Correlation Electromagnetic Analysis on an FPGA Implementation of CRYSTALS-Kyber

Rafael Carrera Rodriguez[1 [0000−0003−4779−6635]], Florent Bruguier[1 [0000−0002−7897−5700]], Emanuele Valea[2 [0000−0001−9804−7250]], and Pascal Benoit[1 [0000−0002−2945−5725]]

[1] LIRMM, University of Montpellier, CNRS, Montpellier, France
firstname.last-names@lirmm.fr
[2] Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
emanuele.valea@cea.fr

Abstract. Post-quantum cryptography represents a category of cryptosystems resistant to quantum algorithms. Recently, NIST launched a process to standardize one or more of such algorithms in the key encapsulation mechanism and signature categories. Such schemes are under the scrutiny of their mathematical security, but they are not side-channel secure at the algorithm level. That is why their side-channel vulnerabilities must be assessed by the research community. In this paper, we present a non-profiled correlation electromagnetic analysis against an FPGA implementation of the chosen NIST key-encapsulation mechanism standard, CRYSTALS-Kyber. The attack correlates an electromagnetic radiation model of the polynomial multiplication execution with the captured traces. With 166,620 traces, this attack correctly recovers 100% of the subkeys. Furthermore, a countermeasure is presented for securing the target implementation against the presented attack.

Keywords: side-channel analysis · correlation power analysis · lattice-based cryptography · FPGA

1 Introduction

The advent of quantum computing represents a change of paradigm from classical algorithms. Its properties like superposition and entanglement allow for quantum algorithms, like Shor's algorithm [22], to speed up certain problems that are difficult for classic computing. In particular, this algorithm allows to break certain number theory problems in subexponential time, such as integer factorization and discrete logarithm. Many widely used asymmetric cryptosystems today, such as RSA or ECC, rely on these problems. Therefore, quantum computing poses a future threat for these systems.

This is why the National Institute of Standards and Technologies (NIST) launched an effort to choose the next cryptosystems that are resistant to quantum computing, called the Post-Quantum Cryptography (PQC) Standardization

Process [14]. The scheme chosen as the standard in the key-encapsulation mechanism (KEM) category is CRYSTALS-Kyber [2], having a mathematical primitive using lattice-based cryptography (LBC).

At first, researchers put their focus on securing the algorithms against mathematical attacks. However, the algorithms are implemented in a physical device and quantities such as power consumption, timing and electromagnetic radiation can be used to perform side-channel analyses (SCA) and obtain sensitive information, namely the plaintext or the secret key. Thus, there is an interest from the scientific community to protect the proposed PQC algorithms against such attacks [3].

Related works. Since the opening of the NIST PQC standardization effort, many SCA against LBC schemes have been presented in the literature. However, most of them are either profiling attacks [9,18,17] or attacks against the Chosen-Ciphertext-Attack (CCA) security [5,15,19,20,23], by instantiating a plaintext-checking oracle or other side-channel assisted oracles. Other works, such as [1,25], perform simple power analysis to partially or totally retrieve the secret key of the KEM. Some of these works assume an attacker model with increased capabilities, such as being able to communicate directly with the attacked server. The works from [5,9,15,19,20,23,25] create invalid ciphertexts, either from small modifications of valid ciphertexts or from completely crafted ciphertexts, in order to induce to decryption errors or to obtain decrypted plaintexts that depend directly on parts of the secret key that will have an effect on the power consumption/electromagnetic radiation.

In [11], the authors perform an unprofiled correlation power analysis (CPA) on an ARM Cortex-M4 implementation of CRYSTALS-Kyber, targeting the polynomial multiplication. In [10], the authors identify the vulnerability of this same operation within the reference implementation of CRYSTALS-Kyber, using a leakage assessment metric called the normalized inter-class variance (NICV) [4]. Recently, the authors of [13], extended this analysis of the polynomial multiplication to all the lattice-based KEMs finalists, targeting their implementations in the `pqm4` library for ARM Cortex-M4. However, all of these attacks and analyses target a software implementation.

Contribution. To the best of the authors knowledge, there are no unprofiled attacks against hardware implementations of CRYSTALS-Kyber in the literature. In this work, such attack is presented against one of the literature reconfigurable hardware (FPGA) implementations [24], using a power model that leverages the attacker's knowledge of the device's algorithm. A cost-effective countermeasure, specific to this implementation and to this attack is also proposed.

Outline. This paper is organized as follows: Section 2 presents the background necessary for this paper, including notation, a brief description of CRYSTALS-Kyber and a presentation of correlation power/electromagnetic analysis. Section

3 presents the outline of the proposed attack. Section 4 exhibits the setup and practical results of the attack. In Section 5, a simple countermeasure is presented. Finally, in Section 6, conclusions are drawn.

2 Background

2.1 Notation

The ring of integers modulo q is denoted as $\mathbb{Z}_q$. The polynomial ring defined over the previously mentioned ring, $\mathbb{Z}_q(x)/\phi(x)$, is represented as R_q, where $\phi(x)$ is the reduction polynomial. When a polynomial f is defined as an element of such ring, the i-th coefficient of such polynomial is denoted as f_i. The representation of such polynomial in the number theoretic transform domain is denoted as $\hat{f}$.

A vector is denoted with bold letters, *e.g.*, $\boldsymbol{a}$. When referring to a matrix, uppercase letters will be used, *e.g.*, $\boldsymbol{A}$. When indexing elements of vectors or matrices, brackets will be used, *e.g.*, $\boldsymbol{A}[i, j]$. When referring to all rows (resp. columns) of a matrix, a point will be used when indexing, *e.g.*, $\boldsymbol{A}[\cdot, j]$ (resp. $\boldsymbol{A}[j, \cdot]$).

2.2 CRYSTALS-Kyber

CRYSTALS-Kyber [2] is a key encapsulation mechanism (KEM) composed of two basic parts:

- An indistinguishable under chosen-plaintext attack (IND-CPA) public key encryption scheme, with its security relying on the hardness of breaking the Module Learning with Errors problem (M-LWE)[12,21].
- A Fujisaki-Okamoto transform [8], in order to convert it to an indistinguishable under chosen-ciphertext attack (IND-CCA) KEM, by re-encrypting in decapsulation and checking if the output is equal to the sent ciphertext.

The last version of CRYSTALS-Kyber uses polynomials in the ring R_q, where $q = 3329$ and the reduction polynomial is a cyclotomic polynomial $\phi(x) = x^{256} + 1$. The flexible parameter is k, which is the rank of the module lattice in this cryptosystem and it dictates the size of vectors and matrices, as well as the value of other subparameters.

In CRYSTALS-Kyber, in order to reduce complexity for the polynomial multiplications, the authors choose to use the Number Theoretic Transform (NTT), which works like a Fast-Fourier Transform over the integers ring. However, because of the modulus q chosen in CRYSTALS-Kyber, the field $\mathbb{Z}_q$ contains n-th primitive roots of unity, but not $2n$-th primitive ones. This means that the defining polynomial of CRYSTALS-Kyber cannot be factorized in 256 polynomials of degree 1, but in 128 polynomials of degree 2. Therefore, the NTT of a polynomial $f \in R_q$ is a vector of 128 polynomials of degree 1. Then, according to the specification, for $\zeta = 17$, as the first 256-th primitive root of unity and $\mathrm{br}_7(\cdot)$ as the 7-bit-reversion operation, NTT is defined as follows:

$$\hat{f}_{2i} = \sum_{j=0}^{127} f_{2j} \zeta^{(2\mathrm{br}_7(i)+1)j}$$
$$\hat{f}_{2i+1} = \sum_{j=0}^{127} f_{2j+1} \zeta^{(2\mathrm{br}_7(i)+1)j}$$

that defines the polynomial:

$$\mathrm{NTT}(f) = \hat{f} = \hat{f}_0 + \hat{f}_1 x + \cdots + \hat{f}_{255} x^{255}$$

As mentioned before, the actual NTT of $f \in R_q$ is a vector of polynomials. However, the authors of the cryptosystem chose to represent the result of NTT as a polynomial in R_q, even though it does not have that algebraic meaning.

NTT in Kyber can be splitted, separating the polynomial f with degree 256 and transform it into two polynomials of degree 128, where the coefficients of the first polynomial are the coefficients with even index of polynomial f, and the coefficients of the second polynomial are the coefficients with odd index of f. Then, NTT is executed for both polynomials obtaining two vectors: $(\hat{f}_0, \hat{f}_2, \hat{f}_4, \ldots, \hat{f}_{254})$ and $(\hat{f}_1, \hat{f}_3, \hat{f}_5, \ldots, \hat{f}_{255})$. Finally, the coefficients are reorganized to obtain the resulting polynomial $\hat{f}$.

Multiplication of two elements $f, g \in R_q$ is a polynomial multiplication. This operation, transformed in the NTT domain, becomes a point-wise multiplication (PWM), which, for this specific way of executing NTT, is defined as follows:

$$\hat{f} \circ \hat{g} = \hat{h} = \hat{h}_0 + \hat{h}_1 X + \cdots + \hat{h}_{255} X^{255}$$

where:

$$\hat{h}_{2i} = \hat{f}_{2i}\hat{g}_{2i} + \hat{f}_{2i+1}\hat{g}_{2i+1} \cdot \zeta^{2br_7(i)+1}$$
$$\hat{h}_{2i+1} = \hat{f}_{2i}\hat{g}_{2i+1} + \hat{f}_{2i+1}\hat{g}_{2i} \quad (1)$$

The previously described operation is executed in the CRYSTALS-Kyber scheme at the decapsulation/decryption routine with the secret key and part of the ciphertext as inputs. Therefore, this is a critical operation that can be the object of side-channel attacks, as explained in Sections 2.3 and 3.

2.3 Correlation Power/Electromagnetic Analysis

A correlation power analysis (CPA) [7] and its analog, correlation electromagnetic analysis (CEMA), is a divide-and-conquer type of attack that targets an operation with a known value and a part of the secret key, using a leakage model. The steps of such an analysis are the following:

1. The attacker retrieves n traces $\boldsymbol{t}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{n-1}$ of m sample points corresponding to the execution of a targeted operation, with $i_0, i_1, \ldots, i_{n-1}$ known inputs. For simplicity, the set of traces will be referred as the matrix $\boldsymbol{T}$ of size $n \times m$, where each row is the trace of an execution.

2. The attacker targets an unknown subkey s_0 and makes a guess g_0 from the set of l possible values of s_0.
3. According to his knowledge of the device, the attacker uses a power consumption model denoted as $M(i_j, g_0)$, depending on a certain known input i_j, for each point k of the trace $\boldsymbol{T}[j, \cdot]$. Then, the attacker will calculate the vector $\boldsymbol{h}_0$ of n points, where $\boldsymbol{h}_0 = M(i_j, g_o)$ for each input given to the device. If the consumption is assumed to be dependent on the set bits of the result of a targeted operation $z = F(i_j, s_0)$, then the used model will be the Hamming weight, where the consumption is modeled as $a \cdot HW(z) + B$, where a is a scaling factor and B is random noise. On the other hand, if the consumption is assumed to be dependent on the switched bits from a reference state r to the result $z = F(i_j, s_0)$, then the used model is the Hamming distance, modeled as $a \cdot HW(z \oplus r) + B$, being $\oplus$ a bit-wise XOR. In practice, the Hamming weight model is more suitable for software implementations, whereas the Hamming distance model is more adapted to hardware implementations, with known switching activity [16].
4. The attacker uses the Pearson's correlation coefficient

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

   to obtain a vector $\boldsymbol{c}_0$ of size m, where each point is obtained as:

$$\rho(\boldsymbol{T}[\cdot, j]^T, \boldsymbol{h}_0)$$

   and $\boldsymbol{T}[\cdot, j]$ is equal to the column vector $(\boldsymbol{T}[0, j], \boldsymbol{T}[1, j], \ldots, \boldsymbol{T}[n-1, j])$. That is, the vector $\boldsymbol{c}_0$ contains the correlation coefficient of each of the sample points with the model vector $\boldsymbol{h}_0$.
5. The attacker repeats steps 2-4 for guesses $g_1, g_2, \ldots, g_{l-1}$, obtaining the vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{l-1}$.
6. The attacker uses a strategy to choose the most likely guess. For example, the attacker can choose the guess g_j that yields the vector $\boldsymbol{c}_j$ that contains the point with the maximum absolute value.
7. The attacker repeats steps 2-5 for the rest of unknown subkeys until the whole key is retrieved.

3 Attack outline

3.1 Attack scenario and target

This attack targets the server part of the compact FPGA implementation of the third round NIST submission of CRYSTALS-Kyber [2], proposed by Xing and Li [24]. It was designed to fit the smallest of the Xilinx Artix-7 devices, with a resource utilization of 7412 LUTs, 4644 flip-flops, 2126 slices, 2 DSPs and 3 BRAMs. Apart from the Keccak core for hash and extendable output functions, most of the operations are done in the NTT core. This core contains two parallel

and pipelined butterfly units used for NTT and inverse NTT, as well as for other operations, such as PWM, decoding, encoding, compressing and decompressing.

Apart from the attacked server device, the other two participants of this setting are a client and an adversary. The server owns a set of generated keys for executing CRYSTALS-Kyber: a private one and a public one. This keys are not regenerated. When the client wants to establish a shared key with the server for a secure communication, they do it using the encapsulation function with the public key of the server, as well as with the ciphertext to be sent. Then, the server decapsulates the shared key with their secret key, and a secure communication with symmetric cryptography can be established. The client (or other clients) repeats the process for creating other shared keys for other sessions.

In this setting, the adversary has access to the channel, capturing the ciphertexts that are being sent through it, along with the messages encrypted with the shared key. Therefore, they desire to retrieve the private key of the server, because it allows them to decapsulate the ciphertexts and decrypt all messages exchanged between the server and the client. However, the adversary is not able to communicate directly with the attacked devices. Therefore, they cannot craft special ciphertexts to attack the Fujisaki-Okamoto transform, as in [20,23].

The adversary also has physical access to the device where the server is being executed. There, they can place an electromagnetic probe and capture the radiation corresponding to the execution of the CRYSTALS-Kyber decapsulation.

3.2 Identifying vulnerable operation

In order to execute CEMA in CRYSTALS-Kyber, an operation that uses a controlled value and a part of the secret key must be chosen in the protocol. Inside the decapsulation algorithm, CRYSTALS-Kyber executes a PKE decryption, before performing the Fujisaki-Okamoto transform. This decryption is done with the secret key of the server and the ciphertext sent by the client. Going deeper, Algorithm 1 describes such decryption.

In such algorithm, the encoding operation is used to serialize byte arrays, polynomials and vectors of polynomials. The compressing operation allows to reduce the size of ciphertexts without having much effect on the correctness of the scheme. The decoding and decompressing operations are the inverses of encoding and compressing, respectively. In Line 4, an inner product between the secret key and part of the ciphertext is done. As they are in the NTT domain, each polynomial multiplication is a point-wise multiplication. This operation is then vulnerable to a CEMA as described before.

In the FPGA implementation from Xing and Li[24], PWM is done k times, according to the parameter k of CRYSTALS-Kyber, which defines the security level of the protocol and the sizes of the vectors $\boldsymbol{s}$ and $\boldsymbol{u}$. The result of each PWM is accumulated, in order to complete the matrix multiplication between $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{u}}$. The implementation's designers allocate the PWM operation described

Algorithm 1 KYBER.CPAPKE.Dec()

Require: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Require: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Ensure: Message $m \in \mathcal{B}^{32}$
1: $\mathbf{u} := \text{Decompress}_q(\text{Decode}_{d_u}(c), d_u)$
2: $v := \text{Decompress}_q(\text{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
3: $\hat{\mathbf{s}} := \text{Decode}_{12}(sk)$
4: $m := \text{Encode}_1(Compress_q(v - \text{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \text{NTT}(\mathbf{u})), 1))$
5: ▷ Secret key multiplied with known value. Thus, vulnerable to CEMA
6: **return** m

in Equation 1 in two clock cycles: PW0 and PW1, shown in (2).

$$\begin{aligned} \text{PWM0: } n_0 = \hat{f}_{2i} + \hat{f}_{2i+1},\ n_1 = \hat{g}_{2i} + \hat{g}_{2i+1},\ m_0 = \hat{f}_{2i} \cdot \hat{g}_{2i},\ m_1 = \hat{f}_{2i+1} \cdot \hat{g}_{2i+1} \\ \text{PWM1: } n_2 = m_0 + m_1,\ m_2 = n_0 \cdot n_1,\ m_3 = m_1 \cdot \zeta^{2br(i)+1}, \\ \hat{h}_{2i} = m_0 + m_3,\ \hat{h}_{2i+1} = m_2 - n_2 \end{aligned} \tag{2}$$

In the case of PWM between two polynomials in the secret vector $\hat{\boldsymbol{s}}$ and the ciphertext $\hat{\boldsymbol{u}}$, the inputs are $\hat{f} = \hat{\boldsymbol{s}}[j]$ and $\hat{g} = \hat{\boldsymbol{u}}[j]$ (or viceversa), for an integer $j \in [0, k-1]$. Now, the first cycle, *i.e.*, PWM0, is of special importance because it executes the operations $m_0 = \hat{\boldsymbol{s}}[j]_{2i} \cdot \hat{\boldsymbol{u}}[j]_{2i}$ and $m_1 = \hat{\boldsymbol{s}}[j]_{2i+1} \cdot \hat{\boldsymbol{u}}[j]_{2i+1}$. These two operations are done between a controlled input by the client and the secret key of the server. Thus, it is the final internal target for the CEMA in this work.

However, both targeted operations are executed at the same time, in one single clock cycle. To circumvent this issue, one can use the same traces for two different analyses. In the first analysis, the attacker can execute the CEMA procedure and obtain a guess for $\hat{\boldsymbol{s}}[j]_{2i}$, assuming that the other operations of the device, including the multiplication $\hat{\boldsymbol{s}}[j]_{2i+1} \cdot \hat{\boldsymbol{u}}[j]_{2i+1}$, generate random noise in the power trace that does not affect the correlation analysis. For the second analysis, the attacker uses the same power traces and obtains a guess for $\hat{\boldsymbol{s}}[j]_{2i+1}$, making the same noise assumption in the first analysis.

3.3 Usage of Hamming distance as consumption model

As the target is a hardware implementation instead of a software one, a more precise model is needed. Therefore, the decision was made to use the Hamming distance model, which requires the knowledge of previous reference values of an operation. In this setting, those reference values can be known by the attacker, because of their knowledge of the algorithm implementation.

In the implementation from Xing and Li [24], the reference values $\boldsymbol{r}[j * 256]$, $\boldsymbol{r}[j * 256 + 1]$ for $j \in [0, k-1]$ can be known for sure. These values are the result of the physical multipliers before modular reduction and before starting the first PWM0 cycle for each PWM. The operation before such cycles is the ending of the NTT of $\hat{\boldsymbol{u}}[j]$. Therefore, r_{j*256} is the result of the last multiplication in the

NTT algorithm of the even values of $\hat{\boldsymbol{u}}[\boldsymbol{j}]$ and $\boldsymbol{r}[j * 256 + 1]$ corresponds to the result of the last multiplication for the NTT of the odd values of $\hat{\boldsymbol{u}}[j]$. Then, for obtaining such result, it suffices to revert the last steps used to obtain $\hat{\boldsymbol{u}}[j]$. Such reversal is presented in (3). The reader can refer to [24] for details on the used NTT algorithm.

$$\begin{aligned} \boldsymbol{r}[j \cdot 256] &= (((\hat{\boldsymbol{u}}[j]_{252} - \hat{\boldsymbol{u}}[j]_{254}) \cdot two' \cdot \omega') \bmod q) \cdot \omega \\ \boldsymbol{r}[j \cdot 256 + 1] &= (((\hat{\boldsymbol{u}}[j]_{253} - \hat{\boldsymbol{u}}[j]_{255}) \cdot two' \cdot \omega') \bmod q) \cdot \omega \end{aligned} \tag{3}$$

The rest of the reference values depend on the success of the attack on the previous subkeys. They correspond to the previous result m_0 for the even values and to m_1 for the odd values in cycle PWM0 in (2). Such reference values are obtained like in (4), for $i \in [0, 127]$.

$$\begin{aligned} \boldsymbol{r}[j \cdot 256 + 2i] &= \hat{\boldsymbol{u}}[j]_{2i} * \hat{\boldsymbol{s}}[j]_{2i} \\ \boldsymbol{r}[j \cdot 256 + 2i + 1] &= \hat{\boldsymbol{u}}[j]_{2i+1} * \hat{\boldsymbol{s}}[j]_{2i+1} \end{aligned} \tag{4}$$

After executing the acquisition campaign, the analysis phase must be carried out sequentially, obtaining at first the subkeys $\hat{\boldsymbol{s}}[j]_0$ and $\hat{\boldsymbol{s}}[j]_1$, in order to obtain the next reference values for establishing the Hamming distance model.

4 Results

4.1 Attack setup

An FPGA evaluation general-purpose board was programmed with the implementation from [24]. The FPGA is configured with the target implementation slightly modified. The modifications consist of a UART interface for communication with the computer and a trigger to start the capture of the traces on the desired PWM operation. The evaluation board is a Digilent Basys-3 with a Xilinx Artix-7 XC7A35TCPG236 FPGA chip. The signal is shaped by a Langer RF-U 5-2 EM probe coupled with a Femto HSA-X-2-40 amplifier and captured with a Tektronix MSO64 oscilloscope configured to use a low-pass filter with a cut-off frequency of 200 MHz.

The security parameter k from CRYSTALS-Kyber was set to 2, the lowest one, for ease of analysis. It should be noted that increasing the security parameter should not make the attack more difficult. The effect of such a modification is to increase the number of subkeys, but because of the divide-and-conquer nature of this attack, this would only lead to a linear increase of the attack runtime.

The sample rate of the oscilloscope was fixed to 1.25 GS/s, whereas the FPGA was programmed to use a clock of 62.5 MHz generated by the Artix-7 chip which in turn, gets its 100 MHz reference clock from the Basys-3 board. Therefore, the number of samples per clock cycle is 20. 15 sets of about 11k traces are used (166,620 traces in total), calculating the correlation incrementally with each set.

Instead of using the full traces for attacking each subkey, it was only selected the clock cycle where the result of the multiplication with the subkey is being updated. This was done to reduce the runtime. Fig. 1 shows one full trace of a PWM execution, and the selected traces for attacking the first subkey, *i.e.*, $\hat{\boldsymbol{s}}[0]_0$.

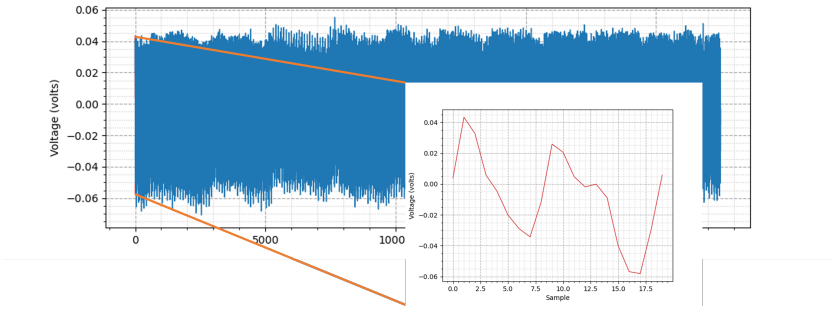


Fig. 1: Full trace of a PWM execution, with the samples selected for the analysis for subkey $\hat{\boldsymbol{s}}[0]_0$.

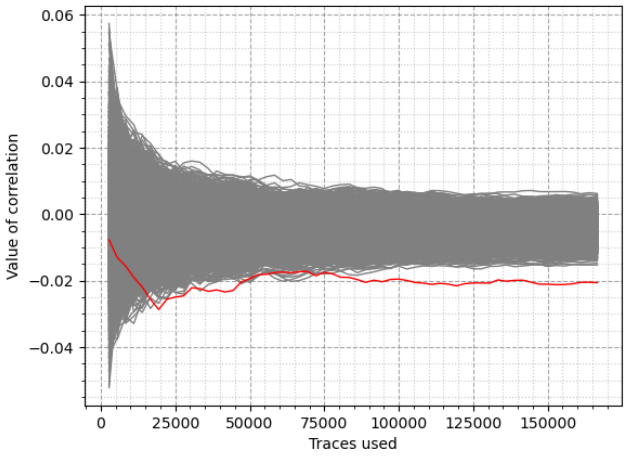
In this test attack, the secret key and all reference values are known beforehand, so it is not sequential like explained in Section 3.3. The attack script is written in Python, with some CPU parallelization provided by the Numba library.

4.2 Attack results

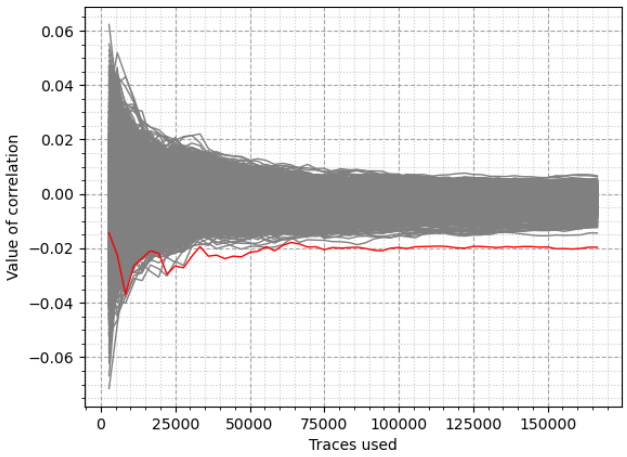
The trace capture campaign took around 6 hours and 45 minutes, with around 2 hours and 45 minutes of runtime for the analysis part, using a laptop with an Intel Core i7-11850H processor. The success rate SR, when executing the test attack in a non-sequential way as explained in Section 4.1, is 100%. That is, the accuracy of finding the correct subkeys when only selecting the guess with the highest correlation coefficient. Even if the attack had been sequential, SR would still hold at 100%, because there is not an incorrect guess that would induce to have false reference values and prevent the analysis of further subkeys.

In Fig. 2, the highest correlation for the first sample, of each key guess in function of the number of sets used for the subkeys $\hat{\boldsymbol{s}}[0]_0$ and $\hat{\boldsymbol{s}}[0]_{128}$ is shown. In red, the highest correlation of the correct key guess is shown, surpassing visibly the other key guesses. In Fig. 3, the correlation coefficient of all samples considered for subkeys $\hat{\boldsymbol{s}}[0]_0$ and $\hat{\boldsymbol{s}}[0]_{128}$ after 166,620 traces is shown. The correct guess is shown in red, and it is visibly above other key guesses in multiple points.

The rank for all subkeys is shown in Fig. 4. After aproximately 80,000 traces, all of the subkeys have a value under 10. Also, it can be seen that some of the subkeys take a bit longer to have the correct guess among the first candidates.



(a) Graph for $\hat{\boldsymbol{s}}[0]_0$

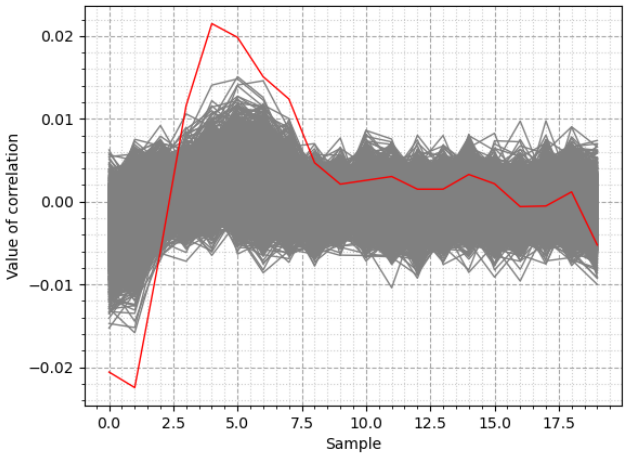


(b) Graph for $\hat{\boldsymbol{s}}[0]_{128}$

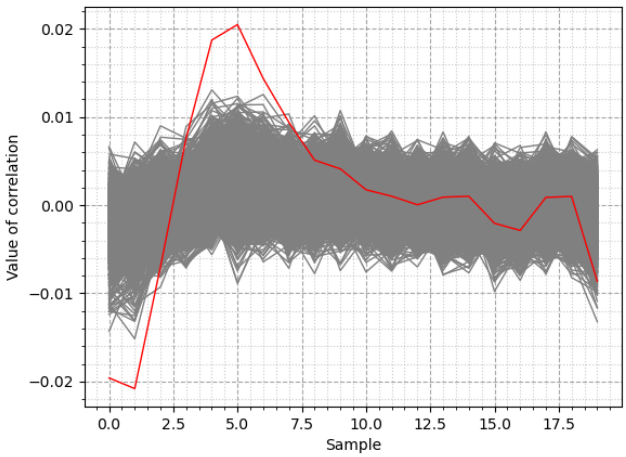
Fig. 2: Highest correlation of the first sample for each key guess in function of the number of sets of traces used for the subkey $\hat{\boldsymbol{s}}[0]_0$ and $\hat{\boldsymbol{s}}[0]_{128}$. In red, correct key guess

5 Cost-effective countermeasure

Masking countermeasures have already been studied to protect CRYSTALS-Kyber implementation against this type of vertical SCA. Bos et al. [6] conduct a complete study on how to mask CRYSTALS-Kyber. For the decryption part,



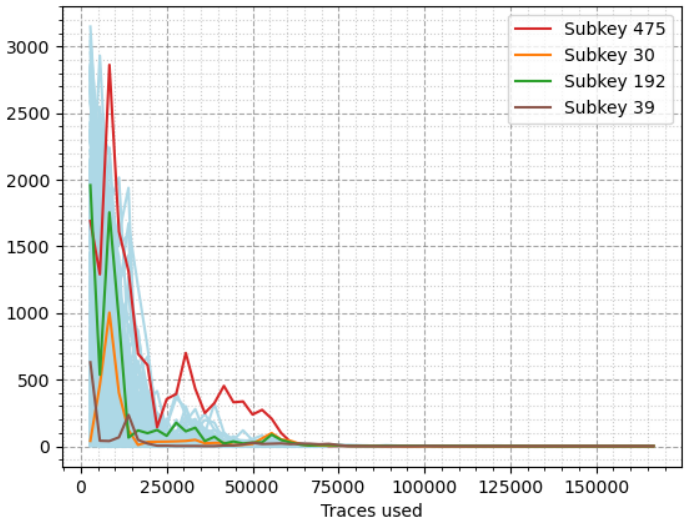
(a) Graph for $\hat{\boldsymbol{s}}[0]_0$



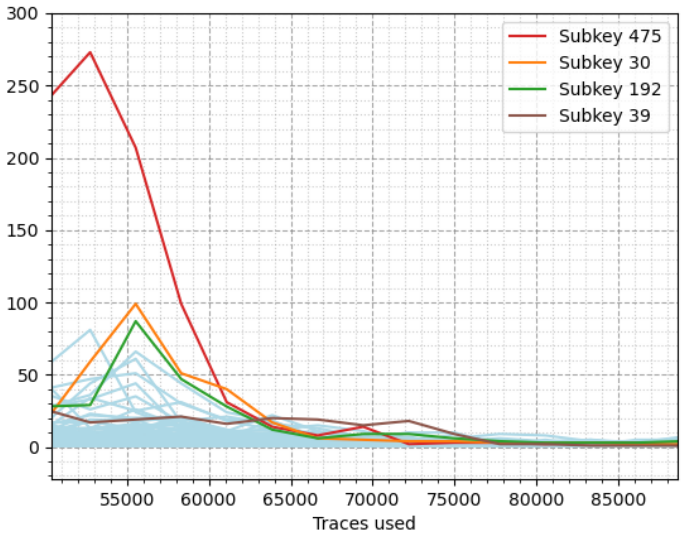
(b) Graph for $\hat{\boldsymbol{s}}[0]_{128}$

Fig. 3: Correlation coefficient for all samples considered for subkey $\hat{\boldsymbol{s}}[0]_0$ and $\hat{\boldsymbol{s}}[0]_{128}$ for each key guess after 166,620 traces are used. In red, correct key guess.

they perform additive masking. On the other hand, Hamoudi et al. [10] propose a multiplicative masking, that is more performing than additive masking when using an Intel general purpose processor. However, because of the nature of the attack proposed in this work, a simpler countermeasure is proposed.



(a) Graph for all of the traces used



(b) Zoomed graph around 50000-90000 traces

Fig. 4: Rank for all the subkeys

This attack leverages the supposition that the device is leaking the Hamming distance of the results of the attacked multiplier. The attacker knows the first reference values for the first subkeys, because the NTT of the ciphertext is performed before the PWM between the secret vector and the ciphertext in the NTT domain. If a random operation is performed in the multiplier after the NTT and before the PWM, the attacker will not know the reference value and hence, the attack as it is presented here will not work.

In this case, the decision was made to use a simple linear-feedback shift register (LFSR) to provide the inputs of the multiplier. This is because those inputs are never used in the algorithm and so, there is no need for a cryptographically-secure pseudorandom number generator. Also, as the attack is performed sequentially, this random multiplication can be performed just k times, before the start of each PWM sequence. This is because if the attacker does not know the reference values for the first subkeys, then they cannot continue with the rest of the key.

The LFSR used is a maximal LFSR of degree 24, in order to use 24 bits of its state. It is initialized at `0xaaaaaa`, started when the device is powered up and never stopped. The first 12 bits of the LFSR state are used as the first input of both of the butterfly units of the implementation in [24]. The last 12 bits are used as the second input of such units, as well as the twiddle factor inputs.

This countermeasure has a time overhead of only k clock cycles. When looking at area, it uses 7694 LUT and 4953 flip-flops, with an overhead of 3.80% and 6.65% respectively, compared to the original implementation in [24].

5.1 Evaluation of countermeasure

For evaluating the countermeasure, the same attack procedure described in Section 3 is used. However, the practical setting varies a little from the one shown in Section 4.1. In this case, instead of using 15 sets of about 11k traces, 150 sets are used, for a total of 1,666,200 traces. That is, ten times more traces. Also, the attack is only done against subkeys $\hat{\boldsymbol{s}}[j]_0$ and $\hat{\boldsymbol{s}}[j]_1$, for $j \in [0, k-1]$, using the samples in the traces where the multiplication with these coefficients is being done.

The attack is not successful for any of the analyzed subkeys, even when using this number of samples. In Fig. 5, the highest correlation, for all considered samples, of each key guess in function of the number of sets used for the subkey $\hat{\boldsymbol{s}}[0]_0$ is shown. In red, the highest correlation of the correct key guess is shown, being visibly similar to other key guesses. In Fig. 6, the correlation coefficient of all samples considered for subkey $\hat{\boldsymbol{s}}[0]_0$ is shown. The correct guess is shown in red, and it does not surpass other key guesses at any point. In Fig. 7, the rank for subkeys $\hat{\boldsymbol{s}}[0]_0$, $\hat{\boldsymbol{s}}[0]_1$, $\hat{\boldsymbol{s}}[1]_0$ and $\hat{\boldsymbol{s}}[1]_1$ in function of the number of used trace sets is shown. As seen, a constant and regular drop in the rank is not present.

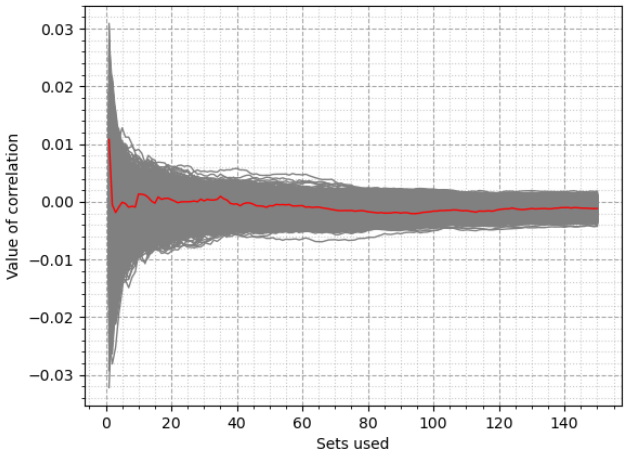


Fig. 5: Highest correlation of all samples for each key guess in function of the number of sets of traces used for the subkey $\hat{\boldsymbol{s}}[0]_0$ after countermeasure. In red, correct key guess.

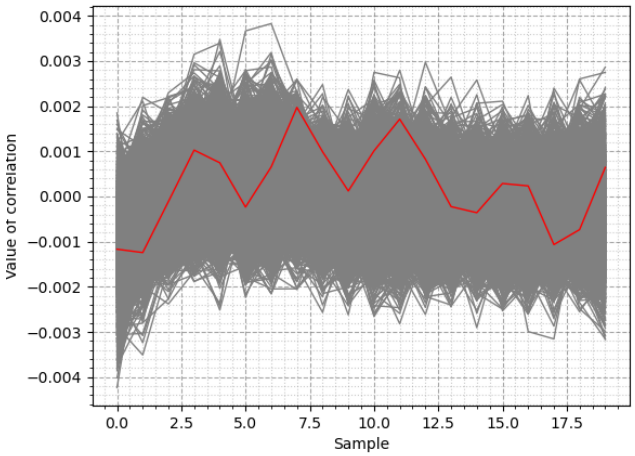


Fig. 6: Correlation coefficient for all samples considered for subkey $\hat{\boldsymbol{s}}[0]_0$ for each key guess after all traces are used. In red, correct key guess.

5.2 Limitations of the countermeasure

This countermeasure works well because this attack assumes the leakage of the Hamming distance of the hardware multiplier result within the FPGA. If this

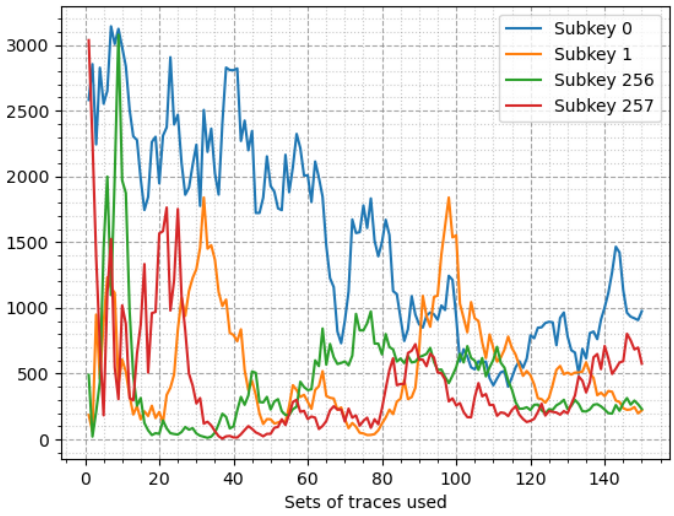


Fig. 7: Rank for considered subkeys in function of the number of sets of traces used for the attack.

attack were to use a power model that only takes into consideration the instant manipulated values (such as the Hamming weight), this countermeasure would be useless. In that case, complete arithmetic masking [6,10] comes as the best countermeasure for this type of attack.

Another limitation is that this countermeasure could be circumvented by a higher order attack, leveraging information from several clock cycles, in order to compensate the lack of information that the attacker has over the switching activity of the circuit. This type of attacks are more complex, but the attacker only needs to execute it for the first subkeys $\hat{\boldsymbol{s}}[j]_0$ and $\hat{\boldsymbol{s}}[j]_1$, for $j \in [0, k-1]$. After that, the attacker can continue executing the attack as proposed in Section 3.

6 Conclusion

A correlation electromagnetic attack on a compact hardware implementation of CRYSTALS-Kyber is presented. It recovers the secret subkeys of the Kyber-512 version with a success rate of 100%, given the knowledge of register reference values. Furthermore, a lightweight countermeasure has been presented against this specific attack and platform, with its limitations against other power consumption models or more advanced attacks.

Even though the number of employed traces may question the practicality of this attack, this work stresses the need of research for securing post-quantum cryptography implementations.

References

1. Askeland, A., Rønjom, S.: A Side-Channel Assisted Attack on NTRU. Cryptology ePrint Archive, Report 2021/790 (2021), https://ia.cr/2021/790
2. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (2020), https://pq-crystals.org/kyber/
3. Bellizia, D., Mrabet, N.E., Fournaris, A.P., Pontié, S., Regazzoni, F., Standaert, F.X., Tasso, E., Valea, E.: Post-quantum cryptography: Challenges and opportunities for robust and secure hw design. In: 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). pp. 1–6 (2021). https://doi.org/10.1109/DFT52944.2021.9568301
4. Bhasin, S., Danger, J.L., Guilley, S., Najm, Z.: NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. In: 2014 International Symposium on Electromagnetic Compatibility, Tokyo. pp. 310–313. IEEE (2014)
5. Bhasin, S., D'Anvers, J.P., Heinz, D., Pöppelmann, T., Beirendonck, M.V.: Attacking and defending masked polynomial comparison for lattice-based cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 334–359 (2021). https://doi.org/10.46586/tches.v2021.i3.334-359
6. Bos, J.W., Gourjon, M., Renes, J., Schneider, T., Vredendaal, C.V.: Masking Kyber: First- and Higher-Order Implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 173–214 (2021). https://doi.org/10.46586/tches.v2021.i4.173-214
7. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Lecture Notes in Computer Science, pp. 16–29. Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
8. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. pp. 537–554. Springer Berlin Heidelberg (1999)
9. Hamburg, M., Hermelink, J., Primas, R., Samardjiska, S., Schamberger, T., Streit, S., Strieder, E., Vredendaal, C.V.: Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 88–113 (2021). https://doi.org/10.46586/tches.v2021.i4.88-113
10. Hamoudi, M., Bel Korchi, A., Guilley, S., Takarabt, S., Karray, K., Souissi, Y.: Side-Channel Analysis of CRYSTALS-Kyber and A Novel Low-Cost Countermeasure. In: Communications in Computer and Information Science, pp. 30–46. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-90553-8_3
11. Karlov, A., Linard de Guertechin, N.: Power Analysis Attack on Kyber. Cryptology ePrint Archive, Report 2021/1311 (2021), https://ia.cr/2021/1311
12. Langlois, A., Stehlé, D.: Worst-Case to Average-Case Reductions for Module Lattices. Des. Codes Cryptography **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4

13. Mujdei, C., Beckers, A., Bermundo, J., Karmakar, A., Wouters, L., Verbauwhede, I.: Side-Channel Analysis of Lattice-Based Post-Quantum Cryptography: Exploiting Polynomial Multiplication. Cryptology ePrint Archive, Report 2022/474 (2022), https://ia.cr/2022/474
14. National Institute of Standards and Technology: Post-Quantum Cryptography Standardization Process, https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization
15. Ngo, K., Dubrova, E., Guo, Q., Johansson, T.: A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 676–707 (2021). https://doi.org/10.46586/tches.v2021.i4.676-707
16. Ouladj, M., Guilley, S.: Side-Channel Analysis of Embedded Systems. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-77222-2
17. Pessl, P., Primas, R.: More Practical Single-Trace Attacks on the Number Theoretic Transform. In: Progress in Cryptology – LATINCRYPT 2019, pp. 130–149. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-30530-7_7
18. Primas, R., Pessl, P., Mangard, S.: Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In: Lecture Notes in Computer Science, pp. 513–533. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-66787-4_25
19. Ravi, P., Ezerman, M.F., Bhasin, S., Chattopadhyay, A., Sinha Roy, S.: Will You Cross the Threshold for Me? Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(1), 722–761 (2021). https://doi.org/10.46586/tches.v2022.i1.722-761
20. Ravi, P., Sinha Roy, S., Chattopadhyay, A., Bhasin, S.: Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 307–335 (2020). https://doi.org/10.46586/tches.v2020.i3.307-335
21. Regev, O.: On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing - STOC '05. ACM Press (2005). https://doi.org/10.1145/1060590.1060603
22. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Comput. Soc. Press (1994). https://doi.org/10.1109/sfcs.1994.365700
23. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 296–322 (2021). https://doi.org/10.46586/tches.v2022.i1.296-322
24. Xing, Y., Li, S.: A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 328–356 (2021). https://doi.org/10.46586/tches.v2021.i2.328-356
25. Xu, Z., Pemberton, O.M., Sinha Roy, S., Oswald, D., Yao, W., Zheng, Z.: Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber. IEEE Transactions on Computers (2021). https://doi.org/10.1109/TC.2021.3122997