Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors

Julius Hermelink[1,2], Erik Mårtensson[3,4], Simona Samardjiska[5], Peter Pessl[2], Gabi Dreo Rodosek[1]

[1] Universität der Bundeswehr München, Munich, Germany
{julius.hermelink,gabi.dreo}@unibw.de

[2] Infineon Technologies AG, Munich, Germany {hermelink.external,peter.pessl}@infineon.com

[3] Selmer Center, Department of Informatics, University of Bergen, Norway
erik.martensson@uib.no

[4] Department of Electrical and Information Technology, Lund University, Sweden
erik.martensson@eit.lth.se

[5] Digital Security Group, Radboud University, Nijmegen, The Netherlands simonas@cs.ru.nl

Abstract. In LWE-based KEMs, observed decryption errors leak information about the secret key in the form of equations or inequalities. Several practical fault attacks have already exploited such leakage by either directly applying a fault or enabling a chosen-ciphertext attack using a fault. When the leaked information is in the form of inequalities, the recovery of the secret key is not trivial. Recent methods use either statistical or algebraic methods (but not both), with some being able to handle incorrect information. Having in mind that integration of the side-channel information is a crucial part of several classes of implementation attacks on LWE-based schemes, it is an important question whether statistically processed information can be successfully integrated in lattice reduction algorithms.

We answer this question positively by proposing an error-tolerant combination of statistical and algebraic methods that make use of the advantages of both approaches. The combination enables us to improve upon existing methods – we use both fewer inequalities and are more resistant to errors. We further provide precise security estimates based on the number of available inequalities.

Our recovery method applies to several types of implementation attacks in which decryption errors are used in a chosen-ciphertext attack. We practically demonstrate the improved performance of our approach in a key-recovery attack against Kyber with fault-induced decryption errors.

Keywords: Kyber · LWE · Belief Propagation · Lattice Reduction · SVP · Implementation Attack

1 Introduction

The advent of quantum computing renders cryptography based on classical hard problems potentially unsafe. In response to this newly arising threat, the National Institute of Standards and Technology (NIST) began to standardize post-quantum cryptography in 2016 [Nata]. Several of the candidates and the algorithms selected for standardization are based on hard lattice problems [Natb]. While classical cryptography has seen extensive cryptanalysis and research on side-channel attacks for decades, implementation security of post-quantum schemes is a comparably new topic. The combination of Chosen Ciphertext

Attack (CCA) with side-channel and fault attacks is a particularly underdeveloped field. The imminent standardization makes understanding such attacks a pressing matter.

A commonly used underlying problem for lattice-based cryptography is the Learning with Errors (LWE) problem. Closely related and reducible to LWE are Ring Learning with Errors (RLWE) and Module Learning with Errors (MLWE). Especially MLWE-based schemes offer competitive key sizes and execution time, making these schemes especially well-suited for embedded devices, and thus a target of implementation attacks. Kyber, a CCA2-secure Key Encapsulation Mechanism (KEM) based on the MLWE problem [BDK+18], has been selected for standardization [Natb]. Therefore, understanding properties of LWE-based schemes in general – and MLWE-based schemes in particular – with regards to side-channel analysis is particularly important.

Several side-channel attacks on LWE schemes, including multiple single trace attacks, have been proposed [PH16, PPM17, PP19, ACLZ20] and methods to at least partially protect against side-channel attacks, such as [RRVV15, RRdC+16, OSPG18, BDH+21, HP21], are already known. A newly emerging topic is the combination of CCA with side-channel analysis allowing for greatly improved key recovery even under the presence of countermeasures as shown in [RRCB20] and [HHP+21] with improvements against countermeasures in [HSST22]. Another common target is the comparison operations of the Fujisaki-Okamoto transform (FO) [FO99], for which several protection mechanisms have already been proposed [BDH+21, DHP+22].

KEMs based on an LWE-type problem are commonly constructed from a Public-Key Encryption (PKE) scheme using an FO to obtain a CCA2-secure KEM [FO99, ABD+21a, AAB+19, BCD+16]. Without such a transform, or when the re-encryption check is disabled by an implementation attack, schemes are vulnerable to CCA. Additionally, if decryption errors occur, an attacker can obtain information on the secret key. In the area of fault attacks, a well-known strategy is to skip the check of the FO [VOGR18, OSPG18, BGRR19, XIU+21]. Another recent approach is a failure boosting type attack using a rowhammer fault injection as presented by Fahr et al. in [FKK+22]. In 2021, Pessl and Prokop [PP21] presented a fault attack targeting the decoder, thereby causing decryption errors. This allows them to obtain one inequality involving the secret key per applied fault. To recover the secret key from that information, they developed a method using continued Bayesian updating. Hermelink et al. [HPP21] then presented an attack using a fault in combination with a chosen-ciphertext to obtain similar information. They also improved upon the recovery method of [PP21] by using a belief propagation based approach to solve systems of inequalities. In contrast to [PP21], the fault used by [HPP21] targets public data and can be applied in several places over a long execution time. In addition, an unreliable fault may be used, but in this case more faults are needed. A follow-up attack was presented by Delvaux in [Del22]. Their attack allows for a more forgiving fault application and uses a newly developed solver also allowing for incorrect inequalities using an approach previously described in [PP21].

Another recovery also leveraging statistical information was presented in [FKK+22]. Their failure boosting attack produces inequalities from which they recover the secret key. In contrast to the above-mentioned attacks [PP21, HPP21, Del22], these inequalities arise from a different kind of attack, therefore carrying different amounts of information. In addition their attack targets FrodoKEM instead of Kyber.

A more generally applicable framework to integrate side-channel information has been presented by Dachman-Soled et al. in [DDGR20]. The framework of [DDGR20] has been used in [BDH+21] to estimate the impact of their attack and in [FKK+22] for a comparison. A very recently published refinement by Dachman-Soled et al.[DGHK22] also covers the decryption errors arising in [FKK+22]. The inequalities arising in [FKK+22] are of the same form but carry more information than those obtained in other recent attacks such as [PP21, HPP21, BDH+21, Del22, DHP+22].

Retrieving the secret key from information coming from decryption errors is a crucial part of several side-channel and fault attacks. The attacks presented in [PP21, BDH+21, BDH+21, HPP21, DHP+22, Del22] all rely on solving the same kind of inequalities. In addition, any kind of implementation allowing observation of decryption errors[1] is vulnerable to an attack similar to [HPP21] using a chosen-ciphertext which then also requires an attacker to solve such inequalities. Several recovery methods exist, but none of the currently known approaches allows for practical, efficient, error-tolerant recovery while giving security estimates for partial attacks – a comprehensive approach unifying all advantages is missing.

Our contribution. We present a new recovery algorithm which uses an error-resistant coding theoretic approach combined with lattice reduction techniques. We thereby leverage statistical as well as algebraic information, improving upon previous methods, and are able to give estimates on the remaining security level in cases where a full key recovery is not practically feasible.

Firstly, we consider inaccurate information, i.e. incorrect inequalities, and integrate them into the solver of [HPP21]. For this, we provide a new type of belief propagation node, factoring in the possibility of inaccurate information. We show that if the fraction of incorrect inequalities stays below a certain threshold, the secret may still be extracted using belief propagation and greatly improve upon the method of [Del22]. Secondly, we embed the information obtained by running belief propagation into a primal attack on the LWE instance. To achieve this, we first integrate fully recovered coefficients, leveraging additional information not available in the more general framework of [DDGR20, DGHK22]. We then use the remaining information to reduce the difficulty of solving the resulting Closest Vector Problem (CVP) instance. This problem in turn is transformed into a unique Shortest Vector Problem (uSVP) instance. Finally, we estimate the security level of the uSVP instance, coming from the partial key recovery in the case of Kyber for several settings. We evaluate our recovery method on the example of Kyber [BDK+18] as Kyber has been selected for standardization in the NIST contest [Natb]. Additionally, by targeting Kyber, our results are comparable to the work of Pessl and Prokop, Hermelink et al., and Delvaux.

Our recovery method applies to several kinds of attacks. This includes previous attacks such as [PP21, BDH+21, HPP21, DHP+22, Del22]. Further, whenever an implementation attack allows to observe decryption errors and can be combined with a CCA, an attacker can obtain information from which the secret key can be recovered using our method. Our work is open-source[2], practical, can be run on widely available hardware, and also allows for an estimate on remaining security if the secret cannot be recovered. We use fewer inequalities, are more resistant to errors, and give refined estimates on the security remaining after belief propagation, compared to previous methods.

Outline. In Section 2, we summarize the preliminaries, especially the work of Hermelink et al. and Delvaux. Section 3 then states our attack including the modified belief propagation, the recovery from partially recovered keys and the methodology of estimating the remaining security. The results section, Section 4, provides evaluations and compares our work to previous approaches. Finally, Section 5 concludes with a summary and a recommendation.

2 Preliminaries

We use lowercase letters to denote elements of a base ring R and lowercase bold letters, $\mathbf{u}, \mathbf{v}, \mathbf{s}, \mathbf{e}, \ldots$, to denote vectors over the ring, i.e. elements of $\mathcal{R}^k$. Upper case bold letters,

[1] Note that decryption errors differ from decapsulation errors.

[2] The code is available under https://github.com/juliusjh/improved_decryption_error_recovery.

e.g. $\mathbf{A}$, denote matrices over the base ring. In the case of plain LWE, the base ring is simply $R = \mathbb{Z}_q$ while in MLWE schemes, most notably Kyber, R is $\mathbb{F}_q[x]/(x^n + 1)$. Since MLWE instances can be seen as LWE instances by ignoring the additional structure, in most of the section, we will work over $\mathbb{Z}_q$.

2.1 LWE and the Primal Attack

We do not give a full summary of LWE-based KEMs, but only a short reminder on the structure of the secret key and the primal attack. For a more detailed introduction to lattice-based cryptography, we refer to [MR09]. LWE-based KEMs work with vectors over a ring $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, where q is a prime in Kyber but not necessarily a prime in other schemes. The secret key is a small vector $\mathbf{s}$ and the public key is commonly given by $\mathbf{s}\mathbf{A}^\top + \mathbf{e}$, where $\mathbf{A}$ is a public matrix and $\mathbf{e}$ is also small and secret, but usually discarded after key generation. The secrets are sampled from a distribution D with small standard deviation σ, such as a central binomial distribution. In RLWE instead of vectors over $\mathbb{Z}_q$, polynomials over $R = \mathbb{Z}_q[x]/(f)$, for some f, are used. MLWE uses vectors over such a ring R, i.e. vectors of polynomials. The polynomial f is often chosen to be cyclotomic. Note that a MLWE/RLWE sample can be seen as an LWE sample by simply ignoring the additional structure; this results in more structured samples compare to a random LWE instance. In the following, we will work over $\mathbb{Z}_q$, instead of over $\mathbb{Z}_q[x]/(f)$.

LWE-based scheme rely on the hardness of lattice problems, such as the CVP and the uSVP. The uSVP asks to find the shortest vector of a lattice while the closely related closest vector problem asks for the closest vector to any given point (which is of course not necessarily a lattice vector). For a full definition and more context, we once again refer to [MR09].

Primal attack. An LWE instance, as typically arising from LWE-based schemes, given by $\mathbf{A} \in \mathbb{Z}_q^{m\times n}, \mathbf{b}, \mathbf{e} \in \mathbb{Z}_q^m, \mathbf{s} \in \mathbb{Z}_q^n$ with $\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b}$ can be embedded into a CVP. The secret vector $(\mathbf{e}, \mathbf{s})$ can be found by looking for an element of the lattice generated by the rows of

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} \\ \mathbf{A}^\top & \mathbf{I}_n \end{pmatrix} \tag{1}$$

close to the target vector $(\mathbf{b}, \mathbf{0})$. To solve the CVP instance, a common method is to embed into a uSVP using Kannan's embedding [Kan87], i.e. searching for the smallest vector in the lattice generated by

$$\mathbf{B}_{\mathrm{svp}} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & 0 \\ \mathbf{A}^\top & \mathbf{I}_n & 0 \\ \mathbf{b} & \mathbf{0} & c \end{pmatrix} \tag{2}$$

where c is set to the standard deviation of the secret distribution or often simply to 1 as an approximation. By now applying a lattice reduction algorithm, such as BKZ, to $\mathbf{B}_{\mathrm{svp}}$, the secret key can be recovered. In [DDGR20], Dachman-Soled et al. also proposed a more sophisticated way of dealing with the choice of c by using an isotropization step. A detailed introduction to the primal attack and generally attacks on and security of LWE can be found in [APS15].

2.2 Soft Analytical Side-Channel Attacks and Belief Propagation

In 2014, Veyrat-Charvillon et al. introduced a new approach to key recovery in side-channel attacks [VGS14]. Inspired by Low-Density Parity Check (LDPC) codes [Gal62], Soft Analytical Side-Channel Attack (SASCA) applies the theory of decoding linear codes to retrieve a secret key from side-channel information. Typically, probability information on key coefficients is obtained by recording power traces or introducing a fault. This is then fed into belief propagation, a message-passing algorithm exchanging information between nodes in a factor graph, e.g. modeling arithmetic relations. If the information obtained from the side-channel was sufficient, BP converges to the correct key. SASCA has been applied against several symmetric schemes such as AES, e.g. in [GS15], hash functions such as Keccak, e.g. in [KPP20], and also against post-quantum schemes [PPM17, PP19, HHP+21, HSST22]. The attacks of [PP21], [HPP21], and [Del22] can be seen as SASCAs using information obtained by faults instead of from a side-channel attack.

Belief propagation, first introduced in [Gal62], here described as in [Mac03], is a message passing algorithm modeling the relationship between unknown variables using a bipartite graph. Unknown random variables $\mathbf{x} = \{x_i\}_{i \in \{1,\ldots,N\}}$ on a set X are represented by *variable nodes* connected to *factor nodes*, where the latter represent the logical connection between two variables. Variable nodes are commonly initialized with a prior, coming from observed information such as measurements. In each step, message from variable nodes are sent to factor nodes and vice-versa. Let the joint mass function be given by

$$\prod_{k=1}^{K} f_k(\mathbf{x}_{I_k})$$

for some K, with $I_k \subseteq \{1,\ldots,N\}$ and f_k functions mapping $\mathbf{x}_{I_k} = \{x_i\}_{i \in I_k}$ to $[0,1]$. Then, denoting incoming messages from i to j by m_{ij} and outgoing messages by $\widehat{m}_{ij}$, variable nodes compute the messages sent to factor nodes by

$$\widehat{m}_{i,j}(x) = \prod_{k \neq j} m'_{k,i}(x)$$

and messages coming from factor nodes are given by

$$m_{j,i}(x) = \sum_{\mathbf{x}, x_i = x} f_j(\mathbf{x}) \prod_{k \neq i} m_{k,j}(x).$$

Upon convergence, the marginal distributions of an unknown variable j is given by

$$x \mapsto \frac{\prod_i m_{i,j}(x)}{\sum_y \prod_i m_{i,j}(y)}$$

and can be computed efficiently.

2.3 Decryption Errors in LWE-based KEM

Common LWE-based KEMs such as Kyber [BDK+18], FrodoKEM [BCD+16, ABD+21a], or NewHope [ADPS16, AAB+19] define a PKE from which the KEM is derived using a (variant of a) FO. Given a PKE, a KEM can be described using the PKE functions.

From PKE to KEM. To establish a shared secret, one party first encrypts the hash of a randomly sampled message using the public key of a previously established key pair. The second party obtains the message using their secret key and re-encrypts it using the public key. The re-encrypted ciphertext is then compared to the originally received ciphertext. If this check passes, then a shared secret derived from the message and the public key is established. Figure 1 shows a simplified version similar to the transform used in Kyber.

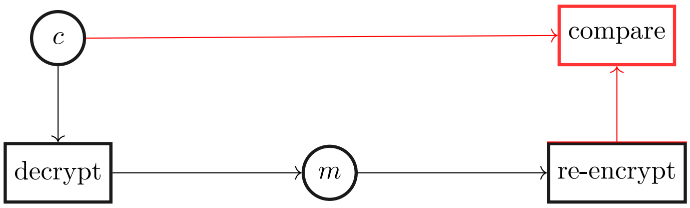

Figure 1: Decapsulation in Kyber as depicted in [HPP21]. An incoming ciphertext is stored, decrypted, re-encrypted and finally the re-encrypted ciphertext is compared to the incoming ciphertext. The areas vulnerable to the attack of [HPP21] are marked in red.

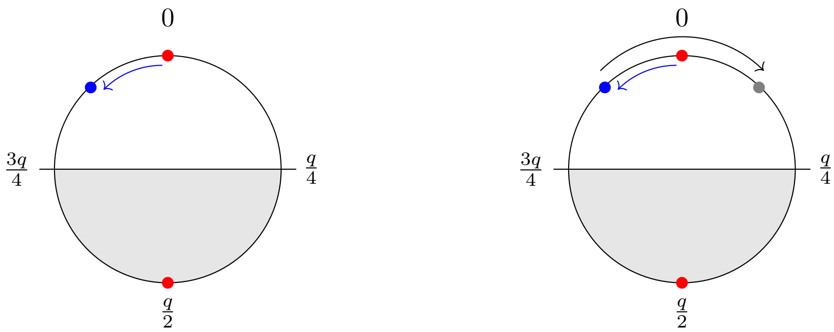

Figure 2: Recovering message bits from noisy coefficients: A noisy coefficient (blue dot) is mapped to a 0-bit if closer to 0 than to $q/2$. An introduced error (grey) moves the point by a quarter rotation. In this example, the addition of $\lceil q/4 \rfloor$ does not cause a decryption error as the initial decryption error term (blue) is smaller than 0[6].

Decryption errors. All LWE-based schemes rely on recovering the messages from a noisy version for decryption (as part of the PKE)[3]. To map message bits to coefficients of a vector over $\mathbb{F}_q$, a decoding routing is called. A bit b is mapped to $m = 0$ if $b = 0$ and to $m = \lceil q/2 \rfloor$ in case of $b = 1$[4]. Then, in the decryption routine, b has to be recovered from a noisy version of m given by $\widetilde{m} = m + v$ with v being a small value[5]. To achieve this, $\widetilde{m}$ is mapped to 0 if $\widetilde{m}$ is closer to 0 than to $\lceil q/2 \rfloor$, and to 1 otherwise. The process is visualized in Figure 2.

The error term v arises out of the decryption routine and is given by an affine linear function of the secret key which can be computed from public parameters of the decryption call. For example, in the case of Kyber, the error term is given by

$$v = \mathbf{e}^\top \mathbf{r}_j - \mathbf{s}^\top (\mathbf{e_1}_j + \Delta \mathbf{u}_j) + e_{2j} + \Delta v_j \tag{3}$$

where $\mathbf{e}, \mathbf{s}$ are the secrets, $\mathbf{r}_j, \mathbf{e_1}_j, e_{2j}$ are known terms of the j-th decapsulation, and the Δ-terms arise from compression and are public as well. Note that an attacker gaining information about the error term immediately obtains information about the secrets $\mathbf{e}$ and $\mathbf{s}$.

Compression. Kyber additionally applies compression to ciphertexts. To reduce the ciphertext size and to increase the security, lower bits of the ciphertexts are discarded. The compression routines with compression factor d are similar to decoding and encoding

[3] Note the distinction between decryption (PKE) and decapsulation (KEM).
[4] With $\lceil \cdot \rfloor$ denoting rounding to the nearest integer with ties being rounded up as in [ABD+21b].
[5] Interpreted in $\mathbb{F}_q$, which means that the value is close to 0 or close to q.
[6] Smaller than 0 when reduced mod q symmetrically around 0.

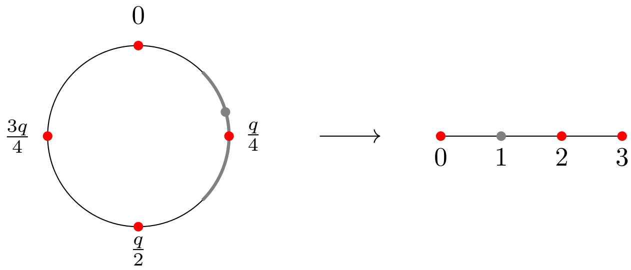

Figure 3: Compression in Kyber visualized with $d = 2$. They gray segment, including the exemplary point near $q/4$, is mapped to 1. In Kyber, ciphertexts are compressed with $d_v \in \{4, 5\}$ and $d_u \in \{10, 11\}$.

of the message bits to coefficients: An uncompressed ciphertext coefficient c_i is mapped to $k \in \mathbb{N}$, where $k\lceil q/2^d \rfloor$ is the multiple of $\lceil q/2^d \rfloor$ which is the closet to c_i. A compressed coefficient k is then mapped back to $k\lceil q/2^d \rfloor$ during decompression. This means that compression and decompression are given by

$$\begin{aligned} \text{compress} &= \lceil (2^d/q)x \rfloor \bmod 2^d \\ \text{decompress} &= \lceil (q/2^d)x \rfloor . \end{aligned}$$

Encoding of message bits is the same as compression with $d = 1$. Figure 3 visualizes Kyber's the compression routine with $d = 2$.

2.4 Fault Attacks on Kyber

Several fault attacks targeting information leakage through the error term have been proposed. In [PP21], Pessl and Prokop presented a fault attack on the decoding in Kyber, which was subsequently improved by Hermelink et al. in [HPP21] and extended by Delvaux [Del22]. We refer to the respective papers for details on the attacks, but give a short explanation on how information is retrieved from applying faults or chosen-ciphertexts with faults.

All three attacks cause an addition of $\lceil q/4 \rfloor$ to the error term by either faults or a combination of faults and chosen-ciphertexts. In some cases this causes a decryption error which can be observed as a decapsulation error either directly or through the use of a fault. Thereby, the FO is used as an oracle for decryption errors. A noisy coefficient is mapped to zero when closer to 0 than to $\lceil q/2 \rfloor$. Thus, an attacker may deduce that an error term must have been less than or equal to 0 when adding $\lceil q/4 \rfloor$ does not cause a decryption error and larger than or equal to 0 when it does cause a decryption error.

For example, the attack of [HPP21] sends a manipulated ciphertext ct' to the attacked device. The manipulation of the ciphertext consists of adding $\lceil q/4 \rfloor$ to a single coefficient to an honestly created ciphertext ct. Due to compression this turns into a one or two bit difference, depending on the originally created ciphertext, which is created offline and under the attackers control. As depicted in Figure 1, the device now stores the ciphertext, decrypts and re-encrypts it, and then compares against the stored ciphertext. The stored ciphertext is faulted back to the original ciphertext ct. This means that the device compares $\mathrm{encrypt}(\mathrm{decrypt}(\mathrm{ct}')) = \mathrm{ct} = \mathrm{encrypt}(\mathrm{decrypt}(\mathrm{ct}))$ which is false if and only if a decryption error happened or the fault failed. Now, assuming a perfect fault, there are two cases, as identified in [HPP21]:

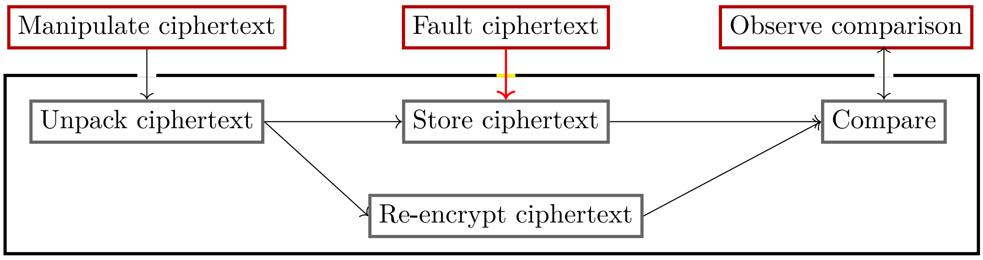

Figure 4: Flowchart of the attack of [HPP21] before recovery. The attacker (top row) sends in a manipulated ciphertext, faults the stored ciphertext, and observes decryption errors by observing decapsulation errors. From the observed successes/failures, they can derive an inequality.

1. A successful decapsulation is observed: This means that adding $\lceil q/4 \rfloor$ does not cause a decryption failure and the error term is less than 0 as visualized in Figure 2.

2. A decapsulation failure is observed: This means that a decryption failure has occured, caused by the addition of $\lceil q/4 \rfloor$.[7] Therefore, the error term is greater or equal to 0.

Thereby, an inequality on the error term can be observed. Recalling Section 2.3, the error term in Kyber has the form $v = \mathbf{e}^\top \mathbf{r}_j - \mathbf{s}^\top(\mathbf{e_1}_j + \Delta \mathbf{u}_j) + e_{2j} + \Delta v_j$ with the notation from (3). The error term can be rewritten as $v = (\mathbf{e}, \mathbf{s})^\top(\mathbf{r}_j, -(\mathbf{e_1}_j + \Delta \mathbf{u}_j)) + e_{2j} + \Delta v_j$, and the two above cases correspond to obtaining one of the following two inequalities[8]

1. $(\mathbf{e}, \mathbf{s})^\top(\mathbf{r}_j, -(\mathbf{e_1}_j + \Delta \mathbf{u}_j)) \leq -e_{2j} - \Delta v_j + \left\lceil \frac{q}{4} \right\rfloor$

2. $(\mathbf{e}, \mathbf{s})^\top(\mathbf{r}_j, -(\mathbf{e_1}_j + \Delta \mathbf{u}_j)) \geq -e_{2j} - \Delta v_j + \left\lceil \frac{q}{4} \right\rfloor$.

Thus, as the error term entails the secret key, an inequality carrying information about the secret key can be derived.

In the attacks mentioned above, the value of Δv_j and b_j are under the control of the attacker and can be chosen to be small which improves recovery of the secret key. In case of an unreliable fault[9], [HPP21] makes use of only successful decapsulations as a successful decapsulation is not possible without a working fault[10]. Note that in schemes without compression, an attacker can obtain equations instead of inequalities. In this case several queries on the same coefficient with different values for the error term, instead of $\lceil q/4 \rfloor$, result in an equation. The attack of [HPP21] is a special case of a class of attacks. Whenever an attacker can observe decryption errors by using an implementation attack, they can derive information about the secret key by using a chosen-ciphertext. In [HPP21], such a decryption error oracle is obtained by using a fault, but various other methods are possible. For example, an attacker could use side-channel analysis to differentiate whether a ciphertext was correctly decrypted or lead to a decryption failure.

2.5 Recovering the Secret Key

After having recovered sufficient information about the secret key, either in the form of equations or inequalities, an attacker is left with solving for the secrets $\mathbf{e}$ and $\mathbf{s}$. In contrast to a purely algebraic system of inequalities, in this case, additional statistical information is available. As the attacker is aware of the coefficients of the key being samples from a

[7] With extremely low probability, a decryption error could also occur naturally.
[8] Note that depending on the message bit, one of the inequalities is strict.
[9] A fault not faulting [the correct value] in every case
[10] With very high probability.

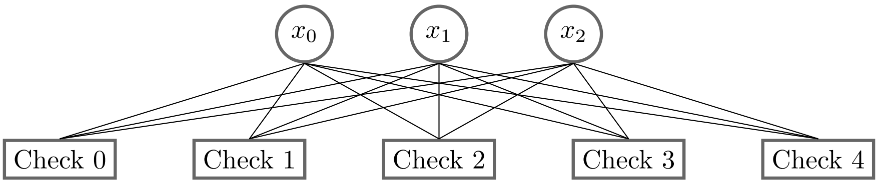

Figure 5: An example for a check graph as used in [HPP21] with 3 variables and 5 inequalities.

certain distribution, usually binomial, they may include this in the solving process. In addition, an attacker may also use the equations or inequalities recorded to improve upon or solve the LWE instance posed by the crypto-system.

2.5.1 Solving Systems of Inequalities using Statistics

Several approaches, starting with the work of Pessl and Prokop in [PP21], exist. An attacker ignoring the information given by the LWE problem can retrieve the secret key by solving the system of inequalities obtained from observing decryption errors, leveraging statistical information.

Pessl and Prokop [PP21] developed an algorithm recovering the key by repeated Bayesian updating. They initialize a vector containing an entry for each key coefficient with the distribution the key was sampled from. Then, in each iteration, the key is updated using the system of inequalities. With enough inequalities available, this converges to a vector of distributions with the correct key coefficients being the most likely ones in each position.

Hermelink et al. [HPP21] improved upon Pessl and Prokop's method by using belief propagation. A factor graph similar to those used in decoding of LDPC codes (introduced in [Gal62]) is constructed where each key coefficient is represented by a variable node and initialized by the distribution the coefficients were sampled from. Each inequality is translated to a factor node which, in each iteration, performs a similar updating as in the algorithm of Pessl and Prokop. Figure 5 shows an example graph with 3 variables and 5 inequalities, similar to an example from [HPP21]. A check node representing the inequality

$$\sum_i a_i x_i \leq b \tag{4}$$

where x_i denotes the unknown coefficients of the secret vector $\mathbf{x} = (\mathbf{e}, \mathbf{s})$ works as follows: Incoming messages m_i, coming from the variable nodes, represent the distributions on x_i, and we denote the corresponding random variable by X_i. First, all distributions

$$S_j = \sum_{i \neq j} a_i X_i \tag{5}$$

are computed. To compute the distributions of leave-one-out sums S_j, the convolutions of the m_i have to be computed. In [HPP21], this is achieved efficiently using Fourier transforms and a tree structure to avoid recomputations. Then, the outgoing messages are given by

$$\widetilde{m}_j = \{c \mapsto \mathrm{P}(S_j + a_j c \leq b)\} \tag{6}$$

where c runs over all possible values of coefficients, i.e. the domain of the binomial distributions e and s were samples from.

This means that a factor node representing an inequality of (4), first computes the distributions S_j of all sums leaving out exactly one coefficient x_j (5). Then, for each

Table 1: Approximate number of inequalities needed to recover the key with success rates 1. In all attacks ciphertexts are filtered. Pessl and Prokop use an older version of Kyber allowing for easier recovery in the case of Kyber512.

Method	Kyber512	Kyber768	Kyber1024
Hermelink et al. [HPP21]	6000	7000	8500
Delvaux [Del22]	9000	9300	12100
Pessl and Prokop [PP21]	8000	10500	13000

random variable X_j, for each possible value c, the probability of the sum plus the remaining term $a_j x_j$ with the unknown coefficient set to c, i.e. $x_j = c$, is computed (6). This then represents the probability distribution used for updating in the variable nodes.

Using belief propagation, the recovery method in [HPP21] achieves improvements in the recovery itself, i.e. fewer inequalities are needed for successful recovery, as well as in RAM usage, allowing an attacker to use widely available and cheap hardware for the recovery. This recovery method is also used by [DHP+22].

Delvaux [Del22] presents a third method which is similar to both the methods used by Pessl and Prokop in [PP21] and Hermelink et al. in [HPP21], but allows for the inclusion of probably incorrect inequalities. This is important for their attack and improves both previous attacks. As a failing decapsulation in the attack of [HPP21] either means that a decryption error happened or that the fault failed, they have to discard those inequalities assuming that the fault is unreliable. Delvaux in [Del22] uses Bayesian updating which includes the possibility of the inequality sign being switched with the probability of the fault failing for failing decapsulations. Their improvement also brings the runtime from minutes to seconds, but in turn, more inequalities are needed, as shown in Table 1[11].

2.5.2 Integration into LWE

In 2020, Dachman-Soled et al. presented a framework for the integration of hints into LWE which can be seen as a generalization of the primal attack on LWE [DDGR20]. The LWE instance is first embedded into the Distorted Bounded Distance Decoding Problem (DBDD). Side-channel information is then integrated by subsequently modifying the DBDD instance. After all side-channel information has been applied, the DBDD instance is embedded into a uSVP which can be solved using lattice reduction techniques such as the Blockwise Korkine-Zolotarev (BKZ) algorithm. The attack of [BDH+21] uses the framework of Dachman-Soled et al. to estimate the remaining security. The framework allows for integration of several types of hints. Equations can be directly integrated as perfect hints using the framework while inequalities need to be converted to approximate hints. Both variants have been used for estimates in [BDH+21] in a slightly different setting.

In [DGHK22], Dachman-Soled et al. presented an extension of their framework focusing on inequality type hints. They use a geometric approach to integrate inequality hints in a way consistent with their previous framework. To evaluate their work, they use inequalities as in [FKK+22] coming from a failure boosting attack, potentially carrying more or different information than the inequalities used to evaluate the other methods. Their work delivers estimates in long but reasonable computation time, but the integration of hints to fully recover the key from inequalities carrying little information seems to be impractical for an average attacker on large LWE instances. In addition, they do not perform a full key recovery and do not use Kyber. Therefore, we do not compare their

[11] Delvaux published their results as "success probability" which seems to mean the average number of correctly guessed key coefficients, leading to a "success probability" of about 0.3 for zero inequalities, i.e. for plain Kyber. Therefore, we only compare success rates of 1.

work to the works of [PP21], [HPP21] and [Del22] here.

3 Improved Integration using Belief Propagation

Recall from Section 2.5.1 that Pessl and Prokop in [PP21] developed a practical algorithm to recover the secret key from error-term inequalities. This was improved in terms of the required number of inequalities and practicability in [HPP21]. However, both methods are unable to handle incorrect inequalities which lead to the method of [Del22]. Their recovery process allows recovering the secret key even with incorrect information present, but in turn requires a greatly increased number of inequalities and therefore faults or measurements. Neither of those methods allow for an estimate of the remaining security – either the attack succeeds or fails. The more general framework of [DDGR20] can be used to estimate remaining security, but only includes a small part of the information and therefore overestimates the remaining security (c.f. estimates in [BDH⁺21]) in this case. In addition, it seems to be of limited practical use for recovery in the case of large LWE instances (c.f. [DHP⁺22]).

We strive to achieve a general approach that applies to the class of attacks of [PP21, BDH⁺21, HPP21, DHP⁺22, Del22]. We do this by using the combination of a statistical and a lattice reduction approach. To achieve error-resistance, our belief propagation needs to be able to deal with unreliable information; we describe how to do this in Section 3.1. To employ a lattice reduction approach, we need to integrate statistical information coming from belief propagation outputs into an algebraic lattice setting. This requires two steps, namely integrating fully recovered coefficients in an optimal way and then using the remaining information to reduce the complexity of the CVP instance, which is described in Section 3.2. Our experiments, shown in Section 4, confirm that we improve upon the number of needed inequalities compared to [HPP21], can give an estimate on the remaining security based on a practical recovery method and are more error-resistant than [Del22].

3.1 Adapting to Incorrect Inequalities

The error-resistance in [Del22] is achieved by considering the unreliability of inequalities into the Bayesian updating step. We also include this information, but use it in the updating step in each check node of the belief propagation. Our graph is similar to the graph of [HPP21] described in Section 2.5.1, but we use different check nodes using a different updating function. As in Section 2.5.1, we describe how messages are updated in a check node that represents an inequality

$$\sum_i a_i x_i \leq b, \tag{7}$$

where x_i denote the unknown coefficients of the secret vector $\mathbf{x} = (\mathbf{e}, \mathbf{s})$. In each iteration, the check node receives messages m_i coming from the variable nodes. The message m_i represents the distribution of the secret coefficient x_i and we denote the random variable by X_i. As in [HPP21], we first compute the leave-one-out distributions s_j for the random variables

$$S_j = \sum_{i \neq j} a_i X_i. \tag{8}$$

We know that for each j

$$a_j x_j + \sum_{i \neq j} a_i x_i \leq b \tag{9}$$

and S_j is the random variable representing $\sum_{i \neq j} a_i x_i$. Therefore, we can update the distribution of X_j using S_j.

During initialization, we pass the information on the correctness of each inequality to each check node representing it. This information is a property of the concrete attack, e.g. coming from the reliability of a fault. Note that the probability differs with each inequality and is not a global constant, but a property of each individual check node. Let p be the probability of the inequality (7) being correct. For a value c in the domain of the initial distribution, the probability of x_i being c is given by

$$
\begin{aligned}
&\mathrm{P}(x_j = c) && (10)\\
=&\mathrm{P}(x_j = c \mid \sum_i a_i x_i \leq b) + \mathrm{P}(x_j = c \mid \sum_i a_i x_i > b) && (11)\\
=&p\,\mathrm{P}(x_j = c \mid S_j \leq b - c) + (1-p)\,\mathrm{P}(x_j = c \mid S_j > b - c). && (12)
\end{aligned}
$$

Thus, our updated outgoing messages are given by

$$\widetilde{m}_j = \{c \mapsto p\,\mathrm{P}(S_j \leq b - c) + (1-p)\,\mathrm{P}(S_j > b - c)\} \tag{13}$$

and the variables nodes update the distributions accordingly in the next step.

As the update as described in (10) computes more terms than a node representing a certainly correct inequality, our computational effort is higher than in [HPP21]. However, since the belief propagation is done within a couple of minutes to a few hours, practically this penalty is not relevant and negligible compared to the other part of the key recovery, namely the lattice reduction.

3.2 Integrating Partially Recovered Keys to LWE

In case sufficiently many inequalities are available, the belief propagation converges, and the attacker does not need any additional tools. In practical scenarios, however, this is often not the case: An attack may have to be aborted early, a device under attack might shut down, or side-channel information may be very expensive to obtain. Therefore, realistically, an attacker may be left with insufficiently many inequalities for a pure belief propagation based recovery.

In this section, we present an approach to efficiently integrate the output of partial belief propagation runs into the LWE instance. By using both statistical and algebraic information, we maximize the likelihood of success. Additionally, we estimate the remaining security that remains after partial belief propagation has been performed.

Our attack works in two stages. Firstly, we reduce the dimensionality of the LWE instance by integrating recovered coefficients. Secondly, we reduce the complexity of the resulting CVP instance by finding a closer target vector. The whole integration process is depicted geometrically in Figure 6.

3.2.1 Belief Propagation Output

Partially converged belief propagation gives probability distributions at the variable nodes, i.e. for each coefficient of the secret vector $(\mathbf{e}, \mathbf{s})$. This means that we have $2n$ probability distributions in addition to the LWE equation system $\mathbf{s}\mathbf{A}^\top + \mathbf{e} = \mathbf{b}$. Unfortunately, these distributions cannot be assumed to be independent and the covariance matrix is unknown, at least with our methods. Therefore, we see the distributions as rankings rather than proper independent probability distributions. We additionally have a metric on how likely a coefficient is correct by looking at the min-entropy[12].

[12] The logarithm of the highest probability value.

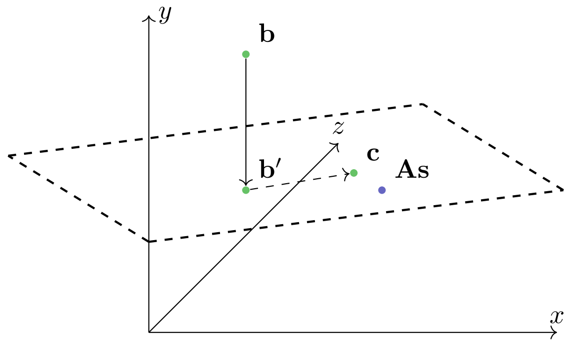

Figure 6: Simplified integration with one recovered coefficient in 3 dimensions. The closest vector problem is first (solid arrows) projected onto the hyperplane (indicated by the dashed rectangle) generated by a recovered coefficient. Then, integrating the remaining information (dashed arrows) leads to a CVP instance with a closer point in the same dimension.

In [HPP21], Hermelink et al. sort the coefficients by several metrics while we use min-entropy only. Throughout these sections, we assume coefficients to be sorted. We call coefficients *correct* if the highest ranked value is the actual value of that coefficient. That means, if D_i is the distribution of the i-th coefficient, and

$$\tilde{x}_i = \operatorname{argmax}_v P_{D_i}(X_i = v),$$

then the i-th coefficient is correct if and only if $x_i = \tilde{x}_i$. In this metric and with this definition of correctness, let the first r coefficients be correct and let the $(r+1)$-th coefficient be incorrect.

We then assume that an attacker has enough computational power to perform the attack $r+1$ times. As one of those attacks being successful suffices, we can assume that the attacker knows that the first r coefficients are correct and call them the *recovered coefficients*. Note that the computational complexity of our algorithm is very low, except for the lattice reduction. The belief propagation needs to run a single time only and the subsequent lattice reduction can be run only when the estimated β matches the available computational resources. In practice, we propose the following procedure for an attacker who can run at most BKZ-β'

- runs belief propagation,
- starts off with a heuristic guess r' for r,
- estimates β and increases r' until $\beta \leq \beta'$,
- runs the lattice reduction.

If the attack fails with r', it was chosen too large; but a larger BKZ-β cannot be handled and the recovery would fail with every smaller r' as well. Thereby, finding the correct r in a practical attack has almost no additional cost.

3.2.2 Integrating Recovered Coefficients

A generally applicable way to integrate recovered coefficients is decribed in [DDGR20]. But our situation is not entirely covered as we have additional information on the reliability of our remaining coefficients. We therefore apply our recovered coefficients not only in the right order, but also in a way eliminating the most unreliable remaining coefficients. We assume that the attacker has carried out $r+1$ tries on recovery and therefore r to be the number of recovered coefficients. We denote $r = r_s + r_e$, where r_s is the number of recovered coefficients of $\mathbf{s}$ and r_e is the number of recovered coefficients of $\mathbf{e}$. Recovered coefficients are denoted by $\hat{e}_i$ and $\hat{s}_j$. Let $\{i_1, \ldots, i_n\}$ denote the ordering on $\mathbf{e}$, i.e. coefficients x_{i_k} for $k \in \{1, \ldots, r_e\}$ are recovered. Similarly, let $\{j_1, \ldots, j_n\}$ denote the ordering on $\mathbf{s}$, i.e. coefficients x_{j_k+n} for $k \in \{1, \ldots, r_s\}$ are recovered.[13] In contrast to [DDGR20], we do not apply the homogenization and isotropization steps and embed the LWE instance directly into a uSVP after integration.

Integrating recovered coefficients of s. This is straightforward by intersecting with the subspace given by the recovered coefficients $\hat{s}_{j_k}$ with $k \in \{1, \ldots, r_s\}$. In other words, the corresponding columns of the LWE matrix $\mathbf{A}$ are deleted and subtracted from $\mathbf{b}$. For each $\hat{s}_{j_k}$ with $k \in \{1, \ldots, r_s\}$ we delete the j_k-th column and subtract $A_{i,j_k} \cdot \hat{s}_{j_k}$ from b_i for all $i \in \{1, \ldots, n\}$. This can also be written as

$$\mathbf{A}' = \mathbf{A}\mathbf{T}_\mathbf{s} \text{ and } \mathbf{b}' = \mathbf{b} - \mathbf{t}_\mathbf{s} \tag{14}$$

where $\mathbf{T}_\mathbf{s}$ is the $m \times n - r_\mathbf{s}$ matrix obtained from the unit matrix by removing each column j_k for $k \in \{1, \ldots, r_\mathbf{s} - 1\}$ and

$$\mathbf{t}_\mathbf{s} = \left(\sum_{k=1}^{r_\mathbf{s}} \hat{s}_{j_k} \cdot A_{i,j_k} \right)_{i \in \{1, \ldots, m\}}. \tag{15}$$

Geometrically the whole process can be seen as intersection with r_s hyperplanes. To keep the notation simple, we also denote the resulting LWE instance after integrating the recovered coefficients of $\mathbf{s}$ to be given by $\mathbf{A}$ and $\mathbf{b}$ with the same dimensions as before. An implementer also needs to keep updating indices whenever a recovered coefficient is integrated; to keep our description concise, we do not describe this process.

Integrating recovered coefficients of e. Integrating a recovered coefficient of $\mathbf{e}$ also allows us to eliminate a coefficient of $\mathbf{s}$ from the equality system. Given a recovered $\hat{e}_i$, the i-th LWE equation yields

$$s_j = A_{i,j}^{-1}(b_i - \hat{e}_i - \sum_{k \neq j} A_{i,k} s_k), \tag{16}$$

assuming that $A_{i,j}$ is invertible. We may therefore save (16) and remove the j-th row as well as the i-th column from $\mathbf{A}$ while adjusting all other entries of $\mathbf{A}$ and $\mathbf{b}$ accordingly. Naturally, we aim at eliminating the least reliable coefficients. Therefore, we first choose $j = j_{m-1}$, continue with the index j_{m-2}, and so on up until and including j_{m-r_e} possibly changing the order to assure invertible entries of $\mathbf{A}$. We may again interpret this geometrically as intersection with hyperplanes given by equations of the kind in (16). In contrast to when integrating recovered coefficients of $\mathbf{s}$, we also remove a row of the matrix, namely the row of the recovered $\hat{e}_i$. In matrix-vector notation, integrating $\hat{e}$ can be written as

$$\mathbf{A}' = \mathbf{T}_\mathbf{e}(\mathbf{A} - \mathbf{T}'_\mathbf{e}) \text{ and } \mathbf{b}' = \mathbf{b} - \mathbf{t}_\mathbf{e} \tag{17}$$

where $\mathbf{T}_\mathbf{e}$ removes the corresponding columns,

$$\mathbf{T}'_\mathbf{e} = \left(\sum_{l=1}^{r_e} A_{i,j_l}^{-1} A_{i,j} \right)_{i,j} \quad \text{and} \quad \mathbf{t}_\mathbf{e} = \left(\sum_{l=1}^{r_e} A_{i,j_l}^{-1} (\hat{e}_i - b_i) \right)_i \tag{18}$$

with i running over $\{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. When having recovered (half of) the remaining coefficients of $\mathbf{e}$ and $\mathbf{s}$ later, in most cases after having run lattice reduction, we may use (16) to recover s_j as well.

[13]Note that coefficients of $\mathbf{s}$ correspond to columns in the LWE matrix while those of $\mathbf{e}$ correspond to rows.

3.2.3 Integrating Remaining Coefficients

Let $\mathbf{A}'$ and $\mathbf{b}'$ denote the transformed versions of $\mathbf{A}$ and $\mathbf{b}$ after having integrated both types of recovered coefficients. The dimensionality of the problem is now reduced. We have $n - r_e$ remaining equations, $n - r$ unknown values in $\mathbf{s}$ and $n - r_e$ unknown values in $\mathbf{e}$. Additionally, we still have estimated probability distributions for all the remaining coefficients of $\mathbf{e}$ and $\mathbf{s}$. Let us use $\mathbf{e}'$ and $\mathbf{s}'$ to denote the remaining positions in $\mathbf{e}$ and $\mathbf{s}$.

A naïve approach to solve this transformed problem would be to use a key enumeration algorithm as described in e.g. [BLvV15, MOOS15, GGP+15, PSG16]. Thereby, one might be able to improve slightly on edge cases. In the majority of cases, the belief propagation either recovers all coefficients or only very few to none at all. Thus, applying key enumeration will only be successful in very few cases and obtaining a few more inequalities will often be the attacker's preferred option.

Instead, we perform a variant of the well-known primal attack as also used in the framework of [DDGR20]. Embedding our LWE instance into a CVP yields the lattice given by the rows of

$$\mathbf{B}_{\text{CVP}} = \begin{pmatrix} q\mathbf{I}_{n-r} & \mathbf{0} \\ \mathbf{A}'^{\top} & \mathbf{I}_{n-r_e} \end{pmatrix}. \tag{19}$$

By searching for vectors close to the target vector $\mathbf{t} = (\mathbf{b}, \mathbf{0})$ we will find the lattice vector $(\mathbf{b}, \mathbf{0}) + (-\mathbf{e}', \mathbf{s}')$ and thereby recover the secret key. Leveraging the additional information coming from the belief propagation, we can refine the CVP instance. Denote the best guesses, i.e. taking the most likely coefficient of $\mathbf{e}$ and $\mathbf{s}$ in each position, by $\hat{\mathbf{x}}' = (\hat{\mathbf{e}}', \hat{\mathbf{s}}')$. Then, $\mathbf{x}'$ is closer to $\hat{\mathbf{x}}'$ than it is to $\mathbf{0}$. Thus, $\mathbf{c} = (-\mathbf{b}, \mathbf{0}) + \hat{\mathbf{x}}'$ is much closer to a lattice point and the CVP can therefore be solved more efficiently for $\mathbf{c}$. Embedding into a uSVP, we are left with finding the shortest vector of

$$\mathbf{B}_{\text{SVP}} = \begin{pmatrix} q\mathbf{I}_{n-r} & \mathbf{0} & 0 \\ \mathbf{A}'^{\top} & \mathbf{I}_{n-r_e} & 0 \\ \mathbf{b} - \hat{\mathbf{e}}' & \hat{\mathbf{s}}' & 1 \end{pmatrix}, \tag{20}$$

which can be achieved using lattice reduction algorithms such as BKZ. Figure 6 shows a simplified visualization of both parts of the integration process. From the uSVP instance the remaining security may be estimated in terms of the BKZ-β needed to find the secret as shown in [ADPS16, AGVW17] and reiterated in [DDGR20]. Note that the coefficient we chose to be 1 in the lower right corner of $\mathbf{B}_{\text{SVP}}$ is optimally set to the standard deviation of the error distribution as already remarked by [DDGR20]. The choice of c is made irrelevant by applying isotropization, but we apply neither an optimally chosen c nor isotropization to keep our recovery method as simple as possible. Using the tweaks with regards to the primal attack, an attacker may improve upon our method.

3.2.4 Additional techniques

Several techniques we did not employ could allow room for improvement.

Additional Key Enumeration. We may apply key enumeration at several stages of our recovery process. Firstly, we may increase the number of recovered and then integrated key coefficients, i.e. before the integration described in Section 3.2.2), resulting in higher r_e and r_s. Secondly, we may enumerate after the integration of recovered coefficients. While key enumeration in this case only will allow for a direct key recovery in edge cases with almost fully recovered secrets, it allows for improved recovery if used together with lattice reduction. After integrating known coefficients, as described in Section 3.2.2, the dimension of $\mathbf{x}'$, i.e. the dimension of $\mathbf{B}_{\text{CVP}}$ is reduced compared to the original secret. As the attacker is looking for a close vector, guessing a partially incorrect key increases the computational effort required to solve the uSVP instance, but does not stop the attack. Note that compared to the former case, enumeration at the latter stage does not require the attacker to correctly guess the key. A mere improvement in some positions resulting in a closer vector already gives an advantage.

To improve upon the latter case, an attacker may first enumerate a large number of candidates and then group those by their distances. Selecting only the most likely candidate from each group ensures that no unnecessary duplicate calls to a lattice reduction algorithm are made. Using this strategy a particularly determined attacker may slightly improve upon the success rate in practice.

Dual and Hybrid Attacks. We only described our recovery method using the primal attack. Recent developments on the dual attack [GJ21, MAT22, AS22] imply that the dual attack might outperform the primal attack when solving the underlying LWE problem of Kyber. The dual attack performs lattice reduction on parts of the secret and systematic guessing on the other parts. The cost of these steps are additive (see for example [MAT22, Theorem 5.1]). So the dual attack benefits a lot from a sparse secret, allowing for guessing of more positions. For particularly sparse secrets, hybrids between a meet-in-the-middle attack and a dual attack can also be considered [CHHS19].

For our particular setting, the most reliable secret values are very sparse, after belief propagation and transformation of the problem are performed. Figuring out the more precise design and performance of such attacks requires knowing more precisely how sparse the positions are. Here it would for example be useful to know how many of the most reliable coefficients can be considered as binary or trinary.

4 Implementation and Results

In this section, we first describe our implementation and then list the results obtained from estimating the remaining security after running belief propagation and integration, as well as success rates. We evaluate our recovery method using the attack of [HPP21] against Kyber512. Thereby, we not only compare our method to [HPP21]. but also to [PP21], and [Del22], as their attacks all target Kyber. A comparison to the methods of Dachman-Soled et al. [DDGR20, DGHK22] and Fahr et al. [FKK+22] proves to be more difficult due to their nature of inequalities and targeted scheme. Their recovery method was motivated by an attack leading to a different kind of inequalities compared to those arising from the attacks we focus on. We note that an estimate of applying [DDGR20] to inequalities similar to the ones used here can be obtained from [BDH+21], but note that no key recovery was performed. The inequalities in the form used here arise not only in fault attacks, such as [PP21, HSST22, Del22], but also from side-channel analysis such as [BDH+21] and [DHP+22] as well as in the class of attacks using a chosen-ciphertext to exploit a vulnerability in an implementation, which allows observing decryption errors.

4.1 Implementation

The starting point for our implementation is the open source implementation of [HPP21][14] which is written in Python, Rust, and uses calls to PQClean [PQC] to simulate the fault attack. From the output obtained from PQClean, inequalities are derived in Python and fed into a parallelized belief propagation written in Rust. We modified their belief propagation to include the improvements of Section 3.1 and to improve performance in terms of memory usage.

Additionally, instead of aborting after a belief propagation run if not enough coefficients were found, we instead apply the strategy of Section 3.2. To do this, we compute the

[14] Their implementation can be found at https://github.com/juliusjh/fault_enabled_cca.

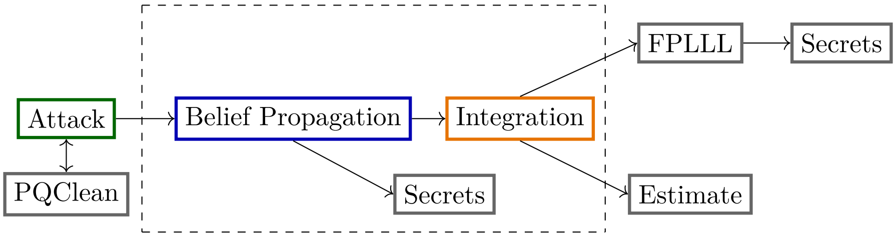

Figure 7: Flow chart of the evaluation using our recovery method on the attack of [HPP21]. The upper row is only used in an actual attack and not needed for estimates. Note that the attack of [HPP21] is merely exemplary for evaluation purposes but our recovery method applies more generally. A flow chart of the attack of [HPP21] (green) is depicted in Figure 4, our belief propagation (blue) is described in Section 3.1, and our integration (orange) is described in Section 3.2. The dashed box shows our recovery method.

matrix $\mathbf{B}_{\text{svp}}$, estimate the required BKZ-β and then either call FPLLL [dt22] or only output the estimate. Figure 7 depicts a flow chart giving an overview of the evaluation.

4.2 Security Estimates

In contrast to [PP21], [HPP21], and [Del22] our recovery method allows for an estimate of the remaining security, similar to the framework of [DDGR20, DGHK22], by applying well-known estimates on the hardness of lattice reduction. In this section, we report our findings with regard to remaining security and compare against previous work in terms of success rates.

After we obtained the uSVP instance as described in Section 3.2.3, we can estimate the remaining security in terms of BKZ-β using an estimate given in [ADPS16, AGVW17] and reiterated in [DDGR20]. Denoting the lattice generated by the rows of $\mathbf{B}_{svp}$ by Λ and the root Hermite factor by δ_β, the BKZ-β needed to solve the uSVP instance satisfies

$$||s||\sqrt{\beta/\dim(\Lambda)} \leq \delta_\beta^{2\beta-\dim(\Lambda)-1}\,\mathrm{Vol}(\Lambda)^{1/\dim(\Lambda)}. \tag{21}$$

The root Hermite factor can be estimated and the volume of Λ is clearly given by q^{n-r}. This allows us to easily estimate the remaining security in terms of the required BKZ-β.

4.3 Results

This section show the results of our recovery runs. We ran 20 samples per number of inequalities for less than 25000 inequalities and 5 runs for higher numbers. We ran up to 50 full belief propagation iterations, where by a full iteration we mean an iteration from variable nodes to factor nodes and then vice-versa. Out of the 50 possible steps, we continue with the data obtained from the step with the most correct coefficients. We note that this leads to an attacker needing to be able to run several instances of lattice reduction. The cost introduced by choosing the best step and guessing the right number of recovered coefficients scales the total cost by at most a linear factor. This can be decreased further by using our statistics on the average number of recovered coefficients and starting with the last step as this is usually among the best steps.

No incorrect inequalities. The situation with only inequalities known to be correct is depicted in Figure 8. For the filtered case, this shows the situation in case of the attack of [HPP21]; in addition we also ran unfiltered inequalities. We then perform our integration of belief propagation data and give the results in terms of BKZ-β in Figure 8. Note

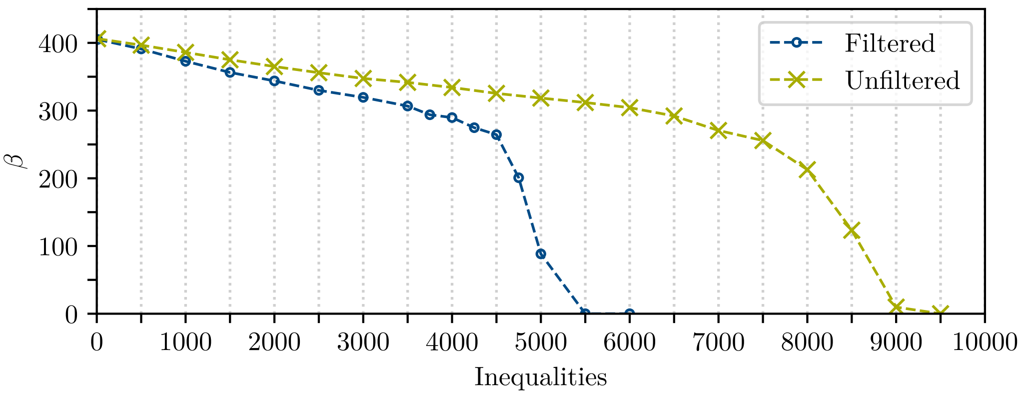

Figure 8: The average remaining security in terms of BKZ-β when all inequalities are correct. Notice the sharp drop corresponding to the belief propagation recovering more coefficients around 4000 inequalities.

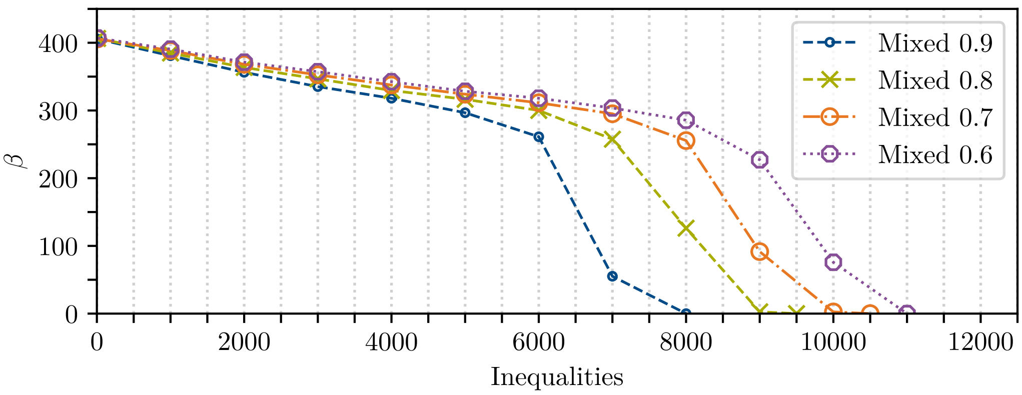

Figure 9: The average remaining security in terms of BKZ-β with incorrect inequalities when half of the inequalities are known to be correct. All inequalities are filtered with the method used in [HPP21].

the sharp drop around 4000 inequalities in Figure 8 coming from the belief propagation recovering more coefficients.

Incorrect inequalities. We distinguish between two settings in which incorrect inequalities can occur. Firstly, an attacker may only have access to potentially incorrect inequalities. Here, we need to consider every inequality to be incorrect with a certain probability. Let those inequalities be correct with probability p. Secondly, an attack may result in some inequalities being certainly correct and others potentially incorrect as e.g. in [HPP21] and [Del22]. In this setting, we can assume half of the inequalities (those coming from a decapsulation success in [HPP21]) to be correct with probability one and the other half to be potentially incorrect with the probability for correctness being p. The former setting is shown in Figure 10 with $p \in \{0.8, 0.9\}$. The latter setting is depicted by Figure 9 with $p \in \{0.6, 0.7, 0.8, 0.9\}$. We note that in case of fault with reliability less than p, i.e. p less than 0.6 in the latter setting, there is no advantage to include inequalities that are not certainly correct. Our evaluations only cover some settings which have already occurred in previous work. New attacks, combinations, or individual circumstances in practice may results in different configurations. Our implementation is flexible in that regard and can easily be adjusted by setting command line parameters.

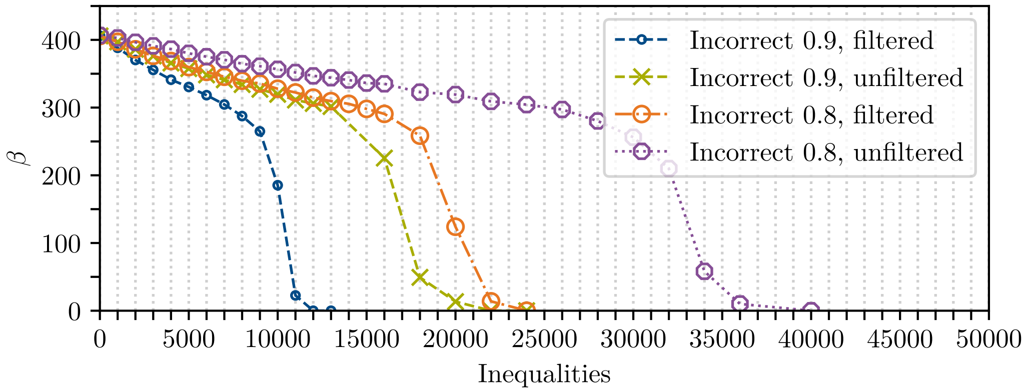

Figure 10: The average remaining security in terms of BKZ-β in two settings with incorrect inequalities. The filtering is performed as in [HPP21].

Table 2: Approximate number of inequalities needed to recover the key with success rates 1 with filtered ciphertexts. [PP21] uses an older version of Kyber allowing for easier recovery. [Del22] uses a slightly different filtering method. Using the method of [Del22], we weren't able to solve the 0.8 incorrect inequalities setting with more than 300000 inequalities.

Method	Correct	Incorrect 0.9	Incorrect 0.8	Mixed 0.9	Mixed 0.6
Pessl and Prokop [PP21]	8000	n.a.	n.a.	n.a.	n.a.
Hermelink et al. [HPP21]	6000	n.a.	n.a.	n.a.	n.a.
Delvaux [Del22]	9000	34000	not solved	12000	21000
This work	5500	12000	24000	8000	11000

Success rates. Assuming an attacker can run BKZ-70, Table 2 lists the number of inequalities needed for a successful attack in different settings with filtered ciphertexts. Note that the filtering in the work of Delvaux differs slightly, but comparing unfiltered results, we assume the impact to be small (c.f [Del22, Figure 4]). In [PP21], Pessl and Prokop can only work with a very small number of incorrect inequalities and [HPP21] does not consider incorrect inequalities at all. The numbers for only potentially incorrect inequalities are not stated by [Del22]. Therefore we slightly modified their implementation to also cover the case where no inequalities are certainly correct.

Belief propagation runtimes. While our method is less efficient than the one of Delvaux, we note that for less than 10000 inequalities we stay below an hour of runtime for the belief propagation on a commonly available CPU. As the belief propagation scales well, an attacker can improve by using a large number of cores. In practice, obtaining traces or applying fault is usually much more expensive than obtaining more CPU cores. Therefore, we believe the drastically reduced number of inequalities justify the increased recovery times. In case the belief propagation is not successful on its own, the lattice reduction will take up the largest portion of the runtime.

4.4 Adapting to Frodo.

We added a variant recovering the key for an attack on FrodoKEM-640 (as specified in [ABD+21a]). While there are no fundamental issues, the large error distribution in Frodo poses a few implementational challenges and requires more resources. In addition to implementing our recovery method proposed in Section 3, we also optimized the already existing parts of the belief propagation. Running with 28 threads working on

4000 inequalities required about 60 GB of RAM and took several hours for a single run. Nevertheless, the computational resources needed to run the belief propagation part of our recovery method for FrodoKEM can easily be obtained; the limiting factor for an attacker is either the lattice reduction or the ability to perform the fault application or the side-channel measurements on an actual device.

Different kinds of inequalities. Using our implementation for Frodo, we applied our recovery method to the inqualities of [FKK+22] which are also used to test the framework of [DGHK22]. This was not successful with the 4000 inequalities used for evaluation in [DGHK22], as the belief propagation quickly tends to a wrong solution with large coefficients. To solve this, we further added message dampening and a random schedule mode to the belief propagation. This also did not lead to any improvements. Nevertheless, our belief propagation implementation now allows for using these features with Frodo and Kyber using a simple flag. In its current form our recovery method is unsuitable for this kind of inequalities and we recommend using the framework of [DGHK22]. Note that we mainly focused on the kind of inequalities arising in the majority of implementation attacks and a combination of both methods may be possible. We leave this for future work.

5 Conclusion

We present a new algorithm to recover the secret key when given access to a decryption failure oracle. In contrast to previous work, we make use of both the information represented by the LWE instance as well as the retrieved inequalities. Using statistical as well as algebraic methods, we combine the advantages of previous approaches. Our method is both practical and also allows for a theoretic estimate of the remaining security if the secret cannot be fully retrieved. We thereby improve upon previous attacks and provide a tool for refining estimates of implementation security. Our method applies to several previously published attacks as well as to the class of attacks combining an implementation attack that allows constructing a decryption failure oracle with a chosen-ciphertext.

Future work. While our methods are practical and efficient, it is yet open if there is a more sophisticated method replacing the proposed key enumeration of Section 3.2.2. By using the information on remaining coefficients more efficiently than by brute-force, further improvements might be achieved. Relying on the dual instead of the primal attack including recent developments, as described in Section 3.2.4, may also allow for a more efficient attack. Another open question is whether the integration of Dachman-Soled et al. in [DGHK22] can be combined with belief propagation. This, to us, seems to be a non-trivial question whose answer holds potential of huge improvements. Further, we do not answer the question if our method can be adapted to inequalities as arising from Fahr et al. [FKK+22]. We only apply the recovery method to Kyber, for which a strong compression makes obtaining equality hints infeasible. This is due to the imminent standardization and to compare against previous works, which all targeted Kyber. Nevertheless, the optimal attack strategy for different schemes is an open question.

Acknowledgements

We would like to thank Hunter Kippen for the helpful discussion on the key recovery of [FKK+22] and the method presented in [DGHK22]. Erik Mårtensson was supported by the project "Kvantesikker Kryptografi" from the National Security Authority of Norway. Julius Hermelink and Peter Pessl were supported by the German Federal Ministry of Education and Research (BMBF) under the project "PQC4MED" (16KIS1041).

References

[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. https://newhopecrypto.org/data/NewHope_2019_07_10.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. 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. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[ACLZ20] Dorian Amiet, Andreas Curiger, Lukas Leuenberger, and Paul Zbinden. Defeating NewHope with a Single Trace. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 2020.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 327–343. USENIX Association, 2016.

[AGVW17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *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*, volume 10624 of *Lecture Notes in Computer Science*, pages 297–322. Springer, 2017.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[AS22] Martin R. Albrecht and Yixin Shen. Quantum augmented dual attack. Cryptology ePrint Archive, Paper 2022/656, 2022. https://eprint.iacr.org/2022/656.

[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 Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018. ACM, 2016.

[BDH+21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and defending masked polynomial comparison for lattice-based cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):334–359, 2021.

[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*, pages 353–367. IEEE, 2018.

[BGRR19] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. Assessment of the Key-Reuse Resilience of NewHope. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 272–292. Springer, 2019.

[BLvV15] Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptol. ePrint Arch.*, page 221, 2015.

[CHHS19] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE. *IEEE Access*, 7:89497–89506, 2019.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020.

[Del22] Jeroen Delvaux. Roulette: A diverse family of feasible fault attacks on masked kyber. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):637–660, 2022.

[DGHK22] Dana Dachman-Soled, Huijing Gong, Tom Hanson, and Hunter Kippen. Refined security estimation for LWE with hints via a geometric approach. *IACR Cryptol. ePrint Arch.*, page 1345, 2022.

[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. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):115–139, 2022.

[dt22] The FPLLL development team. fplll, a lattice reduction library, Version: 5.4.2. Available at https://github.com/fplll/fplll, 2022.

[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 Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 979–993. ACM, 2022.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.

[Gal62] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[GGP+15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015.

[GJ21] Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving lwe with applications to crystals. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 33–62, Cham, 2021. Springer International Publishing.

[GS15] Vincent Grosso and François-Xavier Standaert. Asca, SASCA and DPA with enumeration: Which one beats the other and when? In Tetsu Iwata and Jung Hee Cheon, editors, *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*, volume 9453 of *Lecture Notes in Computer Science*, pages 291–312. Springer, 2015.

[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. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):88–113, 2021.

[HP21] Daniel Heinz and Thomas Pöppelmann. Combined Fault and DPA Protection for Lattice-Based Cryptography. *IACR Cryptol. ePrint Arch.*, 2021:101, 2021.

[HPP21] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. Fault-Enabled Chosen-Ciphertext Attacks on Kyber. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, volume 13143 of *Lecture Notes in Computer Science*, pages 311–334. Springer, 2021.

[HSST22] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. Adapting belief propagation to counter shuffling of ntts. *IACR Cryptol. ePrint Arch.*, page 555, 2022.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

[Mac03] David J. C. MacKay. *Information theory, inference, and learning algorithms.* Cambridge University Press, 2003.

[MAT22] MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack, April 2022.

[MOOS15] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *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, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015.

[MR09] Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Nata] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[Natb] National Institute of Standards and Technology. Status report on the third round of the NIST post-quantum cryptography standardization process. https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical CCA2-Secure and Masked Ring-LWE Implementation. *TCHES*, 2018(1):142–174, 2018.

[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*, pages 1–6. IEEE Computer Society, 2016.

[PP19] Peter Pessl and Robert Primas. More Practical Single-Trace Attacks on the Number Theoretic Transform. In Peter Schwabe and Nicolas Thériault, editors, *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*, volume 11774 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2019.

[PP21] Peter Pessl and Lukás Prokop. Fault Attacks on CCA-secure Lattice KEMs. *TCHES*, 2021(2):37–60, 2021.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017.

[PQC] Contributors to PQClean. PQClean. 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 Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. *TCHES*, 2020(3):307–335, 2020.

[RRdC+16] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptogr. Eng.*, 6(2):139–153, 2016.

[RRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A Masked Ring-LWE Implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *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*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[VOGR18] Felipe Valencia, Tobias Oder, Tim Güneysu, and Francesco Regazzoni. Exploring the Vulnerability of R-LWE Encryption to Fault Attacks. In John Goodacre, Mikel Luján, Giovanni Agosta, Alessandro Barenghi, Israel Koren, and Gerardo Pelosi, editors, *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems, CS2 2018, Manchester, United Kingdom, January 24, 2018*, pages 7–12. ACM, 2018.

[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. *IACR Cryptol. ePrint Arch.*, 2021:840, 2021.