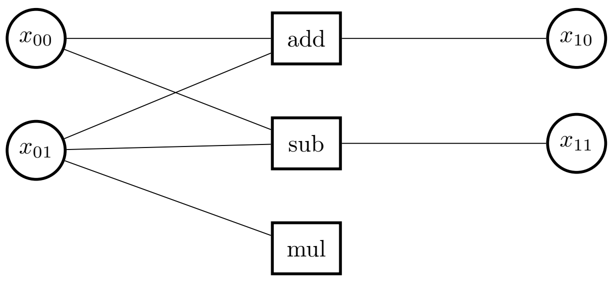


Figure 3.1: A butterfly representing a belief propagation graph as used and depicted in [PPM17]. The nodes $x_{00}$ and $x_{01}$ represent inputs to the butterfly, and the remaining variable nodes (circles) represent outputs. The factor nodes (squared) represent addition, subtraction, and multiplication as well as whether a reduction happened in the respective computation.

by belief propagation nodes as depicted and used in [PPM17]. The node function for addition, $f_{\text{add}}$ is given by

$$f_{\text{add}}(x_{00}, x_{01}, x_{10}) = \begin{cases} 1 & \text{if } x_{00} + x_{01} \equiv x_{10} \bmod q,\ x_{00} + (\omega x_{01} \bmod q) \geqslant q \\ 0 & \text{else} \end{cases}, \tag{3.10}$$

and for subtraction, $f_{\text{sub}}$ is given by

$$f_{\text{sub}}(x_{00}, x_{01}, x_{11}) = \begin{cases} 1 & \text{if } x_{00} - x_{01} \equiv x_{11} \bmod q,\ x_{00} - (\omega x_{01} \bmod q) < q \\ 0 & \text{else} \end{cases} \tag{3.11}$$

where $x_{00}$ and $x_{01}$ are the inputs to the butterfly and $x_{10}$ and $x_{11}$ are the outputs. The first condition in both nodes represents the computation, and the second is the information about if a reduction happened. This means, the first condition corresponds to the EM, and the second condition to the timing leakage.

The authors now build a graph modeling the inverse NTT from these nodes. As a single graph, the belief propagation does not converge well due to multiplications spreading unevenly throughout the NTT and due to varying behavior of the template attack depending on the twiddle factor. Therefore, the NTT graph is split up into three different sub-graphs which do not include the first layer.

**Performance optimizations.** Note that a naive execution of the belief propagation as described previously is computationally expensive; to reduce the runtime of the belief propagation, the authors propose using the Fast-Fourier Transformation (FFT) to speed up the updating process. This is possible due to the fact that the addition of random variables is the convolution of distributions, and pointwise multiplication in FFT-domain is convolution in normal domain. Therefore, for example, given the butterfly calculation $a + b$, the following is computed: Let $A$, $B$ be the random variables corresponding to $a$, respectively $b$, and let $\mu_a$, $\mu_b$ be the corresponding distributions arising in belief propagation. To compute the distributions $\mu_{a+b}$, i.e., the distribution of $A + B$, the Fourier transforms $\hat{\mu}_a = \text{FFT}(\mu_a)$ and $\hat{\mu}_b = \text{FFT}(\mu_b)$ are computed. Then, the inverse FFT of the product of $\hat{\mu}_a, \hat{\mu}_b$

$$\text{IFFT}(\hat{\mu}_a \hat{\mu}_b) = \text{IFFT}(\text{FFT}(\mu_a)\,\text{FFT}(\mu_b)) \tag{3.12}$$

is the distribution of $A + B$, i.e., $\mu_{a+b}$.

**Lattice Decoding**

Using the full belief propagation graph, the inverse NTT would be modeled completely, and the first layer would allow to recover the key in NTT domain. As the sub-graphs do not include all results of the inverse NTT computation, an additional lattice reduction step is required. The NTT is linear; therefore, recovering intermediate values of a layer immediately gives linear equations involving the secret key. The solution space generated by those equations is then intersected with the public key equation reducing the dimension by the amount of equations coming from intermediate values. In practice, this means solving the obtained equations for key coefficients and then substituting the solution into the public key equation. In this scheme, 192 intermediates are available coming from the 3 sub-graphs resulting in a lattice problem of dimension 64, which is solvable in practice on widely available hardware.

**Results**

The authors evaluated their attack on a real device as well as in the Hamming weight leakage model. In the case of the real device, their attack is successful with 20 iterations of belief propagation. In the Hamming weight leakage model, they report a success rate of over 0.9 for both masked and unmasked setting up to $\sigma = 0.4$. Then, the masked setting drops around $\sigma = 0.4$ and the unmasked setting shortly after, at $\sigma = 0.5$. The success rate is 0 for both settings at around $\sigma = 0.7$.

## 3.1.3 The Attack of Pessl and Primas

The attack of [PPM17] is improved by timing leakage and requires a large amount of traces to create the templates. Pessl and Primas [PP19] present an attack on the NTT that targets constant time implementations and requires fewer traces to obtain the templates. This is possible as, firstly, their attack targets the NTT during the encryption step, while [PPM17] targets the decryption step. Secondly, they introduce several improvements to the belief propagation, which leads to a more efficient (in terms of required information) recovery of the secret key. Their improvements allow for a practical, more noise-tolerant attack on an implementation that does not leak timing information. The downside to targeting the encryption step is that the secret key `sk` (often the long-term secret) may not be extracted, but only the message `m` (often the session key) can be recovered.

Instead of targeting the computation of the inverse NTT in

$$\tilde{\mathtt{m}} = v - \mathrm{INTT}(\hat{\mathbf{u}}\hat{\mathbf{s}}^\top) \tag{3.13}$$

during decryption, the authors target the computation of

$$\hat{\mathbf{r}} = \mathrm{NTT}(\mathbf{r}) \tag{3.14}$$

during encryption. The vector of polynomials $\hat{\mathbf{r}}$ is used to compute the ciphertext and recovering it allows recovering the message $\mathtt{m}$.

The belief propagation features three separate improvements: First, they introduce message damping; this may lead to improvements in factor graphs with short loops. Second, they explain how the results may be improved by a different message scheduling. Third, to eliminate the short loops in the factor graph, they introduce a butterfly node unifying the three factor nodes that are used to model a butterfly in [PPM17] into a single node.

### Belief Propagation Improvements

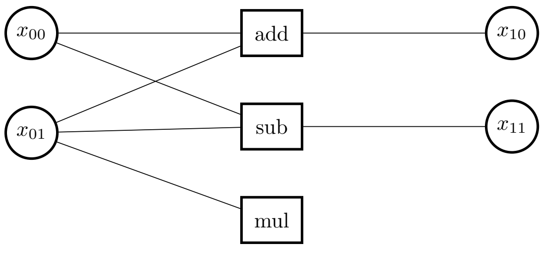
The improvements to the belief propagation incorporated by the authors of [PP19] are inspired by well-known results in the area of decoding algorithms. In particular, the works of Storkey [Sto03] and Yedida [Yed04] provide in-depth analysis of belief propagation in different but comparable contexts. The work of [Sto03] analyses belief propagation in the context of FFTs – which are conceptually similar to NTTs – and [Yed04] provides analysis on Reed-Solomon codes, which can be decoded using an NTT.

The first improvement is message damping with factor of $\alpha = 0.9$. The goal is to reduce oscillation, which may occur as the factor graph is far from being a tree. Message damping computes the average between previous messages with the currently computed distribution where the current message is weighted with factor $\alpha$, and the previous one is weighted with $(1 - \alpha)$.

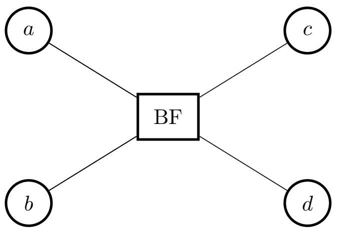
The second improvement is a different message scheduling. The straightforward approach used in [PPM17] is to update all nodes simultaneously. This approach leads to nodes with large distances, which take a long time before influencing each other, while the shortest loops are completed in a single full iteration. The more balanced scheduling that is used in [PP19] consists of first passing messages in only one direction from input to output before reversing and passing in the other direction.

The third improvement is using a factor node that models a butterfly and combines the three factor nodes used in [PPM17] into a single node. This butterfly node represents multiplication with the twiddle factor $\omega$, addition of the inputs, and subtraction of the inputs. Thereby, the joint probability function of a single butterfly is computed in an exact manner and the amount of loops in the factor graph is reduced. The node function for a butterfly node is given by

$$f_{\text{bf}}(x_{00}, x_{01}, x_{10}, x_{11}) = \begin{cases} 1 \text{ if } x_{00} + x_{01} \equiv x_{10} \text{ and } x_{00} - \omega x_{01} \equiv x_{11} \mod q \\ 0 \text{ else} \end{cases} . \tag{3.15}$$



(a) Separate nodes as in [PPM17].



(b) Butterfly node as in [PP19].

Figure 3.2: A butterfly modeled with separate factor nodes, and a butterfly modeled by a single butterfly node. The butterfly node computes the joint probability function of intermediates at one butterfly in an exact manner.

### Targeting Encryption

The work of [PPM17] targets the decryption process recovering the secret key of them Key Encapsulation Mechanism (KEM) (i.e., in many use cases the long-term secret key). The routine targeted in both [PPM17] and [PP19] is the computation of INTT($\hat{\mathbf{u}}\hat{\mathbf{s}}^\top$) (but in [PPM17] only with $k = 1$). But in [PPM17] coefficients of $\mathbf{u}$ are distributed uniformly random over the full range of the base field and not restricted in size (apart from being coefficients of $\mathbb{F}_q$).

On the other hand, when targeting encryption, as in [PP19], the input values are small, as the value $\mathbf{r}$ is sampled from a binomial distribution with small support, $\eta_1$ in Kyber (c.f. Section 2.2.3). This arrows down the support of the prior distributions and, at the same time, gives more precise measurements of intermediate values (c.f. Section 2.3.1).

Note that the encryption subroutine is used in the decapsulation as well, and the key generation features a similar NTT call – this gives an additional target for the attack of [PP19]. During the key generation, the computation which may be targeted is

$$\hat{\mathbf{s}} = \mathrm{NTT}(\mathbf{s}). \tag{3.16}$$

Therefore, the attack of [PP19] may target the session key in encapsulation, decapsulation, and, in exceptions, even the secret key (and not just the message) during key generation.

### Masked Implementations

Masking the NTT takes away a large advantage of the attack described in [PP19]: The inputs to the NTT are no longer small. The authors therefore propose to join the factor graph so that the implicit unmasked result is taken into account. This approach allows retaining some advantage of the small coefficients of the unmasked version of $\mathbf{r}$. Nevertheless, in contrast to [PPM17] the noise tolerance (in terms of $\sigma$ in the noisy Hamming weight model) is more than halved by employing masking.

### Recovering the Shared Secret

In the main scenario, when targeting the message during encryption, the authors do not immediately recover the message. Instead, the value of $\mathbf{r}$ is recovered which is used during encryption to compute the ciphertext. Note that $\mathbf{r}$ is a vector of polynomials, and the attack has to be carried out for each of the $k$ components of $\mathbf{r}$. The message can be computed as follows: With the notation of Section 2.2.3 we have

$$v = \mathbf{t}\mathbf{r}^\top + e_2 + \mathrm{Decompress}_1(\mathrm{Decode}_1(\mathtt{m})) \tag{3.17}$$

and

$$c_2 = \mathrm{Encode}_{d_v}(\mathrm{Compress}_{d_v}(v)) \tag{3.18}$$

where $c_2$ is known to the attacker by additionally observing communications, $\mathbf{r}$ was recovered through the side-channel analysis, and $\mathbf{t}$ can be obtained from the public key. Thus, $\mathtt{m}$ can be obtained by first computing

$$c_2' = \mathbf{t}\mathbf{r}^\top \tag{3.19}$$

and then decoding the difference between $c_2$ and $c_2'$, i.e., computing

$$\mathtt{m} = \mathrm{Decode}_1(c_2 - c_2'). \tag{3.20}$$

As the difference between $c_2 - c_2'$ is approximately

$$e_2 + \mathrm{Decompress}_1(\mathrm{Decode}_1(\mathtt{m})) \tag{3.21}$$

and $e_2$ is also small, this in fact yields the correct value $\mathtt{m}$.

**Results**

The authors evaluate their attack in both the parameter set used in [PPM17] and on Kyber. In both cases, the noisy Hamming weight model with variance $\sigma$ is used. Additionally, they evaluate the belief propagation without employing their improvements. The results for different parameter sets differ only slightly both with a success rate of 1 for $\sigma < 1.4$ and a positive success rate for $\sigma \leqslant 2$ in the unmasked case. In the unmasked case, the success rate for the factor graph without improvements is 1 up to a $\sigma$ of 0.5 and positive up to a $\sigma$ of 1.7. In the masked case, the separate graph does not produce satisfactory results, a success rate of greater than 0.5 is not reached regardless of $\sigma$, and the success rate is 0 beginning with $\sigma = 0.4$. The joined factor graphs achieve a positive success rate for $\sigma \leqslant 0.3$ but drop to zero quickly after; with $\sigma \geqslant 0.5$, no successes are reported.

Additionally, the authors provide analysis on a real device, an STM32F405, using a ChipWhisperer UFO board [CwUfo, Cw32F4, OC14], measuring power consumption. They record 1900 traces to obtain the template and evaluate their attack on 100 traces. For the practical attack on the real device, they report a success rate of 0.57, coming from a success rate of 0.83 per targeted NTT.

### 3.1.4 The Attack of Xu et al.

Xu et al. [XPR+22][2] provide another attack on the inverse NTT targeting the secret key at the same location. Their attack relies on a chosen ciphertext and EM leakage targeting the modular reduction during the inverse NTT. In addition, they propose an attack on the mapping from message bits to polynomials and vice-versa. The authors note that operations after the NTT could be targeted as well but do not carry out these proposals.

The attack of [XPR+22] uses a chosen ciphertext in which components of $\mathbf{u}$ are set to certain constants. In the device under attack, the coefficients of $\mathbf{u}$ are multiplied with the secret key coefficients in $\{-2, -1, 0, 1, 2\}$ during the inverse NTT. The authors choose the coefficients such that this multiplication leads to different Hamming weights depending on whether the respective secret key coefficient is in certain subsets of $\{-2, -1, 0, 1, 2\}$. A subsequent SPA allows partitioning key coefficients into these subsets by their Hamming weight from which the coefficients can be reconstructed after four traces with different constants and subsets. Their attack has been carried out in practice, does not require a profiling device, and can thus be assumed to be more portable (compared to [PPM17, PP19]) because no templates are required. While this is a clear advantage, the approach to target the inverse NTT relies on having few possible values for the secret key; therefore, it is fully prevented by masking countermeasures, which have to be assumed to be in place in a realistic scenario.

### 3.1.5 The Countermeasures of Ravi et al.

The NTT is a major building block of several RLWE and MLWE based schemes, e.g., [ADPS16b, BDK+18], and the usage of an NTT has been proposed for other schemes as well [LS19, CHK+21]. The attacks of [PPM17] and [PP19] show that the (inverse) NTT is a valuable target for side-channel attacks as message `m` as well as the secret key `sk` may be recovered. Therefore, protecting the NTT as well as the inverse NTT against side-channel attacks and understanding the cost of such countermeasures is crucial. Ravi et al. [RPBC20], in response to [PPM17] and [PP19], present several masking and shuffling countermeasures for the (inverse) NTTs used in Kyber

[2]Note that there is a preprint [XPRO20] first published in 2020.

and Dilithium (the latter is a signature scheme introduced in [DKL+18] and currently specified in [BDK+21a]).

Their work proposes several masking countermeasures protecting the atomic operations of the (inverse) NTT by masking the twiddle factor multiplication. The outputs to a butterfly are multiplied with an additional twiddle factor $\omega$, a power of a $2n$-th root of unity, i.e., $\omega = \zeta^k$ for some $k$ where $\zeta$ is the Kyber root of unity as in Section 2.2.3. Thereby, the relationship between the in- and outputs of a butterfly is randomized, and the authors conjecture that this leads to worsened belief propagation performance. The masking countermeasures come in several variants that offer different levels of protection and result in varying performance degradation; the more entropy is introduced and protection is offered, the more expensive is the countermeasure.

As another main contribution, the authors propose shuffling countermeasures randomizing the order of operations. They propose to randomize computations of butterfly nodes or operations inside a butterfly. Thereby, the values an attacker measures by, e.g., using a template do not match the true variable processed in the butterfly that the adversary expects to be computed at a given time. Again, these countermeasures come in several variants that offer different levels of protection and performance degradation.

The authors then provide performance evaluations for their proposed countermeasures. They evaluate a variety of settings of their countermeasures integrated into pqm4 [KPR+]. Depending on the setting and level of protection, they report a performance overhead of about 7%-78% for Kyber and 12-490% for Dilithium. Evaluations of the level of protection and information on whether previous attacks may be adapted to the proposed countermeasures are not provided; this question is partly answered by this thesis in Section 4.5.

### Masking Countermeasures

As described in Section 2.2.3, an unmasked butterfly implementation with twiddle factor $\omega$ commonly computes either the function

$$(a, b) \mapsto (a + \omega b, a - \omega b) \tag{3.22}$$

or

$$(a, b) \mapsto (a + b, \omega(a - b)). \tag{3.23}$$

Twiddle factors are a constant of an implementation and accessible to an attacker. In an unprotected implementation, the twiddle factors belonging to a targeted butterfly can be assumed to be known to the attacker. This knowledge is modeled in the belief propagation by either the multiplication nodes of [PPM17] or the butterfly nodes of [PP19]. Ravi et al. [RPBC20] propose masking with an additional twiddle factor. This means, at a butterfly with twiddle factor $\omega_{\mathrm{bf}} = \zeta^x$ instead of computing Equation (3.23), the device chooses an $\omega_{\mathrm{mask}} = \zeta^y$ – with $y$ depending on the configuration of the countermeasure – and computes

$$\begin{aligned}(a, b) &\mapsto \omega_{\mathrm{mask}}(a + b, \omega_{\mathrm{bf}}(a - b)) && (3.24)\\ &= (\omega_{\mathrm{mask}}(a + b), \omega_{\mathrm{mask}}\omega_{\mathrm{bf}}(a - b)) && (3.25)\\ &= (\zeta^x(a + b), \zeta^{x+y}(a - b)). && (3.26)\end{aligned}$$

Based on this construction, several masked butterfly nodes are defined by setting $\omega_{\mathrm{mask}}$ in different manners. These butterflies either have the same input and same output masks, the same input but different output, or different input and different output masks. This then allows defining coarse and fine masking as well as generic masking as a trade-off between the two variants. *Coarse masking* uses the same input/same output butterflies and the same mask for all inputs, whereas *fine masking* makes use of different input/different output butterflies and uses a different mask for every input. By utilizing combinations of butterfly nodes including the same input/different output nodes, more configurations allowing for a trade-off between security and performance are possible.

### Shuffling Countermeasures

The shuffling countermeasures of [RPBC20] permute the order in which butterfly computations take place. This prevents an attacker from mapping measured values to the correct (intermediate) variables. Thereby, practically, the NTT graph in an attack such as [PPM17, PP19] would assign measurements to incorrect intermediates – the values *actually* measured do not match the variables the measurement is assigned to. The belief propagation after the template attack therefore works on incorrect priors – i.e., the priors do not match the variables, and the arithmetic model of the NTT is incorrect – preventing key recovery. These permutations in the order of execution are again defined in a configurable way, i.e., on different sets of to-be-permuted computations. The first and least protective countermeasure is *fine shuffling*, which permutes in- and outputs on a single butterfly level. The countermeasure called *coarse shuffling* permutes butterflies as a whole and comes in two variants – permuting butterflies in a block, which is called *coarse in-group shuffle*, or in a layer, called *coarse full shuffle*. The latter is the countermeasure offering the most protection but is also the most expensive in terms of execution time.

In an unprotected execution, a layer of the NTT or inverse NTT is computed in a fixed order; e.g., butterflies are computed by the order of the attached coefficients. That means that the first butterfly to be computed is connected to the first coefficient of the previous layer as well as to the coefficient having the layer distance $d$ as index. The second butterfly is connected to the second coefficient and the coefficient at $d + 1$. This is continued until reaching input $d - 1$, and the next $d$ inputs are skipped, because they are connected to the already computed butterflies. Inside the butterfly, the order of operations is fixed as well, commonly computing the output consisting of the addition of the two values first with loads and stores arising just before (directly after) a value is used (is stored). Coarse shuffling now randomly permutes the order of execution of butterflies, and fine shuffling randomizes the load and stores inside a butterfly.

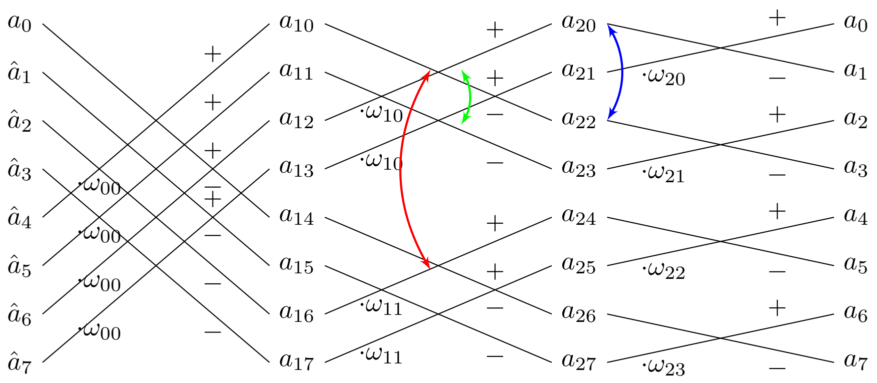


Figure 3.3: Different shuffling methods from [RPBC20] on an inverse NTT graph. Fine shuffling (rightmost arrow in blue) permutes load and stores, coarse in-group shuffling (middle arrow in green) permutes on butterflies with the same twiddle factor, and coarse full-shuffling permutes on all butterflies of a layer. Note that this depiction is only illustrative; it does not show all possible permutations, only permutes stores, and only depicts permutations of the second layer.

**Fine shuffling.** The countermeasure of fine shuffling permutes the load and store operations inside the execution of a butterfly. This means the order in which an input is loaded into a register is randomized; after both inputs are loaded, the butterfly operations are computed, and then finally stored again in a randomized manner. The randomized load operations are achieved using an arithmetic swap operation previously used in [HS13]. This operation has previously been a target of side-channel attacks itself [NCOS16, NC17]. Therefore, a variation of fine shuffling, *bitwise fine shuffling*, is proposed; this variation is not explained here and is not considered in Section 4.5. When applying a template attack targeting load and/or stores, as in [PP19], measurements will be assigned to incorrect intermediates with probability $\frac{1}{2}$ per pair of in- and outputs. During the belief propagation, the priors then do not match the variables they are assigned to, and the modeling of the arithmetic relationship is therefore incorrect and bound to fail. This is exemplarily shown in Figure 3.4. For an (inverse) NTT with $l$ layers and $h$ inputs

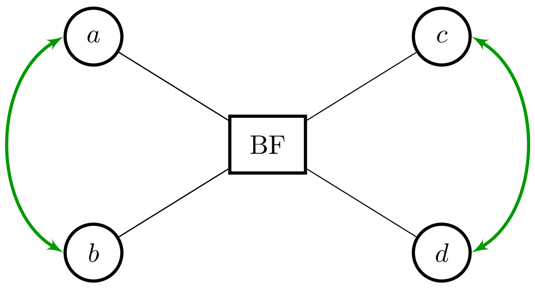


Figure 3.4: Permuted in- and outputs in a butterfly belief propagation node after a template attack on an implementation using fine-shuffling. In a template attack as in [PP19] or [HHP+21] on load and stores with subsequent belief propagation, measurements are randomly permuted as a consequence of the countermeasure.

per layer, fine shuffling has $2^{h/2}l$ possible permutations. Combining the attack of [PP19] with a brute-force approach to adapt to the countermeasure would therefore result in having to run $2^{128} \cdot 7$ belief propagation instances. The authors of [RPBC20] note that a template attack is still feasible if an attacker targets the modular multiplication as for example in [PPM17], but requires a largely increased amount of traces. Also note that the results presented for the attack of [PPM17] made use of additional leakage.

**Coarse in-group shuffling.** In contrast to fine shuffling, coarse in-group shuffling does not permute load and store operations of a single butterfly but instead permutes butterflies as a single unit of operation. This means the order in which butterflies are computed in a single layer is permuted; for in-group shuffling, the permutation is restricted to a single block, i.e., the set of butterflies sharing their twiddle factors. In-group shuffling is shown in Figure 3.5.

In a common NTT a block is given by $\{k2^{l+1}, k2^{l+1}+1, \ldots, k2^{l+2}, k2^{l+2}+1, \ldots, k2^{l+2}-1\}$ where $l$ is the current layer, and $k$ is the index of the block. For an (inverse) NTT the number of permutations per layer with $n$ inputs and $m$ butterfly blocks is $(\frac{n}{2m}!)^m$; the total number of permutations is the product over the complexities of a layer. For Kyber, where the number of groups in the $i$-th layer is $n/2^i$, this results in $(2^{i-1}/m)^m$ different possible permutations. In the case of the layer where a block consists of a single butterfly, the authors suggest using coarse full shuffling for this layer; note that this layer does not exist in a Kyber inverse NTT (c.f. Section 2.2.3).

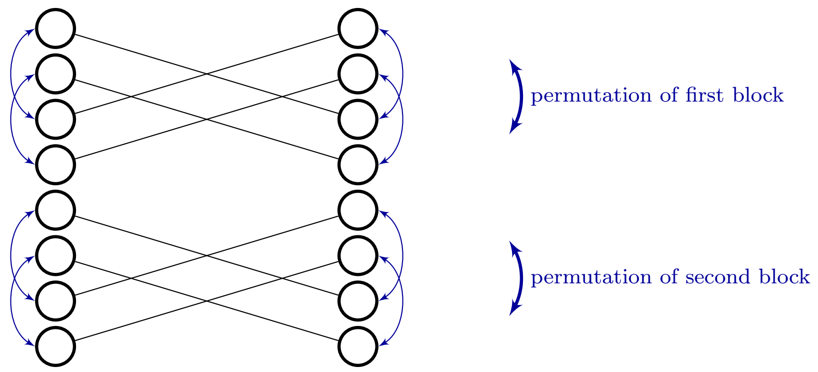


Figure 3.5: Possible permutations of butterflies after a template attack on an implementation using coarse in-group shuffling in a layer with distance 2 and 8 nodes. The permutation on the outputs has to be symmetric to the permutation on the inputs as complete butterflies are permuted.

**Coarse full shuffling.** Coarse full shuffling is similar to coarse in-group shuffling. In contrast to in-group shuffling, as the name suggests, the permutation is not limited to a block but instead all butterflies of a layer may be permuted. The concept of full shuffling is shown in Figure 3.5. This results in a higher brute-force complexity for an attacker, but additional twiddle factors need to be loaded (in a secured way!) per butterfly worsening performance; in addition, loading twiddle factors may leak information which would result in an attacker reaching the brute-force complexity of coarse in-group shuffling again. For an (inverse) NTT with $l$ layers and $h$ inputs per layer, coarse full shuffling results in $(h!)l$ possible permutations.

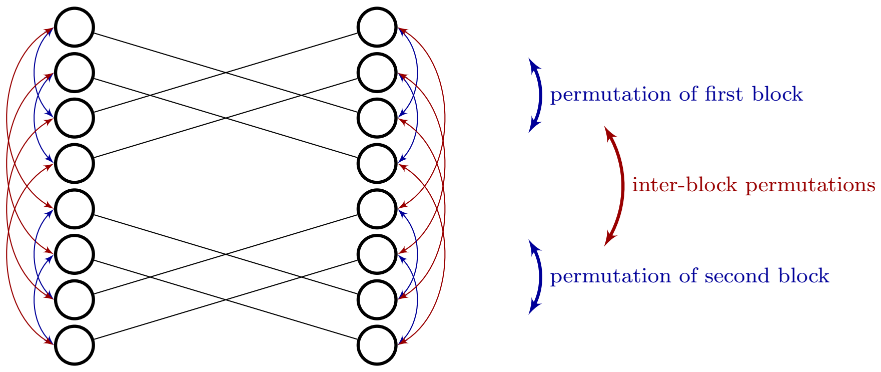


Figure 3.6: Possible permutations of butterflies after a template attack on an implementation using coarse full shuffling in a layer with distance 2 and 8 nodes. The permutation on the outputs has to be symmetric to the permutation on the inputs as complete butterflies are permuted. In addition to the in-block permutations, nodes in different blocks may also be permuted.

### Performance Results

The authors implemented their countermeasures using the Kyber and Dilithium (at National Institute of Standards and Technology (NIST) level 3, i.e., for Kyber768 and Dilithium3) implementations provided by PQM4 [KPR+] and evaluate their countermeasures in comparison to an unprotected implementation. Both masking and shuffling countermeasures are reported to not increase dynamic memory consumption. In terms of performance, the computational overhead depends on the masking or shuffling method employed. The overhead ranges from 6.9% to 77.6% for masking and from 30.2% to 69.5% for shuffling countermeasures. Table 3.1 permutations the results for shuffling countermeasures as reported in [RPBC20].

Table 3.1: Cycle count (in millions) and performance overhead (in brackets, in percent) of shuffling countermeasures as reported in [RPBC20]. Note that [RPBC20] provides data on more masking variants and uses a slightly different naming scheme.

| Countermeasure | Key Generation | Encapsulation | Decapsulation |
|---|---|---|---|
| Unprotected | 1.178 (0) | 1.301 (0) | 1.358 (0) |
| Coarse Masked | 1.259 (6.9) | 1.395 (7.2) | 1.466 (7.9) |
| Fine Masked | 1.979 (68) | 2.229 (71.3) | 2.413 (77.6) |
| Coarse Full Shuffled | 1.534 (30.2) | 1.72 (32.2) | 1.841 (35.4) |
| Coarse In-Group Shuffled | 1.49 (26.5) | 1.664 (27.9) | 1.772 (30.4) |
| Fine Shuffled | 1.468 (24.7) | 1.643 (26.3) | 1.752 (28.9) |
| Bitwise-Fine-Shuffled | 1.878 (59.4) | 2.123 (63.2) | 2.303 (69.5) |

### 3.1.6 Limitations of Prior Attacks

The attack of [PP19] does not target the secret key `sk` (and therefore in many use cases not the long-term secret) and can only recover the message `m` (in many use cases the session key). Taking into account that an application may often use ephemeral session keys or combine session keys with previous keys, this is a heavy restriction. Both attacks, [PPM17] and [PP19], do not take shuffling countermeasures into account. Therefore, an implementation using a shuffled (inverse) NTT is adequately protected against these attacks unless further adaptations are taken by the adversary (c.f. Section 3.1.7). In addition, the noise tolerance of [PPM17] is very low, with a standard deviation of less than 0.7 compared to 1.7 in [PP19], and the noise tolerance of [PP19] is substantially reduced by masking the NTT (see [PP19, Fig. 4]). Depending on the achievable noise tolerance, this may already prevent attacks in practice, especially considering that these template attacks assume access to an almost perfect clone of the device. The attack of Xu et al. [XPR+22] allows for a more portable approach that does not require a profiling device. While this is a major advantage, the attack is prevented by masking countermeasures, and implementations, which are not protected by masking are well-known to be vulnerable.

Summarizing, the noise tolerance in masked implementations is comparably low[3] for both attacks, shuffling countermeasures are not taken into account, and [PP19] does not target the secret key but only the message.

### 3.1.7 Defeating Countermeasures

It is yet unclear how an attacker can evade the proposed countermeasures in a Soft Analytical Side-Channel Attack (SASCA) on a post-quantum scheme using an NTT, but a wide variety of adaptions to countermeasures has previously been proposed in different contexts: In [CCD00] Clavier, Coron, and Dabbous present the techniques Hamming integration and sliding window differential power analysis, which are subsequently used and studied by several attacks on AES.

[3]Compared to an attack on the message `m` in an unprotected setting.

Rivain et al. [RPD09] use both techniques to attack an AES implementation protected by a shuffling countermeasure. Tilich et al. [THM07] employ the windowing approach against a masked and randomized AES implementation, and Tilich and Tilich and Herbst [TH08] study the techniques in practice on a smartcard. Veyrat-Charvillon et al. [VMKS12] provide extensive analysis of shuffling countermeasures and how do adapt in a variety of scenarios using Bayesian updating – this can be seen as similar to adaptations to belief propagation; Azouaoui et al. [ABG+22] further improved upon their work. Bronchain and Standaert [BS20] dissect an AES implementation in practice while proposing several techniques to adapt attacks to countermeasures; this attack is also based on Bayesian updating similar to the techniques used in belief propagation. Guo et al. [GGSB20] provide information theoretic analysis for SASCA that allows for bounds on the amount of leaked information that is obtained in an attack. A theoretical analysis of shuffling countermeasures in the context of Pseudo-Random Functions is provided by Grosso et al. in [GPSG14], and Bruneau et al. [BGNT15, BGNT18] counter local shuffling. Udvarhelyi et al. [UBS21] counter shuffling in an implementation of lightweight cryptography on low-end platforms; they obtain leakage of the shuffling permutation and employ an enumeration technique.

Several adaptations to countermeasures are known, and some occur in settings similar to belief propagation. Nevertheless, it is unclear how to defeat countermeasures such as those presented by Ravi et al. in [RPBC20] in a lattice-based scheme using an (inverse) NTT targeted by an attack using belief propagation. In particular, the impact of shuffling countermeasures in these settings without considering the leakage of permutations is yet unanswered.

## 3.2 Attacks using Decryption Failures

In many LWE-based schemes, decryption failures may occur, i.e., the decryption operation may fail. This is because, as described in Section 2.2.3, the message is recovered from a noisy version of the message polynomial. Every term of the message polynomial, each representing one bit, contains a (with very high probability) small additive noise term – those noise terms are called the *error polynomial* in Kyber. If this noise is too large in any coefficient, a bit is decoded incorrectly – i.e., a 0 bit is decoded to a 1 or vice-versa. In this case, the decryption fails, and an observer may deduce information about the noise term. The noise term itself depends on the secret key and thus leaks information about it. In LWE-based schemes in the context of non-implementation attacks, this was first noted in [Flu16] and used for example in [BBLP18, GJY19, BGRR19]. In the context of NTRU, this principle has previously been analyzed [HNP+03]. Guo, Johansson, and Nilsson [GJN20] first applied this principle in an implementation attack to a lattice-based scheme using an FO-transform by exploiting timing leakage in the comparison operation that allows detecting decryption failures. Ravi et al. [RRCB20] give yet another example of how chosen ciphertexts may exploit the FO-transform but use a different approach with ciphertext that is not close to a valid ciphertext. A generic side-channel methodology targeting the FO-transform of several post-quantum schemes through the creation of a plaintext checking oracle, a deep learning approach, and analysis on the applicability of attacks similar to [GJN20], [BDH+21b], and [PP21] is provided by Ueno et al. [UXT+22].

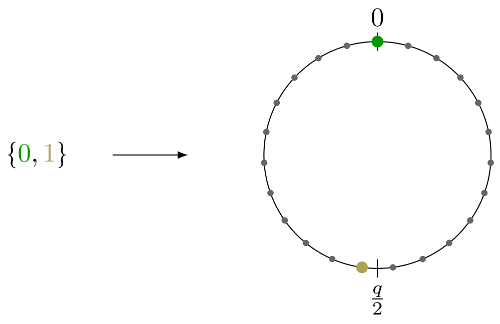
In the following, we reiterate the attacks of [BDH+21b, PP21, DHP+22, Del22] on Kyber. Note that another attack exists as a master's thesis [Wei22] but is unfortunately unpublished; the author uses machine learning to differentiate between decryption failures and successes. The key recovery methods utilized in the attacks presented in this section are described in Section 3.3 because they are relevant in a broader context.

### 3.2.1 Decryption Failures in Kyber

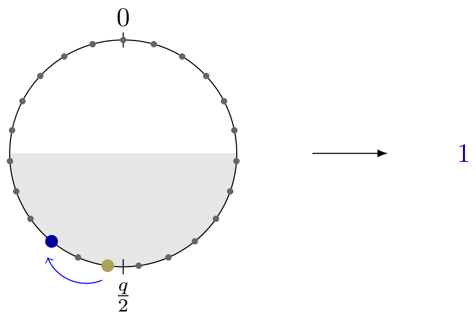
All attacks presented in this section as well as in Chapter 5 depend on the same principle of potentially causing and observing decryption failures and thereby obtaining information. In the context of LWE-based schemes, this was first noted in [Flu16] and first used in a side-channel attack in the context of an FO-transform [GJN20]. In the following, we explain this principle in the context of Kyber as used in [BDH+21b, PP21, HPP21, DHP+22, Del22].

#### Message Bits and Polynomial Coefficients

Recall from Section 2.2.3 that in Kyber bits of a message $\mathtt{m}$ are mapped to polynomial coefficients by mapping a 0 bit to the 0 coefficient and a 1 bit to $\left\lceil \frac{q}{2} \right\rfloor$ during the encryption, which is visualized in Figure 3.7a. During the decryption process, the message bits are recovered from a noisy version of those coefficients. This works by mapping a coefficient $x \in \mathbb{F}_q$ to 0 if and only if $x$ as an is closer to 0 than to $\frac{q}{2}$ and to 1 otherwise; where $x$ is considered to be an integer in $\left\{-\frac{q-1}{2}, \dots, \frac{q-1}{2}\right\}$. When interpreting $\mathbb{F}_q$ as discrete points on a circle, as visualized in Figure 3.7b, this can be seen as mapping the upper half of the circle to 0 and the lower half to 1.



(a) Mapping bits to message coefficients.



(b) Mapping noisy coefficients to message bits.

Figure 3.7: Mapping bits to coefficients and vice-versa with $q = 23$ (adapted from [Her23a]). 0 bits are mapped to 0, and 1 bits are mapped to $\frac{q}{2}$. To recover a bit, the upper half of the circle (closer to 0) is mapped to a zero bit while the lower part of the circle (closer to $\frac{q}{2}$) is mapped to a 1 bit. A small error (blue) does not change the result (c.f. Figure 2.8 and Figure 2.9).

### Error Polynomial

The error polynomial in Kyber is given by

$$\mathbf{e}^\top \mathbf{r} - \mathbf{s}^\top (\mathbf{e_1} + \Delta \mathbf{u}) + e_2 + \Delta v \tag{3.27}$$

with the notation of Section 2.2.3; note that all terms are small, and therefore the noise term can be interpreted as a vector of small integers. If an attacker can observe a (naturally occurring) decryption failure, they may deduce that the absolute value of the noise term in at least one coefficient must have been large. Without manipulation, decryption failures happen with very low frequency as the noise term is small with very high probability, for example in Kyber, the probability for a decryption failure is less than $2^{-139}$ [ABD+21b].

### Causing Decryption Failures

If an attacker can cause an additional noise term in one coefficient occurring before or during the error-correction phase, they may cause decryption failures depending on the noise term which, in turn, depends on the secret key. In a decryption without manipulation, the noisy message coefficient is distributed around either 0 or $\lceil \frac{q}{2} \rfloor$, and the distribution is symmetric and has a small variance. If an adversary adds or subtracts $\lceil \frac{q}{4} \rfloor$ – in the visualization of Figure 3.8b this is a quarter rotation – the noisy message coefficient is distributed exactly around the boundary of where a decryption failure occurs – either $\lceil \frac{3q}{4} \rfloor$ or $\lceil \frac{q}{4} \rfloor$, respectively. This causes decryption failures to happen with probability (almost) $\frac{1}{2}$; whether a decryption failure is caused depends on the sign of the noise term. If the attacker can observe such decryption failures they therefore learn an inequality of the form

$$(-1)^{\mathrm{obs}} (\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta \mathbf{u})^\top + e_2 + \Delta v)_0 \leqslant 0 \tag{3.28}$$

where obs is 1 if a decryption failure occurred and 0 otherwise[4], and the 0-th coefficient was being targeted[5]. Figure 3.8 visualizes the occurrence of decryption failures depending on the noise term: The situation where no decryption failure is caused is shown in Figure 3.8a, and the situation with a positive noise term is depicted in Figure 3.8b.

Clearly, the obtained inequality is linear in the secrets $\mathbf{e}$ and $\mathbf{s}$ and may be written as an inequality over the integers. This means that after $m$ such observations, the attacker obtains a linear system of inequalities

$$\begin{pmatrix} \mathbf{e} & \mathbf{s} \end{pmatrix} \mathbf{M} \leqslant \mathbf{b} \tag{3.29}$$

where $\mathbf{e}$ and $\mathbf{s}$ are flattened to integer vectors with $\mathbf{M} \in \mathbb{Z}^{m \times 2kn}$ and $\mathbf{b} \in \mathbb{Z}^m$.

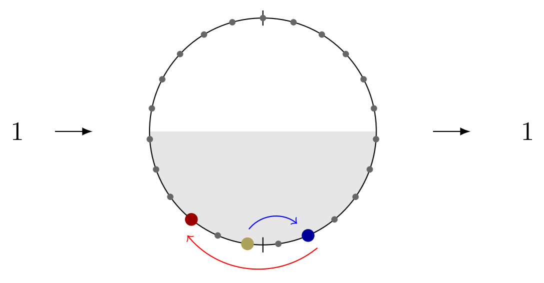
**The impact of compression.** Note that in a scheme without compression, the attacker may obtain equations instead of inequalities as described in [GJN20]: By using different introduced errors, i.e., terms other than $\lceil \frac{q}{4} \rfloor$, they may determine the minimum value causing an error. This is equivalent to finding a value $E$ two inequalities,

$$(\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta \mathbf{u})^\top + e_2 + \Delta v)_0 + E \leqslant 0 \tag{3.30}$$

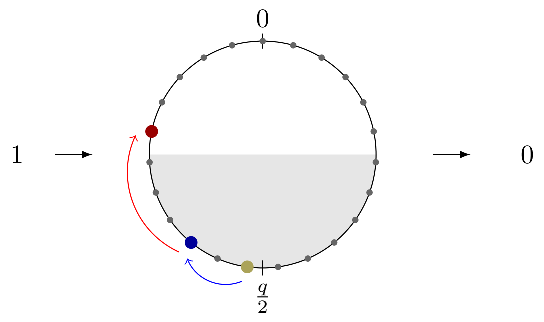
and

$$(\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta \mathbf{u})^\top + e_2 + \Delta v)_0 + E + 1 > 0, \tag{3.31}$$

[4] In the case of a decryption failure, the inequality is strict.
[5] Note that different coefficients may be targeted; 0 is then replaced by the corresponding index.



(a) If the noise term is negative, no decryption failure is caused.



(b) If the noise term is positive, a decryption failure is caused.

Figure 3.8: A maliciously introduced error may cause a decryption failure depending on the noise term (adapted from [Her23a]).

where the 0-th coefficient was targeted. As the compression used on the relevant ciphertext component in Kyber rounds a coefficient, including the introduced error, to the nearest multiple of $\lceil \frac{q}{2^{d_v}} \rfloor = \lceil \frac{q}{2^4} \rfloor$, this is implicitly prevented. Instead, an attacker may obtain more refined inequalities using the same technique as demonstrated in [BDH+21b].

### Impact of the FO-Transform

The additional error that potentially causes a decryption failure may be introduced by a chosen ciphertext: As the decryption process is linear up until the error correction if an attacker adds a $\lceil \frac{q}{4} \rfloor$ term to a ciphertext coefficient $v$, this error propagates into the recovery method; this causes the situation described above. To be precise, the attacker first honestly generates a valid ciphertext $(\mathbf{u}, v)$, adds $\lceil \frac{q}{4} \rfloor$ to coefficient $l$ of the ciphertext component $v_l$. This causes a decryption failure depending on the noise term as described above. If the Public-Key Encryption (PKE) variant of Kyber, KyberPKE, were to be deployed in a non-ephemeral scenario, the following attack would recover the key:

1. Honestly[6] generate a valid ciphertext $\texttt{ct}$ for a random message $\texttt{m}$.
2. Manipulate the first coefficient of $\texttt{ct}$ by adding a $\lceil \frac{q}{4} \rfloor$ term obtaining $\tilde{\texttt{ct}}$.
3. Send the ciphertext $\tilde{\texttt{ct}}$ to the device or algorithm under attack.
4. Observe whether a decryption failure happens, and derive an inequality as described above.

[6]This means: By calling the encapsulation routine without any manipulation.

5. Repeat until a sufficient number of inequalities have been derived.
6. Solve inequalities using the method described in this thesis in Chapter 6[7].

As already described in Section 2.2.3, Kyber uses an FO-transform similar to the one presented in [TU16] to turn the IND-CPA secure KyberPKE into the IND-CCA2 secure KyberKEM, which is commonly called Kyber. Clearly, the FO-transform prevents a pure chosen-ciphertext attack.

In the presence of an FO-transform, the decryption failure is still occurring, as the decapsulation routine calls decrypt on the ciphertext. But regardless of whether a decryption failure occurs, the decapsulation will fail: The comparison of the submitted ciphertext to the re-computed ciphertext yields invalid – if no decryption failure happens, the decryption routine computes the valid ciphertext `ct` and compares against $\tilde{\texttt{ct}}$.

If an implementation attack allows an attacker to observe whether a decryption failure occurred, i.e., replaces step 4., the attack is possible again. This principle, in combination with an implementation attack, is used by, e.g., [GJN20, BDH+21b, DHP+22, HPP21].

## 3.2.2 The Attack of Pessl and Prokop

Pessl and Prokop [PP21] use a fault attack on the decoding routine in New Hope and Kyber. They thereby prevent an addition of $\lceil \frac{q}{4} \rfloor$ to one coefficient. This leads to the leakage of information through inequalities as described in the previous section.

### Attack Target

The attack of Pessl and Prokop [PP19] targets the decoding routine Compress$(\cdot, 1)$. The decoding routine is the function described above mapping a noisy polynomial coefficient back to the original message bit `m` – a coefficient is mapped to a 1 bit if and only if it is in $\left\{\lceil \frac{q}{4} \rfloor, \ldots, \lceil \frac{3q}{4} \rfloor - 1\right\}$. In Kyber, this is described by re-using the compression function with compression parameter $d = 1$. A coefficient $x$ is mapped to

$$\text{Compress}(x, 1) = \left\lceil \frac{2x}{q} \right\rfloor \bmod 2. \tag{3.32}$$

This is directly equivalent to the above description as an easy computation shows.

In the reference implementation (associated with the submission to the NIST contest [NistCfp]), this is implemented in the function `poly_tomsg` which computes

```
(((x << 1) + q/2) / q) & 1;
```

for each coefficient $x$. The bitshift to the left, `<<`, is mathematically equivalent to multiplication with 2. Adding $\lfloor \frac{q}{2} \rfloor$ causes the coefficient bit to be as if it was encoded to $\lfloor \frac{q}{2} \rfloor$ (coming from a 0 bit) or $\lfloor \frac{3q}{2} \rfloor$ (coming from a 1 bit). Dividing by $q$ then thereby causes the least significant bit to be 0 in the first and 1 in the second case. The `& 1` retrieves the least significant bit – which is 0 if a 0 bit was encoded and 1 if a 1 bit was encoded[8].

The authors now target the addition of $\lfloor \frac{q}{2} \rfloor$ and prevent the addition of the term. Then, instead of being close to $\lfloor \frac{q}{2} \rfloor$ (coming from a 0 bit) or $\lfloor \frac{3q}{2} \rfloor$ (coming from a 1 bit), the coefficient is close to either 0 or $\lfloor \frac{q}{4} \rfloor$. This is equivalent to subtracting $\lfloor \frac{q}{4} \rfloor$ from $x$ before calling the function ($x$ is multiplied by 2). This causes the effect described in the previous section: If the decryption noise is negative, a decryption failure happens, otherwise, the decryption is successful.

[7]Note that solving these inequalities is not a trivial task.

[8]With very high probability.

### Attacker Model

The attacker model assumes the attacker to hold the public key `pk` and allows performing arbitrary encapsulations resulting in a ciphertext `ct` and a shared secret `K`. They may then submit the resulting (honestly generated and valid) ciphertexts to the device under attack. In the decapsulation routine, a fault may be injected in the decoding function (`poly_tomsg`) of the decryption routine as described in the previous section. The attacker can observe the result of the decapsulation by observing the output $\mathtt{K}'$ of the decapsulation and comparing it against `K`.

### Countermeasures

The authors of [PP21] take an already published countermeasure into account and additionally state countermeasures potentially preventing the attack.

**The masking countermeasure of Oder et al.** Pessl and Prokop [PP21] take the countermeasures of a masked decoder presented in [OSPG18] into account. They conclude that the attack is neither prevented nor requires more faults. This may not be the case for all masking implementations, but the masking of [OSPG18] does not protect against their attacks. Note that work of [OSPG18] additionally allows for another attack presented in the next section.

**Proposed countermeasures.** The authors also propose several countermeasures: Depending on the use case a trusted third party may only sign ciphertexts or hold the public key. Thereby, the attack is prevented as the possibility of generating or submitting ciphertexts is taken from adversaries. Another generally applicable countermeasure against fault attacks is the introduction of redundancy. The authors propose using double computations and note that using error-correcting codes on the message `m` likely prevents their attack in its current form. The usage of shuffling likely prevents their attack as the faulted coefficients cannot be determined in a shuffled implementation. Finally, control flow integrity prevents their attack as the control flow is altered by a skipping fault.

### Ciphertext Filtering

The information contained in an inequality of the form of Equation (3.28) strongly depends on the size of $\Delta v$ and $e_2$. Both values are under the control of the attacker (c.f. Algorithm 11). Therefore, a straightforward but effective way to increase the obtained information is to filter ciphertexts sent to the device if they surpass a certain threshold in $\Delta v$ and $e_2$. Note that $\Delta v$ is potentially large [9] while $|e_2| \leqslant 3$ (see Section 2.2.3 and Table 2.1). The authors use the filtering condition

$$|\Delta v + e_2| \leqslant 10. \tag{3.33}$$

Note that our work provides an analysis of ciphertext filtering in Section 6.2.3.

### Inefficient Faults

In case of an unreliable fault, the attack may derive an incorrect inequality. This significantly hinders the recovery process in general (see Section 7.2.3 for a precise evaluation), and with the recovery method of this work prevents the attack with just $> 1\%$ of incorrect faults. Therefore, the authors propose using only failing decapsulations as this rules out an inefficient (but not an incorrect) fault. This requires about twice as many fault injections for a perfect fault. For a fault

[9]For example, $\Delta v \in \{-104, \ldots, 104\}$ for Kyber512 and Kyber768.

with $f$ of all tries being effective, $\frac{1-f}{2}$ of all faults lead to a decapsulation failure, and therefore factor $\frac{2}{1-f}$ as many faults are required.

#### Results

To evaluate their recovery method, the authors use a simulation that relies on the Kyber reference implementation associated to submission specified in [ABD+19b]. Their recovery method is described in detail in Section 3.3.3 where we also state the required number of inequalities. In addition, they perform their attack using a ChipWhisperer [OC14] setup. Their experiments confirm the feasibility but show a low success rate with their setup leading to a high number of applied faults.

### 3.2.3 The Attacks of Bhasin et al. and D'Anvers et al.

Bhasin et al. [BDH+21b] and D'Anvers et al. [DHP+22] present secured comparisons for the FO-transform in Kyber as well as attacks on previous, flawed comparison operations in [OSPG18] and [BPO+20] (for [OSPG18] the fix is similar to [BDK+21b]). The attacks of [BDH+21b] are part of a larger class of attacks: If an attacker can use a Side Channel Attack (SCA) to differentiate whether a decryption failure happened, they may use a chosen ciphertext as described in Section 3.2.1 to potentially cause a decryption failure and observe it using the side-channel. This principle has been exploited on the example of timing leakage in the FO-transform of FrodoKEM in [GJN20].

In addition to their countermeasures and attacks, the authors of [BDH+21b] provide a test framework to find such leakage; practical evaluation of their attacks is included as well. We give a brief overview of their attacks and refer to their works for details; the relevant principle for this thesis is already described in Section 3.2.1. The attack of [DHP+22] uses a higher-order attack on [OSPG18] and the fixed versions in [BDH+21b, BDK+21b], and the authors describe how to protect the comparison against such attacks.

#### Attack Targets

The authors of [BDH+21b] target the masked FO-transform comparisons presented in [OSPG18] and [BPO+20] (c.f. Section 3.2.2). The implementation of [OSPG18] leaks through a side-channel while [BPO+20] leaks through a side-channel and is, in addition, vulnerable to a pure chosen-ciphertext attack – without exploiting any implementation vulnerabilities. The targeted comparison operation is part of the FO-transform and checks whether the re-encrypted ciphertext is equal to the originally submitted ciphertext. The authors of [BDH+21b] exploit the fact that both masked implementations unmask partial comparisons and thereby leak information. In addition, they show that the method of [BPO+20] in some cases accepts manipulated ciphertexts and that this allows launching a collision attack.

#### Results

The authors of [BDH+21b] evaluate their attack experimentally on the ARM-Cortex M4 implementation PQM4 [KPR+] with integrated implementation of both comparison operations. They thereby show the feasibility of their attack but do not perform a full end-to-end key recovery. The authors of [DHP+22] perform a full key recovery utilizing the method presented in [HPP21] and in this thesis.

### 3.2.4 The Attack of Delvaux

The attack of Delvaux [Del22] is a follow-up work to the attack strategy presented in this thesis and in [HPP21]. We only very briefly describe the difference regarding the attack target and refer to Chapter 5 as preliminary for this section.

The attack of [Del22] uses a similar principle to [HPP21]: A chosen ciphertext carrying an additional $\lceil \frac{q}{4} \rfloor$ term is submitted to the device under attack, and a subsequent fault removes this in the temporarily stored ciphertext which is used for the comparison operation. In contrast to our attacks, they fault a wider variety of locations and operations circumventing various countermeasures. In addition, they carry out the attack on a physical device confirming that the class of attacks is feasible in practice.

### 3.2.5 Fault Attacks with Different Targets

Using a fault, multiple works suggested countermeasures against or variants of attacks skipping the re-encryption check of the FO-transform as well as multiple other skipping faults that allow for key recovery [VOGR18, OSPG18, BGRR19, XIU+21]. As the FO-transform is the chosen measure to avoid chosen-ciphertext attacks, disabling it again allows for chosen-ciphertext attacks. In turn, the FO-transform comparison step is usually protected mitigating the impact of such attacks. Ravi et al. [RRB+19] target key generation and encapsulation by using a fault to prevent using a domain separator during the expansion of a secret. Valencia et al. [VOGR18] attack plain RLWE schemes in numerous ways; their methods do not target IND-CCA2 secure schemes and do not apply to, e.g., Kyber. Thus, these fault attacks either do not evade known countermeasures or do not target the error correction in Kyber and are therefore out-of-scope in regard to our research questions.

### 3.2.6 Limitations of Priors Attacks

The attack of Pessl and Prokop [PP21] requires that a specific point of attack is insufficiently protected (standard countermeasures prevent the attack). In addition, it targets a specific implementation and requires a reliable fault; while ineffective faults do not prevent the attack (only lead to more faults being required), incorrect faults, i.e., faults faulting the wrong operation. Therefore, shuffling countermeasures thwart the attack even with a low level of incorrect faults/entropy.

The attacks of [BDH+21b] and [DHP+22] have been carried out in practice on a physical device and represent a class of attacks that likely applies in a wide range of scenarios. But those attacks require side-channel leakage and the authors already explain how to prevent – or at least reduce – such leakage in the analyzed comparison operations. The attack of Delvaux [Del22] is a follow-up work to [HPP21], which is part of this thesis and their work makes use of our attack strategy.

## 3.3 Key Recovery Methods

The attacks in Section 3.2 and Chapter 5 show the relevance of decryption failures in LWE-based cryptography. Whenever an implementation attack allows observing whether a decryption failure happens – be it using a side-channel or a fault turning the FO-transform into a decryption failure oracle – an adversary obtains information about the secret key. As we saw in Section 3.2.1, the leaked information is obtained in the form of inequalities. Obtaining the secret key from this kind of information leakage is not a trivial task, and several recovery methods have been proposed: Pessl and Prokop [PP21] developed a method for their attack (c.f. Section 3.2.2). Delvaux [Del22] improved upon the method of [PP21] and the method proposed as part of this thesis in [HPP21]. The attack of [BDH+21b] utilizes the more general framework of Dachman-Soled et al. [DDGR20] for estimates but does not perform a full key recovery. The attack of [DHP+22] uses the method presented in [HPP21] as part of this thesis. In addition, the failure boosting attack of Fahr et al. [FKK+22] on FrodoKEM [BCD+16, ABD+21a] retrieves and solves inequalities; those inequalities differ from the previously described inequalities as the coefficients are strongly correlated to the secret key coefficients. For this attack, another key recovery method has been developed by Dachman-Soled et al. [DGHK22] and published as an extension of the framework of [DDGR20].

### 3.3.1 Attacker Model

Recall from Section 3.2.1 that the attacks described above retrieve inequalities of the form

$$(\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta\mathbf{u})^\top + e_2 + \Delta v)_l < b \tag{3.34}$$

where $l$ is the index of the erroneous coefficient. The coefficients of all coefficients are small enough for no reduction modulo $q$ to happen, and this is clearly a linear equation in $(\mathbf{e}, \mathbf{s})$. Therefore, the inequalities may be written using $\mathbf{M} \in \mathbb{Z}^{2kn \times m}$ as

$$\mathbf{x}\mathbf{M} \leqslant \mathbf{b} \tag{3.35}$$

where $\mathbf{x} \in \mathbb{Z}^{2kn}$ is the flattened key $(\mathbf{e}, \mathbf{s})$ over $\mathbb{Z}$, and $m$ is the number of inequalities. The coefficients of $\mathbf{x}$ are the key coefficients sampled as specified by the Kyber algorithm description [ABD+21b]. Additionally, the adversary is given access to the public key equation

$$\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b} \in R^k. \tag{3.36}$$

From these equations and inequalities, the adversary is asked to find the secret key $\mathbf{x}$. Note that finding half of the coefficients of $\mathbf{x}$ is sufficient, as the public key equation is an $n$-dimensional system of equations over $\mathbf{x}$ with $2n$ unknowns. Therefore, recovering $n$ coefficients of $\mathbf{x}$ gives the remaining $n$ coefficients by solving the system of equations.

### 3.3.2 The Frameworks of Dachman-Soled et al.

The framework of Dachman-Soled et al. [DDGR20] provides a general approach to side-channel information in LWE-based schemes. The authors define several types of *hints* which allow to reduce the security of the LWE instance. To be precise, the LWE instance directly gives a Bounded Distance Decoding (BDD) instance. This instance is then embedded into a Distorted Bounded Distance Decoding Problem (DBDD) instance. The DBDD is related to the BDD problem (see [DDGR20] for details).

Applying a hint, as proposed by the authors, gives another DBDD instance that is easier to solve. After all hints have been integrated, the final DBDD instance is embedded into a unique Shortest Vector Problem (uSVP) instance which can be used to recover the secret key using lattice reduction. The main difference to the standard primal attack, i.e., using Kannan's embedding from BDD to uSVP and then using lattice reduction to solve the uSVP problem, is the integration into DBDD allowing for hints integration.

Given an LWE secret $\mathbf{x} = (\mathbf{e}, \mathbf{s})$, the corresponding lattice $\mathcal{L}$, a known vector $\mathbf{v}$, a value $l$, a modulus $k$[10], and a variance $\sigma$, the authors define the following hints:

[10]This is not the Kyber parameter $k$.

- Perfect hints: $\mathbf{x}\mathbf{v}^\top = l$.
- Modular hints: $\mathbf{x}\mathbf{v}^\top \equiv l \mod k$.
- Approximate hints: $\mathbf{x}\mathbf{v}^\top = l + \mathcal{N}_\sigma(0)$.
- Short vector hints: $\mathbf{v} \in \mathcal{L}$ where $\mathbf{v}$ is small.

The authors explain how approximate hints may be used to model decryption failure information. Note that the decryption failures integrated in [DDGR20] are assumed to occur without manipulation, and therefore the results differ from the inequalities obtained in the attacks we consider; the information obtained in [FKK+22] may be solved with an extension the framework [DGHK22]. Therefore, we here describe the method of [BDH+21b] which relies on [DDGR20] to estimate the security after having obtained inequalities.

The authors of [BDH+21b] use an approach similar to [GJN20] to obtain their inequalities; they use the same honestly generated ciphertext with several varying introduced errors. Due to the Kyber compression, this does not allow them to obtain an equation. Instead, inequalities are obtained, which only give an approximate value for $\mathbf{v} - \mathbf{s}\mathbf{u}^\top$ in a range of length $2^{d_v+1} = 2^5$. Assuming a uniform distribution on the set $S = \left\{-2^{d_v}, \dots, 2^{d_v} - 1\right\}$, the authors of [BDH+21b] use an approximate error distribution with variance $\sigma$ to obtain the hint

$$\mathbf{x}\mathbf{v}^\top = l + \mathcal{N}_\sigma(0), \tag{3.37}$$

where $v$, $l$, and $\sigma$ can be computed from the inequalities.

These approximate values may then be used in the framework of [DDGR20]. The resulting lattice $\mathcal{L}$, after integrating an approximate hint, is given by iterating over all possible values $e \in \text{support}(\mathcal{N}_\sigma(0))$ and intersecting with all

$$I_e = \left\{\mathbf{x}' \in \mathcal{L} \mid \mathbf{x}'\mathbf{v}^\top = l + e\right\}. \tag{3.38}$$

The remaining DBDD parameters are adjusted accordingly.

### Extension to Decryption Failures

The failure boosting attack of Fahr et al. [FKK+22] on FrodoKEM obtains decryption failure information in the form of inequalities. They notice that the framework of [DDGR20] is computationally difficult to use in practice for an end-to-end key recovery in their case. Therefore, the authors of [FKK+22] present their own statistical recovery method. Subsequently, to improve upon the results of [FKK+22], Dachman-Soled et al. [DGHK22] extend the framework of Dachman-Soled et al. [DDGR20] with a method solving inequalities of the form as [FKK+22] obtains them. The decryption failures arising in [FKK+22] conceptually differ from the decryption failures arising in attacks such as [PP21, BDH+21b, BDH+21b] as the failure boosting technique causes inequality coefficients to be strongly correlated to secret key coefficients, which occurs similarly in decryption failures without manipulation.

### Limitations in this Use Case

Pessl and Prokop in [PP21] note that the framework of Dachman-Soled et al. [DDGR20] may be used to solve for the secret key; nevertheless they developed their own method. Later, the authors of [DHP+22] note that the framework seems to be challenging to use for this concrete use case due to computationally complexity; in addition, the estimates of [BDH+21b] yield higher numbers than required for the full key recovery in [HPP21]. This implies that the framework is suboptimal in the case of this kind of decryption failure (c.f. [DHP+22, Section 3.1]), which is not surprising given its generality, and the fact that it was not specifically made for decryption failures on Kyber. Fahr et al. [FKK+22] also develop their own method, but their inequalities differ due to a higher correlation coefficient. Dachman-Soled et al. in [DGHK22] subsequently improved upon the method of [FKK+22], but neither method is directly applicable to decryption failures arising in previous attacks such as [PP21, BDH+21b, BDH+21b] due to the different nature of the inequalities.

### 3.3.3 Recovery Method of Pessl and Prokop.

Pessl and Prokop [PP21] developed a first method to practically solve the arising inequalities. They noticed that more information than the algebraic information from the public key equation, i.e., the lattice problem, and the inequalities are available: For every coefficient, the attacker knows the exact distribution it was sampled from. In Kyber, these distributions are binomial with $\eta = 3$ for Kyber512 and $\eta = 2$ for Kyber768 and Kyber1024[11]. The distributions represent the initial probability for every coefficient; the inequalities allow for an updating of probability distributions. Thereby, using repeated updating, the vector of distributions converges against distributions giving the secret key by taking the likeliest value in every coefficient; that is if enough inequalities are available. The authors also report an unsuccessful attempt at using a linear programming solver.

#### Updating Process

The recovery method work iteratively updates the probability distributions for the unknown key coefficients of $\mathbf{x} = (\mathbf{e}, \mathbf{s}) \in \mathbb{Z}^{2n}$. We denote the vector of random variables representing $\mathbf{x}$ as

$$\mathbf{X}_s = (X_{s,0}, \ldots, X_{s,2n-1}) \tag{3.39}$$

and the corresponding distributions as

$$\mathbf{D}_s = (D_{s,0}, \ldots, D_{s,2n-1}) \tag{3.40}$$

where $i$ is the index of the coefficient, and $t$ is current step. For $t = 0$,

$$\mathbf{D} = \big(\text{Binom}(\eta), \ldots, \text{Binom}(\eta)\big) \tag{3.41}$$

The $i$-th inequality is denoted as

$$\sum_{j=0}^{2n-1} a_j x_j \leqslant b_i. \tag{3.42}$$

In each step $s + 1$, for every inequality and every index $i \in \{0, \ldots, 2n\}$, the *leave-one-out* distribution to the random variable

$$S_{t+1,j} = \sum_{j=0, i \neq j}^{2n-1} a_j X_{t,j} \tag{3.43}$$

is computed. This allows computing

$$P(X_{t+1,i} + \sum_{j=0, i \neq j} a_j X_{t+1,j} \leqslant b_i | X_{t,j} = x') \tag{3.44}$$

[11]Note that at the time of the attack of [PP21], the parameters slightly differed.

for all $x' \in \mathrm{support}(\mathrm{Binom}(\eta))$. The updated distributions,

$$P(X_{t+1,i} = x' \mid \sum_{j=0}^{2n-1} a_j X_j \leqslant b_k), \tag{3.45}$$

are then given by computing

$$\frac{P(\mathrm{Binom}(\eta) = x') \cdot P(X_{t,i} + \sum_{j=0,i\neq j} a_j X_{t,j} \leqslant b_i | X_{t,j} = x')}{\sum_{x''} P(\mathrm{Binom}(\eta) = x'') \cdot P(X_{t,i} + \sum_{j=0,i\neq j} a_j X_{t,j} \leqslant b_i | X_{t,j} = x'')}. \tag{3.46}$$

Thereby all $X_{i,s+1}$ for the $i$-th inequality, Equation (3.42), are computed. Repeating the routine for all inequalities gives updated probabilities for all coefficients. After each step, the likeliest coefficient is plugged into the public key equation – if the equation is fulfilled, the secret key has been found.

**Optimizations.**

Pessl and Prokop [PP21] use several optimizations for improved convergence as well as for performance.

**Clustering.** The first optimization improves the convergence rate: Instead of working with a probability distribution per key coefficient, they consider the joint probability distributions for a *cluster* of coefficients. The clusters are recomputed during the update iterations, and unlikely values are dropped (improving performance). Note that using one cluster with all key coefficients corresponds to an exact computation of the probability, which would result in optimal values but is computationally infeasible.

**Sums of Random Variables** For two random variables, $X$, $Y$ the distribution of $X + Y$ is given by the convolution of the distributions of $X$ and $Y$, i.e., in the discrete case

$$P(X + Y = a) = \sum_{x \in \mathrm{support}(X)} P(X = x) P(Y = a - x) \tag{3.47}$$

Thus, an FFT may be used to efficiently compute the distribution of $X + Y$. The authors use this fact to efficiently compute Equation (3.43).

In addition, they use a binary tree structure to avoid re-computations arising when computing all Equation (3.43); this structure is also used and described in detail in [HPP21]. First, all the Fourier-transforms of all $\mu_{i,j}$ are computed; we denote them $\hat{y}_i$ here for convenience. Then, the upward tree is constructed, and the first layer is initialized with the $\hat{y}_i$ for $i \in \{0, \ldots, 2n\}$. In the following layers, each node has two parents and contains the point-wise product of their parents. The last layer, of index $\log_2(n) - 1$, has only two nodes which are the products of half of the $\hat{y}_i$. Lastly, the downward tree is computed: The top layer of the downward tree is initialized with the two nodes of the upward tree's last layer but in swapped order. Every following layer is computed by multiplying the child node of the downward tree (which is already computed) with the sibling node in the upward tree. The first layer at index $i'$ now holds the products

$$\prod_{i \neq i'} \hat{y}_i, \tag{3.48}$$

which allows obtaining the leave-one-out distributions we aimed to compute.

Table 3.2: Approximate number of required inequalities required for a success rate (SR) greater than 0 and equal to 1 as reported by Pessl and Prokop [PP21]. Note that Kyber512 has since been updated and the required number of inequalities will be higher for the newest version.

| Variant | Inequalities SR $> 0$ | Inequalities SR $= 1$ |
|---|---|---|
| Kyber512 | 5500 | 7500 |
| Kyber768 | 8500 | 10500 |
| Kyber1024 | 11000 | 12500 |

### Results

We differentiate between results regarding performance and regarding the success rate per number of obtained inequalities.

**Performance.** While the recovery method of [PP21] is fairly efficient in terms of computation time, the amount of required RAM is rather large. They report a runtime well below an hour for all their tested scenarios featuring up to 15000 inequalities. The RAM consumption is at a minimum of 24 GB for Kyber512 (with 6300 inequalities) and at a maximum 61 GB for Kyber1024 (with 13000 inequalities).

**Number of inequalities.** The authors of [PP21] report numbers stated in Table 3.2 of required inequalities for a success rate greater than 0 and for a success rate of 1.

**Error Resistance** The method of [PP21] allows for a very small amount of incorrect inequalities but drops to a single-digit success rate when more than 1% of inequalities are incorrect.

## 3.3.4 Recovery Method of Delvaux.

The recovery method of Delvaux [Del22] improves upon the original method presented in [HPP21]. The author simplifies several computations by replacing them with approximations; this brings the runtime down from a few minutes on a multithreaded machine to a few seconds on a single thread. Unfortunately, the performance in terms of required inequalities is also increased by about 50% (see Section 7.2.3).

The main improvement in the method of [Del22] (note that their work additionally improves the fault attack itself) is introducing error resistance. This means that they are able to recover the key in the presence of up to 60% incorrect inequalities. The error resistance is achieved by including the probability of an inequality being incorrect in their Bayesian updating process which is, apart from this, an approximation to the updating process used in [HPP21].

### Recovery Method

The author of [Del22] proposes to use the central limit theorem to approximate the term computed in Equation (3.43). They note that a large number[12] of binomial distributions are summed up in early iterations. Such a sum converges against a normal distribution by the central limit theorem. Thus, the author approximates by a normal distribution, which reduces the computation time. Note that their approximation only holds in early iterations, but they conjecture that the distributions in later stages are close enough to point distributions for this to not be relevant.

[12]To be precise: $2kn$ binomial distributions.

Their practical experiments show that this is true for large numbers of inequalities, but the performance in terms of required inequalities to retrieve the key is worsened.

Apart from the performance improvements, the author integrated error resistance into their solving method. This is achieved by implementing a proposed already mentioned in [PP21]: In every update step, they consider the probability $p$ of an inequality to be correct, and the update process for the $i$-th coefficients computes

$$p \cdot P(X_{s,i} = x' \mid \text{inequality correct}) + \tag{3.49}$$

$$(1-p) \cdot P(X_{s,i} = x' \mid \text{inequality incorrect}). \tag{3.50}$$

This value is then used to update the probability distributions instead of

$$P(X_{s,i} = x' \mid \text{inequality correct}). \tag{3.51}$$

Thereby, the updating process takes incorrect inequalities into account and thus allows retrieving the secret key even in the presence of incorrect inequalities.

### Performance

In terms of memory and computation time, the implementation of [Del22] is extremely lightweight for a key recovery method and is reported to be able to run in a virtual machine on a normal laptop finishing in no more than a few seconds. In terms of required inequalities, the author of [Del22] considers two main scenarios: In the first scenario, all inequalities are correct and assumed to be correct; this is then evaluated with filtered and unfiltered ciphertexts on the example of Kyber512. The filtered scenario is also evaluated for all three Kyber security levels. In the second scenario, they simulate their attack and assume inequalities to be potentially incorrect; this scenario is evaluated for fault correctness probability $f \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$; in this case, only filtered ciphertexts are used to create the inequalities. As the author does not state their success rate but the success probability per coefficient (leading to a success probability of about 0.33 with no inequalities, i.e., without attacking), we only state the number of inequalities leading to a success rate of 1. Note that the attack uses the Kyber512 as specified in [ABD+21b] as opposed to [PP21] which targets an older version [ABD+19b]. Their results for Kyber512 in terms of the number of inequalities required for success rates = 1 are depicted in Table 3.3.

Table 3.3: Approximate number of required inequalities to recover the secret key with success rate 1 for Kyber512 as stated in [Del22]. Note that the $f \in \{0.5, 0.4\}$ are left out as in those scenarios inequalities do not give any information and should be removed; the results for those scenarios can be found in [Del22].

| Filtered | Unfiltered | $f = 0.9$ | $f = 0.8$ | $f = 0.7$ | $f = 0.6$ |
|---|---|---|---|---|---|
| 8500 | 14500 | 11000 | 14500 | 19000 | 20500 |

## 3.3.5 Limitations of Prior Methods

Several types of recovery methods exist as reiterated in this section. Of those methods, three statistical solutions are engineered for the particular use case of inequalities in Kyber as occurring in, e.g., [BDH+21b, PP21]. The framework of [DDGR20, DGHK22] uses an algebraic approach for decryption failure information but is not practical in the case of the aforementioned inequalities. The statistical methods do not allow for security estimates in partial attacks, i.e., they give no information on remaining bit security if the number of inequalities does not suffice to recover the key. Delvaux [Del22] offers error resistance but requires a suboptimal number of inequalities for key recovery. A computationally practical method achieving the current minimum of required inequalities and error resistance that also gives security estimates for partial attacks is yet missing.

## 3.4 Summary

Summarizing, we conclude that while several evaluations of implementation attacks on lattice-based key exchange schemes already exist, some areas are yet underanalyzed, and the questions posed in Section 1.1.2 are still unanswered. This particularly affects the vulnerability of the NTT, the exploitation of information leakage through decryption failures, and key recovery methods to obtain the secret key from decryption failure information. As these building blocks of the decapsulation routine process the secret key, understanding attacks that are not prevented by known or straight-forward countermeasures is highly relevant.

In the area of side-channel analysis on the NTT, the attacks of [PPM17] and [PP19] are powerful – in the sense that they only require a single trace and have been carried out on a physical device in practice – but cannot target the secret key or a masked implementation in a setting with increased measurement noise. The attack of [XPR+22] does not require recording templates but depends on small values as output of the inverse NTT and does not target masked implementations. In addition, no adaptations to shuffling countermeasures in attacks on the NTTs have yet been proposed; therefore, a realistic vulnerability analysis that takes countermeasures into account is missing.

Further, it is yet unclear how decryption failures can be used for generic implementation attacks that do not require an obvious target to be unprotected[13]. The vulnerabilities pointed out in [GJN20], [BDH+21b], and [DHP+22] require either timing leakage or an inadequately protected comparison operation; these works also propose appropriate countermeasures. Pessl and Prokop [PP21] target the error correction and exploit decryption failures using a fault, but the attack is prevented by shuffling the decoder, and the fault need to be reliable.

In the area of key recovery using belief propagation, it is yet unclear how incomplete retrievals may be improved upon using the underlying lattice problem. This is particularly important for attacks in which decryption failure information occurs, as in [PP21, BDH+21b, HPP21, DHP+22, Del22]. While several methods [PP21, HPP21, Del22] and a general framework [DDGR20, DGHK22] exist, a comprehensive method allowing for error-tolerant key recovery in practice that enables security estimates is still missing. This means that in cases where the secret key cannot be recovered fully, the remaining bit security in the instance of the scheme cannot be estimated.

In summary, several questions regarding attacks on the main components of the decapsulation are yet unanswered. These concern the requirements on the capabilities of the adversary, the target of the attack, prevention by single or standard countermeasures, and the availability of security estimates for partial attacks. Table 3.4 summarizes the aforementioned attacks as well as the attacks presented in this thesis in regard to the relevant criteria.

[13]Note that [Del22] provides an answer to this question but is follow-up work to our results in [HPP21].

Table 3.4: Overview of related work regarding the required capabilities to carry out the attack, the target of the attack, if such attacks are prevented by countermeasures or incorrect data, and whether security estimates are available after partial attacks. Note that security estimates for belief propagation based attacks can be obtained using techniques presented in this thesis. Also note that the table, in particular in regard to countermeasures, merely gives a strongly simplified overview, details can be found in the previous chapter. Green indicates more favorable properties.

| Work | Attacker Capabilities | Long-Term Secret | Countermeasures/Error-Tolerance | Security Estimates |
|---|---|---|---|---|
| | | Attacks on the NTT – Section 3.1 | | |
| [PPM17] | Template Attack ($\sigma \leqslant 0.6$) | Yes | Evades Masking, Prevented by Shuffling | No |
| [PP19] | Template Attack ($\sigma \leqslant 2.0$) | No | Evades Masking, Prevented by Shuffling | No |
| [XPR+22] | Simple Power Analysis | Yes | Prevented by Masking | No |
| This Thesis | Template Attack (up to $\sigma \leqslant 3.1$) | Yes | Evades Masking and Shuffling | Yes |
| | | Attacks on the Error Correction – Section 3.2 | | |
| [GJN20] | Timing Attack (on Comparison) | Yes | Prevented by Constant Time | No |
| [BDH+21b] | Power Analysis (on Comparison) | Yes | Requires Faulty Comparison | Yes, but overestimates security |
| [PP21] | Reliable Fault (on Decoding) | Yes | Prevented by Shuffling | No |
| [DHP+22] | Power Analysis (on Comparison) | Yes | Prevented by $2^{\text{nd}}$-order Masking | No |
| [Del22][a] | Unreliable Fault (Multiple Locations) | Yes | Evades Multiple Countermeasures, Prevented by Redundancy and Shuffling | No |
| [FKK+22] | Rowhammer (Key Generation) | Yes | Prevented by Hardware Optimizations, Requires Rowhammer Vulnerability, Requires Access to Key Generation | Yes[b] |
| This Thesis | Unreliable Fault (Multiple Locations) | Yes | Evades Multiple Countermeasures, Prevented by Redundancy and Shuffling | Yes |
| | | Key Recovery from Decryption Failures – Section 3.3 | | |
| [PP21] | Obtain 7500 Inequalities[c] | Yes | Marginally Error-Tolerant | No |
| [Del22] | Obtain 8500 Inequalities[c] | Yes | Error-Tolerant | No |
| [DDGR20] | Obtain $\geqslant$ 10000 Inequalities[d] | Yes | Prevented by Incorrect Inequalities | Yes |
| [DGHK22] | Obtain Correlated Inequalities[e] | Yes | Prevented by Incorrect Inequalities | Yes |
| This Thesis | Obtain 5500 Inequalities[c] | Yes | Error-Tolerant | Yes |

[a]The work of [Del22] is a follow-up up attack to our work.
[b]Using the method presented in [DGHK22].
[c]In Kyber512, uncorrelated with the coefficients of the secret key, for a success rate of 1, obtained using filtered chosen ciphertexts.
[d]As estimated in [BDH+21b] using obtained approximate equations.
[e]Correlated inequalities arise in failure boosting attacks which are out-of-scope for this thesis/our research question

# Chapter 4

# Chosen-Ciphertexts k-Trace Attacks

The Number Theoretic Transform (NTT) enables fast multiplication in some Ring Learning with Errors (RLWE) and Module Learning with Errors (MLWE) based schemes, and proposals for broader usage exist as well [LS19, CHK+21]. In a common construction of MLWE based schemes the inverse NTT processes the secret key; this is also the case for Kyber, which is the primary candidate for Key Encapsulation Mechanisms (KEMs) selected for standardization by the National Institute of Standards and Technology (NIST). Therefore, the NTT is a target for side-channel analysis, and understanding attack strategies against the (inverse) NTT is crucial to develop countermeasures and secured cryptographic devices.

Primas, Pessl, and Mangard [PPM17] introduced a template attack on the inverse NTT during decryption that targets the long-term secret. As this allows only for low noise tolerance, Pessl and Primas [PP19] target the key generation and introduced modifications to the belief propagation of [PPM17]; they thereby improve the noise tolerance. While the latter attack increases noise tolerance by a large amount, it – in most use cases – fails to target the secret key of the KEM. In addition, masking effectively reduces the noise tolerance.

In [HHP+21], we present an attack strategy exploiting the lower dimensionality of the rings in the NTT domain. By exploiting this fundamental property used to speed up the multiplication, our method allows an adversary to reduce entropy during the computation of the NTT by canceling out values using a chosen ciphertext. The ciphertext is sent to the device, and the following decryption process is targeted by a template attack. A subsequent recovery phase consists of belief propagation and an algorithm recovering the key from the obtained coefficients. Thereby, we may target the secret key with substantially increased noise tolerance – the standard deviation of the noise can be increased by a factor of up to greater than 5 (see Section 7.1.2).

In [HSST23], we explain how to adapt the attack strategy to the shuffling countermeasures of [RPBC20]. We introduce novel statistical techniques to adapt belief-propagation-based attacks to shuffling countermeasures. We propose an adaptive belief propagation node, which modifies the algorithm during execution based on observed data, as well as an algorithm that reverts permutations based on interlayer information. Using these techniques, as well as a straightforward adaptation to a masking technique, we show that these countermeasures can be circumvented – but this lowers the noise tolerance –, and we provide an assessment of their impact.

Our methods and techniques improve upon the noise tolerance and allow for a vulnerability assessment of the number theoretic transforms, in particular in regard to countermeasures. We explain our attack strategy on the example of Kyber but similar strategies very likely also apply in different settings as the principle of reducing the entropy during multiplication is not only beneficial when targeting NTTs. Further, our techniques to adapt to shuffling countermeasures is first and foremost a belief propagation technique and not limited to the NTT in Kyber.

## 4.1 Attacker Model

We assume that the adversary has the ability to submit ciphertexts to the device under attack; access to the public key is not necessarily required but likely allows improving upon the key recovery (but this is not part of our work). In addition, the adversary may perform a template attack on the execution of the inverse NTT and correctly assign probability distribution to individual load and store operations of butterfly in-and outputs. In the plain attack without shuffling countermeasures, load operations are only required to leak information in the first layer; parts of the adaptation to shuffling countermeasures require all load and store operations on intermediate butterfly in- and outputs to leak. In [PPM17] and [PP19], a similar attacker model (excluding the chosen ciphertext and only targeting either load or store operations per layer) was used in addition to targeting a real-world device to prove the practicability of the attacks.

## 4.2 Attack Strategy

The attack works in four main phases as depicted in Section 4.2: First, a ciphertext is computed

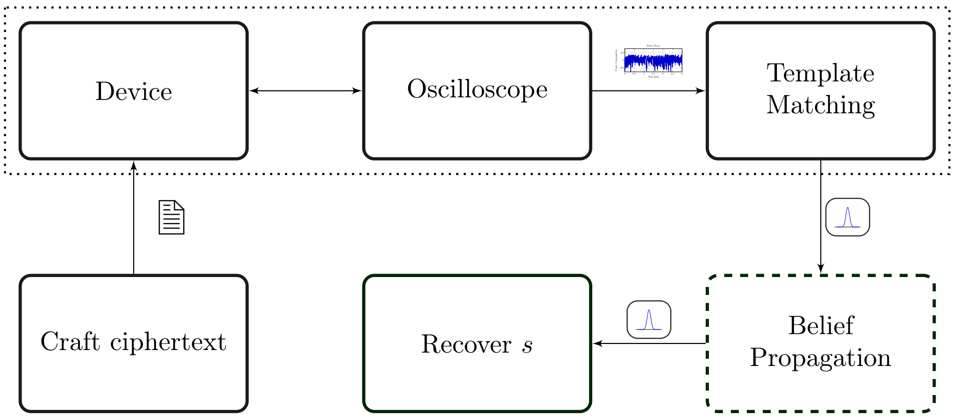


Figure 4.1: Attack strategy of k-trace attacks (adapted from [Her23b]): A chosen ciphertext is sent to a device. The ciphertext has the property that it leads to a large amount of zeros in the NTT. A template attack as in [PPM17] and [PP19] is employed (dashed box in the upper row) and results in probability distributions for intermediate inverse NTT value. These distributions are fed into a belief propagation and the final recovery obtains the secret key.

which has the desired feature to be sparse in NTT domain. This ciphertext is sent to the device under attack, where it is decompressed and fed into, firstly, an NTT. Then, the dot product between the secret key $\mathbf{s}$ and the ciphertext is computed. The result, now containing information about the secret key, is transformed using an inverse NTT – this is the computation we target.

To be precise, the device, given a ciphertext consisting of $(v, \mathbf{u})$, computes

$$v - \mathrm{INTT}(\mathrm{NTT}(\mathrm{Decompress}(\mathbf{u})\hat{\mathbf{s}}^{\top}) \tag{4.1}$$

As our ciphertext was sparse in NTT domain and the multiplication of components during the computation of the dot product is pointwise, the entropy during the inverse NTT (INTT) is reduced. This leads to the second part of the attack, namely the template attack on the inverse NTT targeting load and store operations; the template attack is as in [PP19], and we refer to their work (also summarized in Section 3.1.3) for details. Section 4.2 depicts the point of attack in the decryption routine.

As in previous work, we assume that in the first layer, load operations, and in all following layers, store operations are targeted. From the template attack, we obtain probability distributions for intermediate values.

The third and fourth part deal with recovering the key. As in previous work, we first employ a belief propagation step to recover the parts of the key which are directly contained in the measurements. Then, as the final step, we need to recover from partial information – as the secret key is only partially contained in our data, we need to recover the full key from partial data. This works by exploiting the NTT structure again and thereby using one to eight runs to recover the key.

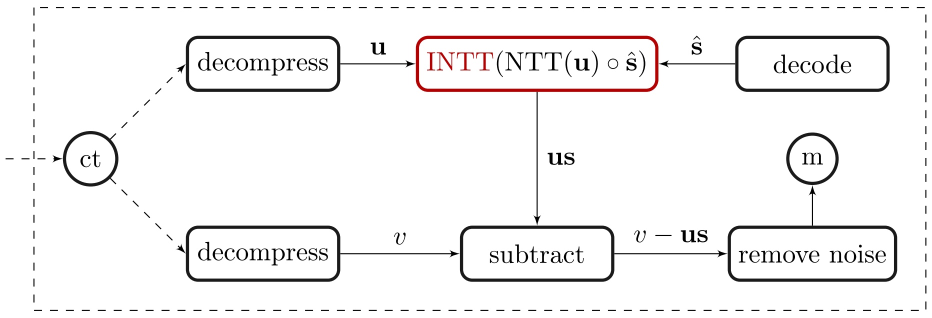


Figure 4.2: Point of attack for the side-channel analysis as part of the attack; the figure details the upper row of Section 4.2. The targeted subroutine, the inverse NTT, is depicted in red.

## 4.3 Construction of the Ciphertext

The first step of the attack is the pre-computation of a ciphertext with the property to reduce entropy in the inverse NTT call during decryption. As reiterated in Section 2.2.3, a ciphertext `ct` in Kyber has two components: $\mathbf{u}$ and $v$. The computation we target is the inverse NTT (INTT) in the following calculation:

$$\mathrm{INTT}(\mathrm{NTT}(\mathbf{u})\hat{\mathbf{s}}^\top)). \tag{4.2}$$

The input to the inverse NTT is

$$\hat{\mathbf{u}}\hat{\mathbf{s}}^\top = \sum_{l=0}^{k-1} \hat{u}_l \hat{s}_l. \tag{4.3}$$

Therefore, to reduce the entropy during the NTT, $\mathbf{u}$ should be component-wise sparse when transformed to NTT domain. We call this property being NTT-sparse, i.e., a polynomial $a \in R$ is *NTT-sparse* on $Z$ if

$$\hat{a}_i = 0 \ \forall i \in Z \tag{4.4}$$

and a vector of polynomials $(a_0, \dots, a_{k-1}) \in R^k$ is NTT-sparse on $Z_0, \dots, Z_{k-1}$ if $a_l$ is NTT-sparse on $Z_l$ for all $l \in \{0, \dots, k-1\}$. In other words, the zero set $Z$ contains indices of $a$ that have to be zero. We require $\mathbf{u} = (u_0, \dots, u_{k-1})$ to be NTT-sparse, this means to have NTT-sparse components with (possibly different) $Z_0, \dots Z_{k-1}$; for example, one component, say $u_l$, could NTT-sparse on a set $Z_l$ and all others might be $0 \in R$, i.e., $Z_{l'} = \{0, \dots, 255\}$ for $l' \neq l$.

Kyber uses a lossy compression routine to compress $\mathbf{u}$. This compression routine is applied coefficient-wise, i.e., every coefficient of every component of $\mathbf{u}$ is compressed separately. The compression prohibits a trivial computation to obtain an NTT-sparse ciphertext component: If we merely compute an NTT-sparse ciphertext component $a \in R$, then

$$a' = \text{Decompress}(\text{Compress}(a)) \neq a \tag{4.5}$$

with very high probability. The NTT which is applied to $a$ after decompression mingles up (almost) all coefficients, and thus no sparseness in NTT-domain is retained. Therefore, if we naively computed an NTT-sparse ciphertext component $\mathbf{u}$, the compression, later decompression, and NTT would result in a non-sparse polynomial $\hat{\mathbf{u}}$; see Figure 4.3. We therefore also require the components of our ciphertext $\mathbf{u}$ to be *compressible*. A polynomial $a \in R$ is compressible if

$$\text{Decompress}(\text{Compress}(a)) = a \tag{4.6}$$

and a vector of polynomials is compressible if all of its components are compressible. If a ciphertext component $\mathbf{u}$ is NTT-sparse **and** compressible, it stays the same and therefore NTT-sparse under compression and decompression. Thus, a NTT-sparse and compressible ciphertext component $\mathbf{u}$ allows us to reduce the entropy during the computation of the targeted inverse NTT. Figure 4.3 shows the functions applied to a chosen ciphertext sent to a device.

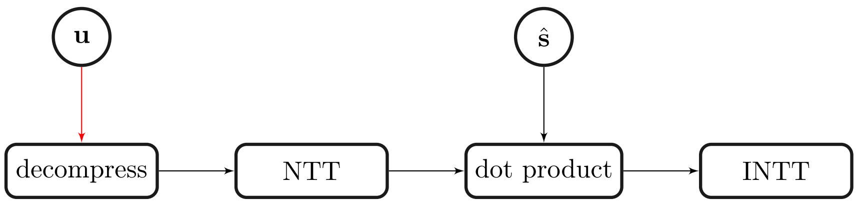


Figure 4.3: Kyber processing an NTT-sparse ciphertext $\mathbf{u}$: The first component of an incoming ciphertext is first decompressed, then transformed into NTT domain where it becomes sparse, multiplied with the secret, and then, finally, the result is re-transformed to normal domain. Sparseness is needed in the multiplication step and needs to be maintained during decompression while still being in normal domain.

### 4.3.1 Compression and Compressibility

We first explain how to obtain compressible polynomials. Compression does not only work component-wise on a vector of polynomials but in fact coefficient-wise, i.e., every coefficient is compressed separately. Therefore, we may define compressibility for a coefficient $c \in \mathbb{Z}$, with $c \in \{0, \dots, q-1\}$, as well:

$$\text{Decompress}(\text{Compress}(c)) = c. \tag{4.7}$$

For an element $c \in \mathbb{F}_q$, compressibility is defined as being compressible as element of $\mathbb{Z}$ in $\{0, \dots, q-1\}$. With this definition, a polynomial is compressible if and only if all coefficients, interpreted as integers, are compressible. As reiterated in Section 2.2.3, compression and decompression in Kyber is given by the functions

$$\text{Compress}(c) = \left\lceil (2^d/q)c \right\rfloor \bmod 2^d, \tag{4.8}$$
$$\text{Decompress}(c) = \left\lceil (q/2^d)c \right\rfloor, \tag{4.9}$$

for a compression factor $d$. In the relevant compression calls, $d$ is chosen to be either 10 or 11. Recall from Section 2.2.3 that during compression, the $q$ values of $\mathbb{F}_q$ are mapped to a multiple of $2^d/q$, and that during decompression, $k2^d/q$, for $k \in \{0, 1, \dots, 2^d - 1\}$ is mapped to $kq/2^d$. Thus, as visualized in Section 4.3.1, $c \in \mathbb{Z}$ (in our case a coefficient of a polynomial $a \in R$) is compressible if and only if it is the closest integer to a multiple of $2^d/q$. The closest integer multiple to $c$ is given by $\text{Compress}(c)q/2^d$. Therefore, to be compressible,

$$2^d c - \text{Compress}(c)q \tag{4.10}$$

needs to be sufficiently small[1].

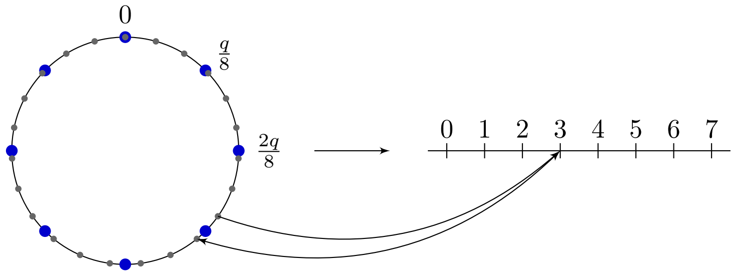


Figure 4.4: Visualization of the compressibility property with $d = 3$ and $q = 23$: Grey polynomial coefficients are mapped to the index of the closest blue compression point. The two arrows show two coefficients mapped to the same index (3), but only the lower point is mapped back to itself and is therefore compressible. In general, only the closest coefficient to a compression point is compressible.

We can now formulate compressibility as a lattice problem: Let $\mathcal{L}_{\text{compr}}$ be the lattice generated by the rows of

$$\mathbf{G}_{\text{compr}} = \begin{pmatrix} 2^d \mathbf{I}_n \\ q\mathbf{I}_n \end{pmatrix}. \tag{4.11}$$

Then for

$$\mathbf{b} = (\mathbf{a}, \text{Compress}(\mathbf{a})) \tag{4.12}$$

for $\mathbf{a} \in \mathbb{Z}^n$ we have

$$\begin{aligned} \mathbf{c} &= \mathbf{b}\mathbf{G}_{\text{compress}} && (4.13) \\ &= (2^d a_0 - q\,\text{Compress}(\text{a}_0), \dots, 2^d a_{n-1} - q\,\text{Compress}(a_{n-1})) && (4.14) \end{aligned}$$

and $\mathbf{a}$ is compressible if and only if $\mathbf{c}$ is sufficiently small. Additionally, we may obtain $\mathbf{a}$ from $\mathbf{c}$ by reducing $\mathbf{c}$ modulo $q$ and divide by $2^d + q\mathbb{Z}$, i.e.,

$$\mathbf{a} + q\mathbb{Z} = \frac{\mathbf{c} + q\mathbb{Z}}{2^d + q\mathbb{Z}}. \tag{4.15}$$

Therefore, to find compressible polynomials in $R = \mathbb{F}_q[x]/(x^n + 1)$, we may block reduce the lattice $\mathcal{L}$ given by $\mathbf{G}$ to obtain a vector $\mathbf{c} \in \mathbb{Z}^n$ which we interpret as an element of $R$ and divide by $2^d + q\mathbb{Z}$.

[1]That is, small enough to fulfill the property above, i.e., $c$ being the closest to a compression point.

### 4.3.2 NTT-Sparseness as Lattice-Problem

If $a, b \in R$ are NTT-sparse on an index set $Z \subseteq \{0, \dots, n-1\}$, then clearly so are $a+b$ as well as $\lambda a$ for $\lambda \in Z$ because the NTT is an isomorphism of modules. Therefore, the vectors $\mathbf{a} \in \mathbb{Z}^n$ which are sparse in $\hat{R}$ when interpreted as elements of $R$ and transformed to $\hat{R}$ by the NTT form a lattice. Let $\mathbf{N}^{-1}$ be the NTT-matrix (see Section 2.2.3) where rows with indices in $Z$ are removed; i.e., the matrix

$$\mathbf{N}^{-1} = (\mathrm{INTT}(\mathbf{e}_i)) \text{ for } i \in \{0, \dots, n-1\} \setminus Z \tag{4.16}$$

where $\mathbf{e}_i$ denotes the $i$-th standard basis vector of the $\mathbb{R}^n$. For $\mathbf{a} \in Z^n$ with corresponding $\hat{\mathbf{a}} \in \mathbb{Z}^n$, we have

$$\hat{\mathbf{a}} \equiv \mathbf{N}^{-1}\mathbf{a} \mod q \tag{4.17}$$

and therefore, for some $\mathbf{k} \in \mathbb{Z}^n$

$$\hat{\mathbf{a}} = \mathbf{N}^{-1}\mathbf{a} + q\mathbf{k}. \tag{4.18}$$

Thus, the lattice $\mathcal{L}_{\text{sparse}}$ of vectors which give sparse polynomials is generated by

$$\mathbf{G}_{\text{sparse}} = \begin{pmatrix} \mathbf{N}^{-1} \\ q\mathbf{I}_n \end{pmatrix}. \tag{4.19}$$

### 4.3.3 Combining Comressibility and Sparseness

We may now formulate the problem of finding an NTT-sparse polynomial $a$ as a lattice problem: For a vector $\mathbf{a} \in \mathbb{Z}^n$ to map to a compressible and NTT-sparse polynomial $\mathbf{a} \in R$, it suffices to be a sufficiently small element of $\mathcal{L}_{\text{compr}}$ and any element of $\mathcal{L}_{\text{sparse}}$. Therefore, clearly, it suffices to be a sufficiently small element of $\mathcal{L}_{\text{compr}} \cap \mathcal{L}_{\text{sparse}}$. As $\mathcal{L}_{\text{compr}}$ is generated by

$$\mathbf{G}_{\text{compr}} = \begin{pmatrix} 2^d\mathbf{I}_n \\ q\mathbf{I}_n \end{pmatrix} \tag{4.20}$$

and $\mathcal{L}_{\text{sparse}}$ is generated by

$$\mathbf{G}_{\text{sparse}} = \begin{pmatrix} \mathbf{N}^{-1} \\ q\mathbf{I}_n \end{pmatrix}, \tag{4.21}$$

the intersection is generated by

$$\mathbf{G} = \begin{pmatrix} 2^d\mathbf{N}^{-1} \\ q\mathbf{I}_n. \end{pmatrix} \tag{4.22}$$

We can easily verify that an element of the lattice generated by $G$ gives an NTT-sparse compressible vector: Let $\mathbf{a} \in \mathbb{Z}^n$ with corresponding NTT-vector $\hat{\mathbf{a}} \in \mathbb{Z}^n$ and let

$$\mathbf{b} = (\hat{\mathbf{a}}, \mathrm{Compress}(\mathbf{a})) \tag{4.23}$$

then

$$\begin{aligned} \mathbf{c} &= \mathbf{bG} && (4.24) \\ &= (2^d a_0 - q\,\mathrm{Compress}(\mathrm{a}_0), \dots, 2^d a_{n-1} - q\,\mathrm{Compress}(a_{n-1})). && (4.25) \end{aligned}$$

If $\mathbf{c}$ is sufficiently small, $\mathbf{a}$ is compressible. In addition, $\mathbf{a}$ is NTT-sparse by definition of $\mathbf{N}^{-1}$. Therefore, we may find compressible NTT-sparse polynomial $a \in R$ by finding a small vector $\mathbf{c}$ in $\mathbf{G}$ and then computing

$$a = \frac{\mathbf{c} + q\mathbb{Z}}{2^d + q\mathbb{Z}} \in R. \tag{4.26}$$

**From a single polynomial to a vector.** The procedure described above results in a single polynomial $a \in R$. This polynomial is then used as a component $u_l$ of $\mathbf{u}$ and, in some cases, all other components of $\mathbf{u}$ are set to 0. In other cases – that is when using our advanced recovery method – choosing $Z_0, \ldots, Z_{k-1}$ for the components of $\mathbf{u}$, $u_0, \ldots, u_{k-1}$ such that more than one $u_l$ is non-zero is beneficial. The number of non-zero inputs to the inverse NTT is the number of indices not in all the $Z_l$ as the input to the NTT is given by

$$\hat{\mathbf{u}}\hat{\mathbf{s}}^\top = \sum_{l=0}^{k-1} u_l s_l. \tag{4.27}$$

Conversely, the coefficients of the input to the inverse NTT set to zero are at the indices which are in all the $Z_l$. Coefficients with an index that is an element of all but exactly on $Z_l$ can be recovered directly. Using our recovery method described in the following section, the directly recovered coefficients may be used to recover the remaining coefficients. Indices being elements of more than one but not all zero sets do not result in a zero (which would reduce entropy) but also do not allow for a coefficient to be recovered. Thus, all zero sets should be disjoint whenever possible. If there are no countermeasures in place checking for repeated usage of a polynomial, $\mathbf{a}$ may be re-used for all components (in different runs); this spares an adversary from having to run more than one lattice reduction, which is computationally expensive.

**Reducing the computation effort.** In Kyber, the ring $R$ does not contain an $n$th root of unity and, therefore, the NTT maps to $\mathbb{F}_{q^2}^{\frac{n}{2}}$ instead of $\mathbb{F}_q^n$. This also means that the NTT has only 7 layers and the computational graph is disconnected. Therefore, we may reduce the computational effort by running the above procedure separately for half of the NTT. In practice, this means taking only the even, respectively odd, indices for every row of $\mathbf{N}$ in Equation (4.22). Clearly, this reduces the dimension of the lattice by a factor of 2.

**Choice of $Z$.** The choice of $Z$ greatly influences the results: We show in Section 7.1 that an optimum in regard to measurement noise is reached if the amount of zeros is as distributed as evenly as possible. The amount of zeros, as we will see in the following sections, influences not only noise tolerance but also the amount of required traces. Therefore, an adversary has to choose $Z$ depending on noise tolerance and the number of traces that can be recorded. In the case of Kyber, for each component of $\mathbf{u}$ a $Z$ can be chosen separately. Note that arbitrary sets of $Z$ may not be possible (or desirable), or they may require a large amount of computational effort; the possibilities and the required computational effort are highlighted in Section 7.1.

**Different methods.** A method avoiding block reduction and the resulting computational complexity is described in [HHP+21]. The lattice reduction is avoided by exploiting the NTT structure. While the computational effort is greatly reduced, it only allows for non-distributed zeros and therefore slightly worse results. The trade-off may be favorable for an adversary with little computational resources. The technique is not described here as the author of this thesis was not involved in its creation.

**Different schemes.** In the case of NewHope, the vector $\mathbf{u}$ is a single, uncompressed polynomial which significantly decreases the effort required to create a sparse polynomial. A single zero set $Z$ is used for all components of $\mathbf{u}$. In fact, for NewHope, the procedure is trivial, and no lattice reduction is required. For FrodoKEM, $\mathbf{u}$ is a matrix over $\mathbb{F}_q$ and uncompressed as well, but as no NTT is used, the usefulness of sparse ciphertexts might differ from Kyber. Saber uses a rounded

**u** which is a vector of polynomial similar to in Kyber, but in the originally specified algorithm, no NTT is used. Note that proposals for NTTs for NTT-unfriendly rings exist which could enable the usage of an NTT in FrodoKEM as well as in Saber [CHK+21]. In addition, other fast multiplication algorithms likely allow for similar attacks and sparse ciphertext can reduce the entropy in such cases as well.

## 4.4 Recovery of the Secret Key

By running a template attack, as described in [PPM17] and [PP19], we obtain information about intermediate values of the inverse NTT computed with $\mathbf{us}^\top$ as input. Assuming loads in the first layer and stores in all other layers are targeted, this information comes in the form of a probability distribution for each in- and output of every butterfly. That means, we have $h(l+1)$ probability distributions where $h$ is the height of the NTT – 256 in Kyber – and $l$ is the number of layers, 7 in Kyber. The distributions in the last or first layer are distributions on the secret key coefficients multiplied with a known polynomial. Unless in settings with exceptionally low noise, we may not recover coefficients of the targeted vector from those layers alone. Instead, the joint probability distributions and the corresponding marginals need to be computed to retrieve $\mathbf{us}^\top$. Similar to prior work, we use belief propagation to achieve this. As the known polynomial that is multiplied with the secret is our manipulated ciphertext component **u**, which is sparse, the resulting recovered coefficients do not hold information about the full secret. In addition, $\mathbf{us}^\top$ is a dot product which, even if recovered completely, does not immediately allow for a full key recovery in a single run (in contrast to previous work where different parts of algorithms were targeted). To account for this and reduce the number of required traces, we introduce an additional algorithm reducing the number of required traces, again relying on lattice reduction. The in- and outputs to the inverse NTT under attack are shown in Section 4.4.

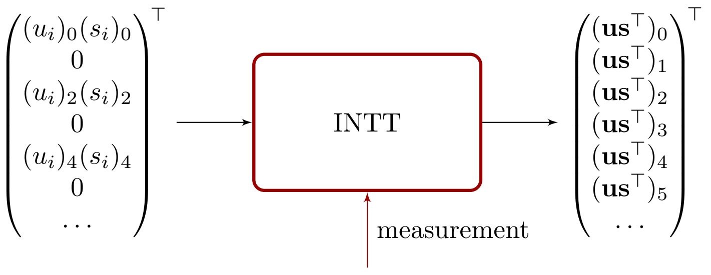

$\begin{pmatrix}(u_i)_0(s_i)_0\\0\\(u_i)_2(s_i)_2\\0\\(u_i)_4(s_i)_4\\0\\\cdots\end{pmatrix}^\top$
INTT
measurement
$\begin{pmatrix}(\mathbf{us}^\top)_0\\(\mathbf{us}^\top)_1\\(\mathbf{us}^\top)_2\\(\mathbf{us}^\top)_3\\(\mathbf{us}^\top)_4\\(\mathbf{us}^\top)_5\\\cdots\end{pmatrix}^\top$

Figure 4.5: In- and outputs to the inverse NTT in the $i$-th component and $Z = \{1, 3, 5, \ldots, 255\}$ with all other component set to zero; the polynomials are written as vectors. The in- and outputs are also the values measured in the first and last layer of the inverse NTT. This figure details the upper middle square (partially red) of Section 4.2 and shows the situation during the attack with a sparse and compressible ciphertext.

### 4.4.1 Belief Propagation

The belief propagation part of the key recovery is similar to [PP19] but applied to our setting. Pessl and Prokop, in [PP19], targeted the key generation and proposed several improvements. We use their improved butterfly node but, as depicted in Section 4.2, target the inverse NTT during the decapsulation, leading to several differences, especially a substantially larger value range for

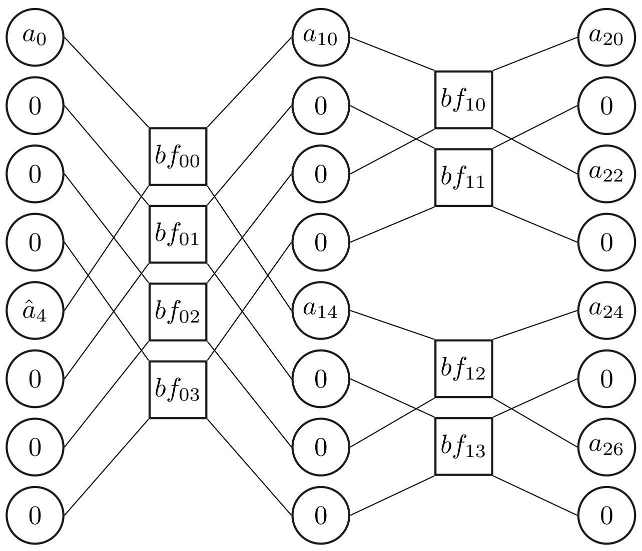


Figure 4.6: A belief propagation graph as arising in our attack but with $n = 8$ instead of $n = 256$ and 6 zeros. Variable nodes are depicted by circles, and the squared nodes represent butterflies (for details on the node types see Section 3.1.3). Zeroed-out nodes contain a zero instead of the variable name (with variable names following the notation in Figure 2.7).

intermediate values (see Section 2.2.3). We do not use message damping or an advanced schedule mechanism. Additionally, our implementation is improved – especially in terms of parallelism – and is not specific to this attack[2]. The implementation details are described in Section 7.1.1, and the generic belief propagation is described in Chapter 7. A typical belief propagation graph as arising in our attack (but in lower dimension) is shown in Figure 4.6.

### 4.4.2 Recovering From Partial Keys

After a successful belief propagation run, we obtain the vector $\hat{\mathbf{u}}\hat{\mathbf{s}}^\top$. Due to the component-wise NTT-sparseness of $\mathbf{u}$, the dot product does not contain any information about some coefficients of $\mathbf{s}$. Additionally, the dot product makes it impossible to *directly* recover $\mathbf{s}$ from $\mathbf{u}\mathbf{s}^\top$, even if $\mathbf{u}$ were not sparse. Previous attacks differed in this as either a different NTT or a different (part of the) algorithm was targeted.

#### Recovery without Lattice Reduction

A straightforward solution to recovering $\mathbf{s}$ is to run the attack again with different $|Z|$-sets for different components of $\mathbf{u}$ multiple times; thereby, we may successively recover all coefficients of $\hat{\mathbf{s}}$, run an inverse NTT on the recovered secret, and obtain $\mathbf{s}$. In more detail, this works as follows: An adversary sets $Z_0$, i.e., the set of zeros for the first component, to some decimation and all other $Z$-sets to contain all indices in the first run. Thereby, as with such an $\mathbf{u}$, we have

$$\hat{\mathbf{u}}\hat{\mathbf{s}}^\top = \sum_{l=0}^{k-1} \hat{u}_l \hat{s}_l = \hat{u}_0 \hat{s}_0 \ (\in R_q) \tag{4.28}$$

[2]It can be used in different attacks as update rules can be exchanged quickly.

and coefficients of $s_0$ can be recovered if their index is not in $\{0,1,\dots,255\}\setminus Z$; this is because $\mathbf{u}$ (and thereby $u_0$) are known, the multiplication is (almost) pointwise, and coefficients of $\hat{\mathbf{u}}$ are invertible where non-zero as $\mathbb{F}_{q^2}$ is a field. Repeating this with different $Z$ allows recovering all coefficients of $s_0$. Then, the same procedure for $u_1$, i.e., with setting the $Z_1$ to what previously were the $Z_0$ and setting all other components of $\mathbf{u}$ to zero, recovers $s_1$. Repeating this for all indices of $\mathbf{u}$, i.e., from 0 to $k-1$, allows the adversary to recover $\mathbf{s}$ completely. While this is a valid strategy, it also requires at least $k\frac{256}{256-|Z|}$ traces. For example, with 128 zeros, we start with $Z = \{0,2,4,\dots,254\}$ for the first component $u_0$ and $u_i = 0$ for $i \in \{1,\dots,k-1\}$; this allows recovering half of the coefficients of $\hat{s}_0$. Then, we set $Z = \{1,3,5,\dots,255\}$, again for $u_0$ and recover the other half of coefficients of $\hat{s}_0$. This is repeated with $u_l$ for $0 < l < k$ to recover $\hat{\mathbf{s}}$ completely, requiring a total of $2k$ traces. Table 4.1 shows the amount of traces required per $k$ and per zero level. An adversary is usually interested in having to perform as few measurements

Table 4.1: Amount of traces with straightforward recovery without using a block reduction algorithm per Kyber-$k$. The values for $k$ correspond to the different Kyber versions. Note that for 64 zeros, the non-zero blocks overlap and do not allow for fewer traces but the noise tolerance is decreased. Therefore, this scenario is not recommended without our recovery algorithm.

| $k$ | 192 zeros | 128 zeros | 64 zeros | 0 zeros |
|---|---|---|---|---|
| $k=2$ | 8 | 4 | 4 | 2 |
| $k=3$ | 12 | 6 | 6 | 3 |
| $k=4$ | 16 | 8 | 8 | 4 |

as possible and the amount of traces required with this recovery strategy is suboptimal.

### Recovery using Lattice Reduction

To reduce the number of required traces, we provide a more sophisticated recovery method. Using our recovery algorithm, an adversary requires merely $k$ traces in most cases and even fewer if only a small noise tolerance is required; this is where the name of the attack derives from. To achieve this, our method recovers $s_l$ from $\nu$ coefficients of $\hat{s_l}$. Recall from Section 2.2.3 that the coefficients of the components of $\mathbf{s}$ are elements of $\{-2,\dots,2\}$ for Kyber512 and Kyber1024, and elements of $\{-3,\dots,3\}$ for Kyber1024. This means the coefficients are already comparably small (in relation to the range from $\{0,\dots,q-1\}$), and if we recover sufficiently many, we obtain a unique Shortest Vector Problem (uSVP). Taking into account that the NTT is disjoint, and we can reduce our problem to half of the keys; there are $5^{128}$ possible half-keys, and for Kyber1024 there are $7^{128}$ possible half-keys. This means if $q^\nu > 5^{128}$ for the first two Kyber security levels and $q^\nu > 7^{128}$, the key can be recovered using lattice reduction (assuming an adversary with sufficient computational resources):

We again denote the NTT-function on the identity matrix rows, with index in $Z$ left out, as matrix $\mathbf{N}^{-1}$. To account for the disconnectedness of the NTT in Kyber, we again halve the dimension by only taking either the even or the odd indices of $\mathbf{N}^{-1}$ in every row, and carry out the routine twice; we will from here on ignore this to keep the description simple. Then, let $\mathbf{a}$ be given by the inverse NTT of $\hat{\mathbf{a}}$ as vector $\mathbf{a}$ over $\mathbb{Z}$, this means

$$\mathbf{N}^{-1}\hat{\mathbf{a}} \equiv \mathbf{a} \mod q. \tag{4.29}$$

Equation (4.29) means that for some vector $\mathbf{k}$ over $\mathbb{Z}$, we have

$$\mathbf{N}^{-1}\hat{\mathbf{a}} + q\mathbf{k} = \mathbf{a} \tag{4.30}$$

which is close to $s_l$ (when interpreted as vector $s_l$ over $\mathbb{Z}$) as it differs only on $Z$. We may therefore recover $s_l$ by solving the Closest Vector Problem (CVP) in the lattice generated by the rows of

$$\mathbf{G}_{\text{INTT}} = \begin{pmatrix} \mathbf{N}^{-1} \\ q\mathbf{I}_n \end{pmatrix} \tag{4.31}$$

for the vector $\mathbf{a}$. We may now, again, translate a CVP into an uSVP (see Chapter 2): Finding a short vector in the lattice generated by the rows of

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_{\text{INTT}} \\ \mathbf{a} \end{pmatrix} \tag{4.32}$$

using, e.g., BKZ allows us to recover $\mathbf{s}$. As before, the computational effort can be reduced by applying the algorithm separately to half of the coefficients if the NTT is disjoint as in Kyber. The procedure has to be carried out for all components of $\mathbf{s}$.

**Employing the recovery method.** The recovery method can be used in the attack as described in the following: We first generate $\rho$ different $\mathbf{u} = (u_0, \dots, u_k)$ each with components being sparse on $Z_0, \dots Z_k$ such that the non-zero values of the $\hat{u}_l$ are disjoint. Each $\mathbf{u}$ is used for a separate run, this means we require $\rho$ traces. The goal is to recover $\nu \geqslant 32$ coefficients of each component of $\mathbf{s}$. With $\mathbf{u}$ chosen such that the non-zeros in the components are disjoint, we can obtain coefficients of different $\hat{s}_l$ in a single run. This works because, due to the disjointedness, the coefficients of the components do not add up with each other in the dot product. Therefore, as before, by multiplying with the coefficient-wise inverse of $\hat{\mathbf{u}}$ we may obtain a total of $256 - |Z|$ coefficients, but not necessarily of a single component.

Without the previously discussed recovery method, this approach does not provide a benefit, as all coefficients of all components have to be recovered. But using the algorithm described above, we can then recover the complete secret key from only $\nu$ coefficients of each component. In the $k$-trace attack, we have $\nu = 32$ and $\rho = k$, as shown in Section 7.1.

For example, for $k = 2$ and $\nu = 128$, we would, in the first run, set $Z_0, Z_1$ choose to contain $n - 64$ indices with the indices not in $Z_0$ being in $Z_1$ and vice-versa. Then, choosing distributed zeros (which is optimal in terms of noise resistance), i.e.,

$$Z_0 = \{i \mid i \bmod 8 \in \{2,3,4,5,6,7\}\} \text{ and } Z_1 = \{i \mid i \bmod 8 \in \{0,1,4,5,6,7\}\}$$

we have

$$\begin{aligned} \mathbf{u}\mathbf{s}^{\top} &= u_0 s_0 + u_1 s_1 && (4.33)\\ &= \big(\blacksquare,\ \blacksquare,\ 0,\ 0,\ 0,\ 0,\ \dots,\ \blacksquare,\ \blacksquare,\ 0,\ 0\big)\, s_0 && (4.34)\\ &+ \big(0,\ 0,\ \bullet,\ \bullet,\ 0,\ 0,\ \dots,\ 0,\ 0,\ \bullet,\ \bullet\big)\, s_1 && (4.35) \end{aligned}$$

where $\bullet$ and $\blacksquare$ denote non-zero values[3]. This way, we have a vector that is sparse in 128 positions and contains information about $\nu = 64$ coefficients of each component. This is enough to recover the secret key as described above; therefore, in this case, we have a single trace attack, i.e., $\rho = 1$. The exact amount of required traces depends on the quality of lattice reduction; for BKZ, this depends on the block size. The minimal amount of required traces (which can be achieved for an adversary with access to commonly available hardware; c.f. Section 7.1) is shown in Table 4.2.

[3]Not to be understood as variables; each $\bullet$ may have a different value (and $\blacksquare$ as well).

Table 4.2: Minimal amount of traces with lattice reduction based recovery per Kyber-$k$ with the number of zeros in the input vector to the inverse NTT. The values for $k$ correspond to the different security levels. The amount of traces without lattice reduction is shown in brackets.

| $k$ | 192 zeros | 128 zeros | 64 zeros | 0 zeros |
|---|---|---|---|---|
| $k = 2$ | 2 (8) | 1 (4) | 1 (4) | 1 (2) |
| $k = 3$ | 3 (12) | 2 (6) | 1 (6) | 1 (3) |
| $k = 4$ | 4 (16) | 2 (8) | 2 (8) | 1 (4) |

## 4.5 Adaptation to Countermeasures

The existence of implementation attacks is well-known and countermeasures protecting devices to a certain degree are commonly employed. A realistic evaluation of vulnerabilities therefore needs to take countermeasures into account. In the case of the attack described in this chapter, first published in [HHP+21], a common masking scheme [RRC+16, OSPG18] is already defeated by the nature of the attack itself – linear masking of the NTT is defeated by running the attack on all shares. But countermeasures presented by [RPBC20], reiterated in Section 3.1, designed to counter [PPM17] and [PP19], also prevent our attack when not adaptations are taken.

We exemplary explain how to adapt to a subset of the presented countermeasures, namely the shuffling countermeasures. This is achieved in software in the case of fine shuffling and by extending the adversary model in the case of coarse shuffling. Note that the masking of [RPBC20], as well as various other types of countermeasures, are out of scope; defeating standard masking and first shuffling countermeasures exemplary explains how to adapt Soft Analytical Side-Channel Attack (SASCA)/belief propagation to countermeasures in the case of post-quantum schemes.

While several works on countering shuffling in various settings exist (see Section 3.1.7), the impact of shuffling countermeasures in the settings of [PPM17], [PP19], and [HHP+21] is yet unclear. Note that we do not attack a real-world protected implementation but instead exemplary explain and evaluate adaptation to hiding countermeasures in belief propagation based SASCA on (inverse) NTTs.

**Adaptation to standard masking.** A straightforward way to mask the (inverse) NTT is to arithmetically mask the input to the (inverse) NTT using $n+1$ shares; this is described, e.g., in [RRC+16] or [OSPG18]. This means instead of a single inverse NTT on the input $\hat{\mathbf{x}}$, here $\hat{\mathbf{x}} = \hat{\mathbf{u}}\hat{\mathbf{s}}^\top$, $n+1$ inverse NTTs are computed on

$$\hat{\mathbf{x}} + \widehat{\mathbf{m}}^{(0)} + \widehat{\mathbf{m}}^{(1)} + \cdots + \widehat{\mathbf{m}}^{(n-1)} \tag{4.36}$$

and

$$\widehat{\mathbf{m}}^{(0)}, \widehat{\mathbf{m}}^{(1)}, \ldots, \widehat{\mathbf{m}}^{(n-1)} \tag{4.37}$$

where $\widehat{\mathbf{m}}^{(i)}$ are randomly sampled masking vectors. As the (inverse) NTT is linear, $\mathbf{x}$ may be computed from the results of the inverse NTTs by subtracting all $\mathbf{m}^{(i)}$ from $\mathbf{x} + \mathbf{m}^{(0)} + \mathbf{m}^{(1)} + \cdots + \mathbf{m}^{(n-1)}$.

In the masked case, the attack described above works by targeting the NTTs on all shares. If the adversary may obtain $\mathbf{x} + \mathbf{m}^{(0)} + \mathbf{m}^{(1)} + \cdots + \mathbf{m}^{(n-1)}$ as well as all $\mathbf{m}^{(i)}$, they can clearly compute the secret themselves; the attack on the single shares is not prevented by standard masking. Whenever the success rate $p_{sr}$ is smaller than 1, the success rate for a masked implementation with $n+1$ shares is reduced to $p_{sr}^{n+1}$.

### 4.5.1 Adaptation to Fine-Shuffling

Adapting to fine shuffling, as proposed by Ravi et al. in [RPBC20] (see Section 3.1.7), is not as straightforward as the adaptation to standard masking. Fine shuffling permutes the load and store operations of butterflies inside the (inverse) NTT. Therefore, if employing the attack on a protected (inverse) NTT, the priors at variable nodes in the belief propagation are assigned incorrectly with probability $p = \frac{1}{2}$. Figure 4.7 shows an example of wrongly assigned priors for the inputs at a butterfly node. We propose two different methods to defeat fine shuffling: *Mixing*

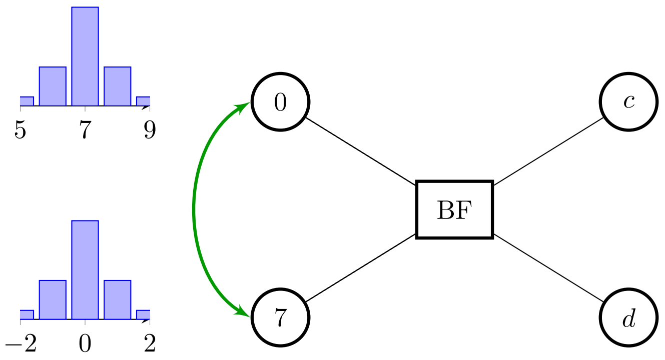


Figure 4.7: Incorrectly assigned priors at a butterfly node of an (inverse) NTT. The prior distributions of the inputs are permuted and therefore the butterfly node models an arithmetic that does not fit the priors.

*priors* and the usage of a *shuffle node.* Mixing priors gives worse results than using a shuffle node but is easy to deploy, as it only requires an inexpensive preprocessing step; therefore, it is also more likely suited to be used in other belief propagation based attacks. Shuffle nodes change the Bayesian updating process by adapting an internal factor to the currently observed beliefs. This then leads to the shuffle node obtaining the permutation applied to the in- and outputs of a butterfly. In contrast to prior work that combines belief propagation with neural networks (see Section 2.3.1), our shuffle node adapts to data observed *during* the belief propagation run but no training phase is required.

#### Mixing Priors

The potentially permuted priors at a butterfly node cause the belief propagation to fail – it runs into an invalid state because a vector with high joint probability with very high probability does not fulfill the NTT equations. To ensure that the belief propagation does not fail and has the chance to converge against the correct key, we may mix up priors. Instead of working with an individual prior per in- and output, a single prior for all possibly permuted priors is computed.

For a butterfly node with input $a, b$ and outputs $c, d$ and measurements resulting in priors $D_a, D_b, D_c, D_d$ (where $D_a$ and $D_b$ as well as $D_c$ and $D_d$ may be permuted), we compute

$$D_{\text{mix,in}} = \left\{ x \mapsto \frac{P_{D_a}(a = x) + P_{D_b}(b = x)}{2} \mid x \in \operatorname{dom} D_a \cap \operatorname{dom} D_b \right\} \tag{4.38}$$

and

$$D_{\text{mix,out}} = \left\{ x \mapsto \frac{P_{D_c}(a = x) + P_{D_d}(b = x)}{2} \mid x \in \operatorname{dom} D_c \cap \operatorname{dom} D_d \right\}. \tag{4.39}$$

Then, the distribution assigned to both inputs, i.e., to model the values of $a$ and $b$, is $D_{\text{mix,in}}$, and the distribution assigned to the outputs is $D_{\text{mix,out}}$. Thereby, the correct value at all in- and outputs certainly has a positive probability if the original priors have not been incorrect in the first place. To be precise, the probability of the correct value of an in- or output is at most halved by fine shuffling when adapting by the use of *mixing priors*. The situation is depicted in Figure 4.8.

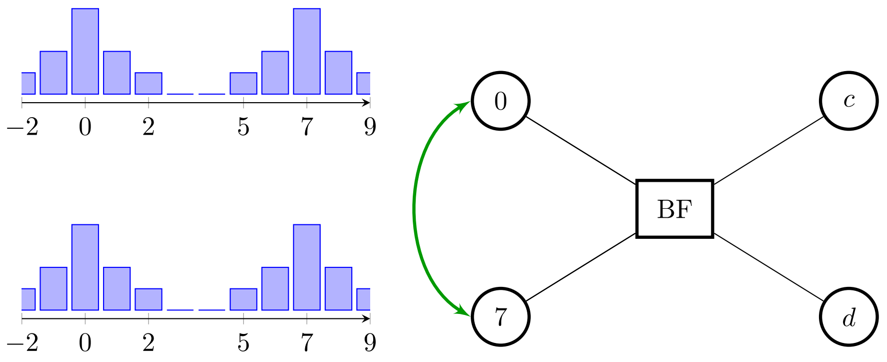


Figure 4.8: Priors at a butterfly node of an (inverse) NTT after mixing priors. The original prior distributions of the inputs are permuted (c.f. Figure 4.7), and therefore the butterfly node models an arithmetic which does not fit the priors – mixing up the priors resolves the inconsistency.

### Shuffle Nodes

Mixing priors is a practical and easy-to-implement adaptation to fine shuffling, but it ignores available information. In an undisturbed (i.e., after attacking an implementation with no shuffling countermeasures) belief propagation, the in- and output distributions of a butterfly node converge against distributions matching with regard to the arithmetic relation represented by the node.

Let $a, b$ again be inputs with outputs $c, d$ and beliefs $D_a, D_b, D_c, D_d$ at a butterfly computing $c = a+b$ and $d = \zeta(a-b)$. In a successful run, $D_x$ with true value $v_x$ converges against a probability distribution that is 1 a $v_x$ and 0 everywhere else, i.e., $v \mapsto \delta_{v_x}(v)$, and we say that $D_x$ converges against $v_x$. With sufficiently[4] precise measurements and in this notation, each distribution will converge against a value with high probability; this value is also likely to have a reasonably high probability in the prior. In a shuffled situation with mixed priors, convergence is less likely, even with precise measurements, but we may leverage additional information by comparing the current distributions against the priors. The Kullback-Leibler (KL) divergence [KL51], expressing the similarity of two probability distributions, between current distributions and priors allows reasoning about the shuffling permutation.

In each step, we compute incoming messages (representing distributions) at two potentially shuffled nodes against the priors of those nodes. Thereby, we compare which permutations of in- and outputs give the best (relative) match. An absolute decision would prevent the belief propagation from converging, as this is not clearly determined except in scenarios with very low noise and a wrong. Therefore, we use a continuous factor representing the current belief in the permutations of in- and outputs. This factor, which we denote by $\xi$, is updated in every factor node step of the belief propagation from incoming beliefs. The updated priors, $P'_a$ and $P'_b$, are

[4]See Section 7.1 for the required noise level.

then computed as

$$\begin{aligned} P'_a &= \xi P_a + (1-\xi)P_b \\ P'_b &= \xi P_b + (1-\xi)P_a \end{aligned}$$

where $P_a, P_b$ are the measured priors and $P'_a, P'_b$ are the priors used in the adapted version. In other words, messages are computed as in an undisturbed run, but priors at variable nodes are mixed according to the current beliefs at two variable nodes. This means, $\xi = 0$ corresponds to an unshuffled situation (at a single node), $\xi = 1$ represents the belief into shuffled priors, and $\xi \in ]0,1[$ corresponds to uncertainty and leads to mixing priors. The technique of *mixing priors* described in the previous section is a special case with fixed $\xi = \frac{1}{2}$.

The (input) shuffling factor $\xi$ is computed by first calculating the Kullback-Leibler divergences of all inputs with all priors, i.e., calculating

$$\mathcal{D}_{a,a} = \mathcal{D}_{\mathrm{KL}}(m_{a,i}, P_a) \tag{4.40}$$
$$\mathcal{D}_{b,b} = \mathcal{D}_{\mathrm{KL}}(m_{b,i}, P_b) \tag{4.41}$$
$$\mathcal{D}_{a,b} = \mathcal{D}_{\mathrm{KL}}(m_{a,i}, P_b) \tag{4.42}$$
$$\mathcal{D}_{b,a} = \mathcal{D}_{\mathrm{KL}}(m_{b,i}, P_a) \tag{4.43}$$

where $P$ denotes the respective priors and $m_{x,i}$ are the messages in the $i$-th step. From this, we may compute the evidence for shuffling/no-shuffling as

$$E_{\text{shuffling}} = D_{a,a} + D_{b,b} \tag{4.44}$$
$$E_{\text{no-shuffling}} = D_{a,b} + D_{b,a} \tag{4.45}$$

and the shuffling factor as

$$\xi = \frac{E_{\text{shuffling}}}{E_{\text{shuffling}} + E_{\text{no-shuffling}}}. \tag{4.46}$$

The computation of the shuffling factor takes place in a shuffle node attached to potentially shuffled variable nodes as shown in Figure 4.9. This node represents the prior of both nodes and changes the priors according to the shuffle factor. The mixing process is visualized in Figure 4.10.

The shuffle node enables our belief propagation to continuously adapt by comparing computed information to information given as priors. This enables us to infer the shuffling permutations during a belief propagation run while gradually adapting and converging against the correct key in the presence of shuffling – if the noise levels are low enough.

### 4.5.2 Adaptation to Coarse Shuffling

The adaptation to coarse shuffling is unexpectedly more difficult because it is the countermeasure providing more entropy and, therefore, achieves a higher level of protection – adapting is computationally more expensive and requires lower noise levels. Coarse shuffling, also introduced by Ravi et al. [RPBC20], in contrast to fine shuffling permutes butterflies instead of inputs to butterflies (see Section 3.1.7). This results in a much higher introduced entropy. As this countermeasure does not permute in- and output distributions of a butterfly, we may not compare distributions against priors and thereby learn the shuffling permutation. Instead, we are missing the knowledge about how butterflies between layers are connected.

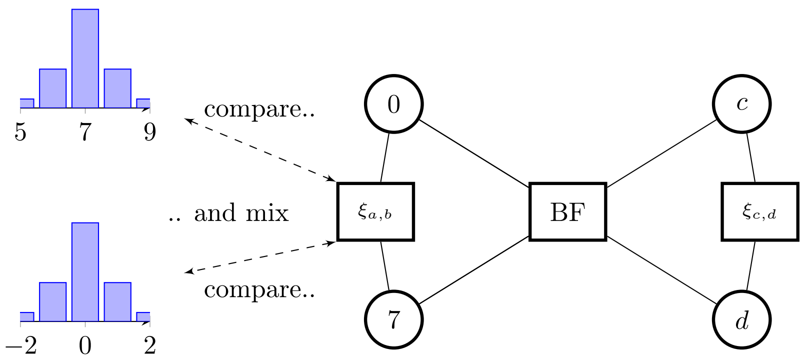


Figure 4.9: Priors at a butterfly node of an (inverse) NTT when using a shuffle node. The original priors of the input nodes are permuted (c.f. Figure 4.7 and Figure 4.8), and therefore the butterfly node models an arithmetic relationship that does not fit the priors – the shuffle nodes resolve this inconsistency by computing a shuffle factor which determines how the distributions are mixed.

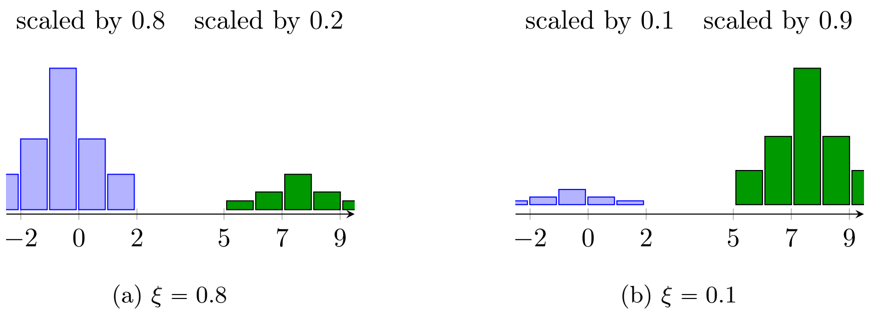


Figure 4.10: Mixed priors at a shuffle node with $\zeta = 0.8$ and $\zeta = 0.1$. The part of the distribution centered around 2 is the measurement taken at the first input while the part of the distribution centered around 7 is the measurement taken at the second input.

### Extended Attacker Model

The attacks of [PP19] and [HHP+21] did not target load and store operations in every layer but load operations in the first layer and store operation in every other layer. In the case of coarse shuffling, we extend the attacker model: To enable our methods, we target load and store operations in all intermediate layers, load operations in the first layer, and store operations in the last layer. This means that we have two measurements of all intermediate values shuffled with different permutations. If we can match intermediate values correctly, we obtain a graph that is correctly connected but the first and last layers may be permuted. Figure 4.11 shows the situation after having obtained measurements of both load and store operations in two layers.

For a successful belief propagation run, the adversary needs to find the correct connections between two layers. This is equivalent to finding the relative permutation of layer $i$ to layer $i-1$ starting with $i$ being the first layer. We achieve this by matching store operations of layer $i-1$ to load operations of layer $i$ – those measurements are measurements of the same values but with different permutations. As no information about the permutation of the first layer is available in our model, it has to be dealt with separately. We first explain how the matching process works in general between single nodes and call this *One-Point Matching*. More information is available –

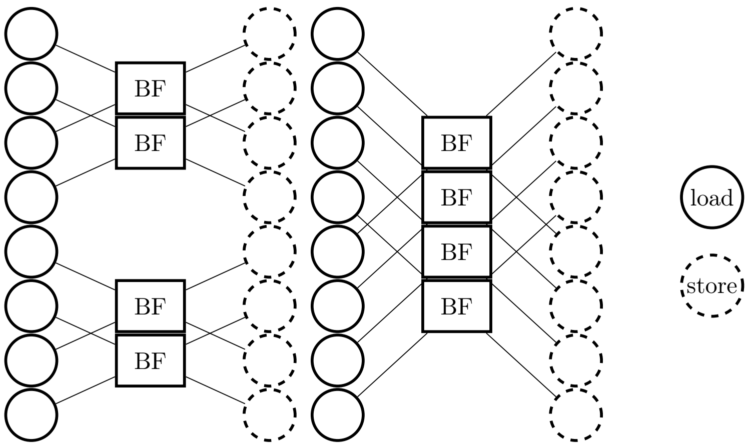


Figure 4.11: The extended attacker model as described and depicted in [HSST23]. Instead of only store operations, load and store operations are targeted for intermediate layers.

butterflies are connected to more than one node – and may be used to improve the accuracy; this improved process is called *Two-Point Matching.*

**Notation.** In the following, we describe the situation in a block between two layers, given store and load measurements, and the task of the adversary is to match those correctly. We denote load distributions as $l_i$ and stores as $s_i$ where $i$ is the index of the node the measurement was taken at. True values, i.e., the value a node at $i$ actually has, are denoted by $v_i$ with Hamming weight $h_i = \mathrm{HW}(v_i)$. Note that the adaptation to shuffling countermeasures assumes the Hamming weight leakage model (see Section 2.3.1); this is also used for evaluation (see Chapter 7). In general, the attack does not necessarily require leakage following the Hamming weight leakage model, but a similar procedure may be carried out for different models.

### One-Point Matching

To match the set of loads $L = \{l_0, \ldots, l_{m-1}\}$, where $m \in \{4, 8, 16, \ldots, n\}$ is the length of the permuted block or layer, to the set of stores $S = \{s_0, \ldots, s_{m-1}\}$, we first compute the probability for all distributions $d \in L \cup S$ with corresponding true value $v$ with hamming weight $h$ and expected value $\mu$. For $\epsilon > 0$, $I_\epsilon = ]\mu - \epsilon, \mu + \epsilon[$ and a Hamming weight $h'$, we have

$$P(h = h' \mid v \in I_\epsilon) = \frac{P(v \in I_\epsilon | h = h') \cdot P(h = h')}{P(v \in I_\epsilon)}. \tag{4.47}$$

This can be computed by using

$$P(v \in I_\epsilon) = \sum_{h''} P(v \in I_\epsilon \mid h = h'') \cdot P(h = h''). \tag{4.48}$$

and

$$P(v \in I_\epsilon \mid h = h') = P_{\mathcal{N}_\sigma(h)}(v \in I_\epsilon). \tag{4.49}$$

The former formula, Equation (4.48), can be computed using the occurrences of Hamming weights on the inputs as well as the latter formula, Equation (4.49), which follows from the assumption of the model. Then, using both Equation (4.48) and Equation (4.49) allows computing Equation (4.47).

From Equation (4.47) we may compute the probability of a true value $v$ when measuring $d$ by taking the limit $\epsilon \to 0$:

$$P(h = h' \mid \text{measured } x) = \lim_{\epsilon \to 0} P(h = h' \mid v \in I_\epsilon). \tag{4.50}$$

In practice, $\epsilon$ is simply chosen to be very small. For two measurements $l_i$ and $s_j$ we may compute the probability of belonging together as

$$\begin{aligned} p_{i,j} &= P(l_i \text{ and } s_j \text{ are measurements of the same variable}) && (4.51)\\ &= \sum_{h'} P(h_i = h' \mid \text{observing } l_i) \cdot P(h_i = h' \mid \text{observing } s_j). && (4.52) \end{aligned}$$

Using the above equations, we compute $P(h = h' \mid \text{measured } x)$ for all $x \in L \cup S$ and all Hamming weights $h'$.

From these formulas, we compute the probability of belonging together for all nodes and store them in a matrix

$$A = \begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,h-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,h-1} \\ \cdots & & & \\ p_{h-1,0} & p_{h-1,1} & \cdots & p_{h-1,h-1} \end{pmatrix}. \tag{4.53}$$

The $i$-th row of $A$ holds the probability of the $i$-th store measurement belonging to the $j$ load measurement in the $j$-th column. In other words: The entry $p_{i,j}$ is the probability of the $i$-th measurement being of the same variable as the $j$-th store measurement. Clearly, $A$ is right stochastic, i.e., the rows sum to 1, and $A$ is symmetric.

The probabilities in $A$ are not independent – if two measurements belong together with high probability this reduces the probability for other measurements to match to these two. Instead, we are looking for the likeliest permutation with respect to $A$. This means we are searching for a matrix with exactly one non-zero entry in every row in such a manner that this represents the likeliest permutation given $A$. In a sense, we simultaneously want to normalize rows and columns, i.e., achieve a fully stochastic matrix.

To achieve this, we employ the Sinkhorn-Knopp algorithm [Sin64], which consists of repeated normalization of rows and columns. First, columns are normalized, which leads to the matrix being left stochastic but losing the right stochastic property; then columns are normalized again. We repeat these steps until either a maximum number of iterations is reached or the entries in the matrix do not change more than a chosen bound.

The result of this procedure is a matrix with a single 1 per row (and column) directly representing a permutation in case of sufficiently precise measurements. In case of insufficiently precise measurements, the result explains how to mix up distributions (see Section 7.1.2); in this case, we do not get a perfect result allowing us to unshuffle the (inverse) NTT, but we may still remove entropy of the countermeasure and thereby inconsistencies in the belief propagation. We call matrices such as $A$ and the result of the above procedure *mix matrices.*

**From a single layer to multiple layers.** The above procedure allows us to find the relative permutation in-between two layers. To unshuffle the complete (inverse) NTT, we may start with the first space between two layers, that is between layers 0 and 1, compute the mix matrix, and apply it to the loads and stores of layer 1. We emphasize that the permutation needs to be applied to the stores of layer 1 (which have not been considered in the matching process) as well.

This is because we match against the first, fixed layer and re-permute the butterflies of the second layers (with index 1). In addition, the mix matrix of layers 0 and 1 needs to be applied to the mix matrices of all subsequent layers. Then, the second and the first layer are considered fixed, and the procedure is applied to layer 2 using the stores of layer 1 and the loads of layer 2. Figure 4.12 visualizes the matching and unshuffling process, and Algorithm 13 states the algorithms as given in [HSST23].

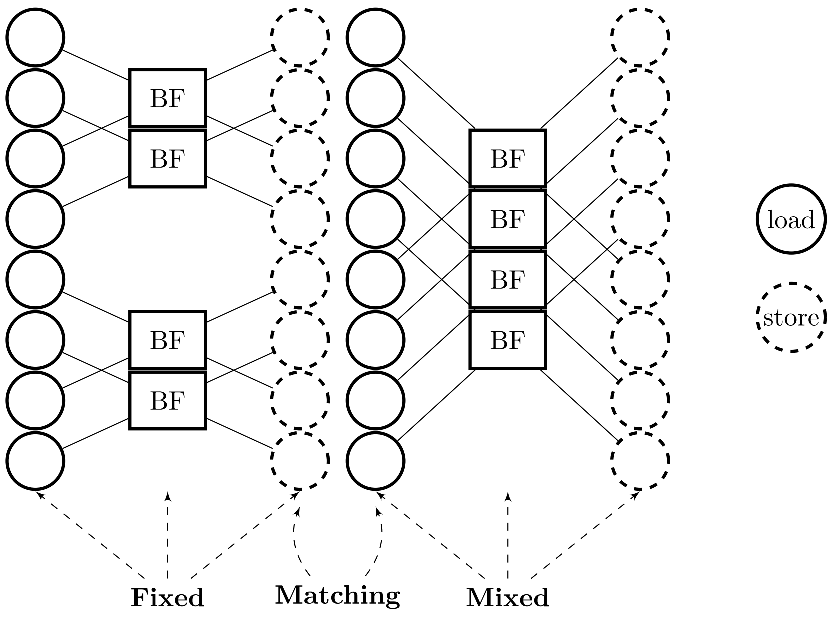


Figure 4.12: The unshuffling process shown in the attacker model as depicted in [HSST23]. The left layer is fixed while the right layer is matched against the left layer and subsequently permuted.

**The effects of zeros.** When applying our technique to the attack of [HHP+21], the zeros applied by using a chosen ciphertext may make matching either impossible (if all values in a block are zeroed out) or harder. Zeroed-out values may not be matched as at these positions, no information is available – the true values are the same in the first place. This leads to a less precise matching and negates previously achieved entropy reduction (which was introduced by the usage of a chosen ciphertext).

### Two-Point Matching

When using One-Point Matching, we ignore that not all permutations are possible. As butterflies are shuffled as a whole, nodes connected to the same butterfly have the same relative position. We may use this to improve upon the probabilities passed into Algorithm 13. Instead of using the probabilities of single measurements coming from the same variable, we consider the probability of $s_i$ belonging to $l_i$ and, simultaneously, of $s_{i+d}$ belonging to $l_{i+d}$. This means the entries of $A$ are given by

$$\begin{aligned} p_{i,j} = \; & P(l_i \text{ and } s_j \text{ are measurements of the same variable}) && (4.54)\\ & \cdot P(l_{i+d} \text{ and } s_{j+d} \text{ are measurements of the same variable}) && (4.55) \end{aligned}$$

**Algorithm 13** One-point matching as stated and depicted in [HSST23].

```
Require: $l_{i,j}, s_{i,j}$ for $i \in \{0 \dots, n-1\}, j \in \{0, \dots, \text{layers} - 1\}$
Ensure: Mix matrices $\tilde{A}_j$ for $j \in \{1, \dots, \text{layers}\}$ and mixed measurements
 1: for all $j \in \{0, \dots, \text{layers} - 1\}$ do
 2:     $A_j \leftarrow$ Compute priors
 3:     $\tilde{A}_j \leftarrow$ Sinkhorn-Knopp$(A_j)$
 4: end for
 5: for all $j \in \{1, \dots, \text{layers} - 1\}$ do
 6:     $l_j = (l_{0,j}, l_{1,j}, \dots, l_{n-1,j})$
 7:     $l_j \leftarrow \tilde{A}_j \cdot l_j$
 8:     $s_{i,j} = (s_{0,j}, s_{1,j}, \dots, s_{n-1,j})$
 9:     $s_{i,j} \leftarrow \tilde{A}_l \cdot s_{i,j}$
10:     $\tilde{A}_{j+1} \leftarrow \tilde{A}_j \cdot \tilde{A}_{j+1}$
11: end for
12: return $\tilde{A}_j$, $l_{i,j}, s_{i,j}$, for all $i, j$
```

for $i$ in the first half of the block and otherwise by subtracting $d$.

Note that this does not only improve upon the priors (used for the belief propagation) but also reduces the effect introduced by using a chosen ciphertext that causes zeros in the inverse NTT: Even without any measurement noise, the usage of One-Point Matching causes noise, because variables that do not belong together may have the same value (recall that we work on Hamming weights). In Two-Point Matching, this occurs with quadratically lower frequency due to matching at two points.

### Exact Permutation Matching

The algorithm described in Algorithm 13 causes additional noise unless it is used on very precise measurements with very few values; this happens regardless of the usage of One-Point or Two-Point Matching due to the mixing process. With very low noise, the resulting matrices may not mix but purely permute, i.e., have a single 1 per row, but in most cases, measurement distributions will be mixed. The noise introduced by this process propagates and increases throughout layers. Therefore, mixing in early layers causes more increase than in later layers. At the same time, mixing in early layers of an inverse NTT happens less often as the blocks have fewer nodes (assuming coarse in-block shuffled inverse NTT).

To avoid mixing, especially in earlier layers, we additionally introduce *exact permutation matching*. After having obtained a mix matrix as in one- or two-point matching, in exact permutation matching, we do not apply it to the measurements. Instead, we select the row with the highest entropy – this belongs to the value that is the least determined – and create a list of all values with a probability higher than a fixed threshold. We then iterate through the list of values and, at each iteration, set that row to contain a single 1 at this position; this means we manually fix a single pair of nodes to be matched to each other. With the resulting matrix we, again, perform two-point matching and return the best result. Thereby, we essentially brute-force the permutation for a single pair. We may repeat the procedure with different values either separately or recursively. The algorithm is stated in Algorithm 14 as in [HSST23].

We note that this exhaustive search approach requires a large amount of computational power. Nevertheless, it may be used on early layers and reduce the noise caused by mixing in those and all further layers. In addition, an adversary with access to more (computational) resources or leakage could improve upon the attack by applying it to later layers as well.

**Algorithm 14** Exact permutation matching as stated and depicted in [HSST23].

```
Require: Prior matrix $A = A_j$ for some layer $j$, depth level $d$, max depth $d_{\max}$, entropy threshold
    $t_e$, probability threshold $t_p$
Ensure: A list of permutations.
 1: if $d > d_{\max}$ then
 2:     return []
 3: end if
 4: result_list ← []
 5: $\tilde{A}$ ← Sinkhorn-Knopp($A$)
 6: $I$ ← Find indices of rows with entropy higher than $t_e$
 7: for all $i \in I$ do
 8:     $J$ ← Find indices of columns with probability greater $t_p$
 9:     for all $j \in J$ do
10:         $\tilde{A}'_{ij}$ ← Set row $i$ to $(\delta_{jk})_k$ where $\delta$ denotes the Kronecker delta.
11:         permutations ← MatchExact($\tilde{A}'_{ij}$, $d+1$, $d_{\max}$, $t_e$, $t_p$)
12:         result_list ← Append permutations to result_list
13:     end for
14: end for
15: result_list ← Sort result_list by likelihood and remove duplicates
16: return result_list
```

### From Block Shuffling to Full Shuffling

The performance of the above algorithms greatly depends on the number of nodes to be matched. In coarse-block shuffling, the number of nodes shuffled, and therefore the length of the permutation is greatly reduced compared to coarse-full shuffling. Therefore, our methods perform better in the case of coarse-block shuffling. We note that additional leakage, for example, caused by different twiddle factors if not protected appropriately, may allow an adversary to differentiate between blocks and therefore diminish the advantage of coarse-block shuffling. In addition, coarse-full shuffling is more expensive in terms of performance (see [RPBC20, Table 3]) because of the high number of twiddle factor loads, and requires additional protection of the load operations.

### Unshuffling the First Layer

Using the previously discussed methods, the permutation of the first layer cannot be found. Because all layers are shuffled, only relative permutations may be found by comparing layers. Therefore, we have to employ a different way to find the permutation applied to the first layer. Note that in the first layer the block size of an inverse NTT as used in Kyber is 4, i.e., only two butterflies may be permuted per block. In the case of coarse in-block shuffling, the brute-force complexity is therefore merely $2^{\frac{n'}{4}}$ where $n'$ is the number of non-zeroed nodes. We propose working with 64 non-zero values, which leads to having to run at most $2^{16}$ belief propagation instances – this is also the scenario we evaluate in Section 7.1. Incorrectly shuffled belief propagation will terminate early and with lower computational effort due to the inconsistencies caused by the first layer. In addition, we note that the convergence of sub-graphs often allows an adversary to draw conclusions about the location of an incorrectly assigned node; this is described in more detail in [HSST23].

## 4.6 Summary

The NTT is a valuable target, and previous work [PPM17, PP19] shows how it may be attacked. But these attacks either do not target the secret key `sk` – but instead the message `m` – or offer only very limited[5] noise tolerance. Moreover, when targeting a masked implementation the noise tolerance in template attacks is low regardless of the target of the attack. In addition, while masking countermeasures are circumvented, shuffling countermeasures fully prevent these attacks in their current form, and no adaptations have yet been provided.

We introduce a novel attack strategy that allows reducing the entropy during the computation of the inverse NTT by use of a chosen ciphertext. The chosen ciphertext that is submitted to a device cancels out values during the NTT. A subsequent template attack is then more noise resistant and achieves similar noise tolerance as [PP19] but targets the secret key of the KEM. We thereby achieve higher, more practical noise tolerance and provide an extended evaluation of the vulnerability of NTTs used in lattice-based schemes. Further, our analysis shows how chosen ciphertext may be used to reduce the entropy for a subsequent attack: Similar strategies could apply against different multiplication methods and on different schemes as well.

We additionally take countermeasures into account: Linear masking of the NTT is countered by applying the attack to both shares separately. To adapt to shuffling countermeasures presented in [RPBC20], we propose two different adaptations that counter the two different types of countermeasures. We show that some countermeasures can be circumvented purely in software, while others can be circumvented using an extended attacker model. Our adaptations to countermeasures likely also apply more generally and extend the currently known techniques for SASCA using belief propagation to attacks on the NTTs in a lattice-based setting.

The NTT is one of the key building blocks of Kyber and processes the secret key during the decapsulation. We show that the inverse NTT is vulnerable to side-channel analysis even in settings with increased measurement noise and with countermeasures in place. Moreover, we show how shuffling countermeasures may be circumvented in SASCA on lattice-based post-quantum schemes. Thus, our findings impact the necessary security considerations regarding devices that run the new NIST standard for post-quantum key exchanges and call for more comprehensive countermeasures.

[5]Compared to when targeting the message or using our technique; c.f. Section 3.2.2 and Section 7.1.

# Chapter 5

# Fault-Enabled Chosen-Ciphertext Attacks

Current schemes based on the Learning with Errors (LWE) problem need to recover the message from a noisy version. During the encryption routine, bits are mapped to coefficients of a vector or a (vector of) polynomials; and during the decryption routine, a noisy version of these coefficients may be retrieved using the secret key. An error correction then recovers a bit message from these noisy coefficients, and if the noise is sufficiently small, this results in the original message. This routine has previously been targeted using a chosen ciphertext or a fault (see Section 3.2).

A Fujisaki-Okamoto (FO)-transform turns an IND-CPA secure Public-Key Encryption (PKE) into an IND-CCA2 secure Key Encapsulation Mechanism (KEM). Thereby, chosen-ciphertext attacks are prevented in the IND-CCA2 model. In particular, the transform prevents an adversary from observing a decryption failure caused by a chosen-ciphertext attack containing an additional error term. Previous attacks, e.g. [GJN20, PP21, BDH+21b], exploited implementation vulnerabilities to launch decryption failure attacks on Kyber and FrodoKEM. Those attacks require an incorrectly implemented comparison operation [GJN20, BDH+21b] or a reliable fault targeting a specific operation in an inappropriately protected decoder [PP21]. While these have been validated in practice on a physical device, it is also known how they can be mitigated, at least in principle; in fact, the works proposing the attacks already state countermeasures.

In [HPP21], we state a fault attack that does not rely on a single attack target and allows for an unreliable fault. As noted in [DHP+22], our attack can be seen as similar to a safe-error attack [YJ00] – we introduce an error that is corrected in a later stage. A ciphertext potentially causing a decryption failure is submitted to the device under attack; the device decrypts the ciphertext, and while it is stored in memory as part of the FO-transform, we reverse the error using a fault. This causes the device to decrypt a faulty ciphertext while comparing the re-encrypted ciphertext against the honestly generated ciphertext. We thereby turn the FO-transform, which is in place to protect against chosen-ciphertext attacks, into a decryption failure oracle.

The attack can be applied at a wide variety of points in execution time; this includes – but is not limited to – routines that are similar to targets of previous attacks. Therefore, the attack surface is enlarged, and single countermeasures do not mitigate the attack. Another advantage of our safe error approach is that a decapsulation success cannot happen if the fault fails. This means the information from decapsulation successes is almost certainly[1] correct. We explain and evaluate our attack strategy on Kyber; however, we note that it applies more generally to LWE based schemes using an FO-transform.

[1]To be precise: An incorrect inequality from a decapsulation success is as likely as finding a SHA3 collision.

## 5.1 Attacker Model

The attacker model requires the adversary to be able to create valid ciphertexts for a device using the public key `pk` belonging to a secret key `sk` similar to the IND-CPA model. In addition, the attacker needs to be able to fault either $\texttt{ct}'$ or `ct` to remove or add, respectively, the error introduced by the chosen ciphertext. The fault is applied on a compressed ciphertext. As the ciphertext is under the control of the attacker, and the bit(s) to be faulted can be chosen freely, it is sufficient to be able to fault

- either a single bit as either set, reset, or flip, or
- multiple bits in an admissible pattern,

of the $d_v = 4$ bit ciphertext component $v$. The section representing the coefficient to be faulted is 4 or 5 bit long (4 for Kyber512 and Kyber768, 5 for Kyber1024) and every pattern changing a 4-bit (5-bit) integer by adding 4 (8) is admissible.

Regarding the fault model, it should be noted that the adversary may choose whether to set, reset, or flip a bit, and the fault can be applied over almost the complete execution time. Moreover, observing the outcome of the comparison of the FO-transform may help with profiling and finding the fault location. Under these conditions, single-bit faults can be considered a realistic threat (depending on the targeted device, the adversary's capabilities, and countermeasures in place) assuming that the adversary has good knowledge and understanding of the device under attack [RSDT13, OGM17].

## 5.2 Attack Strategy

Our attack strategy starts off similar to the general attack description given in Section 3.2.1; we here repeat all steps for the convenience of the reader.

The first part of the attack is performed offline: The adversary honestly generates a ciphertext `ct` for a message `m` using the public key of the device under attack `pk`, which belongs to a secret key `sk`; the latter is the target of the attack. To potentially cause a decryption failure, we add $E$ to one coefficient of the ciphertext creating a manipulated ciphertext $\tilde{\texttt{ct}}$. The targeted coefficient does not matter for the general description of the attack; we conveniently choose the first coefficient. If the ciphertext matches the required properties described in the next section, it is submitted to the device under attack.

The device, as part of the decapsulation routine, first stores the ciphertext to memory and then calls the decryption routine[2] on the ciphertext. The decryption routine obtains a message $\texttt{m}'$, which allows deriving randomness that is used to re-encrypt $\texttt{m}'$ to $\tilde{\texttt{ct}}$. While the decryption and re-encryption are carried out, we fault the ciphertext stored in memory, $\tilde{\texttt{ct}}$, creating the faulted ciphertext $\overline{\texttt{ct}}$; if the fault succeeds, then $\overline{\texttt{ct}} = \texttt{ct}$. Finally, the device compares $\texttt{ct}'$ to $\overline{\texttt{ct}}$ and a decryption failure occurs if $\texttt{ct}' \neq \overline{\texttt{ct}}$; otherwise the correct shared secret is returned[3]. Observing whether a decryption failure happens allows deriving an inequality about the error term (see Section 3.2.1).

Summarizing, to obtain a single inequality, the adversary carries out the following steps:

1. Honestly generate ciphertext `ct` for message `m`.
2. Manipulate ciphertext `ct` to obtain $\tilde{\texttt{ct}}$ by adding error term to a coefficient of $v$ as described in the next section.

[2]Note the difference between decapsulation and decryption routine.
[3]With overwhelming probability

3. Submit ciphertext $\tilde{\texttt{ct}}$ to the device under attack.
4. Fault $\tilde{\texttt{ct}}$ stored in memory back to ensure $\overline{\texttt{ct}} = \texttt{ct}$.
5. Observe whether a decapsulation error happens.
6. Derive inequality about the error term as described in Section 3.2.1.

The procedure is repeated until a sufficient number of inequalities is obtained. We then solve for the secret key using a belief propagation similar to the one used in Low-Density Parity Check (LDPC) codes [Gal62]. The attack strategy (with a key of length 4) is depicted in Figure 5.1.

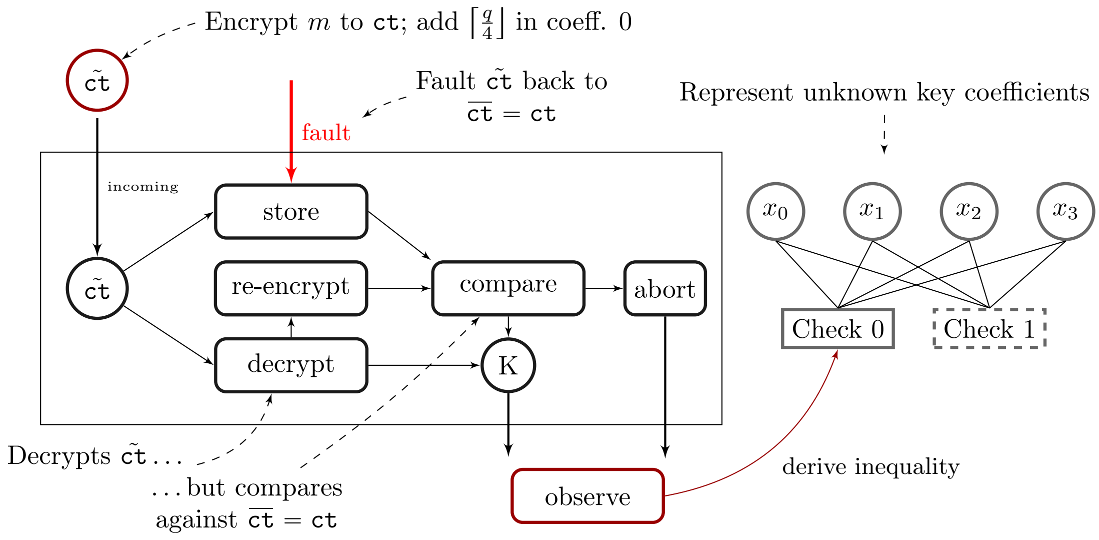

Encrypt $m$ to $\texttt{ct}$; add $\lceil \frac{q}{4} \rfloor$ in coeff. 0
$\tilde{\texttt{ct}}$
fault
Fault $\tilde{\texttt{ct}}$ back to $\overline{\texttt{ct}} = \texttt{ct}$
Represent unknown key coefficients
incoming
store
$\tilde{\texttt{ct}}$
re-encrypt
compare
abort
decrypt
K
$x_0$
$x_1$
$x_2$
$x_3$
Check 0
Check 1
Decrypts $\tilde{\texttt{ct}}$ . . .
. . . but compares against $\overline{\texttt{ct}} = \texttt{ct}$
observe
derive inequality

Figure 5.1: Attack strategy of [HPP21] targeting a secret key with 4 coefficients (adapted from [Her23b]).

## 5.3 Construction of the Ciphertext

The ciphertext is constructed by first honestly generating a ciphertext $\texttt{ct}$. This uncompressed ciphertext is of the form $(v, \mathbf{u})$, and the device under attack will, after it has been submitted, compute

$$\mathbf{m}_{\text{poly}} = v - \mathbf{s}\mathbf{u}^\top \tag{5.1}$$

during the decryption routine. From this, subsequently, the message $\mathtt{m}'$ is obtained by removing the noise using the decoding routine, which is realized as compression in Kyber. The noise on the coefficients on $\mathbf{m}_{\text{poly}}$ is given by the polynomial

$$\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta\mathbf{u})^\top + e_2 + \Delta v \tag{5.2}$$

and if any term of this polynomial is larger than $\lceil \frac{q}{4} \rfloor$, a decryption failure occurs. As all terms of the error polynomial are small, they may be interpreted as integers and linear inequalities can be seen as holding over $\mathbb{Z}$.

## 5.3.1 Introducing an Error

Equation (5.1) is clearly linear in $v$, and $v$ is under the control of the adversary as it is part of the ciphertext. We can therefore introduce an additional error to the error polynomial (in one coefficient) and thereby cause a decryption failure if the sum of the coefficient's error term and the additional error is larger than $\lceil \frac{q}{4} \rfloor$. This allows deriving an inequality, which is linear in the secret key $\mathbf{x} = (\mathbf{e}, \mathbf{s}) \in \mathbb{Z}^{2kn}$. We see the inequalities coefficients as a flattened vector over $\mathbb{Z}$ (c.f. Section 3.2.1). Note that the comment about compression in Kyber from Section 3.2.1 applies here as well and only multiples of $\lceil \frac{q}{2^{d_v}} \rfloor$ may be used as error. Using $E = \lceil \frac{q}{4} \rfloor$ causes the error distribution to be centered around $E = \lceil \frac{q}{4} \rfloor$ instead of 0, which means a decryption failure happens if the original error term was greater than (or equal to) zero. To be precise, if the encoded message bit was 0, a decryption failure happens if

$$(\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta\mathbf{u})^\top + e_2 + \Delta v)_0 > 0, \tag{5.3}$$

and if the encoded message bit was 1, a decryption failure happens if

$$(\mathbf{e}\mathbf{r}^\top - \mathbf{s}(\mathbf{e_1} + \Delta\mathbf{u})^\top + e_2 + \Delta v)_0 \geqslant 0. \tag{5.4}$$

Thus, we create the ciphertext by sampling an honest uncompressed ciphertext $(v, \mathbf{u})$ and calculating $\tilde{\mathtt{ct}}$ as

$$(\mathrm{Compress}(v', d_v), \mathrm{Compress}(\mathbf{u}, d_u)) \tag{5.5}$$

where

$$v' = (v_0 + E, v_1, \ldots, v_{n-1}), \tag{5.6}$$

with $E = \lceil \frac{q}{4} \rfloor$. The manipulated ciphertext $\tilde{\mathtt{ct}}$ is the compressed version of $(v', \mathbf{u})$. The ciphertext is similar to the ciphertext used in [GJN20, BDH+21b], but we only use a single error value $E$. Using multiple values for $E$ to obtain a more restricting inequality, which then holds more information, is of less use due to the compression used in Kyber.

## 5.3.2 Constraints on the Ciphertext

We have two requirements on the ciphertext: The ciphertext should differ by specified bit-distance or pattern and, depending on the fault model, the faulted bits should be either 0 or 1 or do not matter. Additionally, we filter the ciphertext using the condition described by [PP21].

To achieve these properties, we first create a ciphertext, check for the conditions, and re-sample if they are not fulfilled. As the ciphertext generation is done offline and the conditions can also be checked before submitting the ciphertext to the device, these conditions introduce no more than a very small penalty in the (fully parallelizable) offline phase of the attack.

### Ciphertext Filtering

We rely on the technique of ciphertext filtering similar to the one proposed by [PP21]. This technique maximizes the amount of information contained in an inequality by controlling the constant term of the inequality, i.e., by bounding

$$|e_2 + \Delta v| \leqslant c \tag{5.7}$$

for some $c$ which was chosen to be 10 in [PP21]. We analyze the impact of ciphertext filtering in Section 6.2.3. Note that the main impact on the information contained in an inequality comes from $\Delta v$ with an absolute value maximum of $\lceil \frac{q}{2^{d_v+1}} \rfloor = 104$ (for Kyber512, Kyber768; 52 for Kyber1024) [ABD+21b] as $e_2 \in \{-2, -1, 0, 1, 2\}$ is comparably small due to being sampled from a binomial distribution with $\eta = 2$.

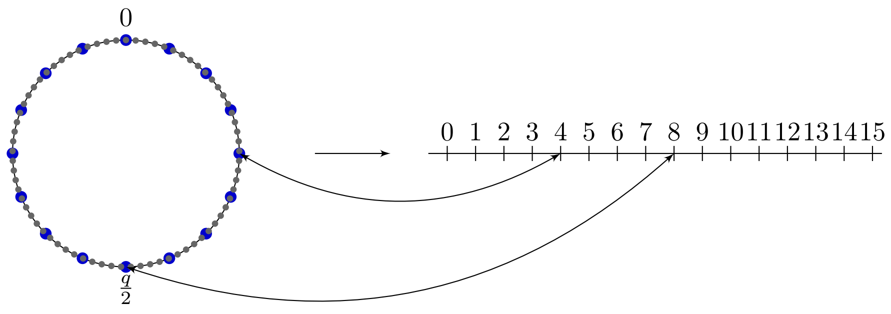


Figure 5.2: Adding $\left\lceil \frac{q}{4} \right\rfloor$ to a coefficient in uncompressed form is the same as adding 4 in compressed form when using a compression factor of $d = 4$; visualized with $d = 4$ and $q = 23$.

**Required Bit Difference**

The adversary is required to remove the additionally introduced error term using a fault. If $\tilde{\texttt{ct}}$ denotes the manipulated ciphertext and `ct` is the honestly generated ciphertext, the Exclusive-Or (XOR) determines the required fault.

The compression function in Kyber is given by

$$\text{Compress}(x) = \left\lceil x \frac{2^d}{q} \right\rfloor \bmod 2^d \tag{5.8}$$

applied coefficient-wise. Therefore, adding $\left\lceil \frac{q}{2^{d'}} \right\rfloor$ in uncompressed form results in the addition of

$$\left\lceil \left\lceil \frac{q}{2^{d'}} \right\rfloor \cdot \frac{2^d}{q} \right\rfloor = 2^{d'-d} \bmod 2^d \tag{5.9}$$

to the compressed ciphertext interpreted as (overflowing) 4-bit integer.

The relevant section of the compressed coefficient of $v$ is 4 bit long as here $d_v = 4$. Interpreting the compressed ciphertext component's coefficient as 4 bit integer, adding $\left\lceil \frac{q}{4} \right\rfloor$ in uncompressed form is adding (with overflow) 4 in compressed form; this is shown in Figure 5.2. In Kyber1024, the coefficient is compressed to $d_v = 5$ bits, and adding $\left\lceil \frac{q}{4} \right\rfloor$ corresponds to adding 8.

Note that in practice, the adversary may compute an honestly generated ciphertext, apply the error in uncompressed form, and then compare the compressed ciphertext to see if it matches the required fault pattern. As the ciphertext generation is performed offline, this is a valid strategy. The ciphertext generation for an adversary requiring a one-bit fault is given in Algorithm 15.

## 5.4 Recovery of the Secret Key

Just as previous attacks [PP21, BDH+21b], our attack strategy allows recovering inequalities that are linear in the secret key. To solve such inequalities, several methods have been suggested as summarized in Section 3.3. In particular, the methods of Pessl and Prokop [PP21] and Delvaux [Del22] have been used in practice in similar attacks. We propose a belief-propagation-based approach in [HPP21] improving upon [PP21]. This was improved on in [Del22] by adding error resistance, i.e., by being able to solve for the secret key in the presence of incorrect inequalities while requiring more inequalities than our previous method. Subsequently, we propose an improved

**Algorithm 15** Generating a ciphertext which may be faulted by a single-bit fault with ciphertext filtering. Encapsulate′ is the encapsulation routine of Kyber but with setting/output of $e_2$ and $v$ in uncompressed form.

**Require:** Public key `pk`, ciphertext filtering value $c$.
**Ensure:** Ciphertext `ct` and $\tilde{\texttt{ct}}$ which differ by one bit.

```
1: repeat
2:     $\texttt{ct}, e_2, v \leftarrow$ Encapsulate'($\texttt{pk}$, _, _)
3:     if $|e_2 + v| \leqslant c$ then
4:         continue
5:     end if
6:     $v' \leftarrow (v_0 + \lceil \frac{q}{4} \rfloor, v_1, \ldots, v_{n-1})$
7:     $\tilde{\texttt{ct}}$, _, _ $\leftarrow$ Encapsulate'($\texttt{pk}, e_2, v'$)
8: until HammingDifference($\texttt{ct}, \tilde{\texttt{ct}}$) == 1 //Replace by required conditions
9: return $\texttt{ct}$, $\tilde{\texttt{ct}}$
```

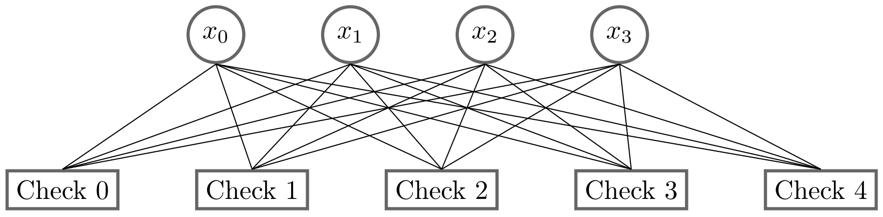


Figure 5.3: A graph used to represent an inequality with 4 variables and 5 inequalities (adapted from [HPP21]).

version in [HMS+23] (see Chapter 6) adding error resistance to the method of [HPP21] while lowering the number of inequalities compared to [Del22]. In [HMS+23], we additionally explain how to integrate the belief propagation output obtained by a method as described here into a lattice problem.

### 5.4.1 Recovery using Belief Propagation

The work of [PP21] uses a Bayesian update method which is related to updating in belief propagation works. Instead of using an ad-hoc updating method, we employ the technique used to solve LDPC codes (introduced in [Gal62]; see e.g. [Mac03] for a comprehensive introduction): A belief propagation graph consisting of *variable nodes* and *check nodes* is used to represent inequalities (equations in LDPC-codes). The variable nodes model an unknown variable while the check nodes model an inequality; an example of a low-dimensional graph is shown in Figure 5.3. An inequality is connected to every variable with a non-zero coefficient; in our case, this means the graph is fully connected/full density (or almost fully connected) in contrast to LDPC-codes which are, by definition, of low density. To keep the recovery practical, we use several optimizations in computations carried out in both variable nodes and check nodes.

**Notation.** In the remainder of the section, we assume to be given a system of inequalities

$$\mathbf{xM} \leqslant \mathbf{b} \tag{5.10}$$

where $\mathbf{x} = (\mathbf{e}, \mathbf{s})$ and the coefficients of $\mathbf{M}$ are small. We denote the $j-th$ inequality by $\mathbf{m}_j = \mathbf{m}_{\cdot,j}$ (as it is the $j$-th column of $\mathbf{M}$) and the $i$-th coefficient of $\mathbf{m}_j$ to be $m_{i,j}$ to be in line with standard matrix notation. The amount of inequalities is denoted by $m$ and $\mathbf{M} \in \mathbb{Z}^{m\times 2kn}$.

### Messages

A message in belief propagation run represents the belief about the true value of a coefficient. A belief $\mu$ is given by a discrete probability distribution on $\{-\eta, \dots, 0, \dots, \eta\}$, i.e., by a function

$$\mu = \{x \mapsto p_x \mid x \in \{-\eta, \dots, 0, \dots, \eta\}\} \tag{5.11}$$

such that

$$\sum_x \mu(x) = 1 \tag{5.12}$$

but in practice the condition of Equation (5.12) is relaxed; before obtaining probabilities the messages then have to be normalized. As $\eta \in \{2, 3\}$ in Kyber, depending on the security level, messages are fairly small and may be stored as floating point arrays of size 5 or 7. The sizes per security level are shown in Table 5.1.

Table 5.1: Sizes in byte of a single message and $2kn$ messages per Kyber security level when using 64-bit floats. Note that in practice, more than $2kn$ messages have to be stored simultaneously, especially when using a multithreaded approach.

| Kyber Security Level | Kyber512 | Kyber768 | Kyber1024 |
|---|---|---|---|
| $\eta$ | 3 | 2 | 2 |
| Message Size | 56 | 40 | 40 |
| Total Message Sizes | 57.344 | 61.440 | 81.920 |

### Variable Nodes

A variable node represents an unknown variable and aims to compute the same value as in the generic description of belief propagation given in Section 2.4. The variable nodes are initialized with Kyber's error distribution as priors, i.e., with binomial distributions centered around 0 with $\eta \in \{2, 3\}$, as the coefficients represented by variable nodes were initially sampled from this distribution (in an honest key generation).

Initially, the priors, $\mu_{\text{prior}}$, are sent out to the factor nodes; after that, in every other step, every variable node receives messages from all check nodes (ignoring that some connections are unnecessary). In step $t \geqslant 1$ a variable node with index $i$ and arriving messages $\mu_{j,i,t}$, $j \in \{0, \dots, 2kn - 1\}$, the variable node computes the messages

$$\mu_{i,j,t+1} = \mu_{\text{prior}} \prod_{i \neq j' = 0}^{2kn} \mu_{j,i,t} = \left\{ x \mapsto \mu_{\text{prior}}(x) \prod_{j' \neq i} \mu_{j,i,t}(x) \right\}. \tag{5.13}$$

Due to the full density, a straightforward computation leads to high run-times – after all $2kn \cdot (2kn - 1) \cdot m$ products have to be computed with $2kn \in \{512, 768, 1024\}$ and `m`, the number of inequalities, commonly being a 4- to 5- digit number. Using the two-binary-tree strategy of [PP21] is valid, but an even easier and comparably efficient algorithm consists of a forward and backward multiplication: An array is first initialized to contain the neutral messages, i.e., the message mapping everything to 1. Then, forward and backward accumulators are initialized to the prior. A loop over $j \in \{0, \dots m\}$ multiplies the message at $j$ with the forward accumulator, the message at $n-1-j$ with the backward accumulator, and the accumulators with the $\mu_{i,j,t}$ and $\mu_{i,n-1-j,t}$, respectively. After the loop finishes, the array contains all $\mu_{i,j,t+1}$ as specified in Equation (5.13). These messages are sent to the variable nodes.

**Factor Nodes**

The messages sent from the variable nodes are received at check nodes. A check node with index $j$ represents an inequality

$$\sum_{i=0}^{2kn} m_{i,j} x_i \leqslant b_j. \tag{5.14}$$

The messages arriving from the variable nodes represent beliefs over the $x_i$, and, to update the belief for $x_i$, the factor nodes aim to compute

$$P(x_i = y \mid \sum_{i'} m_{i;,j} x_{i'} \leqslant b_j). \tag{5.15}$$

We may compute this as

$$\begin{aligned} & P(x_i = y \mid \sum_{i'} m_{i',j} x_{i'} \leqslant b_j,\ x_i' \sim \mu_{i',j,t}) && (5.16)\\ = \ & P(x_i = y \mid m_{i,j} y + \sum_{i' \neq i} m_{i',j} x_{i'} \leqslant b_j,\ x_i' \sim \mu_{i',j,t}) && (5.17)\\ = \ & P(\sum_{i' \neq i} m_{i',j} x_{i'} \leqslant b_j - m_{i,j} y,\ x_i' \sim \mu_{i',j,t}). && (5.18) \end{aligned}$$

To compute Equation (5.18) and thereby Equation (5.18), we compute the *leave-one-out distributions* for all coefficients. The leave-one-out distribution for the $i$-th coefficient is the distributions of

$$s_i = \sum_{i' \neq i} m_{i,j} x_{i'}. \tag{5.19}$$

To compute $s_i$, all the convolutions of all but one $\mu_{i,j}$, representing the beliefs in $x_i$, have to be computed. We originally used the approach two-tree approach of [PP21], but note that are simplified algorithm could be advantageous in terms of memory consumption and copies of large arrays. This approach two-tree approach is described in Section 3.3.3 and allows computing the leave-one-out distributions which are contained in the leaves of the downtree at the end of the computation. It may be replaced by a forward and backward accumulator which is slightly less computationally expensive. The computations taking place in the check node are stated in Algorithm 16.

### 5.4.2 Final Recovery

The work of [PP21] already notes that not all coefficients need to be recovered. Given half of the coefficients of $\mathbf{x} = (\mathbf{e}, \mathbf{s})$, the public key equation

$$\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b} \tag{5.20}$$

can be turned into a $kn$-dimensional (undisturbed) system of linear equations with $kn$ unknowns. Therefore, recovering $kn$ coefficients is sufficient to recover the full key $\mathbf{x}$.

As before in Chapter 4, we define a coefficient to be *correct* if its likeliest value is the true value of the coefficient. A coefficient is called *recovered* if there is some order (known to the adversary)

**Algorithm 16** Computations taking place in a check node for an inequality with coefficients $a_i$ as stated in [HPP21].

```
Require: Incoming messages $m_0, \dots, m_{2n-1}$
Ensure: Outgoing messages $m'_0, \dots, m'_{2n-1}$
 1: for all $i \in \{0, \dots, 2n-1\}$ do
 2:     for all $v \in V$ do
 3:         $m_i[a_i \cdot v]' \leftarrow m_i[x]$                ▷ Distribution of $a_i x_i$
 4:     end for
 5:     $\widehat{m}'_i \leftarrow \mathrm{FFT}(mm_i)$
 6: end for
 7: downtree ← BinaryTrees.compute($\widehat{m}'_0, \dots, \widehat{m}'_{2n-1}$)    ▷ Multiply leaving one out
 8: for all $i \in \{0, \dots, 2n-1\}$ do
 9:     $\widehat{m}''_i \leftarrow$ downtree.leaf($i$)              ▷ downtree.leaf($i$) $= \prod_{j \neq i} \widehat{m}'_j$
10:     $m'' \leftarrow \mathrm{FFT}^{-1}(\widehat{m}''_i)$              ▷ Holds distribution of $\sum_{j \neq i} a_j x_j$
11:     for all $v \in V$ do
12:         $m'_i[v] \leftarrow m''_i$.sum_lesseq_than($b - v$)    ▷ $m'_i[v] = P(\sum_{j \neq i} a_j x_j \leqslant b - v)$
13:     end for
14: end for
15: return $m'_0, \dots, m'_{2n-1}$
```

of the probability distributions in which at least all coefficients up to and including this coefficient are correct. Even without an attack, just taking the priors into account, a lot of coefficients will be correct – but not recovered.

In every step, we apply our orders and check whether the first $n$ coefficients are recovered by solving the public key equation for the likeliest value. The latter can be done in linear time, and we may therefore carry out this step after every belief propagation step. We propose four orders to check for recovered coefficients:

- Entropy per node.
- Change in Entropy per node.
- Min-Entropy[4].
- Entropy and min-entropy.

The recovery algorithm is stated in Algorithm 17.

## 5.5 Impact of Countermeasures

The most general countermeasure preventing decryption failure attacks, including previous attacks as well as ours, is to shut down or regenerate the secret key after a certain number of decryption failures that is lower than required to solve for the secret key. But as already noted in [PP21], this opens the possibility for a Denial of Service (DoS) attack on the device.

Several other countermeasures could mitigate or even prevent a concrete instantiation of our attack. We did not specify the exact point of the physical part of our attack and leave this open as it depends on the vulnerabilities of an implementation or device as well as the possibilities

[4]The logarithm of the probability of the likeliest value.

**Algorithm 17** Recovering the secret value using belief propagation.

```
Require: Belief propagation graph g returning probability distributions $D_i$ for key coefficients $x_i$, step size $s$, orders $\wr_j$ for $j \in \{0, \ldots, l\}$.
Ensure: True and the secret key if the secret key has been found; false otherwise.
repeat
    g . propagate($s$)
    $\mathbf{D} \leftarrow$ g . get_results()
    for all $j \in \{0, \ldots, l\}$ do
        $L \leftarrow$ Order $D_i$ according to $\wr_j$
        for all $i \in \{0, \ldots, kn\}$ do
            $x'_{L(i)} \leftarrow$ Likeliest value of $D_{L(i)}$
            Solve $\mathbf{sA}^\top + \mathbf{e} = \mathbf{b}$ for remaining $kn$ values.
            if Found solution $\mathbf{x}'$ then
                return true, $\mathbf{x}'$
            end if
        end for
    end for
until Abort conditions of g reached
return false
```

of an adversary. Therefore, we may not provide an in-depth analysis of the effect of specific countermeasures; instead, we give an overview of how our attack is impacted by a class of countermeasures.

In general, if a countermeasure prevents the fault from succeeding in all but every $l$-th fault, we require $2l$ times more faults, but the attack is not prevented. This straightforward adaptation to countermeasures is achieved by disregarding decapsulation failures and only working with inequalities resulting from decapsulation successes. As a decapsulation success may not occur with an incorrectly applied fault (be it an inefficient fault or a fault targeting an incorrect value), inequalities obtained from decryption successes are always correct. If the correctness probability for inequalities from decapsulation failures is greater than about 0.6, including these inequalities gives an additional benefit (see Chapter 6 and Section 7.2.3).

### 5.5.1 Shuffling Countermeasures

While the attack of [PP21] is prevented by shuffling the error correction[5], our attack is not affected. This is because we do not target the error correction in the first place but either the stored or the recomputed ciphertext. A potential point of attack against the re-computed ciphertext may be the compression method of the encryption routine during the re-encryption step. In the case of shuffled compression, the difficulty of manipulating the re-computed ciphertext is increased. We may still target the re-encrypted ciphertext, but for $l$ shuffling positions, we require $2l$ times more faults. If we target the originally submitted ciphertext, our attack is not mitigated at all. This means, that shuffling the compression together with additional appropriate countermeasures to ensure the integrity of the recomputed ciphertext, may mitigate the attack when targeting the recomputed ciphertext, but the main attack location is still vulnerable.

[5]Called decoder in the work of [PP21].

### 5.5.2 Redundancy

Storing the ciphertext in a redundant form, differing from the original ciphertext, prevents an adversary form targeting the stored ciphertext. Instead, they are required to be able to fault the re-encrypted ciphertext or fault the comparison itself. A countermeasure could, for example, be the following, also shown in Figure 5.4:

1. Hash the incoming ciphertext before calling the decryption routine.
2. Store hash and ciphertext while performing the re-encryption.
3. Hash re-encrypted ciphertext in the same manner as the submitted cihpertext.
4. Compare ciphertexts and hashes instead of only ciphertexts.

As the hash values will not differ by a single bit and are of much longer length, faulting the hashes is very likely not an option (assuming the adversary may change arbitrarily long values to their liking, the device under attack may be targeted in numerous other ways as well). Therefore, an adversary is required to either also fault the comparison of the hashes, target the re-computed ciphertext, or fault the ciphertext before the hash is performed first. Note that the attack is not prevented, but the attack surface is greatly decreased.

Redundancy in combination with shuffling countermeasures and additional protection of the recomputed ciphertext, a secured comparison, and additional countermeasures such as randomized memory layouts likely prevent our attack in threat models that exclude state-level adversaries. In this case, the adversary needs to target the submitted ciphertext before the computation of the hash takes place.

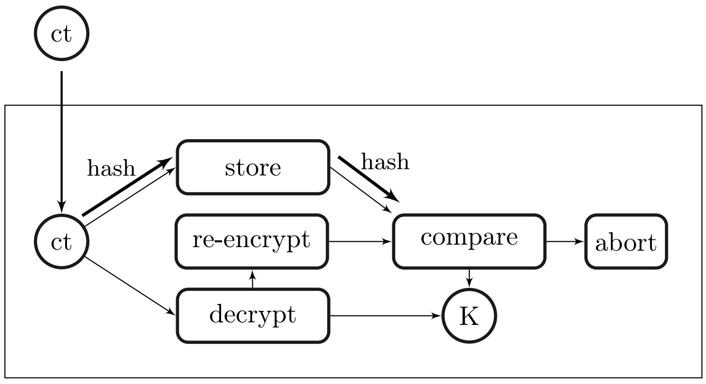


Figure 5.4: A countermeasure adding redundancy and thereby preventing the fault on the stored ciphertext as proposed in [HPP21] (figure adapted from [Her23b]). An additionally computed hash (in bold) is stored and later compared.

Note that [PP21] discuss another countermeasures: A device may shut down or regenerate the key pair after having observed a certain number of decryption failures. They note that this countermeasure causes devices to be vulnerable DoS attacks. If this countermeasure is designed without taking our results into account, our attack may circumvent it, due to the reduced required number of inequalities. Our results in Chapter 6 explain how to deploy this countermeasure to achieve the required security level.

## 5.6 Summary

In lattice-based cryptography, the error correction has been a target in classical as well as in side-channel and fault attacks. In the case of Kyber, the work of [BDH+21b] uses side-channel analysis combined with a chosen-ciphertext, and the work of [PP21] uses a fault to potentially cause a decryption failure and obtain information from observing the outcome. These have been validated in practice and can recover the secret key from a physical device, but the side-channel analysis presented in [BDH+21b] is mitigated by appropriate protection of the comparison – in fact, a main contribution of [BDH+21b] –, and the fault attack of [PP21] requires a reliable fault and a specific point of attack that is insufficiently protected.

We introduce a novel attack strategy: Previous work introduced an error using a chosen ciphertext and then passively observed the decryption result [GJN20, BDH+21b] or used a fault to introduce an error and then observed the decapsulation result [PP21]. Instead, we use a chosen ciphertext to introduce an error (and potentially cause a decryption failure) and a fault to correct the error during storage of the ciphertext or after the re-encryption has been carried out. Thereby, we may target multiple locations over a long execution time, only need to target public data and routines, do not require a reliable fault, circumvent standard countermeasures that prevented previous attacks, and achieve a larger attack surface.

In addition, we explain how belief propagation may be used to recover the secret key from decryption failure information. This technique requires less information to recover the secret key. In comparison to previous attacks, the adversary has to apply fewer faults as they may obtain the secret key from less information. Moreover, countermeasures that shut down the device after a certain number of decryption failures (c.f. [PP21]) could become ineffective. Furthermore, our approach outperforms previous approaches in terms of required computational power and can be carried out on a normal laptop as opposed to requiring large amounts of RAM. In total, our technique relaxes the attacker model, reduces the costs of the attack, and thereby increases the threat the attack poses. We further improve upon this technique in the following chapter by combining belief propagation with an algebraic approach. Evaluations of our attack, in particular the required number of faults to fully recover the key, are given in Section 7.2.

The FO-transform and the error correction are major building blocks of several lattice-based schemes. We show that these components may be targeted with an unreliable fault and in the presence of several standard countermeasures. Moreover, we show that public data has to be protected with countermeasures similar to the measures taken to secure secret data. Thus, our findings impact the security considerations of devices running the new National Institute of Standards and Technology (NIST) standard for post-quantum key exchanges in several aspects: Public data has to be considered vulnerable in the targeted class of schemes, the amount of required information for full key recovery is lower than previously believed, the attack surface is larger than previously known, and devices running such schemes are vulnerable to attacks on the error correction even when only an unreliable fault can be applied.

Chapter 6

# Security Estimates for Error-Tolerant Key Recovery

Implementation attacks commonly require the adversary to recover the key from noisy data. Veyrat-Charvillon [VGS14] introduced the approach to interpret the recovery task as a decoding problem and use coding-theoretic algorithms. For side-channel attacks, they coined the term Soft Analytical Side-Channel Attack (SASCA), and since then, the method has been applied more generally in implementation attacks (e.g. in [PP21, HPP21]). Attack strategies presented in this thesis [HHP+21, HSST23, HPP21] as well as several other attacks targeting post- and pre-quantum cryptography, e.g. [KPP20], utilize belief propagation or a variant thereof.

Many of these attacks rely on the measurement noise level being low enough or an adversary applying a sufficient number of faults. If these conditions are not satisfied, the decoding problem may not be solved and the belief propagation does not converge against probability distributions having the true coefficient as likeliest value. The attacks of, for example, [HHP+21, HSST23, HPP21], abort if the belief propagation has not recovered the secret key after some abort criteria, e.g. a specified number of iterations, has been reached. The public key equation is merely used for testing a potential solution for correctness.

Due to the “all-or-nothing” approach in these attacks, no security estimates are available; this means, that if an adversary can retrieve information that is not sufficient for full key recovery, the remaining security of the instance of the scheme is yet undetermined. In edge cases, coefficients can be assumed to be fully recovered, and estimates may be obtained from a primal attack by taking them into account or by using the more sophisticated framework of Dachman-Soled et al. [DDGR20]. But in many cases, neither can coefficients be directly seen as recovered nor are many coefficients recovered in the first place. As the belief propagation output does not fall into any of the cases covered by [DDGR20], to the best of our knowledge, no method to combine belief propagation with an algebraic attack, e.g. a primal attack, is known. Therefore, these attacks perform suboptimal as available information, i.e., the lattice information coming from the public key equation, is not taken into account. Moreover, the lack of security estimates leads to uncertainty regarding countermeasures which, for example, increase the measurement noise or restrict the number of decryption failures. Even if an adversary may not be able to recover the secret key directly, they could have lowered the bit security enough to launch a successful algebraic attack and then still recover the full secret key.

In [HMS+23], we present a key recovery method for decryption failure; this information occurs in, e.g., [BDH+21b, PP21, HPP21, DHP+22, Del22]. Note that the decryption failure information that occurs in these attacks differs from the information obtained from failures that occur without manipulation or in failure boosting attacks [DVV19, DB22] or failure boosting attacks that manipulate the key generation [FKK+22]. Our recovery method improves upon the work of [HPP21] by achieving error-resistance and combining an approach based on Bayesian updating with lattice reduction. In particular, we give a method to combine a belief-propagation-based approach with a lattice-based recovery. In the first step, we adapt the belief propagation approach from [HPP21] to achieve error resistance. We then integrate belief propagation output into a lattice problem using the public key equation.

While we first and foremost target decryption failure information – due to its high relevance – our method applies more generally. We state a technique to adjust belief propagation to account for incorrect information and, moreover, explain how to combine belief propagation with lattice reduction. Both techniques are highly relevant as belief propagation is used in a wide variety of side-channel attacks, and lattice reduction is used for attacks and security estimates on lattice-based cryptography.

**Notation.** As in the previous chapter, we denote the matrix of inequalities by $\mathbf{M}$ with entries $m_{i,j}$ where $j$ belongs to the $j$-th inequality and $i$ is the index of the key coefficient corresponding to the inequality coefficient. The corresponding vector is denoted by $\mathbf{b}$. The dimension of $\mathbf{M}$ is $m$; this means $\mathbf{M} \in \mathbb{Z}^{m\times 2kn}$ and $\mathbf{b} \in \mathbb{Z}^m$. The inequalities are given as

$$\mathbf{xM} \leqslant \mathbf{b} \tag{6.1}$$

and a single inequality is of the form

$$\mathbf{x}^\top \mathbf{m}_j = \sum_{i=0}^{2kn} m_{i,j} \leqslant b_j \tag{6.2}$$

where $\mathbf{x} = (\mathbf{e}, \mathbf{s})$ is the secret key.

## 6.1 Recovery Model

For our recovery method, we assume an adversary has carried out an attack resulting in decryption failure inequalities in Kyber in which the coefficients are not or only slightly correlated to the ciphertext coefficients as e.g. in [PP21, BDH+21b, HPP21, Del22, Wei22]. They obtained $m$ inequalities each being in the form of

$$(-1)^{\mathrm{obs}}(\mathbf{e}^\top \mathbf{r} - \mathbf{s}^\top(\mathbf{e_1} + \Delta\mathbf{u}) + e_2 + \Delta v)_0 \leqslant 0. \tag{6.3}$$

Inequalities are not necessarily correct, and each inequality with index $i$, $i \in \{0, \ldots, m-1\}$, comes with a probability of correctness $p_i$.

Moreover, the adversary has access to the public key equation. Given this kind of information, they are asked to find the secret key $\mathbf{x} = (\mathbf{e}, \mathbf{s})$. In addition, our method, in particular Section 6.4.1, applies more generally whenever an attack results in belief propagation output and the adversary is given access to a public key equation similar to Equation (6.18).

## 6.2 Recovery Strategy

We provide two main techniques to recover the secret key from given decryption failure inequalities. Firstly, we achieve the error resistance provided by [Del22], also already mentioned in [PP21], in the belief propagation of [HPP21]. Secondly, we use belief propagation output to obtain a computationally less challenging lattice problem from the public key equation.

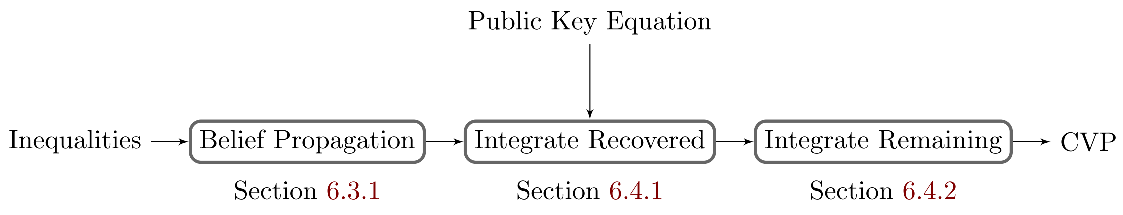


Figure 6.1: Parts of the recovery strategy and sections in which they are described (figure adapted from [HMS+23]).

The error resistance is achieved by taking the correctness probability of every inequality into account. We assign a correctness probability $p_j$ for every inequality (with index $j$); the probabilities are a property of the attack the inequalities are obtained from, for example, a fault success probability or from measurement noise. During the update process (c.f. Section 5.4.1), we compute probabilities for the inequality as well as for the inequality multiplied by negative one. The resulting probabilities are used for the updated message scaled by the correctness probability of the inequality.

After the abort criteria of the belief propagation have been reached (c.f. Section 5.4.2 and Chapter 7), the adversary is left with probability distributions – one for every key coefficient. These probability distributions are integrated in two steps: Firstly, the recovered coefficients are used on the public key equation

$$\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b} \tag{6.4}$$

by substituting given values. This is done in such a way that unknowns are eliminated in the order of reliability; we thereby utilize available information coming from the belief propagation output on non-recovered coefficients. Secondly, we obtain a vector $\mathbf{b}'$ from the belief propagation output which is closer to the lattice vector in the Closest Vector Problem (CVP) instance given by the public key equation. This results in a CVP instance that, when embedded into a Shortest Vector Problem (SVP) problem, proves to be computationally easier to solve and results in the secret key as well. The key recovery process is visualized in Figure 6.1, and a high-level depiction of the integration routine in three dimensions is given in Figure 6.2.

### 6.2.1 Number of Decryption Failures

Previous methods already provide a way to recover the secret key. The method of [PP21] requires a large but not impractical amount of RAM (which can likely be optimized further) and requires a maximum of a lower five-digit number of inequalities (see Section 3.3.3). The algorithm proposed in [Del22] provides error resistance and requires a number of inequalities that is comparable to [PP21]. Assuming an adversary can reliably apply thousands of faults, one could speculate that a few thousand faults more do not matter.

This is not the case when taking a basic countermeasure into account, which has already been proposed in [PP21]: To mitigate attacks that rely on decryption failures, a device may be designed to either shut down or regenerate a key after a certain number of decryption failures happen. The number of decryption failures to shut down after has to be determined based on the security level per number of inequalities (and the type of inequality taking e.g. key manipulation into account, as in [FKK+22]). A lower bound can be obtained by an information theoretic analysis as provided in Section 6.2.3. But the consequences of this kind of countermeasures are that the device becomes vulnerable to a simple Denial of Service (DoS) attack – an adversary merely has

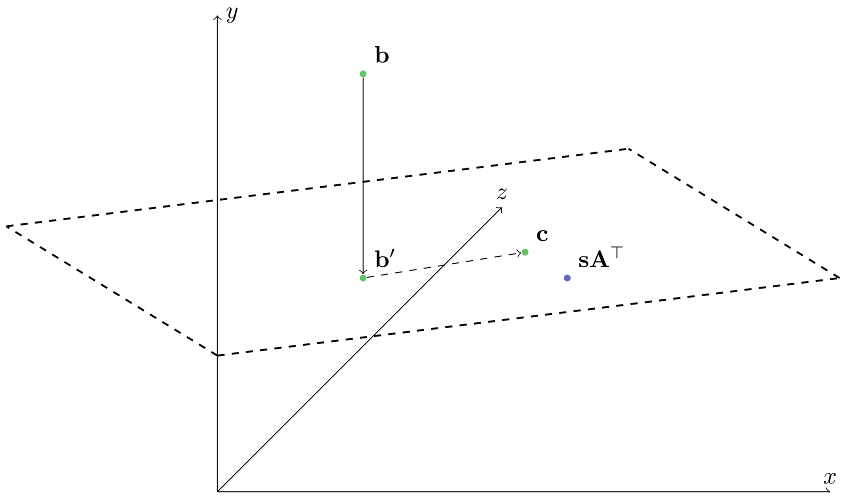


Figure 6.2: A high-level depiction of the integration of belief propagation output into a CVP instance as shown in [HMS+23]: Recovered coefficients reduce the dimension of the problem (from $\mathbf{b}$ to $\mathbf{b}'$), and remaining information gives a closer vector $\mathbf{c}$.

to submit a certain number of manipulated ciphertexts to shut down a device (no key recovery attack needs to be carried out).

Therefore, when considering this countermeasure, regenerating the long-term secret key after as many as possible, i.e., without undermining the targeted security level in practice, inequalities, is likely a favorable scenario. Knowing the minimum required number of decryption failures in practice as well as having security estimates for any number of inequalities is thus crucial to assess the security of systems defended this way. Our method lowers the number of required inequalities compared to previous work and allows for security estimates in partial attacks.

### 6.2.2 Belief Propagation Output

After a belief propagation run, the adversary is left with probability distributions $D_i$ for $i \in \{0, \ldots, 2kn - 1\}$, where $D_i$ belongs to the $i$-th key coefficient $x_i$. We recall the notation used in previous chapters: We say distributions are *converged* if there is a single value with probability 1 (and other values have probability 0). A distribution is *correct* if the likeliest value is the true value of the represented coefficient. A coefficient is *recovered* if the coefficient is correct and there is an order (known to the adversary) in which all coefficients with a smaller index are correct.

We again assume recovered coefficients are known to the adversary, i.e., that the adversary chose an order and knows the first $r$ coefficients are correct. Even though an adversary has no direct way of checking whether a coefficient is actually recovered, they may perform the following algorithm:

1. Order the coefficients.

2. Assume $r'$ coefficients are correct.

3. Run the remaining attack to obtain an estimate $\beta'$ for required BKZ-$\beta$.

4. In case of failure increase $r'$ and go back to 2.

If the attack ends with an estimate $\beta$ for the required BKZ-$\beta$, and the adversary is able to perform BKZ with $\beta'$, a failure is defined as estimated $\beta > \beta'$. In the case that $r'$ is correctly chosen but $\beta > \beta'$, the adversary will in the next step use an incorrectly high $r$; but as the attack is bound to fail in any case due to insufficient computational resources, this does not affect the validity of our assumption. Note that a low computational complexity of step 3 and step 4 (detecting failures) is crucial for our assumption to be correct. As this is the case in our situation, we may assume the adversary to know the correct number $r$ of recovered coefficients.

### 6.2.3 Information Theoretic Analysis

While several works obtain similar inequalities and either solve or estimate the remaining security using a recovery method [PP21, BDH+21b, HPP21, Del22, Wei22], the theoretical information contained per inequality/in a system of inequalities has not yet been stated. We show that with perfect ciphertext filtering (c.f. Section 3.2.2 and Section 5.3.2), every inequality gives exactly one bit, and we explain how to simulate the amount of information contained with different filtering values.

#### Mutual Information

For a random matrix $\mathbf{M}$ of inequalities (as rows) sampled as in an attack, without taking the observations into account (i.e., without fixing the inequality signs), we may see the observations $\mathbf{o}$ and the secret key as random variables and compute the mutual information for key space $X$ and observation space $O = \{0,1\}^m$ as

$$\sum_{\mathbf{x}\in X} \sum_{\mathbf{o}\in O} P(\mathbf{x},\mathbf{o}) \log_2 \left( \frac{P(\mathbf{x},\mathbf{o})}{P(\mathbf{x})P(\mathbf{o})} \right). \tag{6.5}$$

The observation is determined by $\mathbf{x}$ and $\mathbf{M}$; therefore, denoting the correct observation to a key as $\mathbf{o}_{\mathbf{x}}$, this can equivalently be written as

$$\sum_{\mathbf{x}\in X} P(\mathbf{x}) \log_2 \left( \frac{1}{P(\mathbf{o}_{\mathbf{x}})} \right). \tag{6.6}$$

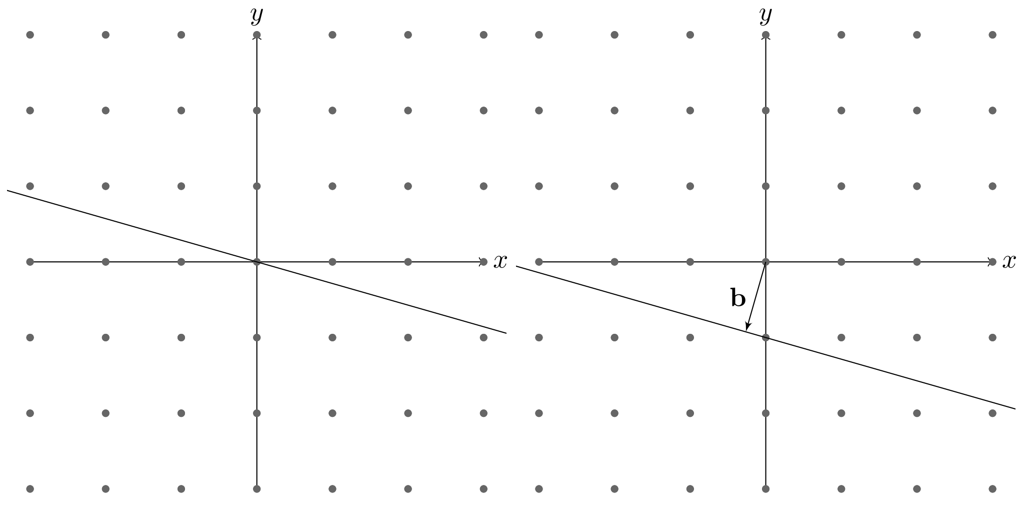
In the case of $\mathbf{b} = \mathbf{0}$, for a random key, every observation is equally likely; thus, in this case, the mutual information is

$$\sum_{\mathbf{x}\in X} \frac{1}{|X|} \log_2 \left(2^m\right) = m. \tag{6.7}$$

In other words, with perfect ciphertext filtering, every inequality holds one bit of information. Note that we implicitly used the linear independence of the columns of $\mathbf{M}$ to obtain Equation (6.7).

#### Intuition and Simulation

The problem can be seen as obtaining an affine hyperplane that partitions the key space as illustrated in Figure 6.3. In the case of perfect filtering, i.e., $\mathbf{b} = \mathbf{0}$, the hyperplane is a subspace and therefore halves the key space. If $\mathbf{b} \neq \mathbf{0}$, the computation of $P(\mathbf{o}_{\mathbf{x}})$ is more difficult as $\mathbf{b}$ and $\mathbf{M}$ influence by what degree the key space is diminished and therefore how likely an observation is. Taking into account that the left side of Equation (3.28) is approximately binomially distributed, the probabilities of $P(\mathbf{o})$ may be simulated for $\mathbf{b} \neq 0$. Another (slightly different) approach is stated in [HMS+23].



(a) $\mathbf{b} = \mathbf{0}$.

(b) $\mathbf{b} \neq \mathbf{0}$.

Figure 6.3: A single inequality in a two-dimensional key space, in the first case with $\mathbf{b} = \mathbf{0}$ and in the second case with $\mathbf{b} \neq \mathbf{0}$.

## 6.3 Error Resistant Belief Propagation

We first augment the belief propagation recovery by error resistance. This allows combining the benefit of being able to work on potentially incorrect inequalities, first provided by [Del22], with the lower number of required inequalities in [HPP21].

### 6.3.1 Error Resistant Check Nodes

To achieve error resistance using the method of [HPP21], we have to modify the check nodes. Recall from Section 5.4.1 that the factor node for the $j$-inequality in [HPP21] computes the probability for the $i$-th coefficient, $\mu_{j,i,t+1}$, based on the beliefs on all other coefficients. Denoting our inequality coefficients by $m_{i,j}$, where $i$ is the coefficient index and $j$ is the inequality index, the outgoing messages are given by

$$\mu_{j,i,t+1}(y) = P(x_i = y \mid \sum_{i'} m_{i',j} x_{i'} \leqslant b_j,\ x_i' \sim \mu_{i',j,t}) \tag{6.8}$$

where the right side may be computed using

$$\begin{aligned} &P(x_i = y \mid \sum_{i'} m_{i',j} x_{i'} \leqslant b_j,\ x_i' \sim \mu_{i',j,t}) && (6.9)\\ =&P(x_i = y \mid m_{i,j} y + \sum_{i' \neq i} m_{i',j} x_{i'} \leqslant b_j,\ x_i' \sim \mu_{i',j,t}) && (6.10)\\ =&P(\sum_{i' \neq i} m_{i',j} x_{i'} \leqslant b_j - m_{i,j} y,\ x_i' \sim \mu_{i',j,t}). && (6.11)\end{aligned}$$

We now additionally store the correctness probability $p_j$ for every inequality in the corresponding check node. Taking into account that inequalities are potentially incorrect, we get

$$P(x_i = y \mid \text{given } m_j \text{ and } p_j,\ x_i' \sim \mu_{i',j,t}) \tag{6.12}$$
$$= pP(x_i = y \mid \sum_{i'} m_{i,j} x_{i'} \leqslant b_j,\ x_i' \sim \mu_{i',j,t},\ m_j \text{ correct}) \tag{6.13}$$
$$+ (1-p)P(x_i = y \mid \sum_{i'} m_{i,j} x_{i'} > b_j,\ x_i' \sim \mu_{i',j,t},\ m_j \text{ correct}) \tag{6.14}$$

and

$$P(x_i = y \mid \sum_{i'} m_{i,j} x_{i'} > b_j,\ x_i' \sim \mu_{i',j,t}) \tag{6.15}$$

may be computed similar to Equation (6.11). The messages coming from check nodes are given as

$$\mu_{j,i,t+1}(y) = P(x_i = y \mid \text{given } m_j \text{ and } p_j,\ x_i' \sim \mu_{i',j,t}). \tag{6.16}$$

The error-resistant variant has to compute the leave-one-out distributions $S_i$, i.e., the distributions of

$$\sum_{i'} m_{i,j} x_{i'}, \text{ where } x_i' \sim \mu_{i',j,t}. \tag{6.17}$$

Therefore, the computation carried out in the check nodes uses similar optimizations compared to [HPP21] to compute the $S_i$. In the computation of Equation (6.14), the check node now, for every coefficient and every possible value, sums over the complete probability density function of $S_i$ instead of only up $b_j - m_{i,j}y$. The sum up to $b_j - m_{i,j}y$ is weighted by $p$, while the remaining sum is weighted by $1 - p$.

### 6.3.2 Computational Complexity

We require about twice the computational effort compared to [HPP21] because every check node has to sum over all values of all the probability distributions of all leave-one-out sums (c.f. Section 5.4.1). Compared to [Del22], the computational overhead is very large; nevertheless, note that we still only require an adversary to run our algorithm for less than a few hours on a laptop or merely minutes to seconds on more powerful Central Processing Units (CPUs). As this part of the attack is purely offline, we argue that such runtimes, especially as they can be reduced by using appropriate hardware[1], are irrelevant.

## 6.4 Belief Propagation and Lattice Reduction

After the abort conditions of the belief propagation have been reached, we are left with $2kn$ probability distributions – one for each coefficient. Those are used for an improved lattice problem compared to the one obtained in classical primal attack.

### 6.4.1 Integration of Recovered Coefficients

We first order all coefficients by some order, for example, min-entropy. In this order, we assume that the first $r$ coefficients are correct. Of these $r$ coefficients, we denote the number of recovered coefficients of $\mathbf{s}$ by $r_s$ and the number of recovered coefficients of $\mathbf{e}$ by $r_e$ (and $r = r_s + r_e$). We

[1] This is because the implementation is parallelized.

could integrate these coefficients using [DDGR20], but thereby we would disregard available information: We know the unknown coefficient's reliability from observed belief propagation output. Moreover, our integration of remaining probability information relies on working on the original Learning with Errors (LWE) equation.

The LWE equation is given by

$$\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b}. \tag{6.18}$$

In this section, the LWE equation is stated over $\mathbb{F}_q$ and not over $\mathbb{Z}$, and we, when integrating recovered coefficients, do work over $\mathbb{F}_q$ (but not $R$). We reuse the variable $n$ to denote the dimension of $\mathbf{e}$ and $\mathbf{s}$, this means we assign $n$ to be $2kn$; in other words, we obtain the LWE problem from the Module Learning with Errors (MLWE) equation and reuse the variable $n$. To simplify notation, we assume the recovered coefficients to come in default order meaning that the most reliable and often the first recovered coefficient has the index 0, and the least reliable coefficient has the index $n-1$. We denote the recovered coefficients as $e'_0, \dots, e'_{r_e-1}$ and $s'_0, \dots, s'_{r_s-1}$.

## Integrating Recovered e

Given a recovered value of $\mathbf{e}$, i.e., $e_j = e'_j$, we may obtain an (undisturbed) linear equation from Equation (6.18). This equation is of the form

$$\sum_{i=0}^{n} s_i A_{j,i} = b_j, \tag{6.19}$$

and as most coefficients $A_{j,i}$ will not be zero, we may solve for most coefficients of $\mathbf{s}$. The chosen coefficient can then be eliminated by using the recovered value of $\mathbf{e}$.

The coefficients of $\mathbf{s}$ are already sorted by some order which we see as correctness reliability. We now use that order again to determine which coefficients to remove first – naturally, we aim to substitute those that are the least converged. This means for recovered $e_j = e'_j$ and least reliable value $s_i$, we eliminate $s_i$ from the system of equation by substituting

$$s_i = A_{j,i}^{-1}(b_j - e'_j - \sum_{i' \neq i} s_{i'} A_{j,i'}) \tag{6.20}$$

and then computing

$$\mathbf{A} \leftarrow (\mathbf{A} - \mathbf{T}_{e,j})_{\hat{i}} \tag{6.21}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \mathbf{t}_{e,j}, \tag{6.22}$$

where $\cdot_{\hat{i}}$ denotes the $i$-th column to be removed,

$$\mathbf{T}_{e,j} = \begin{pmatrix} A_{j,i}^{-1} A_{j,0} \\ A_{j,i}^{-1} A_{j,1} \\ \dots \\ A_{j,i}^{-1} A_{j,n-1} \end{pmatrix}, \tag{6.23}$$

and

$$\mathbf{t}_{e,j} = \left(A_{j,i}^{-1}(b_j - e_j), A_{j,i}^{-1}(b_j - e_j), \dots, A_{j,i}^{-1}(b_j - e_j)\right). \tag{6.24}$$

In matrix-vector notation, the transformation for all recovered values can be written as

$$\mathbf{A} \leftarrow \mathbf{T}'_e(\mathbf{A} - \mathbf{T}_e), \tag{6.25}$$

where $\mathbf{T}'_e$ is the matrix resulting from the identity when removing the columns in $\{0, \ldots, r_{e-1}\}$,

$$\mathbf{T}_e = \sum_j \mathbf{T}_{e,j} \tag{6.26}$$

and

$$\mathbf{b} \leftarrow \mathbf{b} - \mathbf{t}_e \text{ for } \mathbf{t}_e = \sum_j \mathbf{t}_{e,j}. \tag{6.27}$$

**Integrating Recovered s**

The integration of **s** is comparably straightforward as eliminating a coefficient of **s** can be done directly and does not involve a decision that is based on reliability. The elimination of known values of **s** consists of substituting known values $s_j = s'_j$ into Equation (6.18) for $j \in \{0, \ldots, r_s - 1\}$. This means that for every $j$, we set

$$\mathbf{A} \leftarrow \mathbf{A}_{\hat{j}} \tag{6.28}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \mathbf{t}_s \tag{6.29}$$

where $\cdot_{\hat{j}}$ denotes the $j$-th column to be removed and

$$\mathbf{t}_{s,j} = \left(A_{i,0}s'_j, A_{i,1}s'_j, \ldots, A_{i,n-1}s'_j\right). \tag{6.30}$$

In matrix-vector notation, the transformation for all recovered values can be written as

$$\mathbf{A} \leftarrow \mathbf{A}\mathbf{T}_s, \tag{6.31}$$

where $\mathbf{T}_s$ is the matrix resulting from the identity when removing the rows in $\{0, \ldots, r_{s-1}\}$, and

$$\mathbf{b} \leftarrow \mathbf{b} - \mathbf{t}_s \text{ for } \mathbf{t}_s = \sum_j \mathbf{t}_{s,j}. \tag{6.32}$$

### 6.4.2 Integration of Probability Information

After having integrated the recovered coefficients, i.e., after having computed

$$\mathbf{A} \leftarrow \mathbf{T}'_e(\mathbf{A} - \mathbf{T}_e)\mathbf{T}_s \text{ and } \mathbf{b} \leftarrow \mathbf{b} - \mathbf{t}_s - \mathbf{t}_e, \tag{6.33}$$

the dimension of the equation is $n - r$, and we are left with just as many probability distributions – one for each of the remaining coefficients. The vectors **e** and **b** have $n - r_e$ coefficients, **s** has $n - r_s$ coefficients, and **A** is of dimensionality $(n - r_s) \times (n - r_e)$.

A straightforward approach could be to use key enumeration, e.g. as suggested in [PSG16], interpreting the probability distributions as key ranking per coefficient. But the belief propagation algorithm usually recovers either all coefficients or only very few (c.f. Section 7.2.2), i.e., $r$ is usually small and this leads to a large number of potential keys. Therefore, this only allows for recovery in edge cases and does not improve upon the majority of cases where a direct key recovery is not possible.

Instead, we leverage the additional information for a variant of the primal attack[2]. The equation (over $\mathbb{F}_q$) given by (the transformed) **A** and **b** (now interpreted over $\mathbb{Z}$) defines a lattice $\mathcal{L}_{\text{svp}}$ through the basis

$$\mathbf{B}_{\text{svp}} = \begin{pmatrix} q\mathbf{I}_{n-r_s} & \mathbf{0} \\ \mathbf{A} & \mathbf{I}_{n-r_e} \end{pmatrix} \in \mathbb{Z}^{(n-r)\times(n-r)}. \tag{6.34}$$

[2]Note that in [HMS+23] we further discuss the option of using the dual attack.

It can easily be verified that

$$\mathbf{t}_{\text{target}} = (-\mathbf{e}, \mathbf{s}) + (\mathbf{b}, \mathbf{0}) \tag{6.35}$$

is an element of $\mathcal{L}_{\text{cvp}}$ and the primal attack consists of finding a lattice element close to $(\mathbf{b}, \mathbf{0})$ (which is given); this leads to finding $\mathbf{t}_{\text{target}}$ and thus $\mathbf{x} = (\mathbf{e}, \mathbf{s})$.

Through the obtained probability distributions and potentially key enumeration, we are now given guesses for $\mathbf{x}$, which we denote by $\mathbf{x}' = (\mathbf{e}', \mathbf{s}')$. Under the assumption that we found the correct number of recovered coefficients, $\mathbf{x} \neq \mathbf{x}'$. But $\mathbf{x}'$ can be assumed to be closer to $\mathbf{x}$ than $\mathbf{0}$ if we consider the belief propagation to work correctly. Denoting the difference between guess and key by $(\hat{\mathbf{e}}, \hat{\mathbf{s}})$, we have

$$\mathbf{t}_{\text{target}} = (-\hat{\mathbf{e}}, \hat{\mathbf{s}}) + (\mathbf{b}, \mathbf{0}) + (-\mathbf{e}', \mathbf{s}'), \tag{6.36}$$

and

$$||\mathbf{t}_{\text{target}} - ((\mathbf{b}, \mathbf{0}) + (-\mathbf{e}', \mathbf{s}'))|| \leqslant ||\mathbf{t}_{\text{target}} - (\mathbf{b}, \mathbf{0})||. \tag{6.37}$$

Thus, searching for a lattice vector close to $((\mathbf{b}, \mathbf{0}) + (-\mathbf{e}', \mathbf{s}'))$ will be computationally less expensive than searching close to $(\mathbf{b}, \mathbf{0})$.

Embedding into the SVP problem using Kannan's embedding [Kan87], we obtain the lattice $\mathcal{L}_{\text{svp}}$ defined by the basis

$$\mathbf{B}_{\text{CVP}} = \begin{pmatrix} q\mathbf{I}_{n-r_s} & \mathbf{0} & 0 \\ \mathbf{A} & \mathbf{I}_{n-r_e} & 0 \\ \mathbf{b} - \mathbf{e}' & \mathbf{s}' & c \end{pmatrix} \in \mathbb{Z}^{(n-r)\times(n-r)}, \tag{6.38}$$

where $c$ can be chosen in an optimal way as described previously, e.g. in [DDGR20]; for our evaluations, we use $c = 1$ which is close to optimal in Kyber512 and optimal in Kyber768 and Kyber1024. Note that for different schemes, such as, e.g., FrodoKEM, the choice of $c = 1$ is far from optimal. The lattice $\mathcal{L}_{\text{svp}}$ given by $\mathbf{B}_{\text{svp}}$ may be used to find the secret key using lattice reduction, most commonly using BKZ 2.0 [CN11] (implemented e.g. in [Fplll]). Moreover, [ADPS16b, AGVW17] explain how security estimates in terms of required BKZ-$\beta$ may be obtained from this kind of SVP instance (also reiterated and used in [DDGR20]). We reiterate this approach in Section 2.2.1, and our estimates in Section 7.2.3 are obtained by employing it.

**Additional key enumeration.** As already mentioned, taking the likeliest value as key guess $\mathbf{x}'$ is not the only option. Instead, the adversary may use key enumeration to compile a list of likeliest guesses and create an SVP instance from them. Thereby, a guess with fewer incorrect coefficients may be taken into consideration and further reduce security. This approach differs from a purely enumeration-based approach as the enumerated keys do not need to be correct, i.e., the key enumeration is not used to find the true key. Instead, the key enumeration merely improves upon the guess but is not required to find the correct key; this is enabled by the approach based on lattice reduction.

Note that an additional key enumeration also enables a parallelized approach. By enumerating $k$ likely keys for large $k$, and then running lattice reduction all resulting instance, the adversary can run $k$ instances. It is unlikely, that an attacker with widely-available computational resources sees large improvements by this approach – the block size used for the lattice reduction is the determining factor. However, as these instance run completely independent, a state-level adversary may be able to further improve upon the results.

## 6.5 Summary

Several attacks, in particular attacks that exploit decryption failures, rely on statistical algorithms such as belief propagation. Classical state-of-the-art attacks on lattice-based cryptography target the underlying lattice problem and recover the secret by using lattice reduction. In the case of decryption failures, statistical methods are currently used in practice as they perform well in end-to-end key recovery. Those methods, however, are either not error-tolerant or require a larger-than-necessary amount of information, and – most importantly – do not offer security estimates in partial attacks. This not only leads to underestimating attacks that exploit decryption failures, but also hinders proper evaluation of countermeasures and the security of devices.

We introduce a novel method to recover the secret key from decryption failure information. Our method combines statistical and coding theoretic approaches used in previous work [PP21, HPP21, Del22] and the algebraic approach of using information arising from the public key equation used in the classical attacks as well as in [DDGR20, DGHK22]. We first employ an improved belief propagation technique achieving error resistance and then explain how belief propagation output may be integrated into the lattice problem arising from the LWE instance. Thereby, we provide an improved recovery algorithm that is error-resistant, requires less information to recover the secret key, and outputs security estimates under the assumption of an adversary who uses state-of-the-art recovery methods.

The presented method directly improves upon several previously published attacks that target the error correction in the new National Institute of Standards and Technology (NIST) standard for post-quantum key exchange and other lattice-based schemes, e.g., [BDH+21b, DHP+22, Del22]. In addition, our recovery method is also relevant for potential future attacks that exploit decryption failures using a chosen ciphertext. Such attacks arise whenever a side channel allows observation of the comparison operation or a fault allows manipulation of the submitted ciphertext, which could be enabled by various types of leakages in different locations of the algorithm. Thus, understanding the impact of decryption failures is highly relevant to understanding the threat posed by side-channel and fault attacks against lattice-based cryptography.

Moreover, we conjecture that our error-resistant belief propagation techniques and integration of statistical information into a lattice problem may be used more generally to improve upon implementation attacks on lattice-based schemes. This is, e.g., the case for the attacks presented in this thesis, but most likely also applies more generally. Thus, we improve upon several previous attacks, enable novel attacks, and provide concrete security estimates for partial attacks, which allow for designing and evaluating countermeasures.

# Chapter 7

# Evaluation and Results

We evaluate the previously presented attacks and methods on the example of Kyber. We rely on simulations and refer to previous work such as [PP19] or [Del22] for attacks on physical hardware confirming our models. The adaptation of Chapter 4 to countermeasures, described in Section 4.5, as well as the method described in Chapter 6 are only evaluated on the example of Kyber512. The adaptations presented in Section 4.5 are independent of the security level and, therefore, do not require separate evaluations. In the case of Chapter 6, the error distribution in Kyber512 is particularly unfavorable[1], and the performance on other security levels can be inferred from the evaluation of Chapter 5. To access the methods defined by Kyber, our attacks rely on either a modified PQClean [KSSW22, PQClean] implementation or a Python [PyRef] implementation of Kyber, which is close to the reference implementation [ABD+] of Kyber submitted to the National Institute of Standards and Technology (NIST) contest in the second round [NistR2]. In the case of Section 4.5, the effects of countermeasures are directly modeled into the belief propagation instantiation. Due to the varying nature of our attacks and methods, the evaluation models differ.

**Belief propagation implementation.** All attacks and methods rely on belief propagation. This work is accompanied by a generic belief propagation implementation [HSS21], which forms the basis of all our implementations. It was originally developed for the work of [HHP+21] by the author of this thesis with the help of several co-authors. The implementation is written in Rust [MI14, RustRef], deployed as a library, and allows changing the node functions with comparably little effort. The base implementation features a graph class that holds edges and nodes. Nodes are objects containing an abstract node function object (implemented as a trait), which also allows the graph to extract several parameters to, e.g., influence the message scheduling. Based on those parameters and the node function, the graph class offers fully parallelized computation of the node functions and takes care of correct message passing between nodes connected by an edge. To implement a different type of belief propagation, e.g. for one of the here presented attacks, only the node function objects have to be reimplemented.

**Recovered coefficients.** All of our attacks attempt to recover secret key coefficients from belief propagation. Depending on the available information and/or the measurement noise, the belief propagation output varies from having almost no additional information (compared to the priors) to giving distributions where the likeliest value is the correct/true one for all coefficients. When evaluating, we may check whether a coefficient is correctly recovered, but an actual attacker does not have this option as they do not know the true value. Therefore, counting coefficients as

[1]Both other security levels use the same, more narrow error distribution.

"recovered" whenever their distributions have the true value as the likeliest result is obviously incorrect – even when using only the public error distribution as model, this definition would give about a third of "recovered" coefficients in Kyber. Instead, we sort coefficient distributions after a belief propagation run by one or several metrics. In this metric/order, we check how many correct coefficients occur before the first incorrect one. This means we look for the longest chain of correct coefficients we can find in some order. The coefficients in this chain are what we call *recovered*. This method is available to an attacker with a computational overhead that is linear (or even logarithmic using binary search) in the cost of checking whether a key is correct: An attacker can simply run the key recovery for every possible length of the chain and check if this yields the correct result, or use an estimate of hardness for the remaining attack. Checking if a secret key is correct is a linear operation itself, and quickly commutable estimates for the hardness of lattice reduction are available. Therefore, in the case of the attacks and methods presented in this thesis, assuming that an attack may recover these coefficients is valid.

**Hamming weight model.** To evaluate side-channel attacks, i.e., mainly the attacks presented in Chapter 4, we rely on the Hamming weight model, which we already reiterated in Section 3.1.2. This model was shown to be realistic in the similar attack of [PP19] by confirming it against measurements performed on a physical device (c.f. Section 3.1.2). Our attack in general does not depend on the assumed leakage model, but the adaption to shuffling countermeasures does, in parts, assume the measurements to follow the Hamming weight model distributions.

**Fault model.** Our fault model for the evaluation of our fault attacks, i.e., the attack strategy described in Chapter 5, are single bit flips, bit resets, or bit sets. This means an attacker may flip a bit, set a bit to zero, or set a bit to one. Which of the cases applies depends on the ciphertext under (offline) control of the attacker. While requiring an attacker to be able to precisely fault a single bit, we note that in the case of the attack of Chapter 5, the attack can be modified to allow for a more relaxed assumption: If an attacker derives information from decapsulation success alone, they may work with an arbitrary imprecise or unreliable fault. In turn, the number of required faults is multiplied by $\frac{2}{f}$ where $f$ is the fraction of correct faults per total number of faults. Note that in this case, multi-bit faults are also an option an attacker may target; this choice is again taken offline by the attacker. While we did not carry out the attack, the work of Delvaux [Del22] carries out a similar attack but with an even more relaxed fault model, which is achieved by their improvements on our attack.

**Decryption error model.** The model for recovering the secret key from decryption failure information is on its own very straightforward: The method described works on decryption failure information in which the secret key coefficients are not correlated to the inequalities coefficients. The relation to concrete attacks and their properties is more complicated, but not required for evaluation. For example, for an attack such as described in Chapter 5, a possible model would again be the previously described one, which we use to evaluate fault attacks. But for an attack using a side channel to observe decryption failures, the appropriate model may be based on the Hamming weight model. Thus, our method should instead be seen as enabling and improving a wide array of attacks rather than one concrete attack having a specific evaluation model.

## 7.1 Attacks on the Number Theoretic Transform

For the attacks on the inverse Number Theoretic Transform (NTT), we evaluate the success rate of the belief propagation per noise level for all Kyber security levels and different amounts of zeros. We also report on the number of required traces per amount of zeros, taking the practical maximum noise level into consideration. We then analyze the success rate of the final key recovery and derive a final success probability for the complete attack. In addition, we give security estimates for partial attacks and explain how they can be improved using the techniques described in Chapter 6.

### 7.1.1 Simulation

We evaluate the attacks using the Hamming weight model as described above. A Python [PyRef] implementation of Kyber close to the reference implementation of [BDK+18] and [KSSW22, PQClean] gives access to the internal Kyber routines. Most notably, a modified (inverse) NTT implementation allows computing partial (inverse) NTTs and output intermediate values and enables the simulation of the leakage parts of the recovery method.

Our implementation receives several input parameters allowing us to specify the precise variant of the attack. Firstly, a key pair is generated as in a normal Kyber key exchange. To simulate the attack, we then generate a chosen ciphertext component $\mathbf{u}$ (according to the method described in Section 4.3), compute the NTT of $\mathbf{u}$ and the scalar product of $\mathbf{u}$ and the secret key. The result is fed into our modified NTT routine, which records all intermediate values $v_{i,j}$, $i \in \{0, \dots, n\}$ and $j \in \{0, \dots, \text{layers}\}$; the latter serves as basis for the simulated leakage. For each intermediate value as well as in- and outputs of the inverse NTT, we compute a probability distribution according to the Hamming weight model. This means for standard deviation $\sigma$, we compute the array of measured values

$$v'_{i,j} = v_{i,j} + e_{i,j} \tag{7.1}$$

where

$$e_{i,j} \leftarrow \mathcal{N}_\sigma(0). \tag{7.2}$$

The measurements are then given by

$$m_{i,j} = \mathcal{N}_\sigma(v'_{i,j}) \tag{7.3}$$

and used as priors for the belief propagation graph.

We state our results in measurement noise level $\sigma$. A positive success rate for a certain $\sigma$ means that the attack succeeds with this noise level. A higher $\sigma$ means a higher noise level and that the attack is more difficult to carry out. This means that we aim to achieve a $\sigma$ that is as high as possible while still maintaining a positive success rate.

The belief propagation is implemented using the generic belief propagation library, and it models the inverse NTT. Upon convergence, the resulting probability distributions are used to restore the key by assigning the likeliest value to a key coefficient. This means, denoting the resulting distributions by $D_i$ where $i$ is the coefficient index, we retrieve the partial secret key $s'$ by setting

$$s'_i = \operatorname{argmax}_x D_i(x) \tag{7.4}$$

for all positions that were not zeroed out. The full secret key $s''$ is then restored by our recovery method; if both parts of the attack succeed, the secret key $s$ is equal to $s''$.

#### Convergence/Abort Criteria

We run the belief propagation for a maximum of 1000 full iterations (which means for 2000 message passes). Additionally, we abort the belief propagation based on a number of empirically derived convergence criteria: We abort if the Shannon entropy at all nodes is less than 0.1 bit, if the entropy change is less than 0.05 bit in 20 iterations, or if no more coefficients have been recovered in the last 200 steps. The first criterion is mostly a convergence criterion while the second and third criteria do in most cases signal an abort after an unsuccessful run. Note that the last criterion can only be employed for evaluation and is not available to a real-world attacker.

#### Runtime

With these criteria, a belief propagation run takes, on average, 20 minutes on two Intel Xeon E5-2650 v4 2.20GHz (having 24 cores with hyper-threading enabled). The run time greatly differs depending on the priors: With a low noise level, convergence is almost instant, i.e., after very few iterations. With a very high noise level, the runs take more computation time but the second and third abort criteria do cause the belief propagation to fail after in a reasonable time. In settings with a noise level that is close to allowing for a successful run, the consumed time is the highest as neither abort nor convergence criteria are triggered. We emphasize that our belief propagation can be run on an ordinary laptop and does not require specialized hardware.

The final key recovery requires an attacker to run BKZ-70 to BKZ-80, which is computationally more challenging than the belief propagation algorithm. The reduction without lattice reduction (not described in this thesis but only in [HHP+21]) allows for an instant recovery on any ordinary laptop. In addition, instead of running the final recovery method, an attacker may use another trace to recover the remaining key coefficients and thereby trade measurements for required computational power.

### 7.1.2 Results

The success rate of the belief propagation depends on the noise level only – the security level is irrelevant. On the other hand, the results of the final key recovery on the other hand differ between the security level due to differing vector length $k$ and different error distributions.

#### Belief Propagation

After having obtained data from the template attack, the belief propagation computes the marginal distribution for each key coefficient. Recovering coefficients using belief propagation forms the basis for the subsequent recovery algorithms. We first report on its success rate in unprotected, masked, and shuffled settings.

**Masked and unmasked settings.** We first state our results for an unprotected standard linearly masked setting as proposed in e.g. [RRC+16, OSPG18]. We ran our experiments for a variety of distributions of zeros, in steps of $\Delta\sigma = 0.1$, and state them in standard deviation $\sigma$ of the noise. Per noise level, we run 25 experiments and compute the number of recovered coefficients as an average.

The masked and unmasked settings do not differ for all noise levels where the success rate is 1. This is because, in the masked setting, the success rate drops quadratically compared to the unprotected setting as the attack succeeds if and only if both graphs succeed. Figure 7.1 depicts the success rate in an unmasked and unprotected setting, and Figure 7.2 states the result of a masked setting. The number of recovered coefficients is shown in Figure 7.3 for the unprotected scenario and in Figure 7.4 for the standard masked scenario.

**Required traces.** The number of required traces is already shown in Table 4.2 in a masked setting. We re-state the number of required traces in combination with the required noise level. Note that in the case of Kyber512, an attacker with limited computational resources would likely require one extra trace.

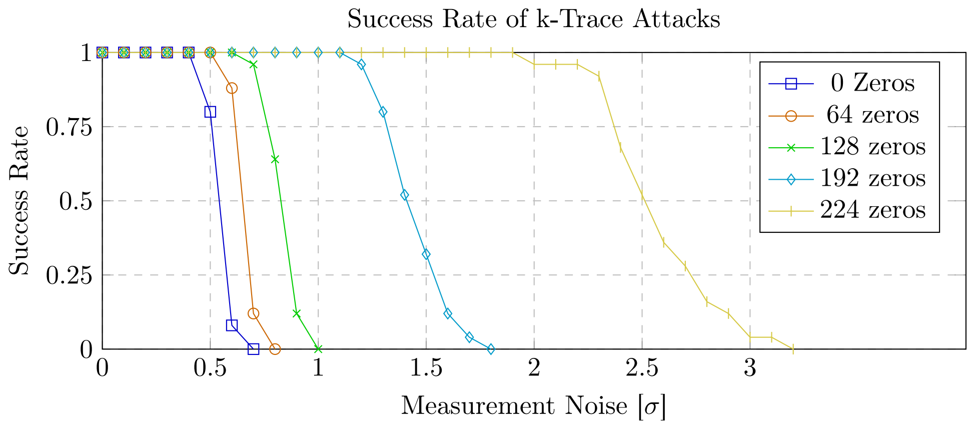


Figure 7.1: The success rate, without countermeasures, of the belief propagation part of the attack on the inverse NTT per measurement noise level (in $\sigma$) and number of zeros as stated in [HHP+21].

Table 7.1: Number of required traces and required noise level in a masked setting as similarly depicted in [HHP+21]. The name of our attacks stems from the setting with 192 zero coefficients.

| Sparseness | Kyber512 | Kyber768 | Kyber1024 | SR $>$ 0.7 | SR $>$ 0 |
|---|---|---|---|---|---|
| # zero coeffs. | # traces | # traces | # traces | max. $\sigma$ | max. $\sigma$ |
| 224 | – | – | 8 | 2.2 | 2.7 |
| 192 | 2 | 3 | 4 | 1.2 | 1.4 |
| 128 | 1 | 2 | 2 | 0.6 | 0.8 |
| 64 | 1 | 1 | 2 | 0.5 | 0.7 |
| 0 | 1 | 1 | 1 | 0.5 | 0.5 |

**Shuffling countermeasures.** In the case of shuffling countermeasures, we take several scenarios into account. We evaluate the effects of fine shuffling when using our shuffle node as well as of coarse in-group shuffling when using our matching algorithm. Success rates in terms of noise level are shown in Figure 7.5. Using the shuffle node, we may target a fine-shuffled implementation with positive success rate with a noise level of $\sigma \leqslant 1$. Two-point matching allows attacking a coarse-in-group shuffled implementation with a positive success rate as long as $\sigma$ is smaller than 0.5. As expected, we lose some noise tolerance in both cases, but may still perform a successful attack on a protected implementation while maintaining a noise level that is comparable to previous attacks on unprotected implementation.

The entropy caused by mixing priors is stated in Section 7.1.2, Section 7.1.2 shows the average entropy per layer when using two-point matching with a mix matrix, and Section 7.1.2 states information about the mix matrix itself. These values correlate with the protection offered by the employed countermeasures and the effectiveness of our adaptation. It can be seen that higher layers are affected to a larger extend as fewer sparse values occur and because noise from lower layers accumulates in higher layers. The rank of the correct permutation, which signifies the computational complexity of exact permutation matching, is depicted in Figure 7.6.

Mixing statistics were obtained by performing 100 runs per sigma, while success rates were

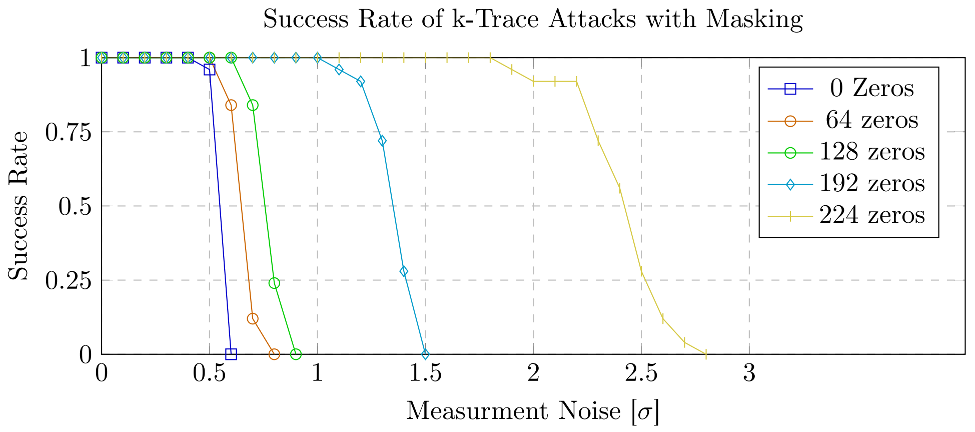


Figure 7.2: The success rate, with standard masking, of the belief propagation part of the attack on the inverse NTT per measurement noise level (in $\sigma$) and number of zeros as stated in [HHP+21].

Table 7.2: Entropy per belief propagation node when applying mixing priors in comparison to an attack against an unprotected inverse NTT as stated and depicted in [HSST23].

| $\sigma$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.00 |
|---|---|---|---|---|---|---|---|---|---|---|
| No shuffling | 8.60 | 8.63 | 8.90 | 9.30 | 9.59 | 9.79 | 9.96 | 10.09 | 10.20 | 10.30 |
| Mixing priors | 9.63 | 9.63 | 9.76 | 10.05 | 10.29 | 10.44 | 10.54 | 10.63 | 10.70 | 10.77 |

computed with 10 runs per sigma. We do not evaluate the effects of coarse full shuffling due to the required computational power and note the high level of protection by these countermeasures. Our methods apply to full shuffling as well but are unavailable for an ordinary attacker without extraordinary computational resources. To the best of our knowledge, possible improvements as well as theoretic security evaluations of these countermeasures are still an open question.

### Final Key Recovery

The final attack step is the key recovery using lattice reduction. We implemented this step using the BKZ implementation of [Fplll]. In Kyber768 and Kyber1024, using BKZ-70, we achieve a success rate of 92.7% in 590 attempts with 64 (out of 256) non-zero values. As the final key recovery works independently on even and odd entries of our vector, we in fact run two key recoveries each working on 32 (out of 128) non-zero values. With a block size of 80, the success rate was 1. Kyber512 uses an error distribution with a larger support and therefore requires more computational effort. Using BKZ-80, we could solve only 54% out of 100 attempts; this can very likely be improved using a larger block size but could be infeasible for an adversary. Therefore, we recommend using more non-zero components or to record more traces. With 96 non-zero coefficients, all 100 attempts to recover the key were successful.

We summarize that for Kyber768 and Kyber1024 with 64 non-zero values, BKZ-70 suffices in over 90% of the cases, and BKZ-80 can solve for the secret key in all cases. For Kyber512, an attacker may need to run one more trace than theoretically required when being able to run BKZ-80; this can be avoided by using more non-zero components.

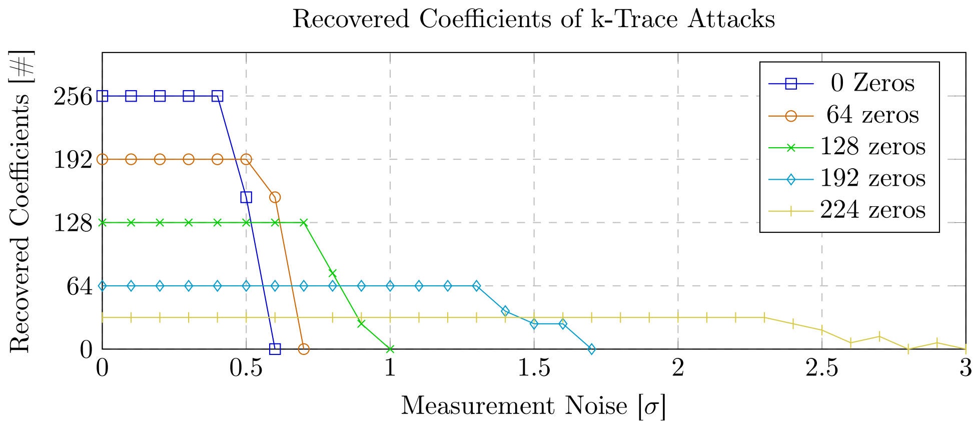


Figure 7.3: The number of recovered coefficients, without deployed countermeasures, of the belief propagation part of the attack on the inverse NTT per measurement noise level (in $\sigma$) and number of zeros as stated in [HHP+21].

Table 7.3: The entropy increase, as stated and depicted in [HSST23], caused by applying two-point matching compared to an attack on unprotected implementation. Note that the sixth layer contains a large amount of butterflies with one input node being zero.

| $\sigma$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 |
|---|---|---|---|---|---|---|
| No shuffling (avg) | 8.69 | 8.71 | 8.97 | 9.35 | 9.66 | 9.85 |
| Layer 1 | 8.70 | 8.71 | 8.97 | 9.38 | 9.67 | 9.86 |
| Layer 2 | 8.68 | 8.70 | 8.94 | 9.43 | 9.72 | 9.92 |
| Layer 3 | 8.79 | 8.83 | 9.10 | 9.61 | 9.96 | 10.14 |
| Layer 4 | 8.92 | 8.94 | 9.19 | 9.75 | 10.07 | 10.33 |
| Layer 5 | 9.01 | 9.00 | 9.28 | 10.06 | 10.48 | 10.78 |
| Layer 6 | 9.82 | 9.83 | 9.98 | 10.51 | 11.13 | 11.48 |
| Layer 7 | 10.21 | 10.21 | 10.37 | 10.89 | 11.32 | 11.49 |

### 7.1.3 Comparison to Prior Work

Our attack strategy improves the attack presented in [PPM17] and [PP19]. Similar to [PPM17], we target the decryption routine and thereby the long-term secret in contrast to [PP19]. As [PPM17] targets a different scheme and different leakage, a direct comparison is hard. Instead, the subset of our attack that does not use a chosen ciphertext, denoted “0 Zeros” (in e.g. Figure 7.2), may be seen as adapting the attack of [PPM17] to our situation while using improvements of [PP19]. Additionally, we provide the chosen ciphertext and the recovery technique to substantially improve upon the noise level (see Section 7.1.2). Using our technique increases the maximal possible noise level, allowing an adversary to launch an attack even if their measurements are less precise. In fact, the required standard deviation $\sigma$ for a successful attack without a chosen ciphertext is about 0.6, whereas it increases to more than 2.5 using our attack strategy. But note that for maximally required noise level, more measurements are required as stated in Section 7.1.2.

In regard to countermeasures, comparable, but more extensive, analysis exists for symmetric cryptography (see Section 3.1.7). But to the best of our knowledge, no adaptations to belief-

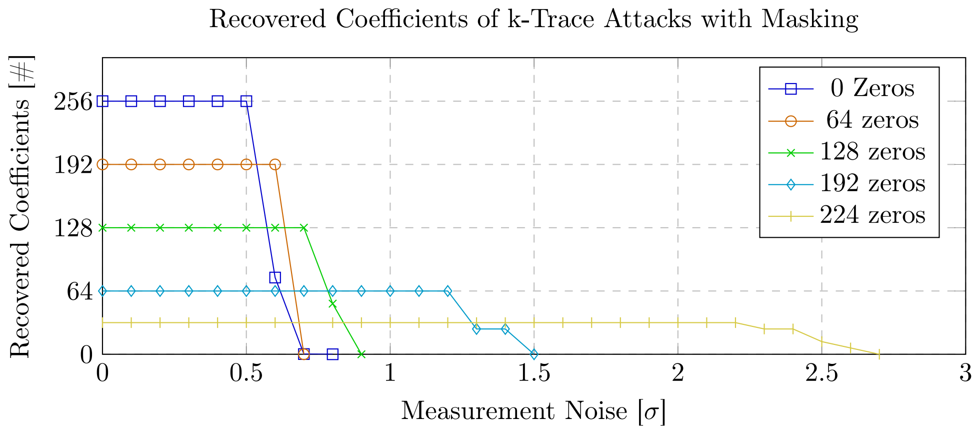


Figure 7.4: The number of recovered coefficients, with standard masking, of the belief propagation part of the attack on the inverse NTT per measurement noise level (in $\sigma$) and number of zeros as stated in [HHP+21].

Table 7.4: Entries in a mix matrix with probability greater than 0.005 when using two-point matching as stated and depicted in [HSST23]. The entropy per row is stated in brackets.

| $\sigma$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Layer 1 | 1.02 (0.02) | 1.02 (0.02) | 1.08 (0.05) | 1.23 (0.12) | 1.36 (0.2) |
| Layer 2 | 1.06 (0.06) | 1.07 (0.06) | 1.23 (0.13) | 1.70 (0.33) | 2.18 (0.58) |
| Layer 3 | 1.15 (0.14) | 1.17 (0.15) | 1.58 (0.30) | 2.96 (0.82) | 4.54 (1.43) |
| Layer 4 | 1.33 (0.03) | 1.37 (0.32) | 2.76 (0.75) | 6.60 (1.94) | 11.58 (2.99) |
| Layer 5 | 8.74 (2.85) | 8.91 (2.87) | 13.07 (3.45) | 25.93 (4.80) | 37.95 (5.50) |
| Layer 6 | 29.13 (4.77) | 28.90 (4.79) | 30.36 (5.35) | 33.52 (6.43) | 26.78 (6.80) |

propagation-based attacks on post-quantum schemes have yet been proposed. Whether shuffling countermeasures prevent attacks on the NTT or to what level these attacks are mitigated is not yet known. Our techniques allow an adversary to target devices protected by shuffling countermeasures with similar noise tolerance as previous attacks achieved against unprotected implementations. We thereby provide previously missing analysis of countermeasures against attacks on the NTT.

Summarizing, our attack strategy in its weakest form is similar to [PPM17] but targets Kyber, assumes a different leakage model, and makes use of improvements from [PP19]. Our attack strategy allows for an increase in the maximal standard deviation of the noise by a factor of up to 5 when targeting the secret key in an attack on the NTT. Moreover, we take countermeasures into account, give techniques to circumvent them, and provide an assessment of their impact.

## 7.2 Decryption Errors and Key Recovery

We evaluate the attack stated in Chapter 5 in the described fault model. We simulate a fault by using a manipulated implementation which manually introduces the required error. Using a chosen ciphertext, we thereby obtain decryption failure inequalities which are solved using belief

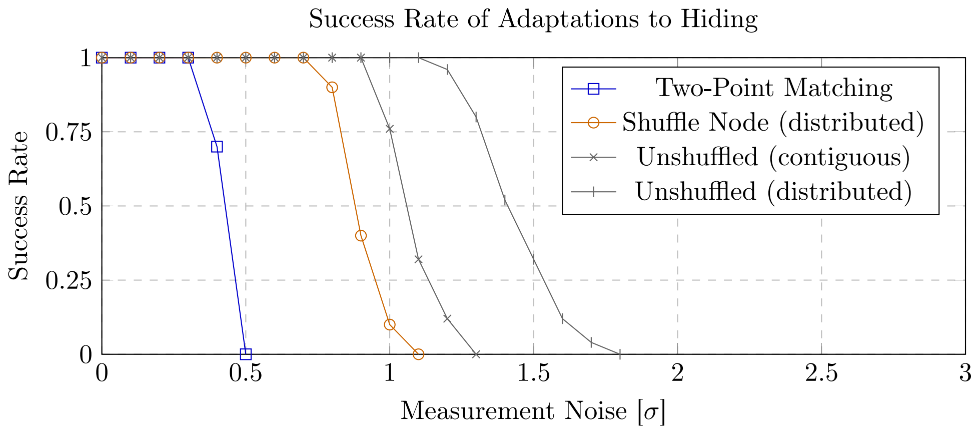


Figure 7.5: The success rate, in the presence of shuffling countermeasures as stated in [HHP+21], of the belief propagation part of the attack on the inverse NTT per measurement noise level (in $\sigma$) and number of zeros used in the chosen ciphertext. Note that the zero distributions differ depending on the countermeasure. Therefore, we re-state the distributed results from Figure 7.1 and, additionally, state the result for contiguous zeros as given (only) in [HHP+21].

propagation. The ciphertexts may be pre-filtered to simulate the technique of ciphertext filtering.

Using a similar method, we may also simulate more generally applicable decryption failures. We use those to evaluate our recovery method described in Chapter 6. By additionally introducing incorrect inequalities in several ways, we obtain inequalities as they occur in a variety of attacks, e.g. [PP21, BDH+21b, HPP21, DHP+22, Wei22]. Moreover, the evaluation of Chapter 6 results in security estimates for partial attacks for the strategy described Chapter 5.

### 7.2.1 Simulation

To simulate a fault or obtain decryption failure inequalities, we use a modified PQClean [KSSW22, PQClean] implementation. We added a manipulated decapsulation routine that has a second ciphertext as a parameter; this second ciphertext is the ciphertext that the re-encrypted one is compared against. Thereby, we may simulate a fault during the storage of the submitted ciphertext. The remainder of the attack is implemented in Python [PyRef] calling the modified PQClean implementation.

To evaluate the attack presented in Chapter 5, we first honestly[2] generate a key pair using the key generation routine which is the target of our attack. Then, we honestly sample ciphertexts using the encapsulation routine with the public key; we store the shared secret that the encapsulation routine returns. The generated ciphertexts are manipulated such that they carry an additional $\lceil \frac{q}{4} \rfloor$ term in the first coefficient; the original ciphertext is stored as well. We check if ciphertext and bit-fault requirements are met before submitting both ciphertexts to the manipulated decapsulation routine. If the ciphertext causes a decryption failure, the decapsulation routine results in a different shared secret than the one obtained from the encapsulation routine.

The implementation cannot only be used to simulate the described attack but also gives a general decryption failure oracle by using the manipulated decapsulation routine. To evaluate our method described in Chapter 6, we merely need to add potentially incorrect inequalities. We add a correctness probability to every inequality and – according to the probability – flip the

[2]This means without manipulation by a call to the key generation.

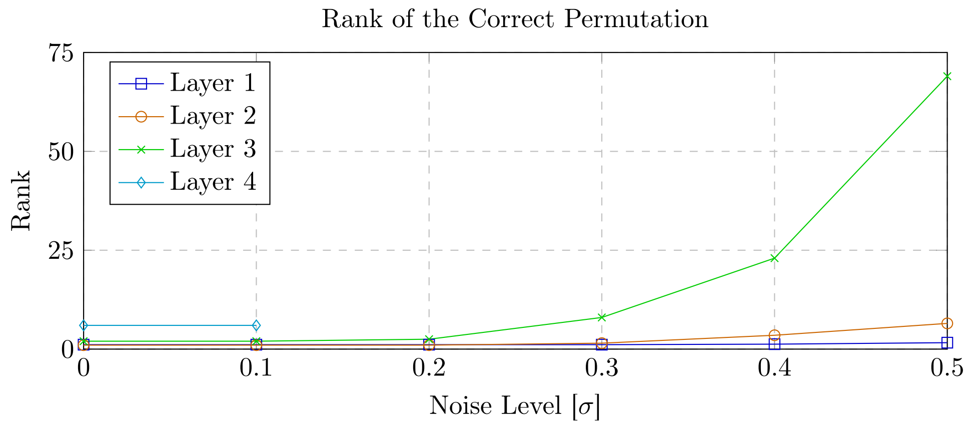


Figure 7.6: The rank of the correct permutation as stated in [HSST23] per noise level and layer when using exact permutation matching.

Table 7.5: Runtimes as stated and depicted in [HPP21] on an Intel(R) Xeon(R) Gold 6242 per security level with a fixed number of inequalities in minutes.

| Kyber security level | Iterations | 32 threads | 8 threads |
|---|---|---|---|
| Kyber512 (6000 inequalities) | 6.8 | 3.25 | 9.3 |
| Kyber768 (7000 inequalities) | 6.75 | 6.7 | 18.6 |
| Kyber1024 (9000 inequalities) | 9 | 16.9 | 39.25 |

sign of the required number of inequalities. For our evaluation, we use three main scenarios with incorrect inequalities: Firstly, we consider the case that all inequalities are incorrect. Secondly, we simulate an attack such as [HPP21] or [Del22] under the assumption of an unreliable fault but with certainly correct decapsulation successes. Thirdly, we assume that half of the inequalities are potentially incorrect and the other half of the inequalities are potentially incorrect.

### 7.2.2 Fault-Enabled Chosen-Ciphertext Attacks

We evaluate the method of Chapter 5 in terms of success rate and recovered coefficients per fault. In this case, we assume the fault to work in all cases with no unreliability. Results for unreliable faults are stated in the next section in a more general setting; moreover, an attacker may use decryption success only. In this case, the number of required faults is increased by a factor of $2\frac{1}{f}$, where $f$ is the fault success rate. Note that faults in Section 7.2.2 correspond to inequalities in Section 7.2.3.

The amount of recovered coefficients is shown in Figure 7.7 and the success rate in Figure 7.8. Additionally, we state the runtime of the belief propagation in Table 7.5 as given in [HPP21]. The attack presented in Chapter 5 is improved by the recovery method stated in Chapter 6; the average number of required faults for an adversary that can run BKZ-70 are shown in Table 7.6 (also as stated in [HMS+23]).

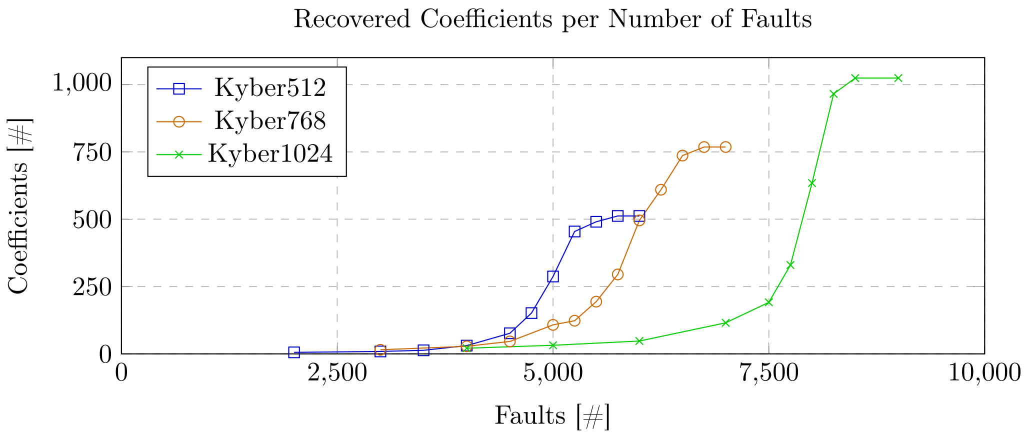


Figure 7.7: Average number of recovered coefficients per administered faults in [HPP21]. Note that recovering 512, 768, or 1024 coefficients, for the respective Kyber security level, is sufficient.

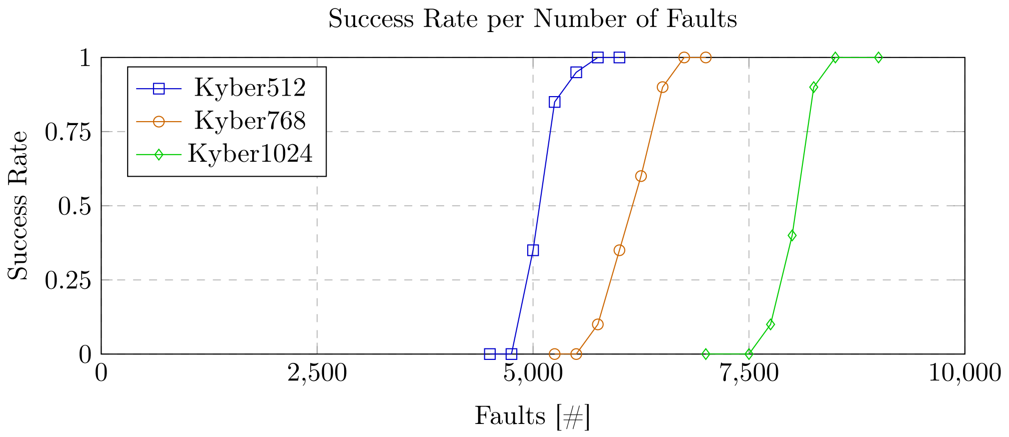


Figure 7.8: Average success rate per administered faults in [HPP21].

### 7.2.3 Improved Recovery

We state the results regarding the improved recovery method presented in Chapter 6 in terms of BKZ-$\beta$ per number of inequalities (for BKZ-$\beta$ and lattice reduction in general, see Section 2.2.1). The inequalities are assumed (and sampled under this assumption) to be the result of an attack where the inequalities coefficients are uncorrelated to the secret key coefficient. We first state the results for a setting in which all inequalities are certainly correct, i.e., $p_i = 1$ for all $i \in \{0, \dots, m-1\}$ where $m$ is the number of inequalities and $p$ is the correctness probability. Then, we give the results for three settings that contain inequalities with $p_i < 1$.

For correct inequalities, we differentiate between two settings: With and without ciphertext filtering. The results are shown in Figure 7.9.

For potentially incorrect inequalities, the first scenario, shown in Figure 7.10, is one where all inequalities are incorrect with $p_i = p$ for $p \in \{0.6, 0.7, 0.8, 0.9\}$. This corresponds for example to a side-channel attack in which the attacker may detect a decryption failure correctly in $p$ of the cases; an example would be the attacks of [Wei22] or attacks similar to [BDH+21b, DHP+22].

Table 7.6: Approximate required number of faults with fault success probability $f$ assuming the adversary can run BKZ-70 as stated in [HMS+23].

| Method | Correct | $f = 0.9$ | $f = 0.8$ | $f = 0.7$ | $f = 0.6$ |
|---|---|---|---|---|---|
| Number of Faults | 5500 | 8000 | 9000 | 11000 | 15000 |

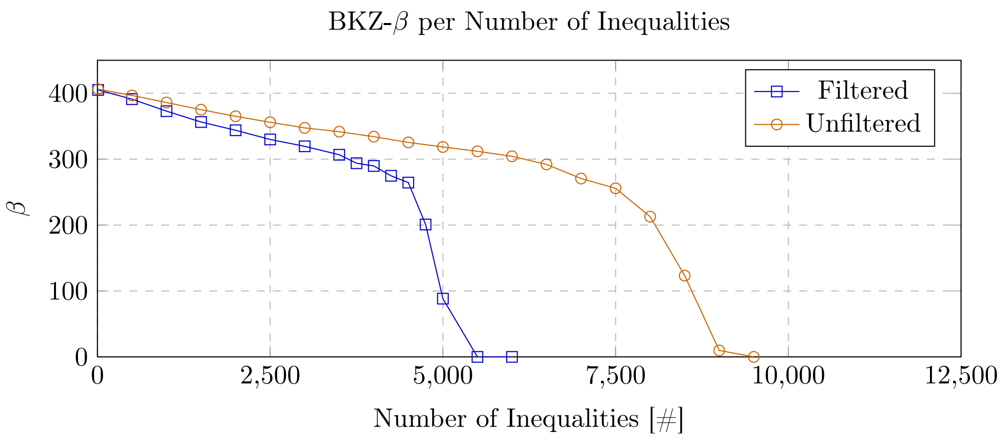


Figure 7.9: Security level per number of inequalities with and without ciphertext filtering as stated in [HMS+23].

Another example would be a fault attack such as [PP21] in which a fault works in $p$ of the cases and no chosen ciphertext is used.

The second scenario, shown in Figure 7.11, corresponds to the attacks of [HPP21] or [Del22] when assuming an incorrect fault but certainty about the result in the case of a decapsulation success. In this scenario, the adversary uses a fault with success rate $f$, and an inequality derived from a decapsulation failure is potentially incorrect; an inequality derived from a decapsulation success has correctness $p = 1$.

In the chosen-ciphertext attacks used in the mentioned attacks, the amount of decryption failures is approximately the same as the number of successes, but as the fault fails in $f$ of the cases, not all decryption successes can be identified. An observed decapsulation failure may either be an actual decryption failure with a successful fault (with probability $\frac{f}{2}$), a decryption success with a failed fault (with probability $\frac{1-f}{2}$), or a decryption failure with an unsuccessful fault (with probability $\frac{1-f}{2}$). Therefore, under the condition of having observed a decapsulation failure, we may deduce that a decryption failure happened with probability

$$p = \frac{(f/2) + (1-f)/2}{(f/2) + ((1-f)/2) + (1-f)/2} = \frac{1}{2-f}. \tag{7.5}$$

The final scenario, Figure 7.12, is stated mainly for evaluation purposes and features half the inequalities potentially incorrect with probability $p_{\text{half}} \in \{0.6, 0.7, 0.8, 0.9\}$. To the best of our knowledge, no currently published attack corresponds to this setting, but it allows for better intuition on the effect of incorrect inequalities per total number of inequalities in a mixed setting (i.e., a setting containing correct and incorrect inequalities).

We also compare our work to the work of [PP21] and [Del22]. Their presented methods do not allow for security estimates – one of our main contributions – but we may compare the

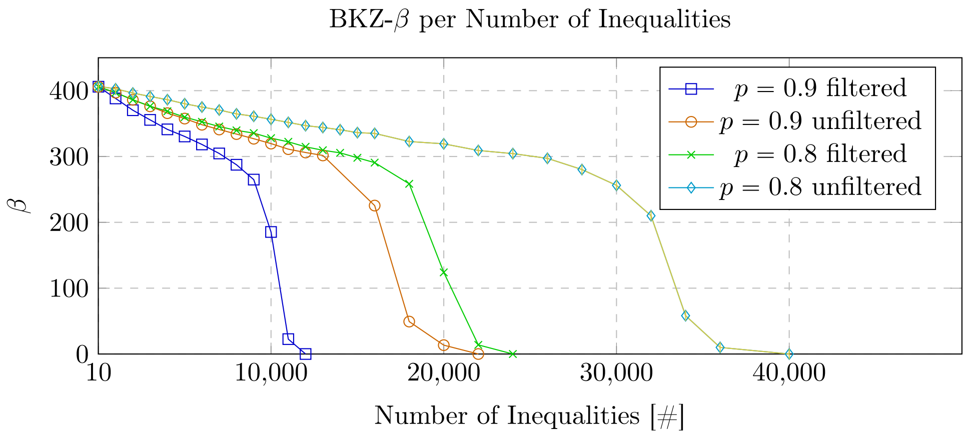


Figure 7.10: Security level per number of inequalities when all inequalities are potentially incorrect with probability $p$ as stated in [HMS+23].

Table 7.7: Comparison of recovery methods with different fault success probabilities (denoted by $f$) and correctness probability (denoted by $p$) as depicted in [HMS+23].

| Method | Correct | $p = 0.9$ | $p = 0.8$ | $f = 0.9$ | $f = 0.6$ |
|---|---|---|---|---|---|
| Pessl and Prokop [PP21] | 7500 | n.a. | n.a. | n.a. | n.a. |
| Delvaux [Del22] | 8500 | 34000 | not solved | 12000 | 21000 |
| This work | 5500 | 12000 | 24000 | 8000 | 15000 |

required number of inequalities with different correctness probabilities. A comparison of required inequalities, as stated in [HMS+23], is given in Table 7.7.

### 7.2.4 Comparison to Prior Work

Several previous attacks [PP21, BDH+21b, DHP+22], as well as the follow-up attack [Del22] to our work, exploit decryption failures. The main improvement to previous attacks in the case of [HPP21] is the relaxed fault model (only requiring an unreliable fault) and that the attack has a large attack surface, which does not depend on a single routine or a faulty comparison. Additionally, we only target public data and points in execution time, some of which did not obviously require protection. These properties are inherited by [Del22] as their work builds upon our attack strategy and improves it further. Our improvements in Chapter 5 are not quantified in this section, as all of these attacks obtain the same kind of inequalities; the main contribution lies in the properties of the attack itself, i.e, in the enlarged attack surface and the relaxed requirements on the fault injection.

In terms of recovery methods, we mainly compare our work to the recovery method of [Del22] as they are the most competitive among the methods which have been used in practice in this setting. We require about 58% to 68% of the inequalities required by [Del22] depending on the setting. This means that we may recover the secret key with slightly more than half of the data required by [Del22], which not only further reduces the requirements on the attacker, but may also circumvent certain countermeasures proposed in [PP21]. In addition, we provide security estimates that consider a practical recovery method, which previously were not available. These

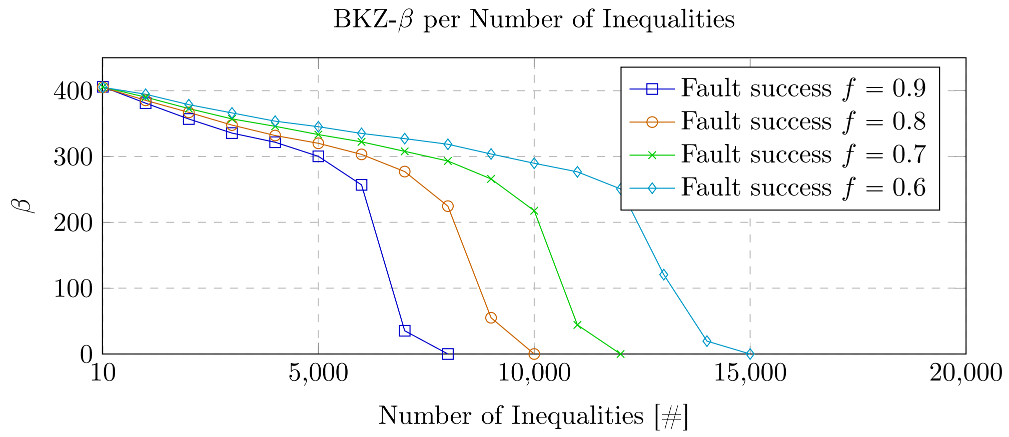


Figure 7.11: Security level per number of inequalities in an attack with fault success rate $f$ and resulting probability $p = (1 - f)/2$ for decryption failures as stated in [HMS+23].

enable the design of countermeasures as well as appropriate evaluation models that consider the usage of lattice reduction after a partially successful attack.

In summary, the main improvements of [HPP21] are not expressed in the evaluations of this section but lie in the nature of the attack itself. In comparison to previous attacks, [HPP21] only requires an unreliable fault, i.e., even if the fault fails in a certain fraction of cases or targets an incorrect value, the attack can still be carried out. This is because with out attack strategy, a successful decapsulation can only happen with a successfully applied fault. Thus, failed faults can be detected and do not prevent the recovery of the secret key.

The improvements of [HMS+23] are a reduced amount of required inequalities and security estimates in case of partial attacks. The former, depending on the setting, reduces the number of required faults or measurements, may circumvent countermeasures, or increases the noise tolerance. The latter gives a basis for designing countermeasures and quantifies the effect of inequalities by providing a translation from required computational power to required inequalities.

## 7.3 Summary

We evaluate our work under the assumptions of the attacker models described in the respective previous sections and simulate our attacks and recovery methods in software. To evaluate the template attack of Chapter 4, we employ the noisy Hamming weight model; to evaluate the impact of countermeasures, we additionally state the noise level and entropy increase due to the countermeasure in the presence of our adaptation.

For Chapter 5, we assume that an attacker may apply a one-bit flip, set, or reset to either the stored ciphertext or the re-computed ciphertext. We simulate the attack and state the success rate as well as the recovered coefficients per number of inequalities. In this case, the main contribution of our work lies in the methodology and expresses itself in the type of faulted data, in the multiple locations that we may target, and in the fact that we can use an unreliable fault.

Chapter 6 is evaluated in a similar model as Chapter 5, but in addition, we assume that some decryptions are classified incorrectly. Moreover, we also provide security estimates for a variety of potential or previous attacks.

Our evaluations show an improvement on the state of the art in all evaluated settings. Using

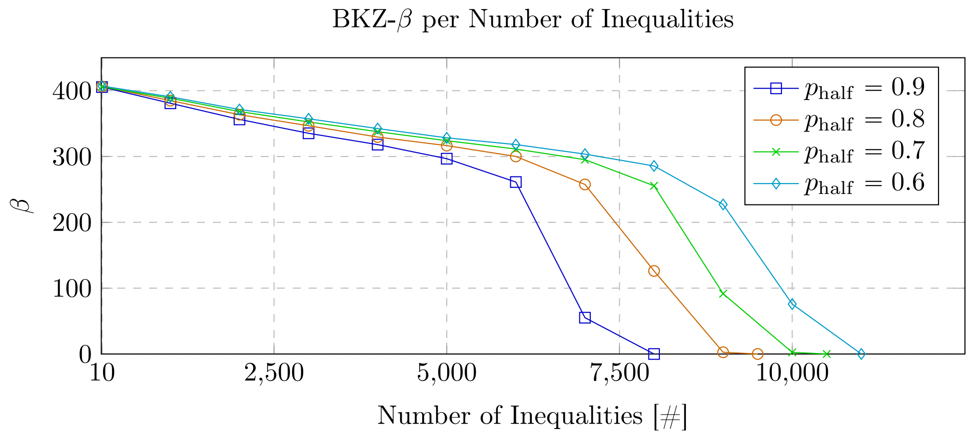


Figure 7.12: Security level in BKZ-$\beta$ per number of inequalities when half of the inequalities are incorrect with probability $p_{\text{half}}$ as stated in [HMS+23].

the attack strategy presented in Chapter 4, we improve upon the noise tolerance the adversary has to achieve for a successful key recovery when targeting the secret key during the inverse NTT. In this setting, previous work required the adversary to achieve a noise level of $\sigma \leqslant 0.6$ for a positive success rate, while we only require the attacker to achieve a noise level from $\sigma \leqslant 1.7$ to $\sigma \leqslant 3.1$, depending on the security level[3]. Thereby, the requirements on the attacker are relaxed. This means targeting the NTT is more realistic using the presented methods. In addition, we take shuffling countermeasures into account and show that our attacks can be adapted in certain settings, depending on the security level of the countermeasure.

In the case of Chapter 5, the main advantage lies in the attack itself, namely the reduced requirements on the attacker. When using our attack strategy, the adversary is only required to be able to apply an unreliable fault in one of several possible locations over a long execution time. Nevertheless, we also improve upon the number of required inequalities for full key recovery, and we greatly reduce the memory requirements of previous methods.

Our key recovery method in Chapter 6 improves upon previous algorithms in terms of required inequalities. The difference is particularly visible in cases where incorrect inequalities have to be considered or no ciphertext filtering is in place. For full key recovery, the number of inequalities the attacker needs is reduced by a factor of 0.73 to 0.35, and we can recover the secret key in settings where previous methods failed. This lowers the requirements on the attacker and may circumvent previously proposed countermeasures. In addition, we offer security estimates for partially successful attacks; this means that the remaining bit security after having obtained any number of inequalities is evaluated.

[3]In the unmasked k-trace attack setting.

# Chapter 8

# Conclusions and Outlook

The standardization of post-quantum cryptography, with Kyber as the current main candidate for key exchanges, is imminent. In the near future, Kyber will likely be used in a vast variety of use cases. Therefore, understanding the vulnerability of Kyber – and lattice-based cryptography in general – in regard to side-channel and fault attacks is crucial. In our work, we present and analyze attack strategies, methods and techniques for current and future attacks, including vulnerability assessment on major building blocks. We conclude this thesis by summarizing our findings, revisiting the research questions, and outlining open questions that result from our work.

## 8.1 Main Findings

We first give an overview of the strategies and techniques used in our work and then revisit our research questions and summarize the answers provided in previous chapters.

### 8.1.1 Attack Strategies

The attack strategies presented in Chapter 4 and Chapter 5 rely on the combination of a chosen-ciphertext attack with either side-channel analysis or a fault attack. Compared to pure side-channel or fault attacks, by using a chosen ciphertext, the adversary may (further) influence the computation taking place in an attacked device. Similar kinds of chosen-ciphertext attacks have previously been used to enable implementation attacks (see e.g. [GJN20, BDH+21b]).

We provide two new attack strategies using chosen-ciphertexts in the context of lattice-based schemes; the first reduces the entropy during the computation of the Number Theoretic Transform (NTT), and the second potentially introduces a decryption failure that can be observed when combined with a fault. We thereby improve previous attacks by allowing for a greater noise tolerance, requiring a less reliable[1] fault, and enhancing the attack surface.

Our attack strategies can be used to improve, adapt, and construct different attacks as well. The method to reduce the entropy arising during the inverse NTT is not dependent on a subsequent template attack. It could be used with any kind of attack that targets the inverse NTT and benefits from reduced entropy. The strategy of using manipulated ciphertext and a correcting fault can be used with other types of invasive implementation attacks. Our strategy has already been used and extended in the follow-up work of Delvaux [Del22][2], who shows that more locations are potentially vulnerable using this strategy.

[1]A fault that can fail and not target the correct or any value.

[2]Note that [Del22] was published after [HPP21] which is part of this thesis.

In conclusion, our methods are not limited to the specific attacks presented in this thesis, and our attack strategies have already enabled novel attacks. Furthermore, awareness of potential vulnerabilities is necessary to adequately protect cryptographic operations and devices.

### 8.1.2 Attack Techniques

All presented methods and techniques rely on belief propagation, which enables our attacks in the first place. Since having been proposed for the usage in Soft Analytical Side-Channel Attacks (SASCAs) [VGS14], belief propagation has been used in a variety of side-channel attacks, e.g. in [PPM17, GRO18, PP19, KPP20]. We additionally use belief propagation in the context of a fault attack retrieving decryption failure information; this is similar to the usage of belief propagation for the decoding of Low-Density Parity Check (LDPC)-codes [Gal62]. For this use case, we show how belief propagation may be used to efficiently (in terms of time and memory complexity) solve a system of inequalities. This – as well as our belief-propagation-based adaptations to countermeasures – further stresses the effectiveness of coding-theoretic perspectives in side-channel and fault attacks on lattice-based cryptography.

Our techniques again apply to a wider variety of situations: In the context of shuffling countermeasures, we show that belief propagation may be extended to adapt an attack to countermeasures. This adaptation is not dependent on the attack we evaluated it on; instead, it may be used more generally in the context of belief-propagation-based attacks to adapt to hiding countermeasures. Our results are in line with previous work (see Section 3.1.7) that shows how shuffling countermeasures may be circumvented in symmetric cryptography.

We propose using belief propagation to solve inequalities arising from decryption failure information. This technique outperforms previous approaches in terms of required inequalities and has already been used in [DHP+22] and [Wei22]. We also show how approaches such as belief propagation may be combined with lattice reduction by the integration of statistical output into a lattice problem. We explain how belief propagation can be made error-resistant in the context of decryption failure information. This technique is also not limited to a specific situation used for evaluation but may be applied whenever side-channel analysis data may not be classified correctly in an attack using belief propagation. Further, attacks not yet relying on belief propagation may use our technique to work with potentially incorrectly classified data.

Moreover, our work shows how to make use of statistical information while also leveraging the algebraic data from the public key equation in lattice-based schemes. This is achieved by the integration of probability information into the lattice problem derived from the public key equation. Thereby, we achieve an improved recovery and may provide security estimates for partial attacks, i.e., attacks that did not obtain sufficient information for a full key recovery. This technique likely applies to a wide variety of belief-propagation-based attacks; it may already be used for improvements and to provide security estimates for the works presented in [BDH+21b, PP21, DHP+22, Del22], and it likely also applies to [HHP+21].

In conclusion, the approaches we presented apply beyond the scope we chose for evaluation. Our methods allow adapting to countermeasures, and they enable current and potential future attacks. In addition, our methods are required to deploy appropriate countermeasures that achieve the required security levels and to understand the security of several types of attacks.

### 8.1.3 Vulnerability Analysis

By applying our attack strategies and techniques and evaluating them on the example of Kyber, we provide vulnerability analyses of major building blocks of modern lattice-based schemes. This includes the number theoretic transform, the error correction, and the Fujisaki-Okamoto

(FO)-transform. These are common building blocks of lattice-based schemes (not only of Kyber), and as the National Institute of Standards and Technology (NIST) selected lattice-based schemes for standardization, understanding their security is crucial for the migration the post-quantum cryptography.

#### Number Theoretic Transform

We provide analysis in terms of required attacker capabilities to perform a template attack on the inverse NTT. Our assessment is given in terms of noise tolerance, i.e., in the standard deviation that an attacker needs to achieve for the probability distributions obtained from the template attack. In addition, we provide an analysis of the impact of a standard masking countermeasure as proposed in [RRC+16, OSPG18] and hiding countermeasures proposed in [RPBC20]. We state the decrease in noise tolerance when these countermeasures are in place and the attack is adapted accordingly.

#### Error Correction

We evaluate the vulnerability of the error correction in terms of information an attacker gains from potentially causing and then observing a decryption failure. We provide an evaluation of the required number to fully recover the secret key, security estimates for partial attacks, and an information-theoretic analysis resulting in an upper bound on the obtained information. These estimates in theory and practice are important not only to understand the security of modern lattice-based schemes but also to deploy appropriate countermeasures that allow for sufficiently secured cryptographic operations and devices. Otherwise, the impact of attacks that exploit decryption failures could be underestimated and countermeasures could fail to reach the targeted security level.

#### FO-Transform

Our analysis in regard to the FO-transform points out that public data may as well be vulnerable to fault attacks targeting the secret key. Even with an unreliable fault on public data alone, we may turn the FO-transform into a decryption failure oracle and obtain reliable information about the secrets. Our assessment provides additional potential points of attack which need to be protected to achieve appropriately secured implementations.

### 8.1.4 Revisiting the Research Questions

In Section 1.1.2, we identified several research questions; given the previously described findings, we may now answer our research questions.

**To what extent is the number theoretic transform vulnerable to side-channel analysis?**

Our work in Chapter 4 shows that the entropy during the inverse NTT may be reduced using a chosen-ciphertext attack. A ciphertext component that is compressible, i.e., unaffected by compression and subsequent decompression, and NTT-sparse, i.e., sparse in NTT-domain, is computed using lattice reduction. This ciphertext is submitted to the device under attack; the device decompresses the ciphertext, and the component is transformed to NTT domain and multiplied with the secret. The result of this computation is the input to a subsequent inverse NTT. As the component is NTT-sparse, so is the input to the NTT; it therefore greatly reduces the entropy and increases the noise tolerance when performing a side-channel analysis – in our case a template attack – on the inverse NTT.

Our strategy enables far more noise tolerant – and therefore more realistic – attacks on the inverse NTT. The technique is easily deployed[3] and only requires widely available resources in terms of computational power. Additionally, all our computations are performed offline, and the ciphertext generation has to be carried out only once. Compared to previous attacks, the additional requirements on an adversary are therefore minimal. As previous attacks have been carried out on physical devices, our attack is realistic using a common setup for template attacks. Moreover, we show that first countermeasures may be circumvented using adaptations either in software or in software and attacker model.

We conclude that a combined side-channel and chosen-ciphertext attack allows for high noise tolerance. Circumventing shuffling countermeasures is possible using our techniques but reduces the noise tolerance. Moreover, our attack strategy allows reducing the entropy in an arbitrary implementation attack on the NTT that can be combined with a chosen-ciphertext attack. Our work suggests that public data should be protected by masking countermeasures as well and provides further evidence that ciphertexts should be checked for irregularities. We thus highlight and evaluate the vulnerabilities of the NTT in modern lattice-based cryptography.

**How can implementation attacks target the error correction and exploit decryption failure leakage in the presence of an FO-Transform?**

In Chapter 5, we present the combination of a chosen ciphertext potentially causing a decryption failure with a fault attack. Our attack strategy causes the FO-transform to function as a decryption failure oracle for the submitted ciphertext – observing a decryption failure allows deriving an inequality in the secret key. These inequalities are solved using belief propagation, which outperforms previous methods in terms of required information.

Our strategy allows for the usage of an unreliable fault and enlarges the attack surface compared to previous attacks. Regardless of the fault success rate, the attacker may still recover information about the secret key – merely the required number of required faults is increased. In addition, the fault may be applied to public data, which is less likely to be protected appropriately. Further, the fault may be applied at a variety of places and is therefore not impacted by a single currently known countermeasure; in particular, it is not prevented by the shuffling countermeasures that prevented previous invasive attacks.

We conclude that our combined method consisting of a chosen ciphertext and a fault attack offers advantages over previous methods and requires careful protection of the decapsulation routine in a wide variety of places. We show that FO-transform is vulnerable to this class of attacks, which use a fault to obtain a decryption failure oracle.

Our analysis shows that even an unreliable fault can be used to target the error correction. Therefore, our work shows that protecting the error correction against side-channel attacks or fault attacks in specific locations is insufficient to prevent attacks exploiting decryption failure leakage. In addition to protecting against previous attacks, further countermeasures that protect the submitted and recomputed ciphertexts from manipulation are required.

**Which techniques allow for key recovery from (partially) leaked information?**

In Chapter 6, we present a recovery method to retrieve the secret key from decryption failure information. We first employ a belief propagation method, which is similar to the algorithm used in Chapter 5, but enhance it with additional probability information to achieve error resistance. The probabilistic output of the belief propagation method is subsequently integrated into a lattice

[3]It merely requires a computationally inexpensive preprocessing step.

problem, similar to the primal attack, which may be solved using lattice reduction. In addition, we provide an information-theoretic analysis of the obtained inequalities.

Our method requires a lower number of inequalities compared to previous methods, is error-resistant, and allows for security estimates. Error resistance enables attacks in which information cannot be classified with certainty – often a realistic assumption –, and our method allows combining this property with a reduced amount of required information. The availability of security estimates is important to deploy countermeasures that prevent decryption failure attacks within a certain security level. Thereby, our technique improves upon the state of the art in recovery methods for decryption failure information and offers valuable results for the deployment of appropriate countermeasures against decryption failure attacks. Moreover, our method generally explains how belief-propagation-based attacks may be combined with lattice reduction.

We conclude that a hybrid method consisting of belief propagation, a statistical/coding theoretic approach, and a lattice-based attack, an algebraic approach, outperforms previous techniques. A hybrid approach enhances the understanding of previous attacks and provides an assessment of their impact that enables the deployment of appropriate countermeasures.

## 8.2 Future Work

Our work opens several questions which are to be answered by future work. In the following, we address those questions sorted by the attack domain.

### 8.2.1 Attacks on the NTT

In regard to attacks on the (inverse) NTT presented in this thesis and [HHP+21], the most pressing open question are more comprehensive countermeasures in real-world use cases. We did not provide any evaluation in this regard due to the lack of a production-ready protected implementation used in a real-world scenario.

We adapted the attacks described in Chapter 4 to a common masking scheme as well as to hiding countermeasures proposed in [RPBC20]. Other masking schemes might also protect the public input including $\mathbf{u}$. In this case, we lose the option to apply zeros using a chosen ciphertext. While our attack is not fully prevented, its main advantage compared to previous attacks is lost unless further adaptations – yet unknown – are taken. We, exemplary, adapted our attack to the hiding countermeasures published in [RPBC20] and a general masking scheme. The work of [RPBC20] additionally defines a specific masking scheme for an (inverse) NTT. The adaptation and the impact of such schemes are out-of-scope for our work. Again, attacking such a countermeasure is interesting mainly on a real device that runs a protected implementation using a variety of countermeasures.

In the case of partial attacks, we only consider attacks that can fully recover a sub-key. While we give security estimates based on the sub-key, neither recovered coefficients nor statistical information in the form of belief propagation output is taken into account. We conjecture that a method similar to the one explained in Chapter 6 allows improving upon the results and, more importantly, offers more precise security estimates for partial attacks.

Previous attacks, including ours as well as [PPM17] and [PP21], perform a profiled attack, which is a stronger attacker model. Another interesting question concerns the application of our technique to non-profiled attacks such as [MBB+22], which could achieve improvements. In addition, applying recent advances in deep learning in the area of side-channel analysis, e.g. [NDGJ21, NWDP22, DNG22] to our setting could lead to a more relaxed attacker model and pose an increased threat to protected implementations.

### 8.2.2 Decryption Failure Attacks

The attack presented in this thesis and [HPP21] has already been extended by the attack presented in [Del22]. In [Del22], the fault model was further relaxed, and the attack surface increased; i.e., their attack can obtain the secret key with less reliable data and works with a fault that can be applied at more locations. Nevertheless, the attack surface and the fault model can almost certainly be improved even further. Additionally, this type of attack can be extended to different lattice-based schemes that achieve IND-CCA2 security using an FO-transform.

The main open question concerns comprehensive countermeasures: A combination of countermeasures likely prevents our attack in its current form, but it is not unlikely that adaptations exist – the work of [Del22] already improves upon the resistance against countermeasures. In [AKSV22], a general countermeasure through ciphertext verification by a third party was presented, but it only applies to use cases where a trustworthy third party is available. The authors of [PP21] already propose shutting down after a certain amount of decryption failures. However, they also note that this leaves a device vulnerable to Denial of Service (DoS) attacks. A more comprehensive and generally applicable countermeasure to decryption failure attacks that neither requires a third party nor allows for DoS attacks is yet missing.

Various models and notions to prove the side-channel security of cryptographic schemes exist. Common models such as $t$-probing security [ISW03] or $t$-strong non-interference [BBD+16], assume that the adversary can observe $t$ intermediate variables. The practicality in some real-world applications of $t$-probing security has previously been questioned (see, e.g., [KGM+21]). In decryption failure attacks targeting the comparison, the attacker is merely interested in whether a decryption failure has occurred or not – accessing single intermediates (or their Hamming weights) is not necessarily required, and the adversary is only interested a single bit of information. Therefore, in this case, a horizontal higher-order attack can be less costly than in other attacks, and different models to assess side-channel security of the comparison of the FO-transform in the presence of decryption failures could be more realistic.

### 8.2.3 Key Recovery Methods

The key recovery method presented in this thesis and [HMS+23] outperforms previous approaches in the area of recovering the secret key from decryption failure information by combining belief propagation with lattice reduction techniques. Nevertheless, we note that our method does not reach the information-theoretic optimum; the code given by inequalities is likely a "good" code (see [CG90]), and the secret key is determined by fewer inequalities (c.f. Section 6.2.3) than we require. While the information-theoretic optimum is a lower bound unlikely to be reached – as decoding random linear codes is NP-hard (see [BMT78]) –, there could be further improvements in terms of lowering the amount of inequalities.

Moreover, our method in its current form is unsuitable for the inequalities arising in failure-boosting attacks such as [FKK+22]. These inequalities have coefficients strongly correlated to the secret key coefficients and may be solved using the method of [FKK+22] or [DGHK22]. Our method is not applicable to inequalities of [FKK+22], and the techniques used to solve those inequalities seems not to apply to the inequalities we considered. An interesting question is whether a unified approach exists that allows solving both kinds of inequalities.

Another important topic is the question for different types of leakage on the error term. Different leakage could lead to difference in the required recovery method. In these question different applications of recovery methods based on belief propagation or algebraic methods could be beneficial.

### 8.2.4 Post-Quantum Signature Schemes

Due to the store-now-decrypt-later threat, secured post-quantum Key Encapsulation Mechanisms (KEMs) are arguably a more pressing matter than signature schemes. However, the signature scheme Dilithium [DKL+18, BDK+21a] will be standardized [NistCfpv4], and has already seen side-channel and fault analysis (see, e.g., [RJH+18, MUTS22, IMS+22, BVC+23, EAB+23]) as well as discussion on countermeasures (see, e.g., [MGTF19, ABC+23, CGTZ23]). Moreover, NIST extended the standardization process for signature schemes [NistSig]. Understanding vulnerabilities and the security against physical attacks of these schemes has been out-of-scope for this thesis, but it is crucial to deploy protocols that also rely on post-quantum signature schemes.

### 8.2.5 Physical Attacks and Deep Learning

Deep learning and neural networks have been proven to enable and simplify physical attacks. This already included several post-quantum schemes (c.f. Section 2.3.1). These techniques could drastically lower the required expertise to carry out side-channel attacks. In addition, neural networks could reduce difficulties that arise in profiled attacks where the device used for profiling differs from the targeted device. Moreover, neural networks that learn from side-channel information while taking the underlying mathematical structure into account could lead to new types of attacks. This could be particularly effective if neural networks can be combined with belief propagation in side-channel attacks. Therefore, these developments make it necessary to assess the impact of attacks and to revise the corresponding threat models.

# Bibliography

[AAB+19] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwabe, and Douglas Stebila. *NewHope – Submission to the NIST post-quantum project.* 2019. URL: https://newhopecrypto.org/data/NewHope_2019_07_10.pdf.

[AB75] Ramesh C. Agarwal and C. Sidney Burrus. "Number theoretic transforms to implement fast digital convolution". In: *Proceedings of the IEEE* 63 (1975), pp. 550–560.

[ABC+23] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. "Protecting Dilithium against Leakage Revisited Sensitivity Analysis and Improved Implementations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.4 (2023), pp. 58–79. DOI: 10.46586/TCHES.V2023.I4.58-79. URL: https://doi.org/10.46586/tches.v2023.i4.58-79.

[ABD+] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-KYBER reference implementation.* Available at https://github.com/pq-crystals/kyber/, commit '8e9308bd0f25fa698e4f37aba216249261f8b352', last accessed 2021-04-11.

[ABD+19a] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 1.0).* 2019. URL: https://pq-crystals.org/kyber/data/kyber-specification.pdf.

[ABD+19b] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 2.0).* 2019. URL: https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf.

[ABD+21a] Erdem Alkim, Joppe W. Bos, Leo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. *FrodoKEM Learning With Errors Key Encapsulation.* 2021. URL: https://frodokem.org/files/FrodoKEM-specification-20210604.pdf.

[ABD+21b] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.02).* 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[ABG+22] Melissa Azouaoui, Olivier Bronchain, Vincent Grosso, Kostas Papagiannopoulos, and François-Xavier Standaert. "Bitslice Masking and Improved Shuffling: How and When to Mix Them in Software?" In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.2 (2022), pp. 140–165. DOI: 10.46586/tches.v2022.i2.140-165. URL: https://doi.org/10.46586/tches.v2022.i2.140-165.

[ACC+21] Erdem Alkim, Dean Yun-Li Cheng, Chi-Ming Marvin Chung, Hülya Evkan, Leo Wei-Lun Huang, Vincent Hwang, Ching-Lin Trista Li, Ruben Niederhagen, Cheng-Jhih Shih, Julian Wälde, and Bo-Yin Yang. "Polynomial Multiplication in NTRU Prime Comparison of Optimization Strategies on Cortex-M4". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.1 (2021), pp. 217–238. DOI: 10.46586/tches.v2021.i1.217-238. URL: https://doi.org/10.46586/tches.v2021.i1.217-238.

[ACD+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. "Estimate All the {LWE, NTRU} Schemes!" In: *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings.* Ed. by Dario Catalano and Roberto De Prisco. Vol. 11035. Lecture Notes in Computer Science. Springer, 2018, pp. 351–367. DOI: 10.1007/978-3-319-98113-0_19. URL: https://doi.org/10.1007/978-3-319-98113-0_19.

[ACLZ20] Dorian Amiet, Andreas Curiger, Lukas Leuenberger, and Paul Zbinden. "Defeating NewHope with a Single Trace". In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings.* Ed. by Jintai Ding and Jean-Pierre Tillich. Vol. 12100. Lecture Notes in Computer Science. Springer, 2020, pp. 189–205. DOI: 10.1007/978-3-030-44223-1_11. URL: https://doi.org/10.1007/978-3-030-44223-1_11.

[ADH+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. "The General Sieve Kernel and New Records in Lattice Reduction". In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II.* Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11477. Lecture Notes in Computer Science. Springer, 2019, pp. 717–746. DOI: 10.1007/978-3-030-17656-3_25. URL: https://doi.org/10.1007/978-3-030-17656-3_25.

[ADPS16a] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "NewHope without reconciliation". In: *IACR Cryptol. ePrint Arch.* (2016), p. 1157. URL: http://eprint.iacr.org/2016/1157.

[ADPS16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum Key Exchange - A New Hope". In: *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.* Ed. by Thorsten Holz and Stefan Savage. USENIX Association, 2016, pp. 327–343. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.

[AGVW17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. "Revisiting the Expected Cost of Solving uSVP and Applications to LWE". In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I.* Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. Lecture Notes in Computer Science. Springer, 2017, pp. 297–322. DOI: 10.1007/978-3-319-70694-8_11. URL: https://doi.org/10.1007/978-3-319-70694-8_11.

[Ajt96] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 99–108. DOI: 10.1145/237814.237838. URL: https://doi.org/10.1145/237814.237838.

[Ajt98] Miklós Ajtai. "The Shortest Vector Problem in $L_2$ is $NP$-hard for Randomized Reductions (Extended Abstract)". In: *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*. Ed. by Jeffrey Scott Vitter. ACM, 1998, pp. 10–19. DOI: 10.1145/276698.276705. URL: https://doi.org/10.1145/276698.276705.

[AK96] Ross Anderson and Markus Kuhn. "Tamper Resistance – a Cautionary Note new". In: *2nd USENIX Workshop on Electronic Commerce (EC 96)*. Oakland, CA: USENIX Association, Nov. 1996. URL: https://www.usenix.org/conference/2nd-usenix-workshop-electronic-commerce/tamper-resistance-cautionary-note.

[AK97] Ross J. Anderson and Markus G. Kuhn. "Low Cost Attacks on Tamper Resistant Devices". In: *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*. Ed. by Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe. Vol. 1361. Lecture Notes in Computer Science. Springer, 1997, pp. 125–136. DOI: 10.1007/BFb0028165. URL: https://doi.org/10.1007/BFb0028165.

[AKSV22] Melissa Azouaoui, Yulia Kuzovkova, Tobias Schneider, and Christine van Vredendaal. "Post-Quantum Authenticated Encryption against Chosen-Ciphertext Side-Channel Attacks". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.4 (2022), pp. 372–396. DOI: 10.46586/tches.v2022.i4.372-396. URL: https://doi.org/10.46586/tches.v2022.i4.372-396.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203. DOI: doi:10.1515/jmc-2015-0016. URL: https://doi.org/10.1515/jmc-2015-0016.

[BB05] David Brumley and Dan Boneh. "Remote timing attacks are practical". In: *Comput. Networks* 48.5 (2005), pp. 701–716. DOI: 10.1016/j.comnet.2005.01.010. URL: https://doi.org/10.1016/j.comnet.2005.01.010.

[BBC+23] Joppe W. Bos, Olivier Bronchain, Frank Custers, Joost Renes, Denise Verbakel, and Christine van Vredendaal. *Enabling FrodoKEM on Embedded Devices*. Cryptology ePrint Archive, Paper 2023/158. https://eprint.iacr.org/2023/158. 2023. URL: https://eprint.iacr.org/2023/158.

[BBD+16] Gilles Barthe, Sonia Belaıd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. "Strong Non-Interference and Type-Directed Higher-Order Masking". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM, 2016, pp. 116–129. DOI: 10.1145/2976749.2978427. URL: https://doi.org/10.1145/2976749.2978427.

[BBLP18] Daniel J. Bernstein, Leon Groot Bruinderink, Tanja Lange, and Lorenz Panny. "HILA5 Pindakaas: On the CCA Security of Lattice-Based Encryption with Error Correction". In: *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*. Ed. by Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Vol. 10831. Lecture Notes in Computer Science. Springer, 2018, pp. 203–216. DOI: 10.1007/978-3-319-89339-6_12. URL: https://doi.org/10.1007/978-3-319-89339-6_12.

[BBPS19] Madalina Bolboceanu, Zvika Brakerski, Renen Perlman, and Devika Sharma. "Order-LWE and the Hardness of Ring-LWE with Entropic Secrets". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11922. Lecture Notes in Computer Science. Springer, 2019, pp. 91–120. DOI: 10.1007/978-3-030-34621-8_4. URL: https://doi.org/10.1007/978-3-030-34621-8_4.

[BCD+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. "Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM, 2016, pp. 1006–1018. DOI: 10.1145/2976749.2978425. URL: https://doi.org/10.1145/2976749.2978425.

[BCLV17] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. "NTRU Prime: Reducing Attack Surface at Low Cost". In: *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*. Ed. by Carlisle Adams and Jan Camenisch. Vol. 10719. Lecture Notes in Computer Science. Springer, 2017, pp. 235–260. DOI: 10.1007/978-3-319-72565-9_12. URL: https://doi.org/10.1007/978-3-319-72565-9_12.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. "Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem". In: *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 553–570. DOI: 10.1109/SP.2015.40. URL: https://doi.org/10.1109/SP.2015.40.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation Power Analysis with a Leakage Model". In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 16–29. DOI: 10.1007/978-3-540-28632-5_2. URL: https://doi.org/10.1007/978-3-540-28632-5_2.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. "New directions in nearest neighbor searching with applications to lattice sieving". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. Ed. by Robert Krauthgamer.

SIAM, 2016, pp. 10–24. DOI: 10.1137/1.9781611974331.ch2. URL: https://doi.org/10.1137/1.9781611974331.ch2.

[BDH+21a] Ward Beullens, Jan-Pieter D'Anvers, Andreas Hülsing, Tanja Lange, Lorenz Panny, Cyprien de Saint Guilhem, and Nigel P. Smart. *Post-Quantum Cryptography: Current state and quantum mitigation.* 2021. URL: https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-mitigation.

[BDH+21b] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. "Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3 (2021), pp. 334–359. DOI: 10.46586/tches.v2021.i3.334-359. URL: https://doi.org/10.46586/tches.v2021.i3.334-359.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26 , 2018.* IEEE, 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032. URL: https://doi.org/10.1109/EuroSP.2018.00032.

[BDK+21a] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation (Version 3.1).* 2021. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[BDK+21b] Michiel Van Beirendonck, Jan-Pieter D'Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. "A Side-Channel-Resistant Implementation of SABER". In: *ACM J. Emerg. Technol. Comput. Syst.* 17.2 (2021), 10:1–10:26. DOI: 10.1145/3429983. URL: https://doi.org/10.1145/3429983.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. "On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)". In: *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding.* Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 37–51. DOI: 10.1007/3-540-69053-0_4. URL: https://doi.org/10.1007/3-540-69053-0_4.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. "Relations Among Notions of Security for Public-Key Encryption Schemes". In: *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings.* Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Springer, 1998, pp. 26–45. DOI: 10.1007/BFb0055718. URL: https://doi.org/10.1007/BFb0055718.

[Ber05] Daniel J. Bernstein. *Cache-timing attacks on AES.* 2005. URL: https://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[Ber21] Daniel J. Bernstein. *S-units attacks.* 2021. URL: https://cr.yp.to/talks/2021.08.20/slides-djb-20210820-sunitattacks-4x3.pdf.

[BGNT15] Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Teglia. "Multivariate High-Order Attacks of Shuffled Tables Recomputation". In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings.* Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 475–494. DOI: 10.1007/978-3-662-48324-4_24. URL: https://doi.org/10.1007/978-3-662-48324-4_24.

[BGNT18] Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Teglia. "Multivariate High-Order Attacks of Shuffled Tables Recomputation". In: *J. Cryptol.* 31.2 (2018), pp. 351–393. DOI: 10.1007/s00145-017-9259-7. URL: https://doi.org/10.1007/s00145-017-9259-7.

[BGR+21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. "Masking Kyber: First- and Higher-Order Implementations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 173–214. DOI: 10.46586/tches.v2021.i4.173-214. URL: https://doi.org/10.46586/tches.v2021.i4.173-214.

[BGRR19] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. "Assessment of the Key-Reuse Resilience of NewHope". In: *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings.* Ed. by Mitsuru Matsui. Vol. 11405. Lecture Notes in Computer Science. Springer, 2019, pp. 272–292. DOI: 10.1007/978-3-030-12612-4_14. URL: https://doi.org/10.1007/978-3-030-12612-4_14.

[BGS+08] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. "Efficient Helper Data Key Extractor on FPGAs". In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings.* Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 181–197. DOI: 10.1007/978-3-540-85053-3_12. URL: https://doi.org/10.1007/978-3-540-85053-3_12.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *ACM Trans. Comput. Theory* 6.3 (2014), 13:1–13:36. DOI: 10.1145/2633600. URL: https://doi.org/10.1145/2633600.

[BJL08] Guoan Bi, Yingtuo Ju, and Xiumei Li. "Fast Algorithms for Polynomial Time-Frequency Transforms of Real-Valued Sequences". In: *IEEE Trans. Signal Process.* 56.5 (2008), pp. 1905–1915. DOI: 10.1109/TSP.2007.913162. URL: https://doi.org/10.1109/TSP.2007.913162.

[BL17] Daniel J. Bernstein and Tanja Lange. "Post-quantum cryptography". In: *Nat.* 549.7671 (2017), pp. 188–194. DOI: 10.1038/nature23461. URL: https://doi.org/10.1038/nature23461.

[Ble98] Daniel Bleichenbacher. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1". In: *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings.* Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Springer, 1998, pp. 1–12. DOI: 10.1007/BFb0055716. URL: https://doi.org/10.1007/BFb0055716.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors". In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 575–584. DOI: 10.1145/2488608.2488680. URL: https://doi.org/10.1145/2488608.2488680.

[BMJ+21] Andrea Basso, Jose Maria Bermudo Mera, Angshuman Karmakar Jan-Pieter D'Anvers, Sujoy Sinha Roy, Michiel Van Beirendonck, and Frederik Vercauteren. *SABER: Mod-LWR based KEM (Round 3 Submission)*. 2021. URL: https://frodokem.org/files/FrodoKEM-specification-20210604.pdf.

[BMT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. "On the inherent intractability of certain coding problems (Corresp.)" In: *IEEE Trans. Inf. Theory* 24.3 (1978), pp. 384–386. DOI: 10.1109/TIT.1978.1055873. URL: https://doi.org/10.1109/TIT.1978.1055873.

[BPO+20] Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. "High-Speed Masking for Polynomial Comparison in Lattice-based KEMs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3 (2020), pp. 483–507. DOI: 10.13154/tches.v2020.i3.483-507. URL: https://doi.org/10.13154/tches.v2020.i3.483-507.

[Bra16] Matt Braithwaite. *Experimenting with Post-Quantum Cryptography*. 2016. URL: https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html.

[BS20] Olivier Bronchain and François-Xavier Standaert. "Side-Channel Countermeasures' Dissection and the Limits of Closed Source Security Evaluations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.2 (2020), pp. 1–25. DOI: 10.13154/tches.v2020.i2.1-25. URL: https://doi.org/10.13154/tches.v2020.i2.1-25.

[BsiPqc] Johannes Buchmann, Juliane Krämer, Nabil Alkeilani Alkadri, Nina Bindel, Rachid El Bansarkhari, Florian Göpfert, and Thomas Wunderer. *Bewertung gitterbasierter kryptografischer Verfahren*. 2017. URL: https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4.

[BsiQu] Frank K. Wilhelm, Rainer Steinwandt, Brandon Langenberg, Per J. Liebermann, Anette Messinger, Peter K. Schuhmacher, and Aditi Misra-Spieldenner. *Status of quantum computer development*. 2020. URL: https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Quantentechnologien-und-Post-Quanten-Kryptografie/Entwicklungsstand-Quantencomputer/entwicklungsstand-quantencomputer.html.

[BVC+23] Alexandre Berzati, Andersson Calle Viera, Maya Chartouni, Steven Madec, Damien Vergnaud, and David Vigilant. "A Practical Template Attack on CRYSTALS-Dilithium". In: *IACR Cryptol. ePrint Arch.* (2023), p. 50. URL: https://eprint.iacr.org/2023/050.

[CA69] Stephen A. Cook and Stål O. Aanderaa. "On the Minimum Computation Time of Functions". In: *Transactions of the American Mathematical Society* 142 (1969), pp. 291–314.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. “Differential Power Analysis in the Presence of Hardware Countermeasures”. In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 252–263. DOI: 10.1007/3-540-44499-8_20. URL: https://doi.org/10.1007/3-540-44499-8_20.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. “Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing”. In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 45–68. DOI: 10.1007/978-3-319-66787-4_3. URL: https://doi.org/10.1007/978-3-319-66787-4_3.

[CFGR12] Claude Carlet, Jean-Charles Faugère, Christopher Goyet, and Guénaël Renault. “Analysis of the algebraic side channel attack”. In: *J. Cryptogr. Eng.* 2.1 (2012), pp. 45–62. DOI: 10.1007/s13389-012-0028-0. URL: https://doi.org/10.1007/s13389-012-0028-0.

[CG00] Jean-Sébastien Coron and Louis Goubin. “On Boolean and Arithmetic Masking against Differential Power Analysis”. In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 231–237. DOI: 10.1007/3-540-44499-8_18. URL: https://doi.org/10.1007/3-540-44499-8_18.

[CG90] John T. Coffey and Rodney M. Goodman. “Any code of which we cannot think is good”. In: *IEEE Trans. Inf. Theory* 36.6 (1990), pp. 1453–1461. DOI: 10.1109/18.59944. URL: https://doi.org/10.1109/18.59944.

[CGTZ23] Jean-Sébastien Coron, François Gérard, Matthias Trannoy, and Rina Zeitoun. “Improved Gadgets for the High-Order Masking of Dilithium”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.4 (2023), pp. 110–145. DOI: 10.46586/TCHES.V2023.I4.110-145. URL: https://doi.org/10.46586/tches.v2023.i4.110-145.

[CHK+21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. “NTT Multiplication for NTT-unfriendly Rings New Speed Records for Saber and NTRU on Cortex-M4 and AVX2”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.2 (2021), pp. 159–188. DOI: 10.46586/tches.v2021.i2.159-188. URL: https://doi.org/10.46586/tches.v2021.i2.159-188.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. “Towards Sound Approaches to Counteract Power-Analysis Attacks”. In: *Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 398–412. DOI: 10.1007/3-540-48405-1_26. URL: https://doi.org/10.1007/3-540-48405-1_26.

[CN11] Yuanmi Chen and Phong Q. Nguyen. “BKZ 2.0: Better Lattice Security Estimates”. In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 1–20. DOI: 10.1007/978-3-642-25385-0_1. URL: https://doi.org/10.1007/978-3-642-25385-0_1.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. “Template Attacks”. In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28. DOI: 10.1007/3-540-36400-5_3. URL: https://doi.org/10.1007/3-540-36400-5_3.

[CRVV15] Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. “Efficient software implementation of ring-LWE encryption”. In: *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*. Ed. by Wolfgang Nebel and David Atienza. ACM, 2015, pp. 339–344. URL: http://dl.acm.org/citation.cfm?id=2755830.

[CT65] James W Cooley and John W Tukey. “An algorithm for the machine calculation of complex Fourier series”. In: *Mathematics of computation* 19.90 (1965), pp. 297–301.

[Cw32F4] NewAE Technology Inc. *CW308 UFO STM32F4*. URL: https://rtfm.newae.com/Targets/UFO%5C%20Targets/CW308T-STM32F/.

[CwUfo] NewAE Technology Inc. *CW308 UFO*. URL: https://rtfm.newae.com/Targets/CW308%5C%20UFO/.

[DB22] Jan-Pieter D’Anvers and Senne Batsleer. “Multitarget Decryption Failure Attacks and Their Application to Saber and Kyber”. In: *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13177. Lecture Notes in Computer Science. Springer, 2022, pp. 3–33. DOI: 10.1007/978-3-030-97121-2_1. URL: https://doi.org/10.1007/978-3-030-97121-2_1.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. “LWE with Side Information: Attacks and Concrete Security Estimation”. In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Springer, 2020, pp. 329–358. DOI: 10.1007/978-3-030-56880-1_12. URL: https://doi.org/10.1007/978-3-030-56880-1_12.

[Del22] Jeroen Delvaux. “Roulette: A Diverse Family of Feasible Fault Attacks on Masked Kyber”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.4 (2022), pp. 637–660. DOI: 10.46586/tches.v2022.i4.637-660. URL: https://doi.org/10.46586/tches.v2022.i4.637-660.

[DGHK22] Dana Dachman-Soled, Huijing Gong, Tom Hanson, and Hunter Kippen. "Revisiting Security Estimation for LWE with Hints from a Geometric Perspective". In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. Lecture Notes in Computer Science. Springer, 2022, pp. 748–781. DOI: 10.1007/978-3-031-38554-4_24. URL: https://doi.org/10.1007/978-3-031-38554-4_24.

[DGKS20] Dana Dachman-Soled, Huijing Gong, Mukul Kulkarni, and Aria Shahverdi. "(In)Security of Ring-LWE Under Partial Key Exposure". In: *J. Math. Cryptol.* 15.1 (2020), pp. 72–86. DOI: 10.1515/jmc-2020-0075. URL: https://doi.org/10.1515/jmc-2020-0075.

[DHP+22] Jan-Pieter D'Anvers, Daniel Heinz, Peter Pessl, Michiel Van Beirendonck, and Ingrid Verbauwhede. "Higher-Order Masked Ciphertext Comparison for Lattice-Based Cryptography". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.2 (2022), pp. 115–139. DOI: 10.46586/tches.v2022.i2.115-139. URL: https://doi.org/10.46586/tches.v2022.i2.115-139.

[Din12] Jintai Ding. "A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem". In: *IACR Cryptol. ePrint Arch.* (2012), p. 688. URL: https://eprint.iacr.org/archive/2012/688/20121210:115748.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.1 (2018), pp. 238–268. DOI: 10.13154/tches.v2018.i1.238-268. URL: https://doi.org/10.13154/tches.v2018.i1.238-268.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. "Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM". In: *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*. Ed. by Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Vol. 10831. Lecture Notes in Computer Science. Springer, 2018, pp. 282–305. DOI: 10.1007/978-3-319-89339-6_16. URL: https://doi.org/10.1007/978-3-319-89339-6_16.

[DNG22] Elena Dubrova, Kalle Ngo, and Joel Gärtner. "Breaking a Fifth-Order Masked Implementation of CRYSTALS-Kyber by Copy-Paste". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1713. URL: https://eprint.iacr.org/2022/1713.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data". In: *SIAM J. Comput.* 38.1 (2008), pp. 97–139. DOI: 10.1137/060651380. URL: https://doi.org/10.1137/060651380.

[DR98] Joan Daemen and Vincent Rijmen. "The Block Cipher Rijndael". In: *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*. Ed. by Jean-Jacques Quisquater and Bruce Schneier. Vol. 1820. Lecture Notes in Computer Science. Springer, 1998, pp. 277–284. DOI: 10.1007/10721064_26. URL: https://doi.org/10.1007/10721064_26.

[DTVV19] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. "Timing Attacks on Error Correcting Codes in Post-Quantum Schemes". In: *Proceedings of ACM Workshop on Theory of Implementation Security, TIS@CCS 2019, London, UK, November 11, 2019*. Ed. by Begül Bilgin, Svetla Petkova-Nikova, and Vincent Rijmen. ACM, 2019, pp. 2–9. DOI: 10.1145/3338467.3358948. URL: https://doi.org/10.1145/3338467.3358948.

[Duc18] Léo Ducas. "Shortest Vector from Lattice Sieving: A Few Dimensions for Free". In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Springer, 2018, pp. 125–145. DOI: 10.1007/978-3-319-78381-9_5. URL: https://doi.org/10.1007/978-3-319-78381-9_5.

[DVV19] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. "The Impact of Error Dependencies on Ring/Mod-LWE/LWR Based Schemes". In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*. Ed. by Jintai Ding and Rainer Steinwandt. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 103–115. DOI: 10.1007/978-3-030-25510-7_6. URL: https://doi.org/10.1007/978-3-030-25510-7_6.

[EAB+23] Mohamed ElGhamrawy, Melissa Azouaoui, Olivier Bronchain, Joost Renes, Tobias Schneider, Markus Schönauer, Okan Seker, and Christine van Vredendaal. "From MLWE to RLWE: A Differential Fault Attack on Randomized & Deterministic Dilithium". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.4 (2023), pp. 262–286. DOI: 10.46586/TCHES.V2023.I4.262-286. URL: https://doi.org/10.46586/tches.v2023.i4.262-286.

[Emd81] P. van Emde-Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Report. Department of Mathematics. University of Amsterdam. Department, Univ., 1981. URL: https://books.google.de/books?id=tCQiHQAACAAJ.

[EMK06] Gal Elidan, Ian McGraw, and Daphne Koller. "Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing". In: *UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence, Cambridge, MA, USA, July 13-16, 2006*. AUAI Press, 2006. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1%5C&smnu=2%5C&%20article_id=1308%5C&proceeding_id=22.

[FBR+22] Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. "Masked Accelerators and Instruction Set Extensions for Post-Quantum Cryptography". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.1 (2022), pp. 414–460. DOI: 10.46586/tches.v2022.i1.414-460. URL: https://doi.org/10.46586/tches.v2022.i1.414-460.

[FFS87] Uriel Feige, Amos Fiat, and Adi Shamir. "Zero Knowledge Proofs of Identity". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. Ed. by Alfred V. Aho. ACM, 1987, pp. 210–217. DOI: 10.1145/28395.28419. URL: https://doi.org/10.1145/28395.28419.

[FKK+22] Michael Fahr, Hunter Kippen, Andrew Kwong, Thinh Dang, Jacob Lichtinger, Dana Dachman-Soled, Daniel Genkin, Alexander Nelson, Ray A. Perlner, Arkady Yerukhimovich, and Daniel Apon. "When Frodo Flips: End-to-End Key Recovery on FrodoKEM via Rowhammer". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM, 2022, pp. 979–993. DOI: 10.1145/3548606.3560673. URL: https://doi.org/10.1145/3548606.3560673.

[Flu16] Scott R. Fluhrer. "Cryptanalysis of ring-LWE based key exchange with key share reuse". In: *IACR Cryptol. ePrint Arch.* (2016), p. 85. URL: http://eprint.iacr.org/2016/085.

[FM97] Brendan J. Frey and David J. C. MacKay. "A Revolution: Belief Propagation in Graphs with Cycles". In: *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*. Ed. by Michael I. Jordan, Michael J. Kearns, and Sara A. Solla. The MIT Press, 1997, pp. 479–485. URL: http://papers.nips.cc/paper/1467-a-revolution-belief-propagation-in-graphs-with-cycles.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *J. Cryptol.* 26.1 (2013), pp. 80–101. DOI: 10.1007/s00145-011-9114-1. URL: https://doi.org/10.1007/s00145-011-9114-1.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Advances in Cryptology - CRYPTO '99, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, 1999, pp. 537–554. DOI: 10.1007/3-540-48405-1_34. URL: https://doi.org/10.1007/3-540-48405-1_34.

[Fplll] The FPLLL development team. "fplll, a lattice reduction library, Version: 5.4.5". Available at https://github.com/fplll/fplll. 2023. URL: https://github.com/fplll/fplll.

[Gab10] Philippe Gaborit. *Noisy diffie-hellman protocols. Slides of a talk in the recent results session at the Third International Workshop on Post-Quantum Cryptography – PQCRYPTO 2010*. 2010.

[Gal62] Robert G. Gallager. "Low-density parity-check codes". In: *IRE Trans. Inf. Theory* 8.1 (1962), pp. 21–28. DOI: 10.1109/TIT.1962.1057683. URL: https://doi.org/10.1109/TIT.1962.1057683.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. "Mutual Information Analysis". In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 426–442. DOI: 10.1007/978-3-540-85053-3_27. URL: https://doi.org/10.1007/978-3-540-85053-3_27.

[GFS+12] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin A. Huss. "On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes". In: *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture

Notes in Computer Science. Springer, 2012, pp. 512–529. DOI: 10.1007/978-3-642-33027-8_30. URL: https://doi.org/10.1007/978-3-642-33027-8_30.

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)* Cambridge University Press, 2013. ISBN: 978-1-107-03903-2.

[GGSB20] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. “Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.4 (2020), pp. 209–238. DOI: 10.13154/tches.v2020.i4.209-238. URL: https://doi.org/10.13154/tches.v2020.i4.209-238.

[GHO15] Richard Gilmore, Neil Hanley, and Máire O’Neill. “Neural network based attack on a masked implementation of AES”. In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015.* IEEE Computer Society, 2015, pp. 106–111. DOI: 10.1109/HST.2015.7140247. URL: https://doi.org/10.1109/HST.2015.7140247.

[GJN20] Qian Guo, Thomas Johansson, and Alexander Nilsson. “A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM”. In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II.* Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Springer, 2020, pp. 359–386. DOI: 10.1007/978-3-030-56880-1_13. URL: https://doi.org/10.1007/978-3-030-56880-1_13.

[GJY19] Qian Guo, Thomas Johansson, and Jing Yang. “A Novel CCA Attack Using Decryption Errors Against LAC”. In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I.* Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11921. Lecture Notes in Computer Science. Springer, 2019, pp. 82–111. DOI: 10.1007/978-3-030-34578-5_4. URL: https://doi.org/10.1007/978-3-030-34578-5_4.

[GK08] Jacob Goldberger and Haggai Kfir. “Serial Schedules for Belief-Propagation: Analysis of Convergence Time”. In: *IEEE Trans. Inf. Theory* 54.3 (2008), pp. 1316–1319. DOI: 10.1109/TIT.2007.915702. URL: https://doi.org/10.1109/TIT.2007.915702.

[GM84] Shafi Goldwasser and Silvio Micali. “Probabilistic encryption”. In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(84)90070-9. URL: https://www.sciencedirect.com/science/article/pii/0022000084900709.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean- Pierre Seifert. “Approximating Shortest Lattice Vectors is not Harder than Approximating Closest Lattice Vectors”. In: *Inf. Process. Lett.* 71.2 (1999), pp. 55–61. DOI: 10.1016/S0020-0190(99)00083-6. URL: https://doi.org/10.1016/S0020-0190(99)00083-6.

[GN08a] Nicolas Gama and Phong Q. Nguyen. “Finding short lattice vectors within mordell’s inequality”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008.* Ed. by Cynthia Dwork. ACM, 2008, pp. 207–216. DOI: 10.1145/1374376.1374408. URL: https://doi.org/10.1145/1374376.1374408.

[GN08b] Nicolas Gama and Phong Q. Nguyen. “Predicting Lattice Reduction”. In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 31–51. DOI: 10.1007/978-3-540-78967-3_3. URL: https://doi.org/10.1007/978-3-540-78967-3_3.

[GP99] Louis Goubin and Jacques Patarin. “DES and Differential Power Analysis (The "Duplication" Method)”. In: *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES’99, Worcester, MA, USA, August 12-13, 1999, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. Lecture Notes in Computer Science. Springer, 1999, pp. 158–172. DOI: 10.1007/3-540-48059-5_15. URL: https://doi.org/10.1007/3-540-48059-5_15.

[GPSG14] Vincent Grosso, Romain Poussier, François-Xavier Standaert, and Lubos Gaspar. “Combining Leakage-Resilient PRFs and Shuffling - Towards Bounded Security for Small Embedded Devices”. In: *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*. Ed. by Marc Joye and Amir Moradi. Vol. 8968. Lecture Notes in Computer Science. Springer, 2014, pp. 122–136. DOI: 10.1007/978-3-319-16763-3_8. URL: https://doi.org/10.1007/978-3-319-16763-3_8.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. “Trapdoors for hard lattices and new cryptographic constructions”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by Cynthia Dwork. ACM, 2008, pp. 197–206. DOI: 10.1145/1374376.1374407. URL: https://doi.org/10.1145/1374376.1374407.

[GRO18] Joey Green, Arnab Roy, and Elisabeth Oswald. “A Systematic Study of the Impact of Graphical Models on Inference-Based Attacks on AES”. In: *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*. Ed. by Begül Bilgin and Jean-Bernard Fischer. Vol. 11389. Lecture Notes in Computer Science. Springer, 2018, pp. 18–34. DOI: 10.1007/978-3-030-15462-2_2. URL: https://doi.org/10.1007/978-3-030-15462-2_2.

[Gro96] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[GS15] Vincent Grosso and François-Xavier Standaert. “ASCA, SASCA and DPA with Enumeration: Which One Beats the Other and When?” In: *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. Lecture Notes in Computer Science. Springer, 2015, pp. 291–312. DOI: 10.1007/978-3-662-48800-3_12. URL: https://doi.org/10.1007/978-3-662-48800-3_12.

[GS66] W. Morven Gentleman and Gordon Sande. “Fast Fourier Transforms: for fun and profit”. In: *AFIPS ’66 (Fall)*. 1966.

[GW22] Ruben Gonzalez and Thom Wiggers. "KEMTLS vs. Post-quantum TLS: Performance on Embedded Systems". In: *Security, Privacy, and Applied Cryptography Engineering - 12th International Conference, SPACE 2022, Jaipur, India, December 9-12, 2022, Proceedings*. Ed. by Lejla Batina, Stjepan Picek, and Mainack Mondal. Vol. 13783. Lecture Notes in Computer Science. Springer, 2022, pp. 99–117. DOI: 10.1007/978-3-031-22829-2_6. URL: https://doi.org/10.1007/978-3-031-22829-2_6.

[Her23a] Julius Hermelink. *Decryption Errors and Implementation Attacks on Kyber*. Talk at the Institute für IT-Sicherheit at Universität zu Lübeck. Apr. 2023.

[Her23b] Julius Hermelink. *Side-Channel and Fault Attacks in Modern Lattice-Based Cryptography*. Oberseminar talk at the Fakultät für Informatik of the Universität der Bundeswehr München. June 2023.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. Lecture Notes in Computer Science. Springer, 2017, pp. 341–371. DOI: 10.1007/978-3-319-70500-2_12. URL: https://doi.org/10.1007/978-3-319-70500-2_12.

[HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. "Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 88–113. DOI: 10.46586/tches.v2021.i4.88-113. URL: https://doi.org/10.46586/tches.v2021.i4.88-113.

[HHP+21p] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. "Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber". In: *IACR Cryptol. ePrint Arch.* (2021), p. 956. URL: https://eprint.iacr.org/2021/956.

[HKL+22] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. "First-Order Masked Kyber on ARM Cortex-M4". In: *IACR Cryptol. ePrint Arch.* (2022), p. 58. URL: https://eprint.iacr.org/2022/058.

[HKM+12] Anthony Van Herrewege, Stefan Katzenbeisser, Roel Maes, Roel Peeters, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. "Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs". In: *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*. Ed. by Angelos D. Keromytis. Vol. 7397. Lecture Notes in Computer Science. Springer, 2012, pp. 374–389. DOI: 10.1007/978-3-642-32946-3_27. URL: https://doi.org/10.1007/978-3-642-32946-3_27.

[HMS+23] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. "Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.4 (2023), pp. 287–317. DOI: 10.46586/tches.v2023.i4.287-317. URL: https://doi.org/10.46586/tches.v2023.i4.287-317.

[HMS+23p] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. “Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 98. URL: https://eprint.iacr.org/2023/098.

[HNP+03] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. “The Impact of Decryption Failures on the Security of NTRU Encryption”. In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings.* Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 226–246. DOI: 10.1007/978-3-540-45146-4_14. URL: https://doi.org/10.1007/978-3-540-45146-4_14.

[HPP21] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. “Fault-Enabled Chosen-Ciphertext Attacks on Kyber”. In: *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15 , 2021, Proceedings.* Ed. by Avishek Adhikari, Ralf Küsters, and Bart Preneel. Vol. 13143. Lecture Notes in Computer Science. Springer, 2021, pp. 311–334. DOI: 10.1007/978-3-030-92518-5_15. URL: https://doi.org/10.1007/978-3-030-92518-5_15.

[HPP21p] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. “Fault-enabled chosen-ciphertext attacks on Kyber”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 1222. URL: https://eprint.iacr.org/2021/1222.

[HPS+20] Julius Hermelink, Thomas Pöppelmann, Marc Stöttinger, Yi Wang, and Yong Wan. “Quantum safe authenticated key exchange protocol for automotive application”. In: *18-th escar Europe : The World’s Leading Automotive Cyber Security Conference (Konferenzveröffentlichung).* 2020. DOI: 10.13154/294-7549.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. “Analyzing Blockwise Lattice Algorithms Using Dynamical Systems”. In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings.* Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 447–464. DOI: 10.1007/978-3-642-22792-9_25. URL: https://doi.org/10.1007/978-3-642-22792-9_25.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. “NTRU: A Ring-Based Public Key Cryptosystem”. In: *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings.* Ed. by Joe Buhler. Vol. 1423. Lecture Notes in Computer Science. Springer, 1998, pp. 267–288. DOI: 10.1007/BFb0054868. URL: https://doi.org/10.1007/BFb0054868.

[HS13] Michael Hutter and Peter Schwabe. “NaCl on 8-Bit AVR Microcontrollers”. In: *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings.* Ed. by Amr M. Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien. Vol. 7918. Lecture Notes in Computer Science. Springer, 2013, pp. 156–172. DOI: 10.1007/978-3-642-38553-7_9. URL: https://doi.org/10.1007/978-3-642-38553-7_9.

[HSS21] Julius Hermelink, Silvan Streit, and Emanuelle Strieder. *Belief Propagation Implementation.* 2021. URL: https://github.com/juliusjh/belief_propagation.

[HSST+23p] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. “Adapting Belief Propagation to Counter Shuffling of NTTs”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 555. URL: https://eprint.iacr.org/2022/555.

[HSST23] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. “Adapting Belief Propagation to Counter Shuffling of NTTs”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1 (2023), pp. 60–88. DOI: 10.46586/tches.v2023.i1.60-88. URL: https://doi.org/10.46586/tches.v2023.i1.60-88.

[IMS+22] Saad Islam, Koksal Mus, Richa Singh, Patrick Schaumont, and Berk Sunar. “Signature Correction Attack on Dilithium Signature Scheme”. In: *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*. IEEE, 2022, pp. 647–663. DOI: 10.1109/EUROSP53844.2022.00046. URL: https://doi.org/10.1109/EuroSP53844.2022.00046.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. “Private Circuits: Securing Hardware against Probing Attacks”. In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27. URL: https://doi.org/10.1007/978-3-540-45146-4_27.

[Jar22] Brian Jarvis. *How to tune TLS for hybrid post-quantum cryptography with Kyber*. 2022. URL: https://aws.amazon.com/blogs/security/how-to-tune-tls-for-hybrid-post-quantum-cryptography-with-kyber.

[Jin12a] Xiaodong Lin Jintai Ding. “A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem”. In: *IACR Cryptol. ePrint Arch.* (2012), p. 688. URL: https://eprint.iacr.org/archive/2012/688/20130303:142425.

[Jin12b] Xiaodong Lin Jintai Ding Xiaodong Lin. “A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem”. In: *IACR Cryptol. ePrint Arch.* (2012), p. 688. URL: https://eprint.iacr.org/archive/2012/688.

[JJ00] Éliane Jaulmes and Antoine Joux. “A Chosen-Ciphertext Attack against NTRU”. In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 20–35. DOI: 10.1007/3-540-44598-6_2. URL: https://doi.org/10.1007/3-540-44598-6_2.

[JSS12] Tibor Jager, Sebastian Schinzel, and Juraj Somorovsky. “Bleichenbacher’s Attack Strikes again: Breaking PKCS#1 v1.5 in XML Encryption”. In: *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Vol. 7459. Lecture Notes in Computer Science. Springer, 2012, pp. 752–769. DOI: 10.1007/978-3-642-33167-1_43. URL: https://doi.org/10.1007/978-3-642-33167-1_43.

[JT12] Marc Joye and Michael Tunstall, eds. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012. ISBN: 978-3-642-29655-0. DOI: 10.1007/978-3-642-29656-7. URL: https://doi.org/10.1007/978-3-642-29656-7.

[Kan87] Ravi Kannan. “Minkowski’s Convex Body Theorem and Integer Programming”. In: *Math. Oper. Res.* 12.3 (1987), pp. 415–440. DOI: 10.1287/moor.12.3.415. URL: https://doi.org/10.1287/moor.12.3.415.

[KCT+20] Jonathan Kuck, Shuvam Chakraborty, Hao Tang, Rachel Luo, Jiaming Song, Ashish Sabharwal, and Stefano Ermon. "Belief Propagation Neural Networks". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/07217414eb3fbe24d4e5b6cafb91ca18-Abstract.html.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors". In: *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014.* IEEE Computer Society, 2014, pp. 361–372. DOI: 10.1109/ISCA.2014.6853210. URL: https://doi.org/10.1109/ISCA.2014.6853210.

[KGM+21] Thilo Krachenfels, Fatemeh Ganji, Amir Moradi, Shahin Tajik, and Jean-Pierre Seifert. "Real-World Snapshots vs. Theory: Questioning the t-Probing Security Model". In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021.* IEEE, 2021, pp. 1955–1971. DOI: 10.1109/SP40001.2021.00029. URL: https://doi.org/10.1109/SP40001.2021.00029.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings.* Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25. URL: https://doi.org/10.1007/3-540-48405-1_25.

[KJJR11] Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. "Introduction to differential power analysis". In: *J. Cryptogr. Eng.* 1.1 (2011), pp. 5–27. DOI: 10.1007/s13389-011-0006-y. URL: https://doi.org/10.1007/s13389-011-0006-y.

[KL51] Solomon Kullback and Richard A Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[KO62] Anatolii Karatsuba and Yu Ofman. "Multiplication of Multidigit Numbers on Automata". In: *Soviet Physics Doklady* 7 (Dec. 1962), p. 595.

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings.* Ed. by Neal Koblitz. Vol. 1109. Lecture Notes in Computer Science. Springer, 1996, pp. 104–113. DOI: 10.1007/3-540-68697-5_9. URL: https://doi.org/10.1007/3-540-68697-5_9.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. "Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.3 (2019), pp. 148–179. DOI: 10.13154/tches.v2019.i3.148-179. URL: https://doi.org/10.13154/tches.v2019.i3.148-179.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. "Single-Trace Attacks on Keccak". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3 (2020), pp. 243–268. DOI: 10.13154/tches.v2020.i3.243-268. URL: https://doi.org/10.13154/tches.v2020.i3.243-268.

[KPR+] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. *PQM4: Post-quantum crypto library for the ARM Cortex-M4*. URL: https://github.com/mupq/pqm4.

[KSS+20] Patrick Knöbelreiter, Christian Sormann, Alexander Shekhovtsov, Friedrich Fraundorfer, and Thomas Pock. "Belief Propagation Reloaded: Learning BP-Layers for Labeling Problems". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 7897–7906. DOI: 10.1109/CVPR42600.2020.00792. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Knobelreiter_Belief_Propagation_Reloaded_Learning_BP-Layers_for_Labeling_Problems_CVPR_2020_paper.html.

[KSSW22] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Improving Software Quality in Cryptography Standardization Projects". In: *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*. IEEE, 2022, pp. 19–30. DOI: 10.1109/EuroSPW55150.2022.00010. URL: https://doi.org/10.1109/EuroSPW55150.2022.00010.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Miklós Lovász. "Factoring Polynomials with Rational Coefficients." In: *Mathematische Annalen* 261 (1982), pp. 515–534. URL: https://gdz.sub.uni-goettingen.de/id/PPN235181684_0261.

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. Ed. by Aggelos Kiayias. Vol. 6558. Lecture Notes in Computer Science. Springer, 2011, pp. 319–339. DOI: 10.1007/978-3-642-19074-2_21. URL: https://doi.org/10.1007/978-3-642-19074-2_21.

[LPR12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors Over Rings*. Cryptology ePrint Archive, Paper 2012/230. https://eprint.iacr.org/archive/2012/230/20120430:153308. 2012. URL: https://eprint.iacr.org/archive/2012/230/20120430:153308.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *J. ACM* 60.6 (2013), 43:1–43:35. DOI: 10.1145/2535925. URL: https://doi.org/10.1145/2535925.

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *Des. Codes Cryptogr.* 75.3 (2015), pp. 565–599. DOI: 10.1007/s10623-014-9938-4. URL: https://doi.org/10.1007/s10623-014-9938-4.

[LS19] Vadim Lyubashevsky and Gregor Seiler. "NTTRU: Truly Fast NTRU Using NTT". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.3 (2019), pp. 180–201. DOI: 10.13154/tches.v2019.i3.180-201. URL: https://doi.org/10.13154/tches.v2019.i3.180-201.

[LSR+15] Zhe Liu, Hwajeong Seo, Sujoy Sinha Roy, Johann Groß schädl, Howon Kim, and Ingrid Verbauwhede. "Efficient Ring-LWE Encryption on 8-Bit AVR Processors". In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16 , 2015, Proceedings*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 663–682. DOI: 10.1007/978-3-662-48324-4_33. URL: https://doi.org/10.1007/978-3-662-48324-4_33.

[Mac03] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003. ISBN: 0521642981. URL: https://www.inference.org.uk/itprnn/book.pdf.

[MBB+22] Catinca Mujdei, Arthur Beckers, Jose Bermundo, Angshuman Karmakar, Lennert Wouters, and Ingrid Verbauwhede. "Side-Channel Analysis of Lattice-Based Post-Quantum Cryptography: Exploiting Polynomial Multiplication". In: *IACR Cryptol. ePrint Arch.* (2022), p. 474. URL: https://eprint.iacr.org/2022/474.

[MCLS23] Loïc Masure, Valence Cristiani, Maxime Lecomte, and François-Xavier Standaert. "Don't Learn What You Already Know Scheme-Aware Modeling for Profiling Side-Channel Analysis against Masking". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1 (2023), pp. 32–59. DOI: 10.46586/tches.v2023.i1.32-59. URL: https://doi.org/10.46586/tches.v2023.i1.32-59.

[MD99] Thomas S. Messerges and Ezzy A. Dabbish. "Investigations of Power Analysis Attacks on Smartcards". In: *Proceedings of the 1st Workshop on Smartcard Technology, Smartcard 1999, Chicago, Illinois, USA, May 10-11, 1999*. Ed. by Scott B. Guthery and Peter Honeyman. USENIX Association, 1999. URL: https://www.usenix.org/conference/usenix-workshop-smartcard-technology/investigations-power-analysis-attacks-smartcards.

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. "Profiling power analysis attack based on MLP in DPA contest V4.2". In: *39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016*. IEEE, 2016, pp. 223–226. DOI: 10.1109/TSP.2016.7760865. URL: https://doi.org/10.1109/TSP.2016.7760865.

[Mes00] Thomas S. Messerges. "Securing the AES Finalists Against Power Analysis Attacks". In: *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*. Ed. by Bruce Schneier. Vol. 1978. Lecture Notes in Computer Science. Springer, 2000, pp. 150–164. DOI: 10.1007/3-540-44706-7_11. URL: https://doi.org/10.1007/3-540-44706-7_11.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems - a cryptographic perspective*. Vol. 671. The Kluwer international series in engineering and computer science. Springer, 2002. ISBN: 978-0-7923-7688-0. DOI: 10.1007/978-1-4615-0897-7. URL: https://doi.org/10.1007/978-1-4615-0897-7.

[MGTF19] Vincent Migliore, Benoıt Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. "Masking Dilithium - Efficient Implementation and Side-Channel Evaluation". In: *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*. Ed. by Robert H. Deng, Valérie Gauthier-Umaña, Martın Ochoa, and Moti Yung. Vol. 11464. Lecture Notes in Computer Science. Springer, 2019, pp. 344–362. DOI: 10.1007/978-3-030-21568-2_17. URL: https://doi.org/10.1007/978-3-030-21568-2%5C_17.

[MI14] Nicholas D. Matsakis and Felix S. Klock II. "The rust language". In: *Proceedings of the 2014 ACM SIGAda annual conference on High integrity language technology, HILT 2014, Portland, Oregon, USA, October 18-21, 2014*. Ed. by Michael B. Feldman and S. Tucker Taft. ACM, 2014, pp. 103–104. DOI: 10.1145/2663171.2663188. URL: https://doi.org/10.1145/2663171.2663188.

[MLB17] Artur Mariano, Thijs Laarhoven, and Christian H. Bischof. "A Parallel Variant of LDSieve for the SVP on Lattices". In: *25th Euromicro International Conference on Parallel, Distributed and Network-based Processing, PDP 2017, St. Petersburg, Russia, March 6-8, 2017*. Ed. by Igor V. Kotenko, Yiannis Cotronis, and Masoud Daneshtalab. IEEE Computer Society, 2017, pp. 23–30. DOI: 10.1109/PDP.2017.60. URL: https://doi.org/10.1109/PDP.2017.60.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007. ISBN: 978-0-387-30857-9. DOI: 10.1007/978-0-387-38162-6. URL: https://doi.org/10.1007/978-0-387-38162-6.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*. Ed. by Claude Carlet, M. Anwar Hasan, and Vishal Saraswat. Vol. 10076. Lecture Notes in Computer Science. Springer, 2016, pp. 3–26. DOI: 10.1007/978-3-319-49445-6_1. URL: https://doi.org/10.1007/978-3-319-49445-6_1.

[MR09] Daniele Micciancio and Oded Regev. "Lattice-based Cryptography". In: *Post-Quantum Cryptography*. Ed. by Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191. ISBN: 978-3-540-88702-7. DOI: 10.1007/978-3-540-88702-7_5. URL: https://doi.org/10.1007/978-3-540-88702-7_5.

[MS23] Loïc Masure and Rémi Strullu. "Side-channel analysis against ANSSI's protected AES implementation on ARM: end-to-end attacks with multi-task learning". In: *J. Cryptogr. Eng.* 13.2 (2023), pp. 129–147. DOI: 10.1007/s13389-023-00311-7. URL: https://doi.org/10.1007/s13389-023-00311-7.

[MUTS22] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. "Profiling Side-Channel Attacks on Dilithium: A Small Bit-Fiddling Leak Breaks It All". In: *IACR Cryptol. ePrint Arch.* (2022), p. 106. URL: https://eprint.iacr.org/2022/106.

[MW16] Daniele Micciancio and Michael Walter. "Practical, Predictable Lattice Basis Reduction". In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016, pp. 820–849. DOI: 10.1007/978-3-662-49890-3_31. URL: https://doi.org/10.1007/978-3-662-49890-3_31.

[MZ13] Zdenek Martinasek and Vaclav Zeman. "Innovative Method of the Power Analysis". In: *Radioengineering* 22.2 (June 2013), pp. 586–594. ISSN: 1805-9600. URL: https://www.radioeng.cz/fulltexts/2013/13_02_0586_0594.pdf.

[NBB16] Eliya Nachmani, Yair Be'ery, and David Burshtein. "Learning to decode linear codes using deep learning". In: *54th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2016, Monticello, IL, USA, September 27-30, 2016*. IEEE, 2016, pp. 341–346. DOI: 10.1109/ALLERTON.2016.7852251. URL: https://doi.org/10.1109/ALLERTON.2016.7852251.

[NC17] Erick Nascimento and Lukasz Chmielewski. "Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations". In: *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*. Ed. by Thomas Eisenbarth and Yannick Teglia. Vol. 10728. Lecture Notes in Computer Science. Springer, 2017, pp. 213–231. DOI: 10.1007/978-3-319-75208-2_13. URL: https://doi.org/10.1007/978-3-319-75208-2_13.

[NCOS16] Erick Nascimento, Lukasz Chmielewski, David F. Oswald, and Peter Schwabe. "Attacking Embedded ECC Implementations Through cmov Side Channels". In: *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*. Ed. by Roberto Avanzi and Howard M. Heys. Vol. 10532. Lecture Notes in Computer Science. Springer, 2016, pp. 99–119. DOI: 10.1007/978-3-319-69453-5_6. URL: https://doi.org/10.1007/978-3-319-69453-5_6.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. "A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 676–707. DOI: 10.46586/tches.v2021.i4.676-707. URL: https://doi.org/10.46586/tches.v2021.i4.676-707.

[NistAes] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*. Nov. 2001. DOI: 10.6028/NIST.FIPS.197-upd1. URL: https://doi.org/10.6028/NIST.FIPS.197-upd1.

[NistCfp] National Institute of Standards and Technology. *Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[NistCfpv4] National Institute of Standards and Technology. *PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates*. URL: https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4.

[NistPqc] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on Post-Quantum Cryptography*. URL: https://csrc.nist.gov/Pubs/ir/8105/Final.

[NistR1] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. *Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process*. URL: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf.

[NistR2] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. URL: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

[NistR3] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. URL: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf.

[NistSig] National Institute of Standards and Technology. *Post-Quantum Cryptography: Digital Signature Schemes*. URL: https://csrc.nist.gov/projects/pqc-dig-sig.

[NistSt21] Contributors to the pqc-forum thread on S-unit attacks. *National Institute for Standards and Technology pqc-forum thread on S-unit attacks*. 2021. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/3mVeyEfYnUY?pli=1.

[NS09] Phong Q. Nguyen and Damien Stehlé. "An LLL Algorithm with Quadratic Complexity". In: *SIAM J. Comput.* 39.3 (2009), pp. 874–903. DOI: 10.1137/070705702. URL: https://doi.org/10.1137/070705702.

[NS13] J. Neukirch and N. Schappacher. *Algebraic Number Theory*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2013. ISBN: 9783662039830. URL: https://books.google.de/books?id=hS3qCAAAQBAJ.

[NSS+17] Matús Nemec, Marek Sýs, Petr Svenda, Dusan Klinec, and Vashek Matyas. "The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, 2017, pp. 1631–1648. DOI: 10.1145/3133956.3133969. URL: https://doi.org/10.1145/3133956.3133969.

[Nus81] H.J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer Series in Information Sciences. Springer Berlin Heidelberg, 1981. ISBN: 9780387101590. URL: https://books.google.de/books?id=88sfAQAAIAAJ.

[NWDP22] Kalle Ngo, Ruize Wang, Elena Dubrova, and Nils Paulsrud. "Side-Channel Attacks on Lattice-Based KEMs Are Not Prevented by Higher-Order Masking". In: *IACR Cryptol. ePrint Arch.* (2022), p. 919. URL: https://eprint.iacr.org/2022/919.

[NY90] Moni Naor and Moti Yung. "Public-key cryptosystems provably secure against chosen ciphertext attacks". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990, pp. 427–437.

[OBr23] Devon O'Brien. *Protecting Chrome Traffic with Hybrid Kyber KEM*. 2023. URL: https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html.

[OC14] Colin O'Flynn and Zhizhang (David) Chen. *ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research*. Cryptology ePrint Archive, Paper 2014/204. https://eprint.iacr.org/2014/204. 2014. URL: https://eprint.iacr.org/2014/204.

[OG17] Tobias Oder and Tim Güneysu. “Implementing the NewHope-Simple Key Exchange on Low-Cost FPGAs”. In: *Progress in Cryptology - LATINCRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20-22, 2017, Revised Selected Papers*. Ed. by Tanja Lange and Orr Dunkelman. Vol. 11368. Lecture Notes in Computer Science. Springer, 2017, pp. 128–142. DOI: 10.1007/978-3-030-25283-0_7. URL: https://doi.org/10.1007/978-3-030-25283-0_7.

[OGM17] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. “Electromagnetic fault injection: the curse of flip-flops”. In: *J. Cryptogr. Eng.* 7.3 (2017), pp. 183–197. DOI: 10.1007/s13389-016-0128-3. URL: https://doi.org/10.1007/s13389-016-0128-3.

[OKPW10] Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. “Algebraic Side-Channel Analysis in the Presence of Errors”. In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 428–442. DOI: 10.1007/978-3-642-15031-9_29. URL: https://doi.org/10.1007/978-3-642-15031-9_29.

[ORSW12] Yossef Oren, Mathieu Renauld, François-Xavier Standaert, and Avishai Wool. “Algebraic Side-Channel Attacks Beyond the Hamming Weight Leakage Model”. In: *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture Notes in Computer Science. Springer, 2012, pp. 140–154. DOI: 10.1007/978-3-642-33027-8_9. URL: https://doi.org/10.1007/978-3-642-33027-8_9.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. “Practical CCA2-Secure and Masked Ring-LWE Implementation”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.1 (2018), pp. 142–174. DOI: 10.13154/tches.v2018.i1.142-174. URL: https://doi.org/10.13154/tches.v2018.i1.142-174.

[OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. “Cache Attacks and Countermeasures: The Case of AES”. In: *Topics in Cryptology - CT-RSA 2006, The Cryptographers’ Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*. Ed. by David Pointcheval. Vol. 3860. Lecture Notes in Computer Science. Springer, 2006, pp. 1–20. DOI: 10.1007/11605805_1. URL: https://doi.org/10.1007/11605805_1.

[Pei09a] Chris Peikert. “Public-key cryptosystems from the worst-case shortest vector problem: extended abstract”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. Ed. by Michael Mitzenmacher. ACM, 2009, pp. 333–342. DOI: 10.1145/1536414.1536461. URL: https://doi.org/10.1145/1536414.1536461.

[Pei09b] Chris Peikert. “Some Recent Progress in Lattice-Based Cryptography”. In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*. Ed. by Omer Reingold. Vol. 5444. Lecture Notes in Computer Science. Springer, 2009, p. 72. DOI: 10.1007/978-3-642-00457-5_5. URL: https://doi.org/10.1007/978-3-642-00457-5_5.

[Pei14] Chris Peikert. “Lattice Cryptography for the Internet”. In: *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings.* Ed. by Michele Mosca. Vol. 8772. Lecture Notes in Computer Science. Springer, 2014, pp. 197–219. DOI: 10.1007/978-3-319-11659-4_12. URL: https://doi.org/10.1007/978-3-319-11659-4_12.

[Pei16] Chris Peikert. “A Decade of Lattice Cryptography”. In: *Found. Trends Theor. Comput. Sci.* 10.4 (2016), pp. 283–424. DOI: 10.1561/0400000074. URL: https://doi.org/10.1561/0400000074.

[PG13] Thomas Pöppelmann and Tim Güneysu. “Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware”. In: *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers.* Ed. by Tanja Lange, Kristin E. Lauter, and Petr Lisonek. Vol. 8282. Lecture Notes in Computer Science. Springer, 2013, pp. 68–85. DOI: 10.1007/978-3-662-43414-7_4. URL: https://doi.org/10.1007/978-3-662-43414-7_4.

[PH16] Aesun Park and Dong-Guk Han. “Chosen ciphertext Simple Power Analysis on software 8-bit implementation of ring-LWE encryption”. In: *2016 IEEE Asian Hardware-Oriented Security and Trust, AsianHOST 2016, Yilan, Taiwan, December 19-20, 2016.* IEEE Computer Society, 2016, pp. 1–6. DOI: 10.1109/AsianHOST.2016.7835555. URL: %7Bhttps://doi.org/10.1109/AsianHOST.2016.7835555%7D.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. “The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.1 (2019), pp. 209–237. DOI: 10.13154/tches.v2019.i1.209-237. URL: https://doi.org/10.13154/tches.v2019.i1.209-237.

[POG15] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. “High-Performance Ideal Lattice-Based Cryptography on 8-Bit ATxmega Microcontrollers”. In: *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings.* Ed. by Kristin E. Lauter and Francisco Rodríguez-Henríquez. Vol. 9230. Lecture Notes in Computer Science. Springer, 2015, pp. 346–365. DOI: 10.1007/978-3-319-22174-8_19. URL: https://doi.org/10.1007/978-3-319-22174-8_19.

[Pol71] John M. Pollard. “The fast Fourier transform in a finite field”. In: *Mathematics of Computation* 25 (1971), pp. 365–374.

[PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners.* Springer, 2010. ISBN: 978-3-642-04100-6. DOI: 10.1007/978-3-642-04101-3. URL: https://doi.org/10.1007/978-3-642-04101-3.

[PP19] Peter Pessl and Robert Primas. “More Practical Single-Trace Attacks on the Number Theoretic Transform”. In: *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings.* Ed. by Peter Schwabe and Nicolas Thériault. Vol. 11774. Lecture Notes in Computer Science. Springer, 2019, pp. 130–149. DOI: 10.1007/978-3-030-30530-7_7. URL: https://doi.org/10.1007/978-3-030-30530-7_7.

[PP21] Peter Pessl and Lukas Prokop. “Fault Attacks on CCA-secure Lattice KEMs”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.2 (2021), pp. 37–60. DOI: 10.46586/tches.v2021.i2.37-60. URL: https://doi.org/10.46586/tches.v2021.i2.37-60.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. “Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption”. In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 513–533. DOI: 10.1007/978-3-319-66787-4_25. URL: https://doi.org/10.1007/978-3-319-66787-4_25.

[PqcFo] Constributors to the pqc-forum. *National Institute for Standards and Technology pqc-forum*. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum.

[PQClean] Contributors to PQClean. *PQClean*. URL: https://github.com/PQClean/PQClean.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. “Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach”. In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science. Springer, 2016, pp. 61–81. DOI: 10.1007/978-3-662-53140-2_4. URL: https://doi.org/10.1007/978-3-662-53140-2_4.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. “A Framework for Efficient and Composable Oblivious Transfer”. In: *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*. Ed. by David A. Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 554–571. DOI: 10.1007/978-3-540-85174-5_31. URL: https://doi.org/10.1007/978-3-540-85174-5_31.

[PyRef] Contributors to the Python Language Reference. *The Python Language Reference*. URL: https://docs.python.org/3/reference/.

[RBRC20] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. “Drop by Drop you break the rock - Exploiting generic vulnerabilities in Lattice-based PKE/KEMs using EM-based Physical Attacks”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 549. URL: https://eprint.iacr.org/2020/549.

[RBRC22] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. “On Exploiting Message Leakage in (Few) NIST PQC Candidates for Practical Message Recovery Attacks”. In: *IEEE Trans. Inf. Forensics Secur.* 17 (2022), pp. 684–699. DOI: 10.1109/TIFS.2021.3139268. URL: https://doi.org/10.1109/TIFS.2021.3139268.

[REB+22] Prasanna Ravi, Martianus Frederic Ezerman, Shivam Bhasin, Anupam Chattopadhyay, and Sujoy Sinha Roy. “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. DOI: 10.46586/tches.v2022.i1.722-761. URL: https://doi.org/10.46586/tches.v2022.i1.722-761.

[Reg05] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603.

[Reg09] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *J. ACM* 56.6 (2009), 34:1–34:40. DOI: 10.1145/1568318.1568324. URL: http://doi.acm.org/10.1145/1568318.1568324.

[Reg10] Oded Regev. “The Learning with Errors Problem (Invited Survey)”. In: *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*. IEEE Computer Society, 2010, pp. 191–204. DOI: 10.1109/CCC.2010.26. URL: https://doi.org/10.1109/CCC.2010.26.

[RJH+18] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. “Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate”. In: *IACR Cryptol. ePrint Arch.* (2018), p. 821. URL: https://eprint.iacr.org/2018/821.

[RPBC20] Prasanna Ravi, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. “On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT - A Performance Evaluation Study over Kyber and Dilithium on the ARM Cortex-M4”. In: *Security, Privacy, and Applied Cryptography Engineering - 10th International Conference, SPACE 2020, Kolkata, India, December 17-21, 2020, Proceedings*. Ed. by Lejla Batina, Stjepan Picek, and Mainack Mondal. Vol. 12586. Lecture Notes in Computer Science. Springer, 2020, pp. 123–146. DOI: 10.1007/978-3-030-66626-2_7. URL: https://doi.org/10.1007/978-3-030-66626-2_7.

[RPD09] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. “Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 171–188. DOI: 10.1007/978-3-642-04138-9_13. URL: https://doi.org/10.1007/978-3-642-04138-9_13.

[RRB+19] Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. “Number "Not Used" Once - Practical Fault Attack on pqm4 Implementations of NIST Candidates”. In: *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*. Ed. by Ilia Polian and Marc Stöttinger. Vol. 11421. Lecture Notes in Computer Science. Springer, 2019, pp. 232–250. DOI: 10.1007/978-3-030-16350-1_13. URL: https://doi.org/10.1007/978-3-030-16350-1_13.

[RRC+16] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. “Masking ring-LWE”. In: *J. Cryptogr. Eng.* 6.2 (2016), pp. 139–153. DOI: 10.1007/s13389-016-0126-5. URL: https://doi.org/10.1007/s13389-016-0126-5.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. "Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3 (2020), pp. 307–335. DOI: 10.13154/tches.v2020.i3.307-335. URL: https://doi.org/10.13154/tches.v2020.i3.307-335.

[RRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. "A Masked Ring-LWE Implementation". In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings.* Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 683–702. DOI: 10.1007/978-3-662-48324-4_34. URL: https://doi.org/10.1007/978-3-662-48324-4_34.

[RS09] Mathieu Renauld and François-Xavier Standaert. "Algebraic Side-Channel Attacks". In: *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers.* Ed. by Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing. Vol. 6151. Lecture Notes in Computer Science. Springer, 2009, pp. 393–410. DOI: 10.1007/978-3-642-16342-5_29. URL: https://doi.org/10.1007/978-3-642-16342-5_29.

[RS10] Markus Rückert and Michael Schneider. "Estimating the Security of Lattice-based Cryptosystems". In: *IACR Cryptol. ePrint Arch.* (2010), p. 137. URL: http://eprint.iacr.org/2010/137.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (Reprint)". In: *Commun. ACM* 26.1 (1983), pp. 96–99. DOI: 10.1145/357980.358017. URL: https://doi.org/10.1145/357980.358017.

[RSDT13] Cyril Roscian, Alexandre Sarafianos, Jean-Max Dutertre, and Assia Tria. "Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells". In: *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013.* Ed. by Wieland Fischer and Jörn-Marc Schmidt. IEEE Computer Society, 2013, pp. 89–98. DOI: 10.1109/FDTC.2013.17. URL: https://doi.org/10.1109/FDTC.2013.17.

[RSV09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. "Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA". In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings.* Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 97–111. DOI: 10.1007/978-3-642-04138-9_8. URL: https://doi.org/10.1007/978-3-642-04138-9_8.

[RustRef] Contributors to the Rust Reference. *The Rust Reference.* URL: https://doc.rust-lang.org/reference/index.html.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. “Optical Fault Induction Attacks”. In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 2–12. DOI: 10.1007/3-540-36400-5_2. URL: https://doi.org/10.1007/3-540-36400-5_2.

[Sch87] Claus-Peter Schnorr. “A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms”. In: *Theor. Comput. Sci.* 53 (1987), pp. 201–224. DOI: 10.1016/0304-3975(87)90064-8. URL: https://doi.org/10.1016/0304-3975(87)90064-8.

[SE94] Claus-Peter Schnorr and M. Euchner. “Lattice basis reduction: Improved practical algorithms and solving subset sum problems”. In: *Math. Program.* 66 (1994), pp. 181–199. DOI: 10.1007/BF01581144. URL: https://doi.org/10.1007/BF01581144.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. “The Graph Neural Network Model”. In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605. URL: https://doi.org/10.1109/TNN.2008.2005605.

[SH95] Claus-Peter Schnorr and Horst Helmut Hörner. “Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction”. In: *Advances in Cryptology - EUROCRYPT ’95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Vol. 921. Lecture Notes in Computer Science. Springer, 1995, pp. 1–12. DOI: 10.1007/3-540-49264-X_1. URL: https://doi.org/10.1007/3-540-49264-X_1.

[Sho94] Peter W. Shor. “Polynominal time algorithms for discrete logarithms and factoring on a quantum computer”. In: *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*. Ed. by Leonard M. Adleman and Ming-Deh A. Huang. Vol. 877. Lecture Notes in Computer Science. Springer, 1994, p. 289. DOI: 10.1007/3-540-58691-1_68. URL: https://doi.org/10.1007/3-540-58691-1_68.

[Sho97] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM J. Comput.* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

[Sig23] Contributors to libsignal. *Release v0.27.0*. 2023. URL: https://github.com/signalapp/libsignal/releases/tag/v0.27.0.

[SIH+23] Emanuele Strieder, Manuel Ilg, Johann Heyszl, Florian Unterstein, and Silvan Streit. “ASCA vs. SASCA - A Closer Look at the AES Key Schedule”. In: *Constructive Side-Channel Analysis and Secure Design - 14th International Workshop, COSADE 2023, Munich, Germany, April 3-4, 2023, Proceedings*. Ed. by Elif Bilge Kavun and Michael Pehl. Vol. 13979. Lecture Notes in Computer Science. Springer, 2023, pp. 65–85. DOI: 10.1007/978-3-031-29497-6_4. URL: https://doi.org/10.1007/978-3-031-29497-6_4.

[Sin64] Richard Sinkhorn. “A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices”. In: *The Annals of Mathematical Statistics* 35.2 (1964), pp. 876–879. DOI: 10.1214/aoms/1177703591. URL: https://doi.org/10.1214/aoms/1177703591.

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. “A Collision-Attack on AES: Combining Side Channel- and Differential-Attack”. In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 163–175. DOI: 10.1007/978-3-540-28632-5_12. URL: https://doi.org/10.1007/978-3-540-28632-5_12.

[SLG07] Eran Sharon, Simon Litsyn, and Jacob Goldberger. “Efficient Serial Message-Passing Schedules for LDPC Decoding”. In: *IEEE Trans. Inf. Theory* 53.11 (2007), pp. 4076–4091. DOI: 10.1109/TIT.2007.907507. URL: https://doi.org/10.1109/TIT.2007.907507.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. “A Stochastic Model for Differential Side Channel Cryptanalysis”. In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. Lecture Notes in Computer Science. Springer, 2005, pp. 30–46. DOI: 10.1007/11545262_3. URL: https://doi.org/10.1007/11545262_3.

[SM12] Charles Sutton and Andrew McCallum. “Improved Dynamic Schedules for Belief Propagation”. In: *CoRR* abs/1206.5291 (2012). arXiv: 1206.5291. URL: http://arxiv.org/abs/1206.5291.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. “A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks”. In: *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 443–461. DOI: 10.1007/978-3-642-01001-9_26. URL: https://doi.org/10.1007/978-3-642-01001-9_26.

[SPH22] Bo-Yeon Sim, Aesun Park, and Dong-Guk Han. “Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER Using the Side-Channel Leakage of Barrett Reduction”. In: *IEEE Internet Things J.* 9.21 (2022), pp. 21382–21397. DOI: 10.1109/JIOT.2022.3179683. URL: https://doi.org/10.1109/JIOT.2022.3179683.

[SSW17] Cyprien de Saint Guilhem, Nigel P. Smart, and Bogdan Warinschi. “Generic Forward-Secure Key Agreement Without Signatures”. In: *Information Security - 20th International Conference, ISC 2017, Ho Chi Minh City, Vietnam, November 22-24, 2017, Proceedings*. Ed. by Phong Q. Nguyen and Jianying Zhou. Vol. 10599. Lecture Notes in Computer Science. Springer, 2017, pp. 114–133. DOI: 10.1007/978-3-319-69659-1_7. URL: https://doi.org/10.1007/978-3-319-69659-1_7.

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. “Post-Quantum TLS Without Handshake Signatures”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1461–1480. ISBN: 9781450370899. DOI: 10.1145/3372297.3423350. URL: https://doi.org/10.1145/3372297.3423350.

[Sto03] Amos J. Storkey. "Generalised Propagation for Fast Fourier Transforms with Partial or Missing Data". In: *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003 , Vancouver and Whistler, British Columbia, Canada]*. Ed. by Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf. MIT Press, 2003, pp. 433–440. URL: https://proceedings.neurips.cc/paper/2003/hash/8c1b6fa97c4288a4514365198566c6fa-Abstract.html.

[SW07] Joseph H. Silverman and William Whyte. "Timing Attacks on NTRUEncrypt Via Variation in the Number of Hash Calls". In: *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*. Ed. by Masayuki Abe. Vol. 4377. Lecture Notes in Computer Science. Springer, 2007, pp. 208–224. DOI: 10.1007/11967668_14. URL: https://doi.org/10.1007/11967668_14.

[SW21] Victor Garcia Satorras and Max Welling. "Neural Enhanced Belief Propagation on Factor Graphs". In: *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 685–693. URL: http://proceedings.mlr.press/v130/garcia-satorras21a.html.

[TH08] Stefan Tillich and Christoph Herbst. "Attacking State-of-the-Art Software Countermeasures-A Case Study for AES". In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 228–243. DOI: 10.1007/978-3-540-85053-3_15. URL: https://doi.org/10.1007/978-3-540-85053-3_15.

[THM07] Stefan Tillich, Christoph Herbst, and Stefan Mangard. "Protecting AES Software Implementations on 32-Bit Processors Against Power Analysis". In: *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*. Ed. by Jonathan Katz and Moti Yung. Vol. 4521. Lecture Notes in Computer Science. Springer, 2007, pp. 141–157. DOI: 10.1007/978-3-540-72738-5_10. URL: https://doi.org/10.1007/978-3-540-72738-5_10.

[Tim19] Benjamin Timon. "Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.2 (2019), pp. 107–131. DOI: 10.13154/tches.v2019.i2.107-131. URL: https://doi.org/10.13154/tches.v2019.i2.107-131.

[Too63] A.L. Toom. "The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers". In: *Soviet Physics Doklady* 3 (May 1963), pp. 714–716.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. "Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms". In: *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. 2016, pp. 192–216. DOI: 10.1007/978-3-662-53644-5_8. URL: https://doi.org/10.1007/978-3-662-53644-5_8.

[UBS21] Balazs Udvarhelyi, Olivier Bronchain, and François-Xavier Standaert. "Security Analysis of Deterministic Re-keying with Masking and Shuffling: Application to ISAP". In: *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings.* Ed. by Shivam Bhasin and Fabrizio De Santis. Vol. 12910. Lecture Notes in Computer Science. Springer, 2021, pp. 168–183. DOI: 10.1007/978-3-030-89915-8_8. URL: https://doi.org/10.1007/978-3-030-89915-8_8.

[UXT+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. "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. DOI: 10.46586/tches.v2022.i1.296-322. URL: https://doi.org/10.46586/tches.v2022.i1.296-322.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. "Soft Analytical Side-Channel Attacks". In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I.* Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Springer, 2014, pp. 282–296. DOI: 10.1007/978-3-662-45611-8_15. URL: https://doi.org/10.1007/978-3-662-45611-8_15.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. "Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note". In: *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings.* Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 740–757. DOI: 10.1007/978-3-642-34961-4_44. URL: https://doi.org/10.1007/978-3-642-34961-4_44.

[VOGR18] Felipe Valencia, Tobias Oder, Tim Güneysu, and Francesco Regazzoni. "Exploring the Vulnerability of R-LWE Encryption to Fault Attacks". In: *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems, CS2 2018, Manchester, United Kingdom, January 24, 2018.* Ed. by John Goodacre, Mikel Luján, Giovanni Agosta, Alessandro Barenghi, Israel Koren, and Gerardo Pelosi. ACM, 2018, pp. 7–12. DOI: 10.1145/3178291.3178294. URL: https://doi.org/10.1145/3178291.3178294.

[WAR+23] Lichao Wu, Amir Ali-pour, Azade Rezaeezade, Guilherme Perin, and Stjepan Picek. *Breaking Free: Leakage Model-free Deep Learning-based Side-channel Analysis.* Cryptology ePrint Archive, Paper 2023/1110. https://eprint.iacr.org/2023/1110. 2023. URL: https://eprint.iacr.org/2023/1110.

[Wei22] Andreas Weik. "Machine-Learning-based Side-Channel Attacks on Lattice-based Key Encapsulation Mechanisms". Master's Thesis at the Technical University of Munich 2022. Oct. 2022.

[Win96] Franz Winkler. *Polynomial Algorithms in Computer Algebra.* Texts & Monographs in Symbolic Computation. Springer, 1996. ISBN: 978-3-211-82759-8. DOI: 10.1007/978-3-7091-6571-3. URL: https://doi.org/10.1007/978-3-7091-6571-3.

[WND22] Ruize Wang, Kalle Ngo, and Elena Dubrova. “A Message Recovery Attack on LWE/LWR-Based PKE/KEMs Using Amplitude-Modulated EM Emanations”. In: *Information Security and Cryptology - ICISC 2022 - 25th International Conference, ICISC 2022, Seoul, South Korea, November 30 - December 2, 2022, Revised Selected Papers*. Ed. by Seung-Hyun Seo and Hwajeong Seo. Vol. 13849. Lecture Notes in Computer Science. Springer, 2022, pp. 450–471. DOI: 10.1007/978-3-031-29371-9_22. URL: https://doi.org/10.1007/978-3-031-29371-9_22.

[WP06] André Weimerskirch and Christof Paar. “Generalizations of the Karatsuba Algorithm for Efficient Implementations”. In: *IACR Cryptol. ePrint Arch.* (2006), p. 224. URL: http://eprint.iacr.org/2006/224.

[WWM11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Federico Menarini. “Practical Optical Fault Injection on Secure Microcontrollers”. In: *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2011, Tokyo, Japan, September 29, 2011*. Ed. by Luca Breveglieri, Sylvain Guilley, Israel Koren, David Naccache, and Junko Takahashi. IEEE Computer Society, 2011, pp. 91–99. DOI: 10.1109/FDTC.2011.12. URL: https://doi.org/10.1109/FDTC.2011.12.

[WZW13] An Wang, Xuexin Zheng, and Zongyue Wang. “Power Analysis Attacks and Countermeasures on NTRU-Based Wireless Body Area Networks”. In: *KSII Trans. Internet Inf. Syst.* 7.5 (2013), pp. 1094–1107. DOI: 10.3837/tiis.2013.05.009. URL: https://doi.org/10.3837/tiis.2013.05.009.

[XIU+21] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. “Fault-Injection Attacks Against NIST’s Post-Quantum Cryptography Round 3 KEM Candidates”. In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13091. Lecture Notes in Computer Science. Springer, 2021, pp. 33–61. DOI: 10.1007/978-3-030-92075-3_2. URL: https://doi.org/10.1007/978-3-030-92075-3%5C_2.

[XPOZ22] Zhuang Xu, Owen Pemberton, David F. Oswald, and Zhiming Zheng. “Reveal the Invisible Secret: Chosen-Ciphertext Side-Channel Attacks on NTRU”. In: *Smart Card Research and Advanced Applications - 21st International Conference, CARDIS 2022, Birmingham, UK, November 7-9, 2022, Revised Selected Papers*. Ed. by Ileana Buhan and Tobias Schneider. Vol. 13820. Lecture Notes in Computer Science. Springer, 2022, pp. 227–247. DOI: 10.1007/978-3-031-25319-5_12. URL: https://doi.org/10.1007/978-3-031-25319-5_12.

[XPR+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David F. Oswald, Wang Yao, and Zhiming Zheng. “Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems With Chosen Ciphertexts: The Case Study of Kyber”. In: *IEEE Trans. Computers* 71.9 (2022), pp. 2163–2176. DOI: 10.1109/TC.2021.3122997. URL: https://doi.org/10.1109/TC.2021.3122997.

[XPRO20] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David F. Oswald. “Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 912. URL: https://eprint.iacr.org/2020/912.

[Yed04] Jonathan S. Yedidia. “Sparse factor graph representations of Reed-Solomon and related codes”. In: *Proceedings of the 2004 IEEE International Symposium on Information Theory, ISIT 2004, Chicago Downtown Marriott, Chicago, Illinois, USA, June 27 - July 2, 2004*. IEEE, 2004, p. 260. DOI: 10.1109/ISIT.2004.1365296. URL: https://doi.org/10.1109/ISIT.2004.1365296.

[YJ00] Sung-Ming Yen and Marc Joye. “Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis”. In: *IEEE Trans. Computers* 49.9 (2000), pp. 967–970. DOI: 10.1109/12.869328. URL: https://doi.org/10.1109/12.869328.

# Acronyms

| | |
|---|---|
| **AES** | Advanced Encryption Standard |
| **ASCA** | Algebraic Side-Channel Attack |
| **BDD** | Bounded Distance Decoding |
| **BKZ** | Blockwise Korkine-Zolotarev |
| **BP** | Belief Propagation |
| **BSI** | Bundesamt für Sicherheit in der Informationstechnik |
| **CA** | Collision Attacks |
| **CCA** | Chosen-Ciphertext Attack |
| **CPA** | Chosen-Plaintext Attack |
| **CPA** | Correlation Power Analysis |
| **CPU** | Central Processing Unit |
| **CRT** | Chinese Remainder Theorem |
| **CVP** | Closest Vector Problem |
| **DBDD** | Distorted Bounded Distance Decoding Problem |
| **DFT** | Discrete Fourier Transform |
| **D-LWE** | Decision Learning with Errors |
| **DoS** | Denial of Service |
| **DPA** | Differential Power Analysis |
| **EM** | Electro-Magnetic Radiation |
| **FFT** | Fast-Fourier Transformation |
| **FO** | Fujisaki-Okamoto |
| **HW** | Hamming Weight |

| | |
|---|---|
| **INTT** | Inverse Number Theoretic Transform |
| **KDF** | Key Derivation Function |
| **KEM** | Key Encapsulation Mechanism |
| **KL** | Kullback-Leibler |
| **LDPC** | Low-Density Parity Check |
| **LLL** | Lenstra, Lenstra, and Lovasz |
| **LR** | Linear Regression |
| **LWE** | Learning with Errors |
| **MIA** | Mutual Information Analysis |
| **MLWE** | Module Learning with Errors |
| **NIST** | National Institute of Standards and Technology |
| **NTRU** | NTRU |
| **NTT** | Number Theoretic Transform |
| **PKE** | Public-Key Encryption |
| **PQ** | Post-Quantum |
| **PQC** | Post-Quantum Cryptography |
| **PRF** | Pseudo-Random Function |
| **RLWE** | Ring Learning with Errors |
| **RNG** | Random Number Generator |
| **RSA** | Rivest–Shamir–Adleman |
| **SASCA** | Soft Analytical Side-Channel Attack |
| **SCA** | Side Channel Attack |
| **SIVP** | Shortest Independent Vector Problem |
| **SNR** | Signal-to-noise ratio |
| **SPA** | Simple Power Analysis |
| **SVP** | Shortest Vector Problem |
| **TASCA** | Tolerant Algebraic Side-Channel Attack |
| **TLS** | Transport Layer Security |
| **uSVP** | unique Shortest Vector Problem |
| **XOR** | Exclusive-Or |

# List of Publications

This thesis builds upon several published works the author contributed to. Summaries and contributions made by the author of this thesis can be found in Section 1.2. We list the publications, the corresponding preprints, and talks given by the author from which figures have been used in this thesis. Note that preprints come in different versions; we do not list them separately. We do not list talks from which no contents were used and that are therefore not cited in this thesis. The work in [HPS+20] is referenced but not fully part of this thesis.

## Peer-Reviewed Journal Publications

- [HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. “Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 88–113. DOI: 10.46586/tches.v2021.i4.88-113. URL: https://doi.org/10.46586/tches.v2021.i4.88-113

- [HSST23] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. “Adapting Belief Propagation to Counter Shuffling of NTTs”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1 (2023), pp. 60–88. DOI: 10.46586/tches.v2023.i1.60-88. URL: https://doi.org/10.46586/tches.v2023.i1.60-88

- [HMS+23] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. “Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.4 (2023), pp. 287–317. DOI: 10.46586/tches.v2023.i4.287-317. URL: https://doi.org/10.46586/tches.v2023.i4.287-317

## Peer-Reviewed Conference Proceedings

- [HPS+20] Julius Hermelink, Thomas Pöppelmann, Marc Stöttinger, Yi Wang, and Yong Wan. “Quantum safe authenticated key exchange protocol for automotive application”. In: *18-th escar Europe : The World’s Leading Automotive Cyber Security Conference (Konferenzveröffentlichung)*. 2020. DOI: 10.13154/294-7549

- [HPP21] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. “Fault-Enabled Chosen-Ciphertext Attacks on Kyber”. In: *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15 , 2021, Proceedings*. Ed. by Avishek Adhikari, Ralf Küsters, and Bart Preneel. Vol. 13143. Lecture Notes in Computer Science. Springer, 2021, pp. 311–334. DOI: 10.1007/978-3-030-92518-5_15. URL: https://doi.org/10.1007/978-3-030-92518-5_15

## Preprints

- [HHP+21p] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. “Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 956. URL: https://eprint.iacr.org/2021/956
- [HPP21p] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. “Fault-enabled chosen-ciphertext attacks on Kyber”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 1222. URL: https://eprint.iacr.org/2021/1222
- [HSST+23p] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. “Adapting Belief Propagation to Counter Shuffling of NTTs”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 555. URL: https://eprint.iacr.org/2022/555
- [HMS+23p] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. “Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 98. URL: https://eprint.iacr.org/2023/098 (accepted at IACR Trans. Cryptogr. Hardw. Embed. Syst. 2023/4)

## Talks

- [Her23b] Julius Hermelink. *Side-Channel and Fault Attacks in Modern Lattice-Based Cryptography.* Oberseminar talk at the Fakultät für Informatik of the Universität der Bundeswehr München. June 2023
- [Her23a] Julius Hermelink. *Decryption Errors and Implementation Attacks on Kyber.* Talk at the Institute für IT-Sicherheit at Universität zu Lübeck. Apr. 2023

# List of Figures

# List of Tables

# List of Algorithms