Side-channel Analysis of CRYSTALS-Kyber and A Novel Low-Cost Countermeasure

Meziane HAMOUDI[1], Amina BEL KORCHI[3], Sylvain GUILLEY[1,2], Sofiane TAKARABT[1], Khaled KARRAY[1], and Youssef SOUISSI[1]

[1] Secure-IC S.A.S Paris - France
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
[3] Secure-IC S.A.S., Cesson-Sévigné, France

Abstract. In this paper, we propose a vertical side-channel leakage detection on the decryption function of the third round implementation of CPA-secure public-key encryption scheme underlying CRYSTALS-Kyber, a lattice-based key encapsulation mechanism, which is a candidate to the NIST Post-Quantum Cryptography standardization project. Using a leakage assessment metric, we show that the side-channel information can be efficiently used to pinpoint operations leaking the secret variable and how masking countermeasures can be applied. We detect leakages in the polynomial multiplication between the secret key and the ciphertext. We propose and evaluate two different masking countermeasures, based on additive and multiplicative masking. To the best of our knowledge, the multiplicative masking has not been proposed before as a countermeasure to CRYSTALS-Kyber vulnerabilities. We demonstrate their efficiency and discuss their impact in terms of performance. Our work is beneficial to assess and enhance the security of Post-Quantum Cryptography against advanced vertical side-channel analysis.

Keywords: Post-quantum cryptography, lattice-based cryptography, CRYSTALS-Kyber, side-channel analysis, masking countermeasure, additive masking, multiplicative masking.

1 Introduction

Public-key cryptography made it possible to share a common secret-key between two entities communicating over a non-trusted channel. The most used public-key cryptographic algorithms are Rivest Shamir Adleman (RSA) [37] and Elliptic Curve Cryptography (ECC) [27]. The security of these two cryptosystems relies respectively on the hardness of the integer factorization for RSA and discrete logarithm problem for ECC. These two problems are supposed to be impossible to solve using classical computers.

However, quantum computers have recently been the subject of a substantial amount of research. They are known to have the potential to provide the power to brute force current public key encryption standards in a relatively short amount of time. These computers can break standard and complex cryptographic algorithms such as RSA and ECC, as reported in [38, 39]. For this purpose, National Institute of Standards and Technology (NIST) called for a proposal to standardize Post-quantum Cryptography (PQC) schemes. The main functions of a typical PQC scheme in the NIST evaluation process are Public Key Encryption (PKE), Key Encapsulation Mechanism (KEM) and digital signature. The NIST submissions rely on different hard problems: error correcting code, Learning With Errors (LWE), hash-based, multivariate and super-singular isogeny. The most important and commonly known families of PQC algorithms are LWE and those based on the hardness of decoding linear codes (also known as code-based algorithms).

PQC schemes based on lattice theory, mainly variant of LWE and Learning With Rounding (LWR) can be efficiently implemented on hardware and software. The best known and competitive algorithms in NIST competition are CRYSTALS-Dilithium for digital signature and CRYSTALS-Kyber for KEM.

In addition to the provided security against quantum computers, submitters must also take into consideration the protection against Side-Channel Attacks (SCA). Several works have been carried out to evaluate the security of PQC against physical attacks based on power acquisition or Electro-Magnetic emanation (EM), and fault injection. Some of them and other works, proposed also a set of countermeasures which can provide security against physical attacks. The NIST desires the PQC schemes to be resistant to side channel attacks at minimal cost.

Contributions. In this paper we evaluate the masking of the linear part of the current (reference implementation of the third round) submission of CRYSTALS-Kyber, especially the CPA-secure decryption function underlying this post-quantum cryptography scheme. After time constancy check using *timecop* tool [28], we proceed to a practical application of ISO/IEC 17825 [19]; Namely, we check the implementation is properly masked by applying a vertical leakage detection and evaluation leveraging the Normalized Inter-Class Variance (NICV) metric [8] on the secret key. This allows to detect any vulnerabilities related to the long-term key. We evaluate an additive masking countermeasure at machine-code level, discuss the results of the leakage detection, and the impact on the performance. As a second contribution, we propose an alternative countermeasure based on a multiplicative masking which is faster and induces less overhead. The overall execution time of the decryption procedure has an overhead of about 15% for the multiplicative masking versus 32% for the additive one. We discuss the provided security level of this proposition to protect the polynomial multiplication as well as its impact on the performance. A comparison between these two schemes is provided, where we explain in which respect our multiplicative masking scheme happens to be better than the additive one in some other implementations.

Outline. This paper is organized as follows: section 2 covers the related works on the lattice based post-quantum cryptography, side-channel attacks on this topic and the possible countermeasures. We summarize the recent analyses performed on lattice based implementation of the NIST competition previous round . Section 3 explains our analysis methodology, our attack scenario and the used leakage detection metric. Section 4 shows our experimental results on the reference implementation of CRYSTALS-Kyber, on the additive masking countermeasure, and on our proposition based on a multiplicative scheme. Finally, Section 5 concludes our paper.

2 Related Works & Background

2.1 Overview

Lattice-based cryptography [26] is a very promising PQC family. It offers a very strong security, and also a great simplicity, flexibility as well as an efficient implementation. Lattices are the most actively studied techniques and are used to construct key exchanges schemes, digital signature schemes, and fully homomorphic encryption schemes.

However, lattice-based cryptosystems have some disadvantages which restricts their usage in practical applications. One of them is the large size of the public key, the secret key, and ciphertexts. Thus, researchers introduce new algorithms based on LWE, which have the same hardness as the worst-case lattice problems.

CRYSTALS-Kyber [4] is a lattice-based KEM. It is one of the official finalist schemes of the NIST third-round competition. The security of this scheme is based on the difficulty of the Ring-LWE (R-LWE) problem. There are three variants of the scheme, namely Kyber512, Kyber768 and Kyber1024, which offer similar levels of security to AES-128, AES-192 and AES-256 respectively.

2.2 Notation

The algebraic structure used in CRYSTALS-Kyber scheme is the polynomial ring $R_q = \mathbb{Z}_q[X]/f(X)$, with $f(X) = X^n+1$. We note $R = \mathbb{Z}[X]/f(X)$ where $\mathbb{Z}[X]$ is the polynomial ring with coefficients in $\mathbb{Z}$ and $f(X)$ is a cyclotomic polynomial. Elements of R are polynomials of degree less than n and coefficients in $\mathbb{Z}$. Elements of $R_q = \mathbb{Z}_q[X]/f(X)$ are polynomials of degree less than n and coefficients modulo q, where $\mathbb{Z}_q[X]$ is the polynomial ring with coefficients modulo q. Elements of the ring R_q are noted in lowercase ($a \in R_q$), we denote by $[a]_q$ the elements in R obtained by computing all its coefficients modulo q, $a + b$ (resp. $a \cdot b$) is the addition (resp. multiplication) of two polynomials a and b in R_q. In the case where a and b are two vectors in R_q^l with elements a_i and b_i in R_q, the addition of a and b is $a + b$ is a vector of l elements $a_i + b_i$ in R_q and the canonical scalar product is used for the multiplication $a \cdot b$ which is a vector of l elements $a_i b_i$ in R_q. For $x \in \mathbb{R}$ we denote by $\lfloor x \rceil$ rounding to the nearest integer, $\lceil x \rceil$, and $\lfloor x \rfloor$ rounding up and down. In practice, $n = 256$ and $q = 3329$ is a prime number.

2.3 LWE/R-LWE Problems

In 2005, Regev [35] introduced the LWE problem, which consists in finding a secret in the middle of noisy linear equations. Regev has shown that solving the LWE problem in the average case by a quantum algorithm involves solving the SIVP [2] and gapSVP [10] problems in the worst case. R-LWE [24] is the polynomial ring version of LWE problem. CRYSTALS-Kyber is based on the decision version of the RLWE problem, which consists in distinguishing between RLWE samples and uniformly random ones. The problem is described in the mathematical ring formed by degree d polynomials over a finite field such as the integers modulo a prime number q. Let $\phi(x)$ be a cyclotomic polynomial of degree d, and $q \geq 2$ a modulus depending on a security level λ. For a random $s \in R_q$ and a distribution $\chi = \chi(x)$ over R, the problem consists in distinguishing (a, $[a \cdot e + s]_q$) from a random pair sampled uniformly from $R_q * R_q$, where a is a random element of R_q and e a noise term from χ.
For any cyclotomic ring R of degree n, modulus $q < 2^{poly(n)}$ and error distribution χ of error rate $0 < \alpha < 1$ where $\alpha q \geq 2\sqrt{n}$, solving the $RLWE_{q,\chi,m}$ problem is at least as hard as quantumly solving the SVP_γ problem on arbitrary ideal lattices in R, for some $\gamma = poly(n)/\alpha$.

2.4 Side-channel Attacks on Lattice-based Cryptography

SCA on lattice-based Post-quantum cryptography has been performed. Xu et al. propose in [41] an SCA with carefully constructed ciphertext on CRYSTALS-Kyber, and demonstrate that special chosen ciphertexts attack allows an adversary to modulate the leakage of a target device and enable full key extraction through Simple Power Analysis (SPA). Pess et al. in [30] introduced several improvements to the usage of belief propagation, which underlies the single trace attack and changed the target encryption instead of decryption which limited attacks to the recovery of the transmitted symmetric key, but in turn, increased attack performance. Ravi et al. demonstrated in [34] a generic and a practical SCA using a chosen ciphertext attack over multiple LWE/LWE-based PKE and KEM secure in the Chosen Ciphertext Attack (IND-CCA).

They showed that the side-channel information can be efficiently used to instantiate a plaintext checking oracle, which provides binary information about the output of decryption, typically concealed within IND-CCA secure PKE/KEM, thereby allowing such attacks. In [32], Ravi et al. reported an important exploitable vulnerability through side-channel attacks for message recovery in five lattice-based PKE and KEM implementations namely NewHope, Kyber, Saber, Round5 and LAC. The reported vulnerabilities exist in the message decoding function which is a fundamental kernel used in lattice-based PKE/KEM. Further analyses of the implementations in the public *pqm*4 library revealed that this function is implemented in a similar manner in all the identified schemes, and thus they all share the common side-channel vulnerability that leaks individual bits of the secret message. They demonstrate that the identified vulnerability can be exploited through a number of practical electromagnetic side-channel attacks, fault attacks and combined attacks on implementations from the *pqm*4 library running on ARM Cortex-M4 microcontroller.

Ravi et al. demonstrated in [33] practical fault attacks over a number of lattice-based schemes based on the hardness of the LWE problem. One of the common traits of all the considered LWE schemes is the use of nonce as domain separators to sample the secret components of the LWE instance, and showed that simple faults targeting the usage of nonce can result in a nonce-reuse scenario which allows for key recovery and message recovery attacks.

2.5 Countermeasures

To prevent SCA, some countermeasures can be adopted and applied to a given algorithm. In the following, we give the most important countermeasures, from the basic ones to prevent timing attacks [21], to the most advanced ones to prevent as well vertical and template attacks.

Constant time. When the execution time of an algorithm is constant whatever the inputs, it becomes impossible to mount any timing attack. It is sufficient to implement constant-time operations only for the sensitive ones. In fact, if the timing variation is independent from sensitive data, or does not involve the key, the attacker cannot learn anything about the secret. Besides, some countermeasures are based on delay insertion, jitter and fake operations to intentionally desynchronize the traces and make vertical attacks more difficult to achieve. Thus, the algorithm is vulnerable only if the timing variation depends on the secret key (or any sensitive data).

Vertical Side Channel Attacks – Masking. The masking countermeasure is used to make the power consumption independent from the processed data. Thus, it makes attacks such as Differential Power Analysis (DPA) and Template attack non-effective. Masking was used in order to construct side-channel resistant implementations for some lattice-based schemes such as CRYSTALS-Dilithium [14], qTESLA [15] and SABER [5]. When the intermediate data are randomized, the attacker cannot build a leakage model that correlates with the physical leakage. Particularly, the power consumption and EM emanations become independent from the secret data. The principle of this countermeasure is to split the sensitive variable into random shares in order to eliminate the computation dependency on the secret data. Usually, the sum of the shares is equal to the secret data. In that case, we talk about additive masking. Besides, another type of masking exists: multiplicative masking, where any value is expressed as a product of shares. Some masking techniques have already been presented in [29,36] for the R-LWE public-key encryption scheme. In parallel to our research, a completely masked implementation of CRYSTALS-Kyber using only additive and Boolean masking for the Chosen Plaintext Attack (IND-CPA) decrypt has been presented in [9]. The main difference of this paper with ours is that we propose a multiplicative masking on this purpose which induces less overhead on the performances.

Countermeasures against fault injection. Deterministic algorithms are more sensitive to fault injection [6, 40]. Many algorithmic countermeasures have been proposed based on redundancy. However, in most of the cases, the overhead is very important. Some of them checks only certain variables and thus, limits the complexity [7]. Hardware-based countermeasures like sensors can also be used to detect any abnormal change on the clock, voltage, or temperature. These countermeasures can detect global and even local fault injection like EM or Laser injection. In the case of signature based on public key encryption, all deterministic protocols are vulnerable to fault attacks [31]. Additional randomness to make signature non-deterministic is an effective way to counteract all proposed Differential Fault Attack (DFA). In the case of Crystal-Dilithium, the authors of [12] showed that additional randomness is proved to be the most effective countermeasure.

3 Analysis Methodology

3.1 Our Objective

Official reference and optimized implementation of CRYSTALS-Kyber are unprotected against SCA. In this section, we describe a test to pinpoint side channel leakages. It plays two roles on the sequel:

- It allows to identifying the lines of code which are leaking.
- In case no leakage is detected it allows to prove that implementation is not leaking information.

We first check the time constancy of the CYSTALS-Kyber implementation indeed SCA requires traces to be aligned. For this purpose we use time constancy checking tool *timecop*. Results showed that the NIST third round submission of CRYSTALS-Kyber is constant-time and does not contain timing leakages.

In our analysis scenario the key generation function of CRYSTALS-Kyber is only executed one time in order to generate the long-term key-pair, the attacker can only acquire one trace. Thus operations performed in the key generation function are vulnerable only to one trace based attacks, such as SPA and Template attacks. without being helped by a cryptanalysis attack, those attacks have a small chance of success then we do not cover them. Once the long-term key-pair is generated, the public key is used to encapsulate a (symmetric) secret key. In the other side, the secret key is used to decapsulate the ciphertext. An attacker that has access to the decryption step, can give different inputs and record EM or power activity. In addition to single trace based attacks, the decryption is also vulnerable to vertical attacks such as DPA [20] and Correlation Power Analysis (CPA) [11].

The most critical operation of the decryption step is the polynomial multiplication denoted by o which multiplies elements of the ciphertext and the secret key and accumulates the result obtained (line 4 of Alg. 1). This function allows to avoid timing leakages by using Montgomery Reduction. However is stills vulnerable to vertical side-channel attacks cited above as presented

in [17, 18]. In this paper, the goal is not to mount a CPA/DPA attacks, but rather to detect potential leakages that could be exploited by these attacks.

Algorithm 1: KYBER.CPAPKE.Dec(sk, c): Decryption

Input: Secret key sk
Input: Cipherext $c = (u, v) \in R_q$
Output: Message m
1 $u := Decompress_q(Decode_{d_u}(c), d_u)$
2 $v := Decompress_q(Decode_{d_v}(c + d_u.k.n/8), d_v)$
3 $\hat{s} := Decode_{12}(sk)$
4 $m := Encode_1(Compress_q(v - NTT^{-1}(\hat{s} \circ NTT(u)), 1))$
5 **return** m

The *Compress* function, when applied to a vector in R_q^k, takes each coefficient in $x \in \mathbb{Z}_q$ and outputs an integer in $0, \ldots, 2^d - 1$ where $d < log_2(q)$. Formally $Compress(x, d) = \lfloor \frac{2^d}{q} \times x \rceil \bmod 2^d$ and $Decompress(x, d) = \lfloor \frac{q}{2^d} \times x \rceil$. $Decode_l$ function is used to transform an array of $32l$ bytes into a polynomial $f = f_0 + f_1 X + \ldots + f_{255} X^{255}$ where $f_i \in 0, \ldots, 2^{l-1}$. The function $Encode_l$ is the inverse of $Decode_l$. The Number Theoretic Transform (NTT) [23] is an efficient way to perform multiplication of two polynomials in R_q. The complexity of a polynomial multiplication using NTT method is $\mathcal{O}(n \log(n))$ instead of $\mathcal{O}(n^2)$ in the case of naïve multiplication.

3.2 Leakage Detection Test

Leakage detection metric. The NICV [8] can be used to detect inherent leakage of a given parameter. The traces are classified with respect to the parameter value to compute the inter-class variance. When normalized by the total variance, only the samples where the parameter is manipulated will yield peaks.

The NICV is defined as follows:

$$NICV(Y, X) = \frac{\mathbb{V}\left[\ \mathbb{E}\ (Y \mid X\ )\ \right]}{\mathbb{V}[Y]} \tag{1}$$

where Y is the traces and X refers to the parameter used to classify Y. The NICV is a bounded quantity: $0 \leq NICV(Y, X) \leq 1$. When the NICV is small (resp. large), the implementation is secure (resp. insecure).

Definition 1. *An implementation is secure if NICV is lower than* 0.3 *on all samples of a traces.*

Discussion: *Indeed, as one will see on the NICV traces, there is some estimation error, which is empirically evaluated to* 0.1 *for* 500 *traces. Thus* 0.3 *is a conservative value: choosing less than* 0.3 *would have led to false alarms, whereas choosing a larger threshold would have led to undetected leaks.*

We performed NICV on the polynomial multiplication function, using the secret key $\hat{s}$ where $\hat{s} := Decode(sk)$, and identified the leaking operations (lines of code). The NICV on the ciphertext u, where $u := Decompress_q(Decode_{du}(c), d_u)$ showed common leaking operations. Thus, those operations are critical and sensitive, when an attacker can acquire many traces with random ciphertexts.

We proposed an additive masking scheme, and showed the efficiency of this countermeasure and its impact on the implementation performance. Finally, we implemented our proposed variant based on a multiplicative masking, which has less impacts in terms of performance.

4 Experimental Results

4.1 Flow

In order to generate traces of the polynomial multiplication we simulate the SCA activity of the target implementation based on the CPU register content. Indeed, while performing step by step execution, all CPU registers are observed and their content is extracted after each executed instruction. Finally the Hamming Distance (HD) is computed over the CPU register content acquired from $instruction_i$ and $instruction_{i+1}$. This simulates efficiently the SCA activity of the implementation which is highly correlated with the number of switching bits. As a result, the number of samples of each obtained trace equals the number of instructions.

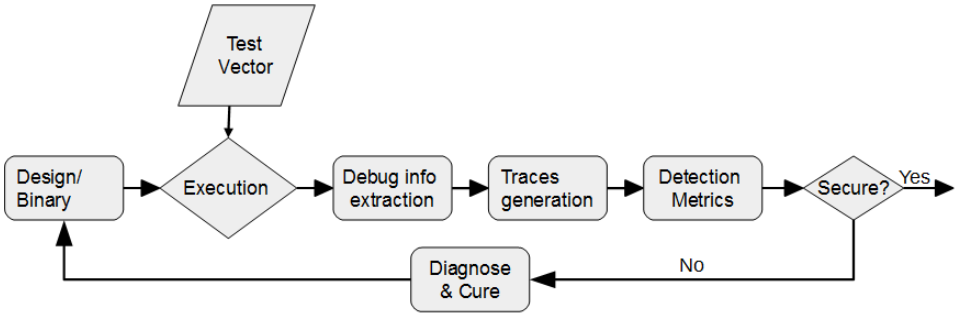

Fig. 1: Workflow of our analysis methodology.

The general workflow of our evaluation methodology from the design implementation to the trace simulation is illustrated in fig. 1:

- We use the binary as design to evaluate the implementation.
- The test vectors are the inputs to the binary and consist of:
  - 500 random ciphertexts to detect leakages depending on the ciphertext.
  - 500 random secret keys to detect leakages depending on the secret key.
- Executing the binary, observing and storing registers of the CPU.
- Simulating traces of EM from the register content using the HD model.
- Using NICV to check the security of the implementation. The implementation is secure if NICV is lower than 0.3.
- If the implementation is not secure, performing diagnose and cure phase in order to provide protection.

4.2 Analysis of CRYSTALS-Kyber – Reference Implementation

Using the reference implementation of the official submission of CRYSTALS-Kyber, we have generated 500 traces of the point-wise multiplication with fixed ciphertext and random secret key in order to locate all the leaking operations. Each trace makes 142761 samples corresponding to each instruction of that function. We then capture the NICV coefficients of the secret key $\hat{s}$ (obtained at line 3 of Alg. 1) which has $256 \times k$ coefficients where $k \in \{2, 3, 4\}$ fixes the lattice

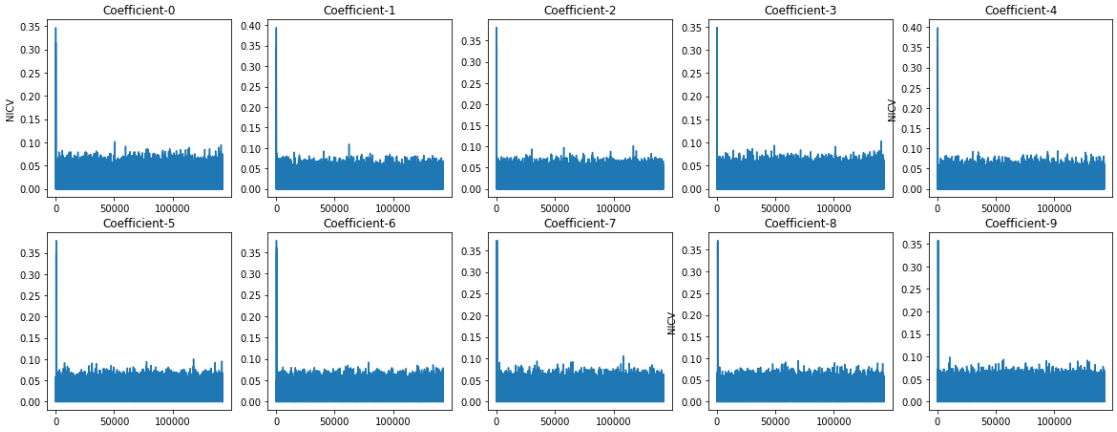

Fig. 2: Leakage peak in NICV of first coefficients of the secret key

dimension, and is the main mechanism in CRYSTALS-Kyber to scale security of different levels.

When targeting the first coefficients of $\hat{s}$, the NICV peak is located at the beginning of the trace (fig. 2). The peak is offset with respect to the targeted coordinate index. This represents exactly the way how the decryption function is implemented: the leakage corresponds to the pointwise multiplication "$\hat{s} \circ NTT(u)$" at line line 4 of Alg. 1. When targeting the last ones (fig. 3),

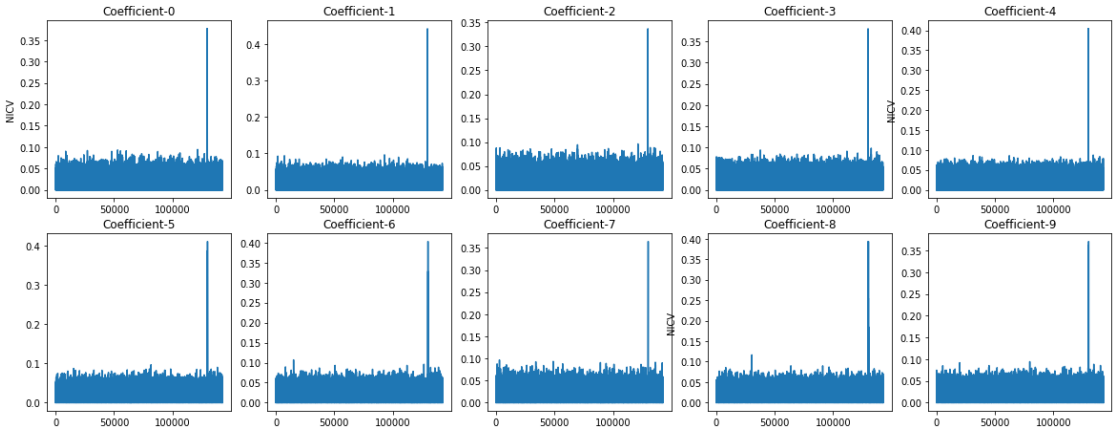

Fig. 3: Leakage peak in NICV of last ten coefficients of the secret key

the NICV peaks are located at the end of the trace. It is easy to see that the peaks are located at equally spaced positions, which is consistent with the $\hat{s}$ coordinates being consumed one after the other. The same results are observed using the ciphertext thus, both parameters are leaking at same samples. The tool used for the analysis allows to map samples to the corresponding lines of code. By applying a threshold of 0.3 on the results of the NICV, we obtain the elementary operations and functions that are leaking. The leaking function are:

– *basemul* called by *poly_basemul_montgomery*, multiplication of two polynomials in NTT domain. *basemul* is a multiplication of two polynomials in $\mathbb{Z}q[x]/(x^2 - \zeta)$, where ζ is $256 - th$ root of unity modulo q, used to multiply two elements in R_q.
– *fqmul* (called by *basemul*) which is a multiplication followed by a *Montgomery reduction*.
– *montgomery_reduction* called by *fqmul* bit integer $a \times R^{-1} \bmod q$, where $R = 2^{16}$ (Beware that R stands here for the Montgomery constant, and it not the lattice ring introduced in section 2.2).

Those functions can be found at the reference source code of CRYSTALS-Kyber implementation [1].

The value of the NICV peaks are upper than 0.3 thus the implementation is not secure we therefore go to diagnose and cure step. To eliminate those vulnerabilities, we implement and evaluate the masking countermeasure. This will be detailed in the next sub-sections.

4.3 Analysis of Masked CRYSTALS-Kyber Implementation – Additive Masking

Additive masking consists of randomly splitting secret data into several shares. This solution will be used to protect sensitive data in the algorithms of the key generation and the decryption of CRYSTALS-Kyber

Key Generation. At the key generation step, the secret key $sk = NTT(s)$ should not be stored in clear. A random sk_1 is picked uniformly in $\beta^{24 \cdot k \cdot n/8+96}$, where β is the set of 8-bit unsigned integers (we denote by β^k the set of byte arrays of length k). Then, the second member sk_0 is computed as

$$sk_0 = sk - sk_1.$$

The secret key is then stored after the key generation as (sk_0, sk_1). Alg. 2 describes the additive masking for key generation.

Algorithm 2: Key generation suitable for additive masking

Input: (sk)
Output: (sk_0, sk_1)
1 Pick a random sk_1 uniformly $sk_1 \in \beta^{24 \cdot k \cdot n/8+96}$
2 $sk_0 := sk - sk_1$
3 **return** (sk_0, sk_1)

Decryption. In the case of decryption, the computation of $\hat{s}^T \circ NTT(u)$ is sensitive as already detected with our methodology in section 4.2. Since the multiplication is performed in NTT domain, the computation of $\hat{s}^T \circ NTT(u)$ is equivalent to the polynomial product $s^T \cdot u$. Remember that $\hat{s} = NTT(s)$, the secret key $sk = sk_0 + sk_1$ and the first element u of the ciphertext $c = (u, v)$ is in $\beta^{du \cdot k \cdot n/8}$. With the additive masking, the polynomial multiplication is performed as:

$$\hat{s}^T \circ NTT(u) = \hat{s_0}^T \circ NTT(u) + \hat{s_1}^T \circ NTT(u).$$

This property leverages the linearity of the NTT.

The total overhead of the additive masking is k point-wise multiplications in R_q, and k additions in R_q. Alg. 3 describes the additive masking for decryption step.

Algorithm 3: Decryption – protected by additive masking

Input: $u, (sk_0, sk_1)$
Output: $\hat{s}^T \circ NTT(u)$
1 $\hat{s_0} := Decode(sk_0)$
2 Compute $\hat{s_0}^T \circ NTT(u)$
3 $\hat{s_1} := Decode(sk_1)$
4 Compute $\hat{s_1}^T \circ NTT(u)$
5 $\hat{s}^T \circ NTT(u) := \hat{s_0}^T \circ NTT(u) + \hat{s_1}^T \circ NTT(u)$
6 **return** $\hat{s}^T \circ NTT(u)$

Each trace of the polynomial multiplication using the additive masking algorithm is made up of 285527 samples which is almost two times the size of the unprotected one (142761), indeed this operation is performed twice (line 5 of Alg. 3). This gives us a first idea about how this protection impacts the implementation overall performance this will be discussed in section 4.5.

The NICV on CRYSTALS-Kyber using additive masking shows the efficiency of this countermeasure. The results show that the secret key is no longer leaking in the polynomial multiplication. Figure 4 shows the NICV on the 10 first coefficients of the secret key and there is no

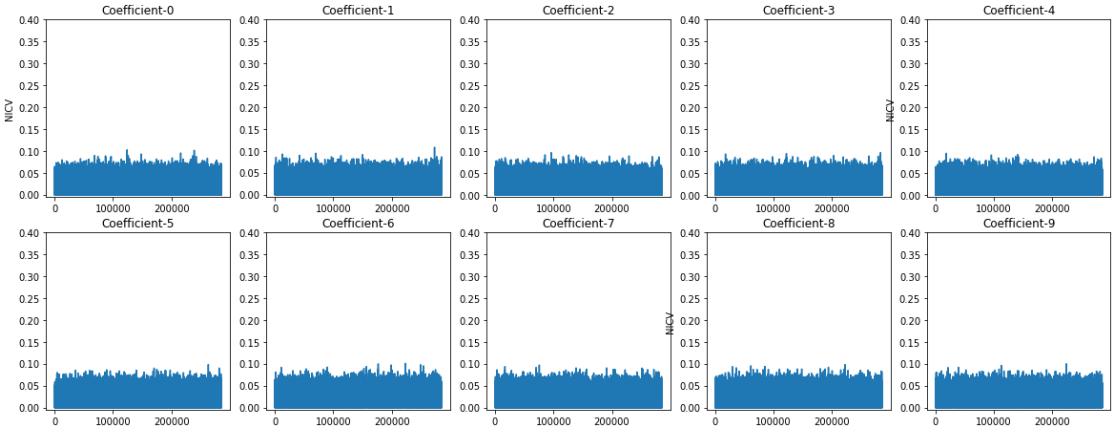

Fig. 4: NICV on the secret key after additive masking countermeasure

significant peak (all peaks are lower than 0.3). Same results are observed for all the coefficient of the unpacked secret key. This should prevent an attacker from building a vertical SCA, such as DPA, to recover the secret key.

Despite that the additive masking offers resistance against DPA, the overall execution time of the implementation is hampered. In section 4.4, we propose an alternative masking countermeasures, providing the same resistance against vertical SCA with less impact on the overall performance of the decryption.

4.4 Analysis of Masked CRYSTALS-Kyber Implementation – Multiplicative Masking

The idea of the multiplicative masking is to write sk as $r^{-1} \times (r \times sk)$. r can be chosen as a random number in $\mathbb{Z}_q^*$. In our case, we suggest to use a random integer r. As q is a prime number, r is invertible in $\mathbb{Z}_q$.

Notice that multiplicative masking employs a non-uniform random (since it cannot be null), hence is subject to first-order attacks. For instance, the multiplicative masking on AES by Akkar and Giraud [3] has been attacked by Golić and Tymen [16], though some repairs are possible (in hardware [25]). However, multiplicative masking on asymmetric algorithms is still common practice, as operands are largers than 8-bits (case of AES). Hence the first-order leakage decreases. For instance, multiplicative masking is suggested by Kocher on RSA [22] (also known as base blinding), then later on by Coron on ECC [13] (also known as randomized projective coordinates). We leverage this countermeasure for lattice-based cryptography.

Key Generation. At the key generation step, the secret key $sk = NTT(s)$ is not returned in this way. The secret key is *randomized* as described in algorithm Alg. 4. A random r is picked uniformly in $\mathbb{Z}_q^*$. The member $rsk \in \beta^{24 \cdot k \cdot n/8+96}$ is computed as:

$$rsk = r \times sk.$$

The member r_inv is computed as $r_inv = r^{-1} \bmod q$. This inversion can be performed with an algorithm like the extended Euclidean algorithm, or using a pre-computed table, that stores the inverse of each element in $\mathbb{Z}_q$. The secret key is then stored after the key generation as (r_inv, rsk).

Algorithm 4: Key generation suitable for multiplicative masking

Input: sk
Output: (r_inv, rsk)
1 Pick a random r in $\mathbb{Z}_q^*$
2 $rsk := r \times sk$
3 Compute r_inv as $r^{-1} \bmod q$
4 **return** (r_inv, rsk)

Decryption. The computation of $\hat{s}^T \circ NTT(u)$ will be performed as:

$$\hat{s}^T \circ NTT(u) = \hat{rs}^T \circ (r_inv \times NTT(u))$$

owing to the invariance of the NTT by scaling by 1 (recall $1 = r_inv \times r$).

In the case of the multiplicative masking, the total overhead is n multiplications in $\mathbb{Z}_q$. Alg. 5 describes the multiplicative masking for the decryption step. The overhead incurred over unprotected version is negligible compared to the overhead in the additive masked decryption.

Algorithm 5: Decryption – protected by multiplicative masking

Input: $u, (r_inv, rsk)$
Output: $\hat{s}^T \circ NTT(u)$
1 $\hat{rs} := Decode(rsk)$
2 Compute $r_inv \times NTT(u)$
3 $\hat{s}^T \circ NTT(u) := \hat{rs}^T \circ (r_inv \times NTT(u))$
4 **return** $\hat{s}^T \circ NTT(u)$

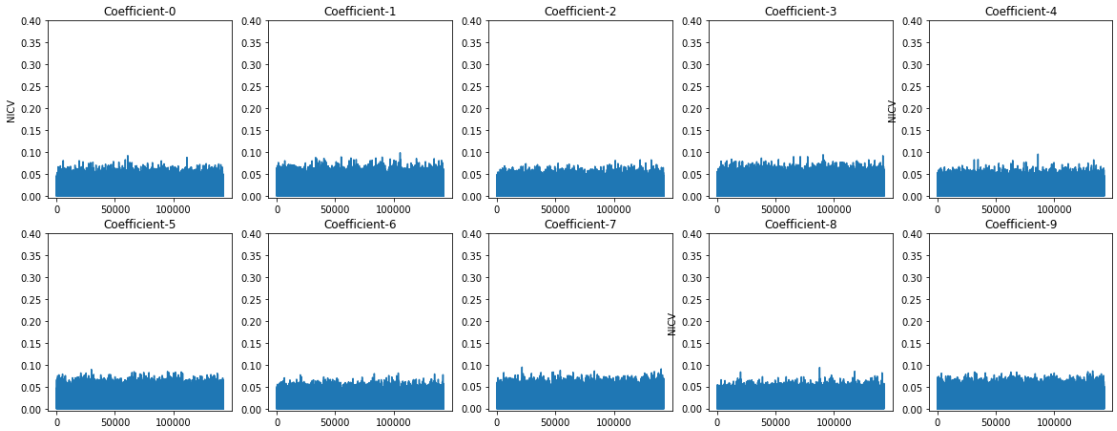

Fig. 5: NICV on the secret key after multiplicative masking countermeasure

The results of NICV on the 10 first coefficients of the secret key are shown in fig. 5. As for the case of the additive masking scheme, no significant peak is detected. Furthermore, this masking countermeasure does not impact the overall cost of CRYSTALS-Kyber implementation because only one polynomial multiplication is performed.

4.5 Discussion

In this work, we have tested the different analysed implementations on a host machine equipped with an Intel core (i7-6700K CPU 4.00 GHz). We have measured the average and the median execution time to see the impact of each countermeasure in terms of performance.

Table 1: Performance benchmarks for different implementation of CRYSTALS-Kyber-512 decryption – Intel (R) Core (TM) i7-6700K CPU 4.00 GHz.

Implementation	#Cycles		Overhead (%)
	Average	Median	
Unprotected (Ref.)	210694	197150	–
Protected (Additive)	278772	260292	32 %
Protected (Multiplicative)	243151	234521	15 %

We reported the performance of each implementation in table 1. The results show that the overall execution time of the decryption procedure has an overhead of about 32% and 15 % for the additive and the multiplicative masking respectively. We notice that the key generation outputs the secret key on NTT domain. Thus, in the decryption stage, we do not need to convert the secret key shares. In the additive masking implementation, only the ciphertext (the variable u) is converted on the NTT domain. This explains why the overhead is not doubled, as we know that the most consuming part is the polynomial multiplication. Only a point-wise multiplication is performed in this case, which is not a full polynomial multiplication for instance. If the secret key is not outputted on NTT domain, the overhead will be more significant in the case of additive

masking, but not for the multiplicative variant, where no addition point-wise (or polynomial) multiplication is needed. This may be more interesting for other variants of LWE algorithms where the usage of NTT is not possible (for example when the modulus in not a prime).

5 Conclusion

In this paper, we identified vertical side-channel vulnerabilities in the current version of the lattice-based PQC algorithm CRYSTALS-Kyber which is a KEM finalist candidate of the NIST standardization project. We experimentally demonstrated with the leakage detection metric NICV, the presence of leakages depending on the secret key in the NTT domain polynomial multiplication, used in the IND-CPA decryption function. We then study additive masking proposed previously in the literature and propose to the best of our knowledge for the first time, a multiplicative masking solution adapted to CRYSTALS-Kyber. We show the efficiency of each masking scheme against vertical attacks, namely by computing and showing that the NICV does not feature any significant peak that may be interpreted as a leakage based on the same number of traces as in the first experiment. Finally, we compare the overhead of each proposed countermeasure (compared to the vanilla reference code available from NIST submission website) and we show that the multiplicative masking performs better for the same security level.

References

1. Implementation of CRYSTALS-Kyber. `https://github.com/pq-crystals/kyber`.
2. Divesh Aggarwal and Eldon Chung. A note on the concrete hardness of the shortest independent vectors problem in lattices. *arXiv preprint arXiv:2005.11654*, 2020.
3. Mehdi-Laurent Akkar and Christophe Giraud. An Implementation of DES and AES Secure against Some Attacks. In LNCS, editor, *Proceedings of CHES'01*, volume 2162 of *LNCS*, pages 309–318. Springer, May 2001. Paris, France.
4. 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. *NIST PQC Round*, 2:4, 2019.
5. Michiel Van Beirendonck, Jan-Pieter D'Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel resistant implementation of saber. Cryptology ePrint Archive, Report 2020/733, 2020. `https://eprint.iacr.org/2020/733`.
6. Noemie Beringuier-Boher, Marc Lacruche, David El-Baze, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Philippe Maurine. Body Biasing Injection Attacks in Practice. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC, Prague, Czech Republic, January 20, 2016*, pages 49–54, 2016.
7. Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE transactions on Computers*, 52(4):492–505, 2003.
8. Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. NICV: normalized inter-class variance for detection of side-channel leakage. In *2014 International Symposium on Electromagnetic Compatibility, Tokyo*, pages 310–313. IEEE, 2014.
9. Joppe W Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First-and higher-order implementations. *IACR Cryptol. ePrint Arch.*, 2021:483, 2021.
10. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
11. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.

12. Leon Groot Bruinderink and Peter Pessl. Differential fault attacks on deterministic lattice signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 21–43, 2018.
13. Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *LNCS*, pages 292–302. Springer, 1999.
14. Direction Générale de l'Armement. Masking dilithium: Efficient implementation and side-channel evaluation.
15. François Gérard and Mélissa Rossi. An efficient and provable masked implementation of qtesla. In *International Conference on Smart Card Research and Advanced Applications*, pages 74–91. Springer, 2019.
16. Jovan Dj. Golić and Christophe Tymen. Multiplicative masking and power analysis of AES. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.
17. Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Correlation power analysis on ntru prime and related countermeasures. *IACR Cryptol. ePrint Arch.*, page 100, 2019.
18. Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Power analysis on ntru prime. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 123–151, 2020.
19. ISO/IEC JTC 1/SC 27/WG 3. ISO/IEC 17825:2016: Information technology – Security techniques – Testing methods for the mitigation of non-invasive attack classes against cryptographic modules. `https://www.iso.org/standard/60612.html`.
20. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.
21. Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.
22. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *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.
23. Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. Cryptology ePrint Archive, Report 2016/504, 2016. `https://eprint.iacr.org/2016/504`.
24. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
25. Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative Masking for AES in Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):431–468, 2018.
26. Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.
27. Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.
28. Moritz Neikes. Timecop: Automated dynamic analysis for timing side-channels.
29. Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical cca2-secure and masked ring-lwe implementation. Cryptology ePrint Archive, Report 2016/1109, 2016. `https://eprint.iacr.org/2016/1109`.
30. Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In *International Conference on Cryptology and Information Security in Latin America*, pages 130–149. Springer, 2019.
31. Damian Poddebniak, Juraj Somorovsky, Sebastian Schinzel, Manfred Lochter, and Paul Rösler. Attacking deterministic signature schemes using fault attacks. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 338–352. IEEE, 2018.
32. Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. Drop by Drop you break the rock-Exploiting generic vulnerabilities in Lattice-based PKE/KEMs using EM-based Physical Attacks. *IACR Cryptol. ePrint Arch.*, 2020:549, 2020.

33. Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Number "not used" once-practical fault attack on pqm4 implementations of nist candidates. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 232–250. Springer, 2019.
34. 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*, pages 307–335, 2020.
35. Oded Regev. The learning with errors problem (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 191–204. IEEE Computer Society, 2010.
36. Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. Cryptology ePrint Archive, Report 2015/724, 2015. `https://eprint.iacr.org/2015/724`.
37. Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
38. Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
39. Shor, Peter W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. volume 41, pages 303–332, 1999.
40. Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential fault analysis of the advanced encryption standard using a single fault. In *IFIP international workshop on information security theory and practices*, pages 224–233. Springer, 2011.
41. Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David F Oswald. Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber. *IACR Cryptol. ePrint Arch.*, 2020:912, 2020.