Overview and Discussion of Attacks on CRYSTALS-Kyber

Stone Li

The Country Day School, King City, Canada, stoneli2006@gmail.com

Abstract. This paper reviews common attacks in classical cryptography and plausible attacks in the post-quantum era targeted at CRYSTALS-Kyber. Kyber is a recently standardized post-quantum cryptography scheme that relies on the hardness of lattice problems. Although it has undergone rigorous testing by the National Institute of Standards and Technology (NIST), there have recently been studies that have successfully executed attacks against Kyber while showing their applicability outside of controlled settings. These include, but are not limited to, fault injections and side-channel attacks. This paper will discuss the effectiveness and details of common attacks, side-channel attacks, side-channel assisted chosen-ciphertext attacks, and fault-injection attacks, as well as possible defenses and their applicability against these attacks on Kyber. This paper aims to provide future researchers insight into what areas should be focused on to strengthen current as well as future cryptosystems. Some attacks discussed include chosen power analysis, timing attacks, primal and dual attacks on the underlying learning-with-errors problem, fault injections, and electromagnetic attacks.

Keywords: Lattice-based Cryptography · Kyber · Side-Channel Attacks · SCA assisted CCA · Fault-Injection Attacks

Contents

1 Introduction

In recent years, as quantum computing has developed at an unprecedented pace, the need and importance of post-quantum cryptography have also grown significantly. With Shor's proposal of an algorithm that can solve prime factorization and discrete-logarithm problems in polynomial time on a quantum computer [Sho94], classical asymmetric cryptography is no longer considered secure against large-scale quantum computer attacks. Due to this, NIST has held a competition for post-quantum cryptography schemes. Kyber, Dilithium, and SPHINCS+ are some of the participants who have made standardization after round 3, and others are undergoing round 4 evaluation at the moment.

The lattice problem is one of the most promising candidates for cryptography schemes for post-quantum cryptography. This problem is also what Kyber's security depends on, more specifically, it uses an intermediate problem proposed by Regev called the Module-Learning-With-Errors Problem [Reg09]. Mathematically, this problem is believed to be a hard problem, making it the basis for many post-quantum cryptography schemes. Although the problem is difficult, NIST has recently realized the vulnerability of the proposed cryptography schemes against side-channel attacks, making NIST send out requests to update schemes to defend against such attacks.

Kyber has now been standardized for its performance in memory storage, computing time, and its improvement in security in comparison to previous lattice-based cryptography schemes, such as NewHope [Nia22]. However, many studies have proven the feasibility of side-channel attacks, and even its feasibility over Kyber with defenses against such attacks. Recently, studies have also reported Kyber's vulnerability to fault injections.

This paper will review Kyber's ability to defend against common attacks, side-channel assisted chosen-ciphertext attacks, side-channel attacks, and fault injections, thus giving researchers an updated view on the current state of security of post-quantum cryptography schemes.

2 Common attacks

2.1 Attacks on module-LWE

For Kyber, since only $m = (k+1)n$ LWE samples will be available for the adversary[ABD+21], BKW-type attacks and linearization attacks can be deemed inapplicable, leaving only BKZ attacks - primal and dual attacks. Recall that the 2 algorithmic approaches for the small vector problem (SVP) oracle in BKZ are enumeration and sieving algorithms [ABD+21].

For Kyber, primal attacks work by constructing a unique smallest vector problem (uSVP) instance from the given LWE problem, and then solving it using a lattice reduction algorithm like BKZ. Kyber states in its supporting documents that the security level is estimated to be a minimum of 151.5, which is Kyber 512 with all circumstances in the adversary's favor [ABD+21]. However, an updated sieve cost and using the more effective dual attack [MAT22][WX22] demonstrates the inaccuracy of the reported security level. Using MATZOV's estimated sieve costs [MAT22] and the asymptotic model [Duc18], the security is brought to 138.2 [MAT22], and using the same estimated sieve cost but with the G6K model [ADH+19], this is brought even lower to 137.5 [MAT22]. Note that the required security level demands a minimum of 143 [oS]. This certainly brings into question the effectiveness of Kyber's chosen parameters against attacks targeting its underlying hard problem.

2.2 Attacks on symmetric primitives

Kyber's symmetric building blocks are all instantiated from KECCAK, more notably, *XOF*, *H*, *G with SHAKE*, and *SHA3*. All of which are modeled and expected to act as random oracles. An attack on any of these instantiations would mean a breakthrough in either KECCAK or random-oracle model proofs in general. However, in the case that any of these instantiations are found to be exploitable, it is possible to switch to other functions of the same effect. Therefore, it can be concluded that attacks against Kyber's symmetric primitives are inviable.[ABD+21]

2.3 Multi-target attacks

Although Kyber does not make any formal claims about security in a multi-target scenario, the design of Kyber has implemented 2 factors that could help defend against this type of attack. First, Kyber adopts the "against-all-authority" approach similar to NewHope, where they re-generate the matrix A for each public key [ABD+21]. This ensures the adversary can not break multiple keys by acquiring the details of 1 key. Second, in the CCA transform, the public key is hashed into the pre-key K and a coin r. By making r dependent on the public key, this defends against precomputed attacks [ABD+21].

2.4 Attacks exploiting decryption failures

Decryption failures typically are the results of either chosen ciphertext attacks or valid ciphertext attacks. When using chosen ciphertext attacks, the adversary generates ciphertexts that intentionally trigger decryption failures based on the properties of the secret key [DB22][DGJ+19]. By repeating the process, the adversary can recover the secret key. However, by using the Fujisaki Okamoto (FO) transform [1] during decryption, Kyber avoids this attack by rejecting the ciphertext input and thereby not allowing the adversary to gain any information on the secret key. Valid-ciphertext attacks, on the other hand, can happen naturally. For example, in Kyber, this could be due to flaws in randomness when generating error terms [ABD+21]. In the case of this attack, the adversary is assumed to recognize the event of a failure from a valid ciphertext. In either case, when a decryption failure occurs, the adversary will gain knowledge about the secret key.

Normally, to find all weak ciphertexts that will most often produce failures, it is needed to run through all possible ciphertexts. However, this is very impractical as the probability of a valid ciphertext decryption failure occurring is extremely low. In turn, adversaries will often try to "failure boost". Failure boosting is a useful technique in at least three scenarios: no multi-target protection, limited queries to the decryption oracle, and access to a quantum computer for the adversary [DGJ+19]. Since Kyber offers minimal multi-target protection and we are assuming that the adversary has access to a quantum computer, failure boosting is a valid technique to improve the applicability of decryption failure attacks.

2.4.1 Weak ciphertext attack

A weak ciphertext attack relies on how often a naturally occurring ciphertext will trigger a decryption failure. To execute the attack, x decryption failures will be collected using failure boosting techniques. Then, the exact error positions and type of failure will be determined for all failure vectors. Using this information, the secret can be estimated and used to simplify the LWE problem. To finish the attack, one just needs to solve the simplified LWE problem. With Kyber's chosen parameters making failure probabilities be $< 2^{-128}$ for security level 1, and $< 2^{160}$ for security levels 3 and 5 [ABD+21], the attack will need more than NIST's set decryption query limit to be considered safe(2^{64}) for a valid ciphertext failure deeming the attack invalid. [DGJ+19]

[1]FO transform first decrypts the ciphertext, then re-encrypts it, and compares the re-encrypted ciphertext to the original ciphertext. Since Kyber's mathematical operations are deterministic, by comparing the two ciphertexts, FO transform can test for its validity.

To boost the ability to find the weak ciphertext, the adversary can use directional failure boosting [DRV20], where the adversary will use a previously found weak ciphertext to improve searches for new ciphertexts, or Grover's algorithm. However, this does not increase the possibility enough for the attack to become viable. [DGJ$^+$19][DB22]

2.4.2 Weak key attack (multi-target attack)

In a weak key attack model, the adversary can only query a limited amount to a single user, but multiple users can be queried. Similar to how certain ciphertexts will fail more often, some keys have a higher tendency to cause failures. The attack procedure can be narrowed down to 3 steps. [DGJ$^+$19][DB22]

1. A precomputation step establishing messages and their ciphertexts. Store all the error vectors in F. The error vectors should be chosen with the goal of being sorted based on their properties.

2. When the ciphertext in F is sent, assume that the adversary learns the decrypted message and that there is a subset in F that causes a decryption failure. The number of subsets causing failures could be above average depending on the properties of the secret key. The adversary then submits a set of ciphertexts to each node holding a public key and selects the node giving the most decryption failures.

3. The adversary will then do statistical testing on the subset causing failures to establish a relationship between the secret key and the noise vectors causing decryption failures. By analyzing the correlation, it is possible to recover partial secrets, which reduces the hardness of the underlying lattice problem considerably. The adversary will finally finish the attack by performing full key recovery attacks using classic algorithms on the underlying problem.

For multi-target attacks, it is also important to keep in mind that Kyber uses rounding, making its distribution curve for the noise polynomial, secret key, and error terms uneven, forcing the adversary to modify some of their failure boost techniques. Generally, for uneven distributions, the adversary can use uneven failure boosting, uneven directional failure boosting, and meet-in-the-middle speed-up. To further speed up uneven directional failure boosting, it is possible to remove bias. D'Anvers et al.'s research [DB22] has also shown that depending on the type of Kyber implemented (Kyber512, Kyber768, Kyber1024), this attack will be inviable due to the number of queries required.

3 Side channel attacks

Side channel attacks are attacks that attempt to extract secrets from a chip or system by analyzing physical parameters. Common side channel attacks include but are not limited to execution time, electromagnetic emission, and power.

3.1 Timing attack

Timing attacks generally work as a signal detection problem, the "signal" consisting of the timing variations. Given i messages, and j time measurements, we will take a guess on the first x bit of the key, and compare it to each other to find the right one. [Koc96]

Since Kyber does not use any secret-dependent branches or any table look-ups, it is time-constant and free from the 2 most notorious sources of timing leakage. However, there is another possible passage for timing leakage, non-time-constant multipliers. However, since Kyber only uses 16-bit inputs, and most non-time-constant multipliers only show time variations for large inputs, such as 32-bits, Kyber is safe from this time of timing leakage as well. Lastly, timing leakage may also be possible through modular reductions, often using conditional statements, but this can also be avoided by using Montgomery and

Barrett reductions. It is also important to try to implement Kyber with a constant-time implementation of AES if hardware support is not available and AES must be used. [ABD+21]

3.2 Soft-Analytical Side-Channel Analysis (SASCA)

In a single trace, the adversary has to extract as much information regarding the target variable as possible from it. In this case, SASCA is a very potent tool to perform single-trace attacks. SASCA are profiled attacks, which work by templating leakage from sequential operations and then processing the secret value. To execute the attack, the adversary obtains a single trace, matching it with the templates, and then using the template-matching information to acquire the secret key. During this process, belief propagation (BP) is also often used, a message-exchange algorithm that exchanges information between nodes in a factor graph. With enough information, BP will converge to the correct key [HMS+23]. Although initially studied for symmetric key cryptography, it has also found application in lattice-based cryptography. [RCDB22]

3.2.1 Targeting NTT

Primas et al. showed in their paper [PPM17] that it is possible to recover a secret key by recovering a single trace of an NTT operation operating over the secret key. More precisely, Primas et al. [PPM17] focused on the public key encryption algorithm, where NTT is computed over several sensitive intermediate values. Thus, this NTT operation can be exploited by SASCA to reveal its inputs.

During the profiling phase, side-channel templates are created from leakages from the clone device, for intermediate operations such as storing input and output of butterfly operations and modular addition, subtraction, and multiplication within the NTT.

Once the templates have been completed, the key recovery phase begins. The adversary can take a single trace related to the leakage from the target NTT, which will then be sorted based on its targeted internal operation and appropriate template. Then, the results from the template that matches are modeled into a factor graph based on the NTT implementation, which will be fed to a BP algorithm to recover the secret key.

Primas et al. [PPM17] first proposed this attack on a generic Ring-LWE-based PKE scheme, implemented on an ARM Cortex-M4 microcontroller using Kyber's round 2 submission values (q = 7681, $\eta = 4$) [PPM17]. This, however, was incredibly inefficient as it took over a million templates for a successful recovery. Still, in their subsequent work [PP19], they showed that this approach could be optimized to require just a few hundred (around 200) templates, and tested this proposal using Kyber's round 2 (q = 7681, $\eta = 4$) and round 3 submission values (q = 3329, $\eta = 2$), showing its applicability [ABD+21][PP19].

3.2.2 Targeting KECCAK

Kyber uses KECCAK in its key generation, encapsulation, and decapsulation process. During key generation, KECCAK is used as a PRNG which takes a secret seed as input and outputs a pseudo-random bit string. Since KECCAK operations are sequential, it makes it an ideal target for SASCA. [ABD+21]

Kannwischer et al. [KPP20] were able to take advantage of this feature and demonstrated single trace SASCA on KECCAK instances. Therefore, targeting the KECCAK operation over the secret seed can be used to recover the secret key. The detailed process of the attack is similar to the 2 phases shown in the attack targeting NTT. Although they only executed the attack over simulated traces, this attack should also remain applicable in real-world circumstances, such as software implementations on embedded micro-controllers.

Furthermore, since KECCAK is mostly used as a PRF and PRNG in many post-quantum cryptography schemes, including Kyber, this SASCA attack on KECCAK would apply to those schemes as well.

Although we have only discussed Targeting KECCAK and NTT during the encryption phase using SASCA, it is also possible to use SASCA during the decryption and re-encryption phase of Kyber as these also require lots of NTT operations and KECCAK operations.[RCDB22]

To combat the SASCA attack, Ravi et al. proposed generic shuffling and masking countermeasures with varying granularity [RPBC20]. These countermeasures can be extremely effective, although, with the trade-off of efficiency, as Hermelink et al. have demonstrated in their paper [HSST22]. It is fairly important to note that although potent, SASCA does suffer from several aspects, which limit its performance. These aspects are the requirement of elaborate profiling (SASCA may require hundreds of carefully prepared templates using detailed information from the target and its internal operations), the requirement of high-signal to noise ratio (SASCA attacks typically need a high signal-to-noise ratio to be successful, so it is likely that a low-cost countermeasure such as jitter[2] is enough to defend against this attack), and the applicability of an attack to noisy devices (most studies have only executed their attacks on embedded micro-controllers such as the ARM Cortex-M4 with high sound-to-noise ratio, so it is unclear of the applicability of the attack on a low sound to noise ratio device. Also, since most hardware implementations will also use Kyber with parallelism, this will create a lot of algorithm noise, making the attack much less viable). [RCDB22]

3.3 Targeting message encoding

Message encoding and encapsulation in Kyber works as follows. The message of length 256 bits is first hashed using SHA3-256, then K and *seed* are generated by concatenating the hashed message and SHA3-256 hashed private key, then hashing it using SHA3-512. Then, r and 2 small error terms are generated from *seed*, and the whole ciphertext is created by compressing the module-LWE equations. The shared key is then generated using SHAKE-256 on K concatenated with SHA3-256 hashed ciphertext. [ABD+21][SKL+20]

The attack presented by Sim et al. targets the encoding process of Kyber [SKL+20]. During this process, a *mask* value is determined by a sensitive bit μ. The *mask* value will be either 0x0000 or 0xffff. Since the power consumption model is typically based on the Hamming weight of an intermediate value, power traces here can be classified into two sets. When *mask* is equal to 0x0000, its power consumption is proportional to 0, whereas if *mask* is 0xffff, the power consumption should be proportional to 16 as it is a 16-bit integer. Since there are significant changes in performance depending on the position of the attack, Sim et al. [SKL+20] selected points where *mask* is used and stored and defined these points as points of interest (POI). The POIs are then separated into 2 groups using clustering algorithms. By doing so, the adversary can classify power consumption. Since power consumption is related to the Hamming weight of intermediate values, the average values of the 2 groups will be different. By assuming the larger being the Hamming weight, and the lower being the power consumption, the adversary can determine which *mask* value each group corresponds to based on the average. The result is that the message can be recovered, which can then also be used to generate the secret shared key.

Sim et al. [SKL+20] proved the attack by measuring 500 power consumption traces for different messages at a sampling rate of 29.54 MS/s, with Kyber's message encoding algorithm running on the ChipWhisperer UFOSTM32F2 target board, using 5 different optimization levels for the compiler, and applying the *k-means* clustering algorithm. In all 5 different optimization levels, there were no observed error rates as the 2 sets were easily

[2]Non-uniform delays that can cause packets to arrive and be processed out of sequence

distinguishable, meaning that the possibility of retrieving the message is possible with a success rate of 100%.

Since this is a single-trace attack, masking does not serve as a viable option as a defense against attacks targeting message encoding. Ngo et al. have proved this in their paper [NWDP22], although it was tested on Saber, the ideology could be transferred to be applied to Kyber as well, as the underlying ideology of message encoding and masking is similar. As well, since message encoding and decoding work similarly in ideology, the attack could be extended to target message decoding as well [RCDB22].

3.4 Using deep-learning

Artificial intelligence (AI) has been used to break into first [NDGJ21], second, and third-order [NWDP22] masked implementations of Kyber before. However, any higher-order masked implementations were incredibly difficult to break with standard AI profiling and training [DNGW23]. Dubrova et al. [DNGW23] were able to overcome this difficulty by using a different type of deep learning and using rotations on the intercepted message to increase the bits' leakiness and therefore the chance of a successful attack. Dubrova et al. [DNGW23] proposed the attack on a C implementation of a first-order-masking of Kyber [HKL+22], where **masked_poly_frommsg()** is modified to extend to higher-order masking. This procedure is called during the re-encryption phase of Kyber, and the power consumption on this procedure is what will be targeted.

During the profiling stage, unlike previous works, recursive learning will be implemented on the AI. Essentially, to train a w-order masked implementation, the starting network M^w is created by copying the weights of the input Batch Normalization layer of the model M^{w-1} trained on the $(w-1)$-order masked implementation, then extending the layer to include one more share [DNGW23]. When $w \leq 3$, the AI is trained from a network with standard random weight distribution, once $w > 3$, recursive learning is used. By using the cut-and-join training traces byte-wise with the technique of Ngo. et al's paper [NDGJ21], 2 universal models, M_0^w and M_1^w are acquired. These recover the first and second bit of each message byte, which has the strongest leakage. Also, the AIs are trained to recover the message directly without extracting the random masks at each execution, and message bits '0' and '1' are used as labels.

The attack stage uses a cyclic rotation method. This is used because the leakage from **masked_poly_frommsg()** is not uniformly distributed, easily seen by the difference of 9% in the probability of a successful recovery between bit 0 and bit 7 [DNGW23]. This is also possible because module-LWE are extensions of ring-LWEs, whose messages can be cyclically rotated by manipulating their ciphertext. In this attack, the message is negacyclically rotated 3 times by 2 bits, where the last 6 bits of each byte are rotated to the first 2 bits, making these bits leak out more information without the trade-off of excessive time consumption compared to other cycling techniques.

Dubrova et al. [DNGW23] tested the proposed attack using a Chipwhisperer-list board, a CW308 UFO board, and a CW308T-STM32F4 target board running at 24MHz containing an ARM Cortex-M4 CPU with STM32F415-RGT6 device, which implements a Kyber in C with arm-none-eabi-gcc with at optimization level -O3. For each w-order masked implementation, 2.5K messages are selected at random, and with each trace containing 3 2-bit negacyclic rotations of the message, this comes out to 10K traces total. On a first-order masked implementation using 1 trace, the average message recovery probability is 0.127% without cyclic rotations. With cyclic rotations, this probability becomes 78.866%. Using a single trace on a fifth-order masked implementation using cyclic rotations the probability is 0.56%; with 3 traces, the probability is 54.53%; with 5 traces, the probability is 87.085% [DNGW23].

3.5 Low-density-parity-check code

Most attacks discussed previously only retrieve the secret coefficient 1 bit at a time, which can be incredibly costly performance-wise and information-wise. Guo et al. proposed a new framework [GNNJ23] for side-channel attacks that recover information using a low-density parity check of secret coefficients from a single side-channel measurement. The sparse system, which defines the low-density-parity-checking (LDPC) code using matrix H, can then be solved using iterative decoding methods such as BP. This approach allows the attainment of both source compression benefits and error correction advantages due to the combination of secret coefficients, leading to a more uniform extraction of information from a single trace. Also, the correction of erroneous decisions depends on the correct recovery of coefficients through spare parity-check relations. On top of this, it significantly reduces the number of side-channel measurements needed for the correct recovery of the coefficient.

The idea of the general attack is as follows. For a good linear code with the sparse parity-check matrix H_{r*n}, there are $k = n - r$ secret positions to recover. With Kyber being a lattice-based KEM, the value k is divided into b equal-sized blocks. The overall goal is to retrieve the first k secret entries s_i. Each parity check equation will allow the adversary to obtain 1 check variable c_i, $i \in \{k+1, ..., k+r\}$. In lattice-based KEMs, the secret entries s_i are generated according to a central binomial distribution B_μ, which can be used as the prior information for s_i. Furthermore, additional side-channel measurements can be gathered to update the distribution. Here, new ciphertexts are then designed to obtain leakages of sparse linear combination c_i of s_j for j in a finite set, which could reveal an empirical probability for c_i. Now, through a noisy discrete channel, the problem of recovering s_i for $i \in \{i, ..., k\}$ is transformed into a coding problem. An example of the attack is illustrated below in figure 1.

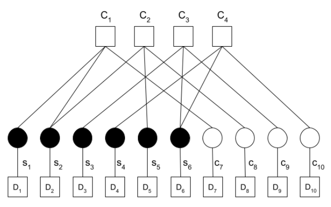

Figure 1: Tanner graph example [GNNJ23]

Assuming 6 secret coefficients need to be recovered, for each s_i, the adversary has more traces to get a better understanding of its distribution. 4 parity checks are shown in the example. With the Tanner graph, iterative decoding can be used to recover the secret coefficients. To generate the code, an LDPC code with mb blocks of the parity-check matrix is first created:

$$H_{ini} = \begin{bmatrix} H_{11} & ... & H_{1b} \\ ... & ... & ... \\ H_{m1} & ... & H_{mb} \end{bmatrix}$$

Vectors H_{ij} are randomly generated with the only constraint being only distances of the multiplicity of 1 are allowed in its distance spectrum [GJS16]. Then, r rows are selected from H_{ini} to form a sub-matrix H' and append $-I_{r*r}$ where I_{r*r} is the identity matrix. In a concatenated construction, the finished code can serve as the outer shell, while the inner codes can be filled by any linear code [GNNJ23].

More specifically for Kyber, the basic key recovery attack that targets message encoding [SKL+20] can be improved upon using the proposed framework [GNNJ23]. Since the basic attack already allows computing for each secret coefficient using inner code, the adversary just needs to create an outer LDPC code. To create the code, the adversary needs information about parity checks c_i, which can be done by implementing another inner code for that purpose. Essentially, the structure against Kyber768 uses 2 inner codes, one which solves for secret coefficients, and one which solves for check variables. Let $u_r = k_u * \sum x^{i^r}$, $r \in \{0, 1, 2\}$. With the help of Oracle O_i, ciphertext (u, v) returns information every 256 parity checks. The outer LDPC code parity-check oracle is, in this case, the form of $H_{ini} = [H_0|H_1|H_2]$, where H_j are negacyclic matrices acquired from the vector connecting c_0 and s_j. The success probability can be increased by increasing the length of either and both inner codes and increasing the number of check blocks.

Guo et al. tested the proposed framework [GNNJ23] on Kyber768 using a ChipWhisperer-Lite board, a CW308 UFO board, and a CW308T-STM32F4 target board with a 32-bit ARM Cortex-M4 CPU running at 24MHz. Kyber was implemented from the mkm4 library with first-order masking [HKL+22] compiled at -O3 optimization level. The procedure **masked_poly_frommsg()** was targeted. During the profiling stage, 100K power traces are collected and used to train 8 neural network models, one for each byte. These models are trained up to 100 epochs. After the profiling stage, the attack stage is executed. Guo et al. [GNNJ23] found the framework to have an average accuracy of 95% and uses 86% fewer traces compared to the majority voting approach with $t = 11$ votes. Guo et al. [GNNJ23] also noted it is possible to improve the overall performance further by using more advanced code-construction methods with improved decoding performance, or heavy post-processing such as lattice-reduction.

4 Side channel attack assisted CCA

Side channel attacks assisted CCAs refer to attacks that use side-channel attacks to improve the effectiveness of adaptive attacks, such as plaintext-checking oracles. These often involve the adversary submitting carefully chosen ciphertext or plaintext, then retrieving the secret key or the message by analyzing the information acquired from side-channel leakages.

4.1 Electromagnetic emission attack

Electromagnetic emission attacks (EM attacks) use the difference in emissions for different bit values when processed and incur an efficient plaintext checking oracle. When the bit value 1 is processed, it will have a different emission than when the bit value 0 is processed. This difference allows the adversary to retrieve the secret key or the message. This attack can be easily used on error-correcting codes, but since Kyber uses the FO transform, the tactic of the attack needs to be modified. For Kyber, Ravi et al. [RSRCB20] proposed targeting a hash function operative over the input message, more specifically, the end of the hash function as the diffusion of the modified bits makes distinguishing much easier due to the induced differential behavior.

Ravi et al.'s proposal [RSRCB20] works as follows. Since the EM attack depends on distinguishing different emissions bit by bit, and the decryption procedure in Kyber takes as input 2 ciphertext parts, u and v, and a secret key module s [ABD+21], the adversary can manipulate u and v bit by bit, then compare EM results. More specifically, to find the first bit, the adversary can set $u[0] = k_u$ and $v[0] = k_v$, and all other coefficients to 0. According to the decryption algorithm of Kyber, the chosen bit(the first bit in this example) can be calculated by $k_v - k_u * s_0[0]$, and all other x bits can be found using

$-1 * k_u * s_0[x]$ [ABD+21][RSRCB20]. This ensures the value of the decrypted message depends solely on the polynomial s_0, which means the adversary now just needs to collect enough traces (5 to cover all possible combinations in this case) of such type of ciphertext to identify the value of $s_0[0]$ based on the decrypted message. By repeating the process for all bits, the adversary will successfully recover the entire key. The true number of traces can be calculated by $5 * n * k$, with n being the size of the secret key polynomial, and k being the number of possible values to test [RSRCB20].

For Kyber512, Ravi et al. [RSRCB20] found the number of traces needed to be just 2560 ($5 * 256 * 2$), with an average success rate of 99% and 3 repetitions for a full key recovery. With each iteration taking around 230 seconds, the total average time was found to be 10 minutes and 50 seconds [RSRCB20], placing EM as a very potent attack. However, the setup to perform such an attack is very delicate, which is a drawback of the EM attack. Countermeasures such as masking complete decryption are valid defenses but are extremely costly in performance.

Furthermore, Rajendran et al. [RRD+23] improved the applicability of this binary Plaintext-Checking (PC) oracle-based side-channel attack by proposing a method to parallelize the attack. By retrieving P number of bits of information from the secret key in a single trace, it cuts down the number of queries required for a successful attack by a factor of P. During their experiment, they also showed their ability to successfully defeat low-cost defenses like shuffling. On top of this, Xu et al. [XPR+20] demonstrated the attack on different parts of the Kyber procedures, such as during its NTT operations or inverse-NTT operations, as well as for differently optimized Kyber implementations.

4.2 Correlation power analysis (CPA)

Polynomial multiplication algorithms such as Toom-Cook and Number Transform Theory (NTT) are fundamental building blocks for lattice-based post-quantum cryptography. Kyber heavily relies on NTT during its encapsulation and decapsulation phase [MBM+22][ABD+21]. Although extremely fast, these algorithms pose the problem of differing power consumption for different polynomial coefficients. It is due to this that we can execute CPA attacks on Kyber.

Generically, CPA works as follows [BCO04]. The adversary first finds an operation that involves the secret value and a public value, then collects n power samples using different known inputs. The adversary will then guess the secret value and compute an intermediate value for all known inputs. Lastly, the adversary will correlate Hamming weight for the intermediate values and power traces for all possible secret values, then pick the secret value with the highest correlation.

Karlov et al. [KdG21] were able to use CPA during the decapsulation process under a semi-static setting[3]. This mode is desirable, especially in embedded devices for its fast key exchange and lack of dependency on randomness.

Karlov et al. [KdG21] executed their attack on an unprotected instance of Kyber-512 from the library pqm4 using the ChipWhisperer Pro Toolkit, which includes an STM32F3 board with an ARM Cortex-M4 micro-controller and the ChipWhisperer-Pro with CW308 UFO target. Karlov et al. [KdG21] performed a vertical CPA for every **basemul** operation during decapsulation. The **basemul** operation is a vector multiplication operation that takes as input 2 x 4 bytes of the recipient's secret key. Kyber from the pqm4 library uses **doublebsemul** which takes as input part of the key $kk' = k_0k_1k_2k_3k'_0k'_1k'_2k'_3$ and part of the cipher text $cc' = c_0c_1c_2c_3c'_0c'_1c'_2c'_3$, and computes 2 **basemul** operations, one with positive ζ and the other negative. Only 4 bytes of the key need to be guessed, $k = k_0k_1k_2k_3$. The steps which Karlov et al. [KdG21] followed were such:

1. Guess values for k_2k_3 and compute Hamming weight for all ciphertexts

[3]One set of keys will remain unchanged throughout the process.

2. Compute the Pearson correlation coefficient (PCC) between the cipher text and the Hamming weight, keeping the highest in absolute value PCC_{k2k3}

3. Repeat steps 1 and 2 for all possible key values and keep a sample of the keys with the highest PCC value.

4. Make guesses for k_0k_1 and compute the result Hamming weight for all of the ciphertexts. Then compute the Pearson correlation between the Hamming weight and the power traces, keeping the largest value in absolute $PCC_{k0k1k2k3}$

5. The part of the key $k_0k_1k_2k_3$ with the highest $PCC_{k0k1k2k3}$ is the correct guess.

Karlov et al. [KdG21] experimented with CPA for different amounts of ciphertexts intercepted and found starting from 50 ciphertexts, the chance of finding the right key starts to rise, and after 80-100 ciphertexts, the chance of finding the right key reaches $> 99\%$, making it another strong candidate for attacks.

Similar to typical CPA attacks, masking serves as a viable option for defense against these styles of attacks, with the trade-off of a slowdown ranging from $2-3.5$ times [BGR+21]. Karlov et al. [KdG21] instead suggested two possible options to defend against this attack. One is to implement Kyber only in ephemeral settings, the other is to regenerate the recipient's secret keys for every x message where $x < 50$. As of current, both of these options work against the discussed CPA but are powerless against single-trace attacks.

5 Fault injection

Fault injections are when the adversary intentionally makes the decryption process faulty, often by faulting a specific operation or skipping an operation as a whole. In the case of Kyber, the use of FO transform means a single-bit error in the message could trigger a huge amount of errors after the hashing function. The most general case of Fault attacks works by comparing the effectiveness of faults by weighing the errors they cause, and the faultiest message would be the correct message with every bit flipped. However, with Kyber using the LWE problem, this general attack must be modified.

5.1 Roulette

Roulette is a family of Fault attacks developed by Delvaux [Del22] based on the fault attack proposed by Hermelink et al. [HPP21] at Indocrypt 2021, where a system of inequalities over the secret key is generated and solved. The general methodology of Roulette works by considering a keyed cryptographic algorithm A : S × I → O, where S is the secret, I is the input, and O is the output, which may not necessarily be public, but the adversary will know whether it is correct or not. Algorithm A is shown in figure 2. Using a constant

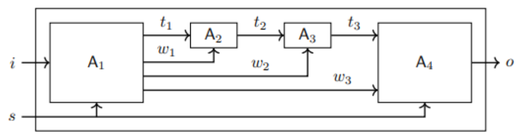

Figure 2: Algorithm A [Del22]

input, the adversary will repeatedly fault either $t_1, t_2, t_3, A_1, or A_2$ such that t_3 is not constant(does not follow a one-point distribution). If for all possible values of t_3, the probability that A is unable to return the correct output depends on the secret S, the adversary retrieves information on the secret. For every fault, one inequality over the secret key is obtained [Del22]. This attack should work on a variety of programs, including those that are protected by blinding or masking. However, for those that are masked or blinded, there will be different fault models to achieve the ideal distribution of t_3.

To instantiate an attack on Kyber, Roulette targets the decapsulation process [PP21]. The attack can be seen as an extension of the fault attack proposed by Hermelink et al. [HPP21] at Indocrypt 2021, which recovers the secret key s by faulting the re-encryption process. To demonstrate the effectiveness of Roulette, Delvaux showed the attack on a masked software on the ARM Cortex-M4 implementing Kyber from the *pqm4* library [Del22]. Although the implementation is unprotected, Delvaux focused primarily on the linear functions written in assembly, and the double GS butterfly in the last layer of the INTT. They found that for 5 out of the 9 instruction skips, the faulted output coefficient is uniformly distributed, the ideal case. It is also quite challenging to defend against Roulette as masking and blinding only require the adversary to modify the roulette attack, and other countermeasures merely slow down the attack with a significant cost. Furthermore, Delvaux developed their linear system solver, which reduces the computational time and increases the error tolerance. The code can be found on their GitHub: `https://github.com/Crypto-TII/roulette` [Del22].

Delvaux then experimented with software-simulated faults and found that using the filtering technique proposed by Pessl and Prokop [PP21], the number of inequalities needed to achieve 100% success probability is nearly halved [Del22]. Delvaux also tested the attack with corrupted inequalities and found that even with 50% of inequalities corrupted, the secret key can still be recovered entirely. The resulting overall error tolerance is found to be 25% [Del22], a huge step up from what was reported by Pessl and Prokop [PP21].

To show the applicability of Roulette, Delvaux set up an attack using Coron et al.'s [CGMZ22] implementation of Kyber implemented on an ARM Cortex-M4. Delvaux [Del22] used a ChipWhisperer board to generate and glitch a 24 MHz clock, which either causes instruction skips or corruptions. Using an optimized implementation for Kyber, an attack would be possible in hours, but Delvaux chose to use an unoptimized implementation, and it took around 5 days to finish. If guided by data from the experiment with corrupted inequalities, tolerating misclassifications of 50% of the decapsulation successes, Roulette requires only a maximum of 20 fault injections [Del22]. This shows that even with a cheap setup and a side-channel attack-protected target, Roulette can still deliver a solvable system of linear inequalities.

5.2 Error-tolerant key recovery

Similar to Delvaux's work [Del22], Hermelink et al.'s proposal [HMS+23] for a new algorithm to retrieve the secret key using decryption faults is based on the fault attack proposed at Indocrypt 2021 [HPP21]. However, unlike Roulette, Hermelink et al. [HMS+23] proposed an algorithm that focuses more on error tolerance rather than performance, which made Hermelink et al. focus more on the process after receiving information from the faults.

This approach suggests first using belief propagation (BP), which could recover the entire key given sufficient information, or at the least, partially recover the key. To retrieve the remaining data, the information obtained from BP will be embedded into a primal attack on the instance [HMS+23]. This is used to reduce the difficulty of the closest vector problem (CVP), which is commonly solved by embedding it into a unique smallest vector problem (uSVP) using Kannan's embedding[Kan04] and using a lattice reduction algorithm using algorithms such as BKZ. In short, the algorithm has 3 parts, the error-tolerant BP, the integration of fully recovered coefficients(bits of the secret key), and the integration of the leftover information [HMS+23].

To make the BP error tolerant, it is important to recognize the unreliability of inequalities. Unlike Roulette, which uses it in the Bayesian updating step [Del22], the proposed approach uses this information during the updating step in each check node of BP [HMS+23]. In previous works, at the end of BP, the first n coefficients are now considered correct and are used to solve for the remaining n coefficients. Here, the remaining information is used to construct a lattice problem as integrating the information into a lattice problem, as long as lattice reduction is not performed, is inexpensive. This also has the effect of setting up later steps for possibly non-convergent BP [HMS+23].

To integrate partially recovered keys to LWE, the dimensionality of the LWE is first reduced by using the recovered coefficient, and then the complexity of the resulting CVP is reduced by finding a closer target vector. Partially recovered BP gives probability distributions at variable nodes, but unfortunately, these distributions can not be assumed to be independent and the covariance matrix is unknown. So, this algorithm sees the distributions as rankings instead and judges the likelihood of a coefficient being correct using min-entropy. For increasing the number of consecutive correct coefficients, an estimate of β, the block size for the BKZ, will be acquired, and when β matches the available computational resources, a lattice reduction algorithm will be run. When implementing the recovered coefficients, r_s and r_e are treated differently and handled separately. If the LWE equation system is set up as such[4]: $sA^T + e = b$, it is easy to integrate recovered coefficients r_s. The adversary simply needs to intersect the subspace given by the recovered coefficients s_{j_k} with $k \in \{1, ..., r_s\}$. This can be written as

$$A' = AT_s$$

$$b' = b - t_s$$

T_s is the $m * n - r_s$ matrix obtained from removing each column j_k where $k \in \{1, ..., r_s - 1\}$, and

$$t_s = (\sum s_{j_k} * A_{i,j_k})_{i \in \{1,...,m\}}$$

Then, to integrate recovered coefficients of e, the equation is

$$A' = T_e(A - T'_e)$$

$$b' = b - t_e$$

where $T'_e = (\sum A^{-1}_{i,j_x} * A_{i,j})_{i,j}$ and $t_e = (\sum A^{-1}_{i,j_x}(e_i - b_i))_i$ [HMS+23].

After having integrated both types of recovered coefficients, the dimensionality of the problem is now reduced to $n - r_e$ remaining equations, $n - r$ remaining unknown values for s, and $n - r_e$ unknown values for e. As well, the probability distribution for the remaining bits of s and e have also been estimated. Here, Hermelink et al. [HMS+23] chose to perform a version of the primal attack shown in the attack at Indocrypt 2020 [DSDGR20], where they were able to obtain s.

To further improve the algorithm, Hermelink et al. [HMS+23] suggested using key enumeration to improve the number of recovered and integrated key coefficients, and to enumerate after the integration of recovered coefficients. It is also possible to use a dual or hybrid attack rather than a primal attack due to studies showing their proficiency at solving the LWE problem of Kyber.

Compared to Roulette, the number of correct inequalities required is nearly halved, showing the effectiveness of this algorithm error tolerance-wise. Hermelink et al. [HMS+23] also pointed out that although the performance may not outperform Roulette, the attack is still very fast and can be further accelerated using techniques noted before, ciphertext-filtering, as well as using physically more cores for BP.

6 Summary and conclusions

This paper has reviewed attacks against CRYSTALS-Kyber, the attacks include common attacks, side-channel attacks, side-channel assisted chosen-ciphertext attacks, and fault-injections.

[4] A^T means the transform of the matrix A

For common attacks, the paper reviewed attacks on module-LWE, the underlying problem that Kyber's security depends on, and showed how Kyber's security is possibly below standard when the adversary uses a dual attack. The paper then reviewed attacks exploiting decryption failure attacks using weak ciphertexts and weak keys, which can be extremely costly and can be therefore deemed inviable. Lastly in the common attack section, attacks on symmetric primitives and multi-target attacks were reviewed. Both of which Kyber has a sufficient defense against.

In the side-channel assisted chosen-ciphertext attacks section, EM attacks and CPA attacks were reviewed. Both attacks are viable but require careful setup. It is also important to note that defenses against these attacks are incredibly costly in performance.

In the side channel attacks section, timing attacks are first reviewed, which is inviable due to the constant-time nature of Kyber. After, SASCA attacks targeting NTT and KECCAK were reviewed, where both proved their ability to attack Kyber, but required specific conditions to be met. As well, defense against these attacks can be easily implemented with minimal drawbacks. Attacks targeting message encoding were then reviewed, where it proved to be very effective, even against masked Kyber implementations. At this point, some new approaches to side-channel attacks are discussed, such as using deep learning and the use of LDPC codes. Using deep learning, the adversary can attack higher-order masked implementations of Kyber, which is very useful for the adversary. As well, the effectiveness of using deep learning against Kyber has been proven to work well. LDPC code, a very new framework against post-quantum KEMs, has also proven itself to be very competent compared to other attacks, even with many possible paths to develop further.

With error injections, both Roulette and error-tolerant key recovery have proven their strengths in their respective fields, and both are very effective at recovering the secret key. Roulette can attack Kyber effectively even with a cheap setup and a well-protected target, showing its applicability in the real world. While Roulette focuses on efficiency and effectiveness, an error-tolerant key recovery attack focuses on making the overall attack error-tolerant, making it very effective with the number of correct inequalities needed to recover the key.

The attacks presented in the paper will likely be improved upon, with new attacks being proposed in the future as well. This speaks to the purpose of this paper, to give researchers and developers an idea of possible attacks against Kyber, and lattice-based programming as a whole, and to give incentive to the researchers and developers to find new ways to improve upon the defenses of Kyber, or perhaps defenses applicable to all cryptography schemes in the post-quantum era.

References

[ABD$^+$21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Algorithm specifications and supporting documentation, January 2021.

[ADH$^+$19] Martin Albrecht, Leo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn Postlethwaite, and Marc Stevens. *The General Sieve Kernel and New Records in Lattice Reduction*, pages 717–746. Springer International Publishing, 04 2019.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BGR$^+$21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking Kyber: First- and Higher-Order Implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):173–214, August 2021.

[CGMZ22] Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order Polynomial Comparison and Masking Lattice-based Encryption. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(1):153–192, November 2022.

[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*, page 3–33, Berlin, Heidelberg, 2022. Springer-Verlag.

[Del22] Jeroen Delvaux. Roulette: A Diverse Family of Feasible Fault Attacks on Masked Kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):637–660, August 2022.

[DGJ$^+$19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 565–598, Cham, 2019. Springer International Publishing.

[DNGW23] Elena Dubrova, Kalle Ngo, Joel Gärtner, and Ruize Wang. Breaking a fifth-order masked implementation of crystals-kyber by copy-paste. In *Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop*, APKC '23, page 10–20, New York, NY, USA, 2023. Association for Computing Machinery.

[DRV20] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. (One) Failure Is Not an Option: Bootstrapping the Search for Failures in Lattice-Based Encryption Schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 3–33, Cham, 2020. Springer International Publishing.

[DSDGR20] 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, page 329–358, Berlin, Heidelberg, 2020. Springer-Verlag.

[Duc18] Léo Ducas. Shortest Vector from Lattice Sieving: A Few Dimensions for Free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 125–145, Cham, 2018. Springer International Publishing.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on mdpc with cca security using decoding errors. In *Advances in Cryptology – ASIACRYPT 2016*, pages 789–815, 12 2016.

[GNNJ23] Qian Guo, Denis Nabokov, Alexander Nilsson, and Thomas Johansson. Sca-ldpc: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes. Cryptology ePrint Archive, Paper 2023/294, 2023. https://eprint.iacr.org/2023/294.

[HKL$^+$22] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. First-order masked kyber on arm cortex-m4. Cryptology ePrint Archive, Paper 2022/058, 2022. https://eprint.iacr.org/2022/058.

[HMS$^+$23] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Rodosek. Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 287–317, 08 2023.

[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*, pages 311–334, Cham, 2021. Springer International Publishing.

[HSST22] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. Adapting Belief Propagation to Counter Shuffling of NTTs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(1):60–88, November 2022.

[Kan04] Ravindran Kannan. Minkowski's convex body theorem and integer programming. 1 2004.

[KdG21] Alexandre Karlov and Natacha Linard de Guertechin. Power analysis attack on kyber. Cryptology ePrint Archive, Paper 2021/1311, 2021. https://eprint.iacr.org/2021/1311.

[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*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KPP20] Matthias Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 243–268, 06 2020.

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

[MBM$^+$22] Catinca Mujdei, Arthur Beckers, Jose Maria Bermudo Mera, Angshuman Karmakar, Lennert Wouters, and Ingrid Verbauwhede. Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication. Cryptology ePrint Archive, Paper 2022/474, 2022. https://eprint.iacr.org/2022/474.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):676–707, August 2021.

[Nia22] Mojtaba Bisheh Niasar. Lattices and kyber pqc presentation, 2022.

[NWDP22] Kalle Ngo, Ruize Wang, Elena Dubrova, and Nils Paulsrud. Side-channel attacks on lattice-based kems are not prevented by higher-order masking. Cryptology ePrint Archive, Paper 2022/919, 2022. https://eprint.iacr.org/2022/919.

[oS] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process.

[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*, pages 130–149, Cham, 2019. Springer International Publishing.

[PP21] Peter Pessl and Lukas Prokop. Fault Attacks on CCA-secure Lattice KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):37–60, February 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*, pages 513–533, Cham, 2017. Springer International Publishing.

[RCDB22] Prasanna Ravi, Anupam Chattopadhyay, Jan Pieter D'Anvers, and Anubhab Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. Cryptology ePrint Archive, Paper 2022/737, 2022. https://eprint.iacr.org/2022/737.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), sep 2009.

[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*, page 123–146, Berlin, Heidelberg, 2020. Springer-Verlag.

[RRD$^+$23] Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):418–446, March 2023.

[RSRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):307–335, June 2020.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[SKL+20] Bo-Yeon Sim, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Taeho Lee, Jaeseung Han, Hyojin Yoon, Jihoon Cho, and Dong-Guk Han. Single-trace attacks on the message encoding of lattice-based kems. Cryptology ePrint Archive, Paper 2020/992, 2020. https://eprint.iacr.org/2020/992.

[WX22] Han Wu and Guangwu Xu. Enhancing the dual attack against mlwe: Constructing more short vectors using its algebraic structure. Cryptology ePrint Archive, Paper 2022/1661, 2022. https://eprint.iacr.org/2022/1661.

[XPR+20] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. Cryptology ePrint Archive, Paper 2020/912, 2020. https://eprint.iacr.org/2020/912.