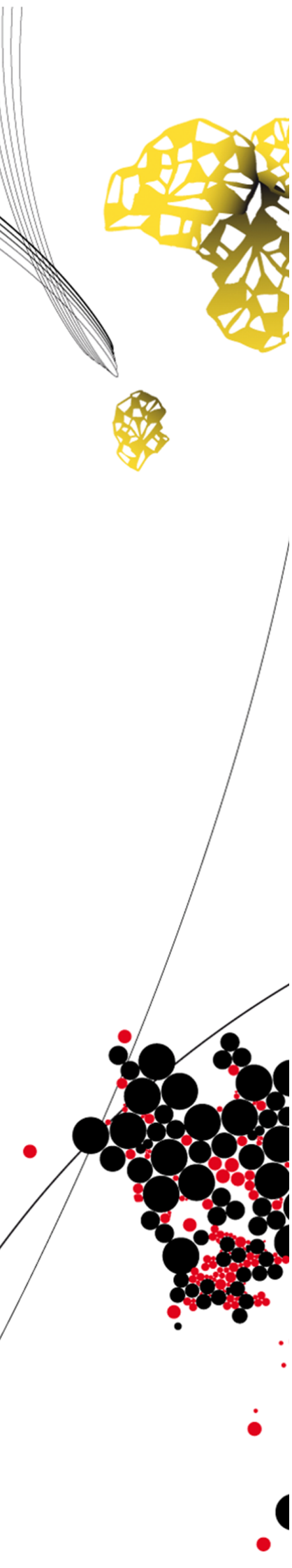


UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Towards Future Proof Cryptographic Implementations: Side-Channel Analysis on Post-Quantum Key Encapsulation Mechanism CRYSTALS - Kyber

J.J.W. Meijer

M.Sc. Thesis - Embedded Systems
June 2023

Supervisors:
dr.ir. M. Ottavi
B. Endres Forlin
Committee:
prof.dr.ir A.L. Varbanescu
dr.ir. A. Continella
L. Mariot

Computer Architecture for Embedded Systems
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

List of acronyms

NIST	National Institute of Standards and Technology
PQC	post-quantum cryptography
KEM	key encapsulation mechanism
PKE	public-key encryption
IoT	internet of things
EM	electromagnetic
SCA	side-channel analysis
SPA	simple power analysis
DPA	differential power analysis
CPA	correlation power analysis
SASCA	soft-analytical side-channel analysis
SNR	signal-to-noise
ISA	instruction set architecture
LWE	learning with errors
R-LWE	ring learning with errors
M-LWE	module learning with errors
CVP	closest vector problem
IND-CPA	indistinguishability under chosen plaintext attack
IND-CCA	indistinguishability under chosen ciphertext attack
FO-transform	Fujisaki-Okamoto transform

NTT	number-theoretic transform
XOF	extendable output function
PCC	Pearson correlation coefficient
DUT	device under test
USB	universal serial bus
GPIO	general purpose input-output
SoC	system-on-a-chip
FPU	floating-point unit
MCU	microcontroller
SAD	sum of absolute difference
TVLA	test vector leakage assessment
NICV	normalized inter-class variance
PGE	partial guessing entropy

Acknowledgement

I would like to express my sincere gratitude to all my supervisors for providing me with valuable feedback and guidance. I am grateful for the opportunities that I have been given and for the educational experience that this turned out to be. I would especially like to thank my daily supervisor Bruno for contending with my daily questions and helping me navigate through the scientific work that was required for this thesis. His guidance and expertise have been instrumental in shaping the outcome of this thesis.

Chapter 1

Introduction

1.1 Context

Modern-day digital infrastructure depends heavily on cryptographic algorithms, which allow for secure communication. Many digital activities require security guarantees, which cryptographic algorithms can provide. A common usage of cryptography is to encrypt messages and guarantee their integrity and authenticity. Often parties use a symmetric cryptography method. In symmetric cryptography, both parties have the same private key, which they use to encrypt and decrypt their communication. In asymmetric encryption, there are different keys. A key pair consisting of a public key and a private key is used to encrypt and decrypt data. This is also known as PKE. PKE tends to require more resources than symmetric cryptography. Often PKE is used to establish a symmetric key after which symmetric cryptography is used to communicate. Cryptographic primitives can provide security through the use of specific mathematical problems, which are deemed hard to solve. The integer factorization problem, discrete logarithm problem, and elliptic-curve discrete logarithm problem are some examples of commonly used mathematical problems. One of the most used PKE algorithms is RSA [1], which makes use of the integer factorization problem.

In the last decade, significant progress has been made in a new computing concept, namely quantum computing. Quantum computers leverage quantum physics to significantly increase computational power. Theoretically, quantum computing will be able to solve particular problems with orders of magnitude faster than classical computing. In 1994 Shor [2] proposed a quantum algorithm that is able to solve the integer factorization problem and discrete logarithm problem within polynomial time. With the advent of quantum computing some of our most used cryptographic primitives have become threatened. Fortunately, as of today (2023), no quantum computer is powerful and stable enough to break current cryptographic standards.

Algorithm	Cryptographic Scheme	Mathematical basis
CRYSTALS-Kyber	Key Encapsulation Scheme	Lattice-based
CRYSTALS-DILITHIUM	Signature Scheme	Lattice-based
FALCON	Signature Scheme	Lattice-based
SPHINCS+	Signature Scheme	Hash-based

Table 1.1: Selected candidates to be standardized (2022)

The field of quantum computing is constantly advancing, and thus the threat to cryptography is ever looming. With the future in mind, we should look for new encryption schemes that are not vulnerable to quantum algorithms. This area of study is called post-quantum cryptography (PQC). With this goal in mind, the National Institute of Standards and Technology (NIST) initiated the PQC standardization process. The goal of this process is to find suitable post-quantum secure encryption algorithms. The NIST PQC competition is currently in its fourth round. Already, four algorithms have been selected for standardization. Three of which are digital signature schemes (CRYSTALS-DILITHIUM [3], SPHINCS+ [4], FALCON [5]). Digital signature schemes are designed to verify the authenticity and integrity of digital messages. The remaining candidate CRYSTALS-KYBER [6] is a key encapsulation mechanism (KEM). The goal of a KEM is to facilitate the establishment of a shared symmetric key between two parties. PQC algorithms leverage mathematical problems that are hypothesized to be computationally hard for quantum algorithms. Table 1.1 gives an overview of the four candidates that have been chosen for standardization by the NIST PQC process.

However secure an encryption algorithm might be, an insecure implementation can still be compromised. A special attack can be designed that does not focus on the algorithm itself, but instead on the way that the algorithm has been implemented on a device. When data is processed physical properties can be observed. For example, when many lines of a data bus are set to logic level 'high', one might notice an increase in power consumption. These physical properties can be leveraged by an attacker to possibly reveal secrets used by an encryption scheme. Possible sources of information are timing, power consumption, electromagnetic radiation temperature, and sound. Attacks utilizing these unintended sources of information are referred to as SCA attacks [7]. They aim to reveal secrets by cleverly associating leakage channels with secret variables. Successful side-channel attacks require significant setup and expertise. Physical access to the device is often necessary to perform successful attacks. Historically, for practical reasons, this was often too large a boundary to overcome. However, with the ever-increasing number of internet of things (IoT) [8] devices and embedded devices, this threshold has decreased

significantly. Due to the pervasive nature and safety-critical position of these devices security is of great importance [9]. An important area of research, therefore is the susceptibility to SCA attacks of these devices [10].

Many of the NIST PQC standardization competition candidates' SCA characteristics have already been studied. It has been found that PQC implementations without SCA countermeasures in mind are vulnerable to possible SCA attacks [11] [12] [13] [14].

1.2 Research objective

The previous section has highlighted the importance of finding new quantum-resistant cryptographic schemes. The section also discussed how new cryptographic implementations should be evaluated on their resistance to SCA attacks. Currently, Kyber seems one of the most promising PQC candidates. The scheme is currently the only KEM that has been selected to be standardized by the NIST PQC competition. The SCA resilience of IoT implementations is especially important. Not only is physical access to IoT devices often easier, they are often used in pervasive and safety-critical applications. As of today (2023), the SCA resistance of Kyber implementations on RISC-V processors has not been extensively studied. To investigate these issues this thesis will try to answer the following research questions:

> ***RQ 1.*** *How resistant are RISC-V Kyber implementations to power side-channel analysis attacks?*
>
> ***RQ 2.*** *What protection mechanisms can be implemented on RISC-V Kyber implementations to protect against power side-channel analysis attacks?*

The research questions mentioned above were formulated at the outset of this thesis. Answering the research questions stated above is non-trivial as many different aspects are involved. While quantitative metrics such as normalized inter-class variance (NICV) [15] and test vector leakage assessment (TVLA) [16] can provide some insights and help answer **RQ 1**, they do not provide a comprehensive understanding of how leakages can lead to exploitation. Relying solely on metrics has its shortcomings as metrics only focus on side-channel leakage and not on the exploitation of these leakages [17]. To overcome these limitations, a more explorative approach was adopted in this study. This approach involved analyzing Kyber for weaknesses and then attempting to exploit these weaknesses through a SCA. By

developing an attack, a better comprehension of the exploitation of identified weaknesses can be obtained, partially addressing **RQ 1**. Furthermore, a new implementation with countermeasures can be developed that should mitigate the derived attack. Which in turn addresses **RQ 2**. Although this methodology might not yield a quantitative review of SCA characteristics of Kyber implementations, it will contribute to a deeper understanding of how SCA can attack Kyber, which in turn will aid in ensuring the security of possible future Kyber implementations.

1.3 Outline

In chapter 2 the necessary background for understanding this thesis will be given. Most importantly the Kyber algorithm and side-channel analysis attacks will be described. Chapter 3 lays out the experimental setup and argues why the *PQCLEAN* implementation has been chosen as a target. Then simple power analysis (SPA) is performed on the target implementation for both an ARM and RISC-V microcontroller (MCU). SPA could identify leakage on the ARM MCU, but not on the RISC-V MCU. In chapter 4 power side-channel leakage is identified through the use of correlation power analysis (CPA) for both MCUs. Furthermore, a chosen-ciphertext CPA attack on the decryption procedure is proposed and experimentally validated. In chapter 5 possible countermeasures to the attack are considered and experimentally verified. In chapter 6 the relevance and implications of the results are discussed. Furthermore, the limitations of the research are considered. Finally, in chapter 7 the thesis is concluded and future avenues of research are suggested.

Chapter 2

Background

In this chapter general background is presented to provide the reader with the necessary preliminary understanding of the various subjects discussed in this thesis. Section 2.1 provides an introduction to the Kyber algorithm. Section 2.2 introduces the concept of side-channel analysis and its potential for compromising cryptosystems. Section 2.3 introduces the RISC-V instruction set architecture (ISA). Lastly, section 2.4 will list related work and their corresponding findings.

2.1 CRYSTALS - Kyber

Kyber [18] is currently the sole KEM that has been selected to be standardized in the PQC standardization competition. The purpose of a KEM is to facilitate establishing a shared symmetric secret-key between two parties. Kyber falls into the category of lattice-based cryptography. Currently, three out of the four algorithms selected to be standardized in the NIST process belong to this category [3], [5], [18]. Lattice-based cryptosystems have been shown to have attractive properties for cryptography, as they allow for the construction of multiple ”hard” computation mathematical problems. Examples of such problems include *SVP*, *GapSVP*, *CVP*, and *GapCVP*. O. Regev states that one might even conjecture that lattice-based problems are hard for quantum computers [19]. Adding to the appeal of lattice-based cryptosystems for PQC. However, the only evidence for this is that as of today (2023), no quantum algorithms exist that solve lattice problems faster than classical algorithms. This conjecture is an important area of study within lattice-based PQC.

2.1.1 Learning With Errors

In 2005, O. Regev [20] introduced the learning with errors (LWE) problem. It is hypothesized that there exists no quantum algorithm that can solve LWE in polynomial

time. Regev's main result was the reduction of the worst-case *GapSVP* problem to the LWE problem. His reduction required the use of a quantum algorithm. The requirement of a quantum algorithm for his proof implies that if a quantum algorithm can efficiently (polynomial complexity) solve LWE, then there also exists a quantum algorithm capable of efficiently solving the *GapSVP* problem. Not only did O. Regev prove the reduction in his paper, but he also constructed a PKE scheme that is based on the LWE problem. In doing so, Regev provided an initial foundation for constructing quantum-resistant cryptosystems.

Problem Description The LWE can be described as a system of equations with errors. Given $n \geq 1$ and a modulus $q \in \mathbb{Z}$ We have an error distribution $\chi : \mathbb{Z}_q \to \mathbb{R}^+$ and a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$. One can request polynomially many equations with errors. The equations are of the following form:

$$\begin{aligned} \mathbf{a_1}^T\mathbf{s} + e_1 &= b_1 \mod q \\ \mathbf{a_2}^T\mathbf{s} + e_2 &= b_2 \mod q \\ &\vdots \end{aligned}$$

For every i'th equation $\mathbf{a}_i$ is sampled uniformly from $\mathbb{Z}_q^n$ and the error e_i is sampled from the error distribution χ. The goal is to determine $\mathbf{s}$ when only given $\langle \mathbf{a_1}, \mathbf{a_2}, ... \rangle$ and $\langle b_1, b_2, ... \rangle$.

Without the introduced error term this problem would be trivial to solve using Gaussian Elimination. Figure 2.1 provides a visualization of the closest vector problem (CVP) in two dimensions. This problem can be reduced to the LWE problem and serves as an illustrative example. The basis vectors form a lattice. The basis vectors are highlighted in blue in the figure. Furthermore, a point on the lattice is given with some error added to it. This point has been highlighted in red. The goal of CVP is to find the point on the lattice that is closest to the point with error. In this case, the green point is closest to the point with error. Although this might seem like a trivial problem in two dimensions. In higher dimensions, this problem becomes significantly more difficult.

Multiple variations of the LWE problem exist. The problem described above is the search variant of LWE. A decision variant also exists, where one has to distinguish whether the input is taken from the problem above or uniformly sampled. It is also possible to take a discrete error distribution, such as the discrete Gaussian distribution. A disadvantage of using LWE to build a cryptosystem is that it requires large key sizes. An alternative version of LWE is ring learning with errors (R-LWE).

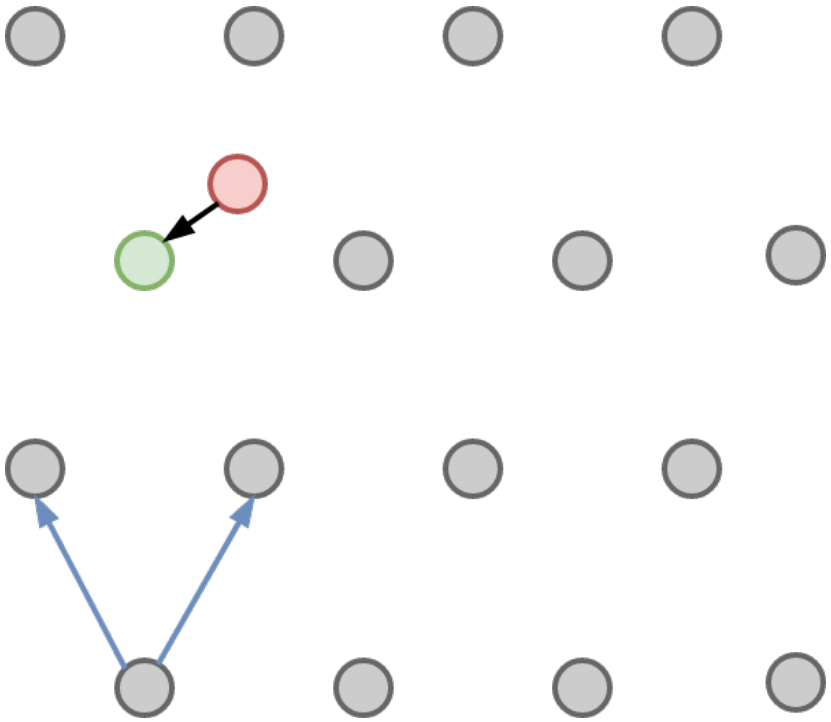


Figure 2.1: CVP in 2-dimensional lattice

The advantage of R-LWE is that it requires fewer resources than LWE. The disadvantage of R-LWE is that an additional structure is introduced into the problem. This additional structure might be leveraged by an attacker. The CRYSTALS-Kyber algorithm is based on module learning with errors (M-LWE). M-LWE provides a middle ground between LWE and R-LWE. It has larger resource requirements compared to R-LWE but is also thought to be more secure than R-LWE. Yet it does not have the stringent resource requirements that LWE requires. Because of these reasons the M-LWE problem is a suitable candidate for developing practical PQC cryptosystems. Because of the addition of random noise, there is a chance that the key exchange procedure fails. The probability of failure is referred to as correctness probability. By choosing appropriate parameters, this probability can be minimized, rendering it practically negligible.

2.1.2 Kyber's Building Blocks

Polynomial Rings Kyber's basic building blocks are elements from the polynomial ring denoted by R_q:

$$R_{3329} : \mathbb{Z}_{3329}[X]/(X^{256} + 1)$$

An element from the polynomial ring above will be a 255-degree polynomial with all coefficients belonging to $\mathbb{Z}_{3329}$. Mathematically, $\mathbb{Z}_{3329}$ is the group of integers modulo 3329. Since the coefficients belong to $\mathbb{Z}_{3329}$, their values belong to the range of

$\{0, 3328\}$. These polynomials are closed under addition, subtraction, and polynomial multiplication. After polynomial multiplication, it is possible to have a polynomial with a degree bigger than 255. These polynomials will be reduced by the polynomial $X^{256} + 1$. In essence, $X^{256} + 1$ acts as a modulus. One can compute the result of the reduction by performing a long division of the polynomial by $X^{256} + 1$ and taking the corresponding remainder. Kyber also makes use of vectors and matrices. The elements of these objects belong to the polynomial ring R_{3329}. Vectors and matrices will be denoted by $\mathbf{x}$ and $\mathbf{X}$ font respectively. The dot product and matrix multiplication are defined where the inner elements are multiplied using polynomial multiplication.

Binomial Distribution Kyber also makes use of the centred binomial random distribution. It is defined for some positive integer η to be:

Sample uniformly $\{(a_i, b_i)\}_{i=1}^{\eta} \leftarrow (\{0, 1\}^2)^{\eta}$ and output $\sum_{i=1}^{\eta}(a_i - b_i)$.

This means that if $\eta = n$ the possible range of values is equal to $\{-n, n\}$ centred around 0. The binomial distribution is denoted by β_η. If a variable is taken from the binomial distribution β_η this is denoted by $v \leftarrow \beta_\eta$. If a k-sized vector $\mathbf{v}$ is sampled by the binomial distribution this is denoted by $\mathbf{v} \leftarrow \beta_\eta^k$

Compression and Decompression To reduce the size of the public key and ciphertext Kyber applies compression. Kyber defines the functions $Compress(x, d)$ and $Decompress(x, d)$ for compression and subsequent decompression. These functions are defined as:

$$Compress_q(x, d) = \lceil (2^d/q) \cdot x \rfloor \mod^+ 2^d$$

$$Decompress_q(x, d) = \lceil (q/2^d) \cdot x \rfloor$$

Where d is a parameter that indicates how many bits should be used to represent x. The purpose of these compression functions is to get rid of the least-significant bits. This can be done, since these bits do not have a large influence on the correctness probability δ.

Montgomery Reduction Kyber's basic elements belong to the polynomial ring R_{3329}. This means that every coefficient is an element of the multiplicative group of integers modulo 3329. The general approach to multiplying two coefficients is to perform multiplication followed by modular reduction. However, a naive implementation

of modular reduction requires costly operations like integer division. Montgomery introduced a modular reduction technique that allows efficient modular reductions [21]. His clever idea was to replace expensive divisions with inexpensive divisions by R, where $R = 2^r$. Divisions by R can be implemented with bit shifts instead of an expensive multi-cycle division operation because R is a power of two. The only difference to using the Montgomery reduction is that the result will be congruent to $a \cdot R^{-1} \mod Q$, instead of $a \mod Q$. This difference can be easily negated by multiplying a with R before performing the Montgomery reduction, as the result will be congruent to $a \cdot R \cdot R^{-1} \equiv a \mod Q$. Algorithm 1 presents the Montgomery reduction operation as described in Montgomery's paper. It is worth noting that the division $(a + mQ)/R$ can be implemented using bit shifts if R is a power of two. In the Kyber implementation, some optimizations were made to the Montgomery reduction. These optimizations are possible because R and Q are known beforehand.

Algorithm 1 `montgomery_reduce(`a`)`

1: $m \leftarrow (a \mod R) \cdot Q^{-1} \mod R$
2: $t \leftarrow (a + mQ)/R$
3: **if** $t \geq Q$ **then**
4: **return** $t - N$
5: **else**
6: **return** t
7: **end if**

2.1.3 The Fujisaki-Okamoto transform

The basic form of Kyber has a notion of security known as indistinguishability under chosen plaintext attack (IND-CPA) [22] This is considered to be a requirement for provably secure asymmetric encryption schemes. Intuitively this notion of security requires that an attacker cannot get additional information from the ciphertext. This notion is a weaker version of indistinguishability under chosen ciphertext attack (IND-CCA) introduced by [23]. Since IND-CCA has stronger security than IND-CPA it is desirable for Kyber to adhere to this IND-CCA notion of security. This is achieved through the Fujisaki-Okamoto transform (FO-transform) [24] The FO-transform lifts Kyber from IND-CPA to IND-CCA security. This is achieved by wrapping the IND-CPA version of the algorithm by another algorithm. Because of this, we are dealing with two different versions of Kyber which we will refer to as IND-CPA Kyber and IND-CCA Kyber. IND-CPA Kyber is a PKE cryptosystem whereas IND-CCA Kyber is a KEM. The specific KEM variant of FO-transform with 'implicit rejection' used for Kyber is described in [25].

2.1.4 Parameter Sets

Kyber offers multiple different parameter sets [6]. These various versions of Kyber are referred to as Kyber512, Kyber768, and Kyber1024. The purpose of parameter sets is to offer more flexibility by choosing between different security and performance trade-offs of Kyber. The parameters need to be considered carefully as they have a significant influence on the accuracy, security, and performance of the algorithm. Furthermore, wrongly chosen parameters can make the algorithm completely invalid. The various parameters are listed in table 2.1. Table 2.2 lists sizes of secret-key, public-key, and ciphertext corresponding to the parameter sets. The next section will explain the different parameters in more detail.

	n	k	q	η_1	η_2	(d_u, d_v)	δ
Kyber512	256	2	3329	3	2	(10,4)	2^{-139}
Kyber768	256	3	3329	2	2	(10,4)	2^{-164}
Kyber1024	256	4	3329	2	2	(11,5)	2^{-174}

Table 2.1: Kyber parameters sets

	secret-key	public-key	ciphertext
Kyber512	1632	800	768
Kyber768	2400	1184	1088
Kyber1024	3168	1568	1568

Table 2.2: Packet sizes for Kyber (bytes)

The most notable difference is the k parameter, which refers to the dimension of the vectors and matrices in the algorithm. A higher k is considered to be more secure but comes at the cost of higher resource requirements, both in key size and execution time. This higher cost increases linearly with k.

The parameter q refers to the modulus of the coefficients. This parameter needs to be considered very carefully as it has a big influence on the polynomial multiplications done in the algorithm. This is chosen to be 3329 for all versions of Kyber. Both η_1 and η_2 refer to the η used for the binomial distribution. The difference between the two is that different distributions need to be used for different variables.

Compression and decompression are influenced by (d_u, d_v). Both parameters indicate how many bits are used to represent different parts of the ciphertext. Smaller

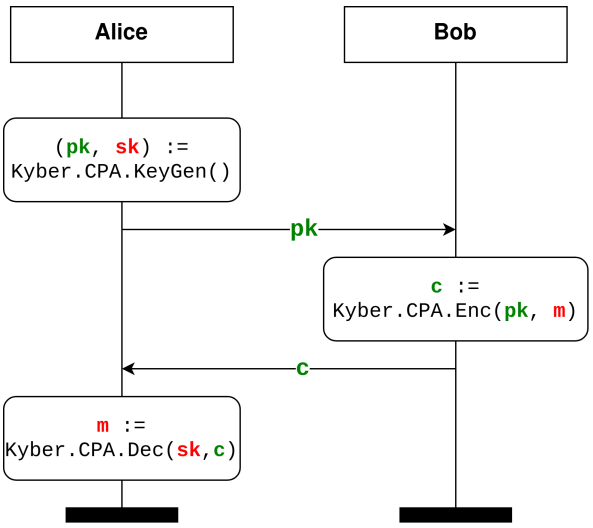


Figure 2.2: IND-CPA Kyber PKE

values result in more least-significant bits being discarded.

The final parameter δ refers to the correctness probability of Kyber. Kyber has a small probability of failing. Failure is defined as being the case where both parties obtain different keys. When parameters are chosen correctly the probability of this happening is so small that the failure probability is considered negligible.

2.1.5 The IND-CPA Kyber PKE algorithm

IND-CPA Kyber can be considered to be PKE. A secret message is sent from one party to another using asymmetric cryptography. This scheme requires three main steps. Key generation, encryption, and decryption. For illustration purposes say Alice wants to send a secret message $m \in M = \{0,1\}^{256}$ to Bob. Kyber achieves this with three main steps. Key-pair generation, encryption, and decryption. In the key-pair generation step, Bob will generate a public-key pk and a private-key sk. Then Bob will publish the public-key pk and keep the secret-key secret. In the second step, Alice will use Bob's public-key to encrypt the secret message m and create ciphertext c. Alice will publish the ciphertext. Finally, in the third and last step, Bob will use his secret-key to decrypt the previously obtained ciphertext and obtain the original message m. In Kyber these three steps are done by the algorithms `Kyber.CPA.KeyGen`, `Kyber.CPA.Enc` and `Kyber.CPA.Dec` respectively. The following sections will introduce and explain each of the previously mentioned algorithms. Section 2.1.6 will give an intuition and general overview of how the encryption and decryption procedures work. Figure 2.2 shows the full encryption scheme and how each of the three algorithms should be applied.

IND-CPA Key-pair Generation Algorithm 2 lists the steps required for key generation. The result is a public-key pk and a secret-key sk. Firstly, ρ and σ are bit-strings that are uniformly sampled. These are used as inputs for an extendable output function (XOF) that is used to generate pseudo-random bit-strings of any desired length. To denote that we would like to use an XOF to sample a y from a distribution S or uniformly from a set S we use $y \sim S := Sam(x)$. It should be noted that the function $y = Sam(x)$ is deterministic and always produces the same y for a given x.

$\mathbf{A} \sim R_q^{k \times k} := Sam(\rho)$ denotes that $\mathbf{A}$ is sampled uniformly from the set $R_q^{k \times k}$ with the use of the XOF $Sam(\rho)$. $\mathbf{A}$ is part of the public-key $pk = (\mathbf{A}, \mathbf{t})$. Since $Sam()$ is deterministic, we can instead send ρ to the other party. The other party can then perform $Sam(\rho)$ to generate the $\mathbf{A}$. Effectively reducing the public-key to $pk = (\mathbf{t}, \rho)$. This is an optimization done to reduce key size.

$v \leftarrow \beta_\eta$ denotes that v is generated from a distribution where all coefficients are sampled from β_η. Similarly $\mathbf{v} \sim \beta_\eta^k := Sam(\alpha)$ denotes a k-dimensional polynomial vector $\mathbf{v}$ where every polynomial is generated by β_η. Both $\mathbf{t}$ and $\mathbf{e_1}$ are generated this way. The error term $\mathbf{e}$ is the first error (or alternatively thought of as noise) that forms an integral part of the LWE problem. It can be seen that the algorithm relies heavily on randomness. Every implementation must make sure that the (pseudo-random) generation of elements does not introduce any weaknesses that can be leveraged by an attacker.

Algorithm 2 `Kyber.CPA.KeyGen()`: key generation

1: $\rho, \sigma \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} \sim R_q^{k \times k} := Sam(\rho)$
3: $(\mathbf{s}, \mathbf{e_1}) \sim \beta_{\eta_1}^k \times \beta_{\eta_1}^k := Sam(\sigma)$
4: $\mathbf{t} := \mathbf{As} + \mathbf{e_1}$
5: **return** $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

IND-CPA Encryption Algorithm 3 lists the steps required for encryption. This requires that Alice received the public-key pk from Bob. Alice will encrypt the secret message m into ciphertext c using Algorithm 3. Firstly, the exact same $\mathbf{A}$ is generated as in Algorithm 2. Secondly, three random variables are sampled from the binomial distribution. Of which $\mathbf{r}$ and $\mathbf{e_2}$ are polynomial vectors. And a single polynomial e_3. Again we see two error terms being introduced into ciphertext, which is integral to the LWE problem. After this the two parts of the ciphertext $c = (\mathbf{u}, v)$ are created. Most importantly the message is encoded into v. This is done through

$\lceil \frac{q}{2} \rfloor \cdot m$, where $\lceil x \rfloor$ is rounding. Every coefficient v_i will have $\lceil \frac{q}{2} \rfloor$ added to it depending on whether $m_i = 1$. The actual message bits are hidden because v also has $\mathbf{t}^T\mathbf{r}$ and e_3 added to it. Finally, Both parts of the ciphertext will be compressed with different compression parameters d_u and d_v.

Algorithm 3 `Kyber.CPA.Enc`$(pk = (\mathbf{t}, \rho),\ m \in \{0,1\}^{256})$: encryption

1: $r \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} \sim R_q^{k \times k} := Sam(\rho)$
3: $(\mathbf{r}, \mathbf{e_2}, e_3) \sim \beta_{\eta_2}^k \times \beta_{\eta_2}^k \times \beta_{\eta_2} := Sam(r)$
4: $\mathbf{u} := Compress_q(\mathbf{A}^T\mathbf{r} + \mathbf{e_2}, d_u)$
5: $v := Compress_q(\mathbf{t}^T\mathbf{r} + e_3 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
6: **return** $c := (\mathbf{u}, v)$

IND-CPA Decryption Algorithm 4 shows the steps for decryption. Firstly the two parts of the ciphertext are decompressed with the same decompression parameters d_u and d_v. Afterwards the secret-key of Bob $\mathbf{s}$ is combined with the two parts of the ciphertext. Finally, the result is decompressed with parameter 1. This is a trick that will convert every coefficient closer to $\lceil \frac{q}{2} \rfloor$ than 0 to a 1. It can be seen that $m = Compress_q(\lceil \frac{q}{2} \rfloor \cdot m, 1)$, since $Compress_q(\lceil \frac{q}{2} \rfloor \cdot m, 1) = \lceil \frac{2}{q} \rfloor \cdot \lceil \frac{q}{2} \rfloor \cdot m \mod^+ 2^d \approx m$. The result r of $v - \mathbf{s^T u}$ contains $\lceil \frac{q}{2} \rfloor \cdot m$ with some additional error. Since this error is small, the actual coefficient values r_i will be close to 0 if $m_i = 0$ and close to $\lceil \frac{q}{2} \rfloor$ if $m_i = 1$. Hence after the final compression step, we will get our original message m back. Section 2.1.6 gives a general explanation of how the encryption and decryption step will result in the original message.

Algorithm 4 `Kyber.CPA.Dec`$(sk = \mathbf{s}, c = (\mathbf{u}, v)$: decryption

1: $\mathbf{u} := Decompress_q(\mathbf{u}, d_u)$
2: $v := Decompress_q(v, d_v)$
3: **return** $Compress_q(v - \mathbf{s^T u}, 1)$

2.1.6 Overview of Kyber Decryption and Encryption

Figure 2.3 shows how the key-pair should be generated for Kyber512. The secret-key and error term are generated from the binomial distribution $\beta_{\eta_1}^k$. In the case of Kyber512, we have $\eta_1 = 3$ and $k = 2$ as can be seen in the parameter set listed in table 2.1. Therefore, we know that the coefficients of the polynomials in $\mathbf{s}$ and $\mathbf{e_1}$ belong to the range of $[-3, 3]$. The matrix $\mathbf{A}$ is uniformly sampled from the set $R_{3329}^{2\times 2}$. Matrix multiplication of $\mathbf{A}$ with the secret-key $\mathbf{s}$ and some added additive noise will

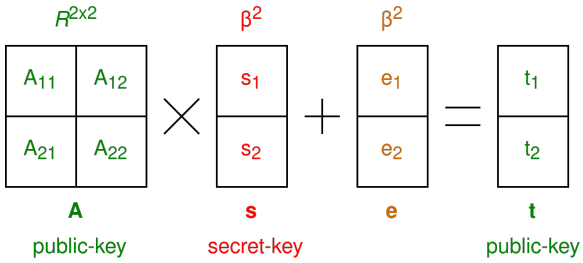
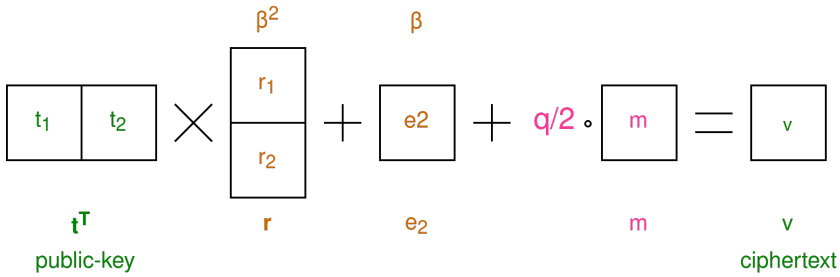
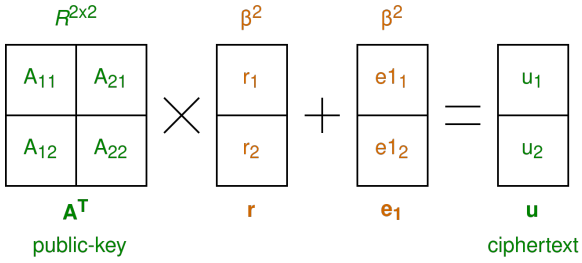


Figure 2.3: Key-pair Generation (IND-CPA Kyber512)



(a) ciphertext v



(b) ciphertext $\mathbf{u}$

Figure 2.4: Creation of ciphertext (IND-CPA Kyber512)

be sent as part of the public-key. Because of the added error $\mathbf{e}$, it is non-trivial to recover the secret-key $\mathbf{s}$ through $\mathbf{t}$ according to the LWE problem.

Bob can now create the ciphertext which consists of two parts, v and $\mathbf{u}$. The message bits will be encoded in v, whereas $\mathbf{u}$ is constructed in a specific way that aids Alice in recovering the original message. Figure 2.4 illustrates how Bob creates v and $\mathbf{u}$. Again the distributions are listed on top of the elements. Note that $\mathbf{A}$ and $\mathbf{t}$ have been provided by Alice. Again we can see the application of the LWE problem where noise is added to hide the elements $\mathbf{r}$ and m in the ciphertext.

After the ciphertext has been transmitted to Alice. Alice can use the secret-key $\mathbf{s}$ to retrieve the original message m. This is done in the final step of algorithm 4, where Alice will compute $v - \mathbf{s}^T\mathbf{u}$. Why this process results in something close to

the original message can be seen in the following equations:

$$\begin{aligned} v - \mathbf{s}^T \cdot \mathbf{u} &= (\mathbf{As} + \mathbf{e_1})^T \mathbf{r} + e_3 + \lceil \frac{q}{2} \rfloor \cdot m - \mathbf{s}^T (\mathbf{A}^T \mathbf{r} + \mathbf{e_2}) \\ &= (\mathbf{As})^T \mathbf{r} + \mathbf{e_1}^T \mathbf{r} + e_3 + \lceil \frac{q}{2} \rfloor \cdot m - \mathbf{s}^T \mathbf{A}^T \mathbf{r} - \mathbf{s}^T \mathbf{e_2} \\ &= (\mathbf{As})^T \mathbf{r} + \mathbf{e_1}^T \mathbf{r} + e_3 + \lceil \frac{q}{2} \rfloor \cdot m - (\mathbf{As})^T \mathbf{r} - \mathbf{s}^T \mathbf{e_2} \\ &= \mathbf{e_1}^T \mathbf{r} + e_3 + \lceil \frac{q}{2} \rfloor \cdot m - \mathbf{s}^T \mathbf{e_2} \\ &\approx \lceil \frac{q}{2} \rfloor \cdot m \end{aligned}$$

The final result is: $\mathbf{e_1}^T \mathbf{r} + e_3 + \lceil \frac{q}{2} \rfloor \cdot m - \mathbf{s}^T \mathbf{e_2}$. Since $\mathbf{s}, \mathbf{r}, \mathbf{e_1}, \mathbf{e_2}$, and e_3 are all taken from the binomial distribution we know that they are small. Therefore, the final result will be $\lceil \frac{q}{2} \rfloor \cdot m$ with some additional noise. Alice can now find the bits of message m by checking whether the coefficients are close to zero or close to $\lceil \frac{q}{2} \rfloor$. The values of the corresponding bits will be 0 and 1 respectively.

2.1.7 The IND-CCA Kyber KEM algorithm

IND-CPA Kyber is a PKE with IND-CPA notion of security. It would be preferable for Kyber to have stronger security guarantees. The FO-transform facilitates the lifting of IND-CPA to IND-CCA. The authors of Kyber apply this transformation to the IND-CPA version of Kyber to obtain the IND-CCA version of Kyber. They use a variant of the FO-transform which also transforms Kyber from a PKE to a KEM. This means that IND-CCA cannot be used to send secret messages. Instead, it is a method of obtaining a shared secret-key between two parties. From now on IND-CCA Kyber will be referred to as Kyber.

IND-CCA Kyber makes use of algorithms 5 and 6. Key generation is still done using algorithm 2 with the only difference being that the secret-key sk also includes the public-key pk and a secret random seed z.

IND-CCA Key Encapsulation Key Encapsulation is the process of forming the ciphertext with the secret-key hidden in the ciphertext. Algorithm 5 shows the steps required for encapsulation. We also require two hash function $\mathbf{G} : \{0,1\}^* \rightarrow \{0,1\}^{2\times 256}$ and $\mathbf{H} : \{0,1\}^* \rightarrow \{0,1\}^{256}$. Contrary to algorithm 3, Bob now has to generate a random message m instead of providing m as input. Firstly, Bob will generate two hashes $\hat{K}$ and r. The r will be used as a random seed when calling the `Kyber.CPA.Enc` function in the next step. In the next step, Bob will generate the ciphertext c using the public-key and randomly generated message m. Finally, he

can generate the shared secret-key K by hashing $\hat{K}$ together with a hash of the ciphertext c.

Algorithm 5 `Kyber.Encaps`$(pk = (\mathbf{t}, \rho))$: encapsulation

$m \leftarrow \{0,1\}^{256}$
$(\hat{K}, r) := \mathbf{G}(\mathbf{H}(pk), m)$
$(\mathbf{u}, v) := \texttt{Kyber.CPA.Enc}((\mathbf{t}, \rho), m; r)$
$c := (\mathbf{u}, v)$
$K := \mathbf{H}(\hat{K}, \mathbf{H}(c))$
return (c, K)

IND-CCA Key Decapsulation Algorithm 6 shows the decapsulation procedure for Kyber. The most notable change to the input of the function is that the secret-key besides $\mathbf{s}$ now also includes the public-key $pk = (\mathbf{t}, \rho)$. Additionally, the secret-key also contains a secret random seed z. Furthermore, the same hashes $\mathbf{G}$ and $\mathbf{H}$ are used. The decapsulation procedure decrypts the ciphertext and re-encrypts it. This re-encryption is to verify that the ciphertext is valid. If re-encryption fails then the algorithm returns a pseudo-random key, which has been generated using z. If re-encryption succeeds Alice can now apply the final hash $\mathbf{H}$ on a hash of the ciphertext $\mathbf{H}(c)$ and $\hat{K}$ to obtain the shared secret-key K.

Algorithm 6 `Kyber.Decaps`$(sk = (\mathbf{s}, z, \mathbf{t}, \rho),\ c = (\mathbf{u}, v))$: decapsulation

$m' := \texttt{Kyber.CPA.Dec}(\mathbf{s}, (\mathbf{u}, v))$
$(\hat{K}', r') := \mathbf{G}(\mathbf{H}(pk), m')$
$(\mathbf{u}', v') := \texttt{Kyber.CPA.Enc}((\mathbf{t}, \rho), m'; r')$
if $(\mathbf{u}', v') = (\mathbf{u}, v)$ **then**
    return $K := \mathbf{H}(\hat{K}', \mathbf{H}(c))$
else
    return $K := \mathbf{H}(z, \mathbf{H}(c))$
end if

With the above KEM, it is possible to create multiple protocols that facilitate key exchange. One can construct an unauthenticated, one-sided authenticated, and authenticated key exchange protocol.

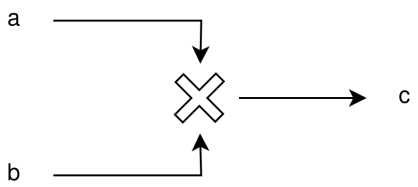
2.1.8 Fast Polynomial Multiplication using the Number Theoretic Transform

Kyber requires the multiplication of polynomials in the polynomial ring R_{3329}. After two polynomials have been multiplied they will be reduced to a polynomial of maximally degree 255 by the modulus $X^{256}+1$. Normally, polynomial multiplication is a costly operation. A naive implementation of polynomial multiplication would be $\mathcal{O}(n^2)$, where n is the degree of the polynomials. Given that Kyber needs to multiply multiple polynomials of 255-degree this is not desirable. Fortunately, under special circumstances, it is possible to significantly speed up polynomial multiplication using the number-theoretic transform (NTT). The NTT can be applied to polynomials to convert a polynomial to the NTT domain. Polynomial multiplication in the NTT domain can be done point-wise and is thus much faster than regular polynomial multiplication. When NTT is applied the complexity is reduced to $\mathcal{O}(n \cdot \log n)$.Kyber's parameters have been chosen in such a way that it facilitates the NTT operation. Specifically for this reason the choice of the prime modulus $q = 3329$ has been made. As it is a suitable candidate for the NTT. Note that the NTT for Kyber is a little bit more involved because of the choice for $q = 3329$. The NTT can only be applied partially and the multiplication of polynomials is not exactly point-wise. The specifics of the mathematics for NTT are omitted from this thesis. Please refer to [26] [27] for introductory reading on how to apply the NTT for polynomial multiplication.

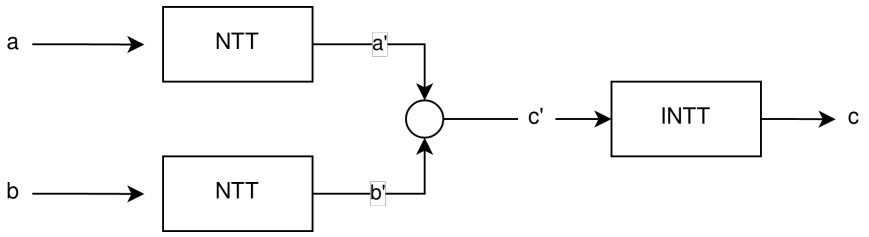
$$NTT(a) \rightarrow \hat{a} \tag{2.1}$$

$$NTT^{-1}(\hat{a}) \rightarrow a \tag{2.2}$$

Intuitively the NTT can be thought of as a discrete integer version of the Fast Fourier Transform. Say we want to apply the NTT on 255-degree polynomials (256 coefficients). By doing this we can convert a single polynomial a into 256 0-degree polynomials. It is also possible to think of the 256 0-degree polynomials as a single 255-degree polynomial in the NTT domain. We will denote a as a polynomial in the NTT domain by $\hat{a}$. If we do this for two polynomials, we will have two sets of 256 0-degree polynomials. Now that we have converted both polynomials to the NTT domain we can multiply them together. To multiply two 0-degree polynomials together we only need to do a single multiplication. We can do this for every pair. This only requires 256 operations. Finally, we can apply the inverse NTT operation to get back the result of the multiplication of the two original polynomials. Figure 2.5 gives an overview of how two polynomials a and b can be multiplied into c using the NTT. The process is akin to how the Chinese Remainder Theorem can be used to form a Residue Number System. Where one can operate on smaller elements



(a) Polynomial multiplication $\mathcal{O}(n^2)$



(b) Polynomial multiplication using NTT and point-wise operator $\mathcal{O}(n \cdot \log n)$

Figure 2.5: Multiplication of polynomials

to perform a computation on a bigger element. In this case, we can multiply many smaller-degree polynomials in order to multiply two larger-degree polynomials.

2.1.9 Implementations

Currently, there are multiple Kyber software implementations available for study. The PQC library PQCLEAN [28] offers multiple clean implementations for Kyber. One notable implementation is a clean reference implementation that is not architecture specific. Additionally, PQCLEAN contains implementations that are optimized for specific ISAs. Another PQC library called PQM4 [29] also provided PQC implementations, including Kyber. These implementations have been specifically optimized for the ARM Cortex-M4 embedded processor.

2.2 Side-Channel Analysis

A lot of attention is given to ensuring that cryptosystems are hard to break theoretically. Messages and secret keys should be hidden from possible observers. At some point, practical cryptosystems need to be implemented on physical hardware. If the design of an algorithm or cryptographic protocol is flawless a specific implementation still can cause security issues. Hardware processing data can reveal possibly secret information through data-dependent side-channels. Examples of possible side-channels are *power consumption, electromagnetic radiation, timings, optics, thermal radiation, and acoustics*. The field of study called side-channel

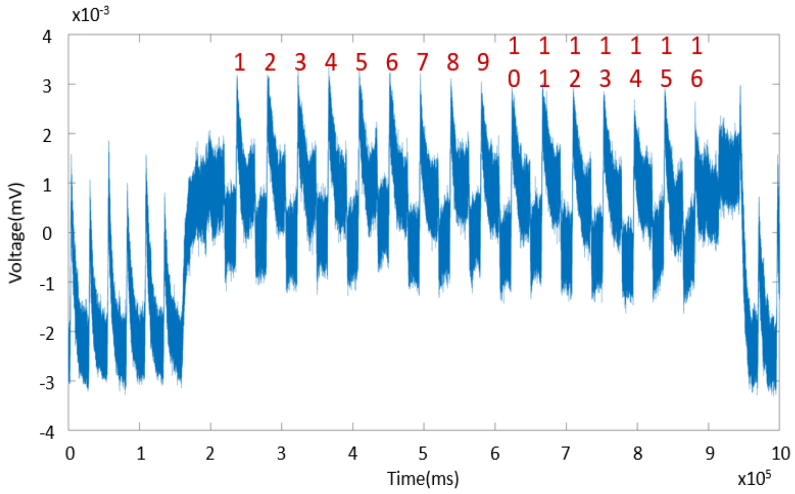


Figure 2.6: Power trace showing 16 DES rounds (CC 4.0)

analysis (SCA) focuses on the study of these side-channels. Using these channels a possible attacker could perform a SCA attack, where side-channels are used to retrieve secret information. Figure 2.6 shows a power trace of the DES encryption process. The 16 rounds of the algorithm are very clearly visible in the power trace. One can imagine that in some cases this might leak information about the data being processed. SCA attacks are not the only form of attacks that can be used to compromise implementations. One can also do invasive attacks by physically disturbing the device. These attacks are referred to as fault-injection attacks. Examples of fault-injections are clock glitching and disturbing the supply voltage. A fault-injection may cause the device to behave in unexpected unintended ways. One example is an instruction jump, which can be used to skip a conditional check. In some cases this allows an attacker to bypass security checks. Side-channel attacks often require physical access to the device in question. Historically, side-channels were not really considered when building cryptosystems. Side-channel attacks have become increasingly more viable in the last couple of decades. Firstly, the rise of IoT has made side-channels increasingly more accessible. Secondly, increases in computing power led to the development of stronger SCA techniques.

In this thesis, only power side-channel analysis is considered. Power side-channel analysis focuses on the power consumption of a device to retrieve secret information. Section 2.2.1 gives a small overview of the developments in the field of side-channel analysis. Section 2.2.2 will introduce different power analysis techniques that can be applied to attack a victim. Finally, section 2.2.3 will introduce countermeasures that can be applied to mitigate SCA attacks.

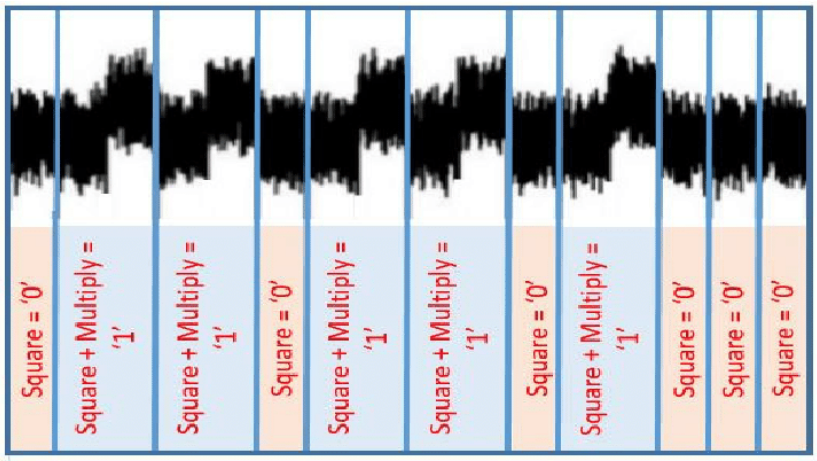


Figure 2.7: Modular exponentiation using square and multiply (CC 4.0)

2.2.1 Side-Channel Analysis History

In 1996 Kocher showed that it was possible to break cryptographic implementations using a timing attack [30]. By measuring the power consumption and inspecting how long it takes to perform the modular exponentiation algorithm. The trace in figure 2.7 clearly shows a difference in the trace between the square operation and the square-and-multiply operation. This observable difference leaks information about the data being processed. The reason for this is that the operation used depends on the bits used for input. Thus by observing this behaviour, it is possible to retrieve bits of the secret input. This example is both a timing attack and a simple power analysis attack. This vulnerability can be easily removed by using a constant-time algorithm.

A possible attack scenario is shown in figure 2.8. There is an embedded device with a secret-key stored on it. When operating the device causes observable changes in side-channels. Possible examples of observable side-channels are power consumption, electromagnetic radiation, acoustics, and temperature. The attacker can observe these side-channels using a measurement device. If the device leaks secret information through these side-channels the attacker might be able to obtain the secret-key. In this scenario, the attacker is also able to provide input and observe the output of the device (this is not always the case). Being able to provide input and observe output can be very valuable for an attacker. The main reason for this is that it becomes easier for the attacker to correlate the side-channel observations to the processed data. Possible examples of inputs in the case of a PKE implementation is the plaintext or ciphertext. Attacks utilizing this are referred to as chosen-ciphertext side-channel analysis and chosen-plaintext side-channel analysis attacks respectively.

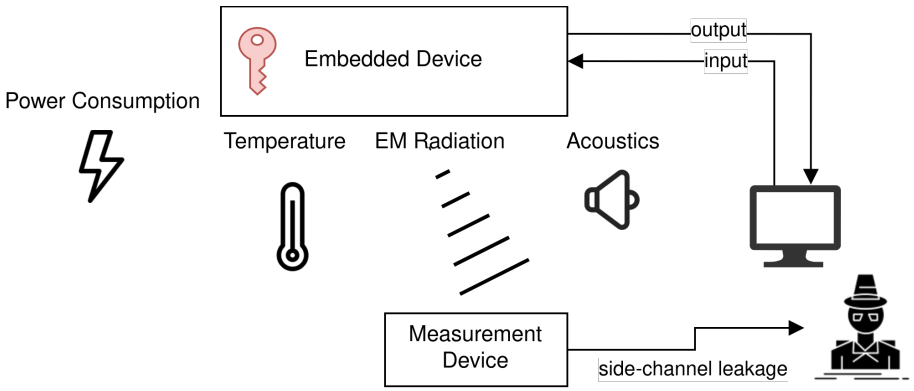


Figure 2.8: SCA attack scenario

2.2.2 Power Side-Channel Analysis Techniques

By placing a resistor in series with the power or ground input it is possible to measure the power consumption of a device. The voltage drop across the resistor can be measured with an oscilloscope and is directly related to the power consumption of the device. Through analysis of this power consumption, it is possible to retrieve secret information. This technique is referred to as power side-channel analysis. This form of analysis can be done with equipment that is relatively inexpensive. The biggest downside is that an attacker needs to be able to obtain power measurements. In most cases, this requires physical access to the device in question.

Simple Power Analysis The most simple form of power analysis is SPA. It involves visual inspection of traces and extracting information based on the variance in power consumption. In most cases, simple power analysis is not powerful enough to perform an attack. The variations in power consumption tend to be very small and often do not result in obvious visual differences. The main reason that simple power analysis is possible in figure 2.7 is because of the difference in execution time and power consumption between the square and square multiply operation.

Differential Power Analysis In 1998 Kocher introduced differential power analysis (DPA) [31], which significantly improved the capabilities of side-channel power analysis attacks. Kocher showed that it is possible to break DES using DPA, where SPA would not have been sufficient. Power traces often have too much noise or too little variation in power consumption to make SPA viable. DPA uses a clever technique which highlights the differences and reduces the noise. The main goal of DPA is two bin traces into two classes and compare their statistical differences. DPA requires a selection function that often resembles something like $D(x, k)$, where x is some observable variable and k is a secret-key guess. A selective function is also often

referred to as a leakage model. DPA is a form of a model-based side-channel analysis technique. The selection function will return 0 or 1 to indicate to which class a trace belongs. The selection function needs to be selected carefully according to the algorithm that is being analyzed. The main goal is to bin the traces into classes with different power consumption characteristics. For example, if x is chosen such that it refers to a specific output bit. Then on average, we will expect the class with $x = 1$ to have larger power consumption than the class where $x = 0$. If we correctly bin our traces into the correct class then we can expect the average of one class to have higher power consumption than the other class. Alternatively, if we bin the traces randomly with probability $\frac{1}{2}$ then we can expect the average power consumption of the classes to be approximately equal. And the difference between the averages will tend towards zero. Say we have a collection of m power traces $\mathbf{T}_i$. We also have a leakage model $D(x, k)$ which selects whether a trace belongs to one of two classes. With one class having different power consumption characteristics than the other class. Next, we can compute the differential trace Δ_D at point j by using equation 2.3. If our guess of k is incorrect and the traces are distributed randomly across the classes, then the difference between the averages of the two classes will tend towards zero. Conversely, if our guess of k is correct, then one class will exhibit significantly different power consumption behaviour, leading to a substantial observable difference.

$$\Delta_D[j] = \frac{\sum_{i=1}^{m} D(x,k)\mathbf{T}_i[j]}{D(x,k)} - \frac{\sum_{i=1}^{m} 1 - D(x,k)\mathbf{T}_i[j]}{1 - D(x,k)} \tag{2.3}$$

In equation 2.3 the mean of the power consumption of the two classes is used as a comparison. One can also opt to use a different statistic such as the standard deviation in order to perform DPA [7]. Figure 2.9 shows an example of DPA. The first plot is the average power consumption across the DES operation. The other three plots are differential traces of Δ_D for different guesses of k. The first one has significant spikes and corresponds to the correct guess of k.

Correlation Power Analysis The introduction of DPA led to further developments in the field of SCA. In 2004 Brier introduced a powerful technique called CPA [32]. CPA is a model-based SCA that uses the Pearson correlation coefficient (PCC) to correlate the power consumption with the data that is being processed. The PCC is a normalized version of the covariance. The covariance is a statistic that is a measure of the linear relation between two joint variables. PCC is simply a normalization of the covariance with a range between negative one and positive one. In CPA one correlates a leakage model to the power consumption. The leakage model requires a guess of the secret key or subkey. If both the leakage model and the guess

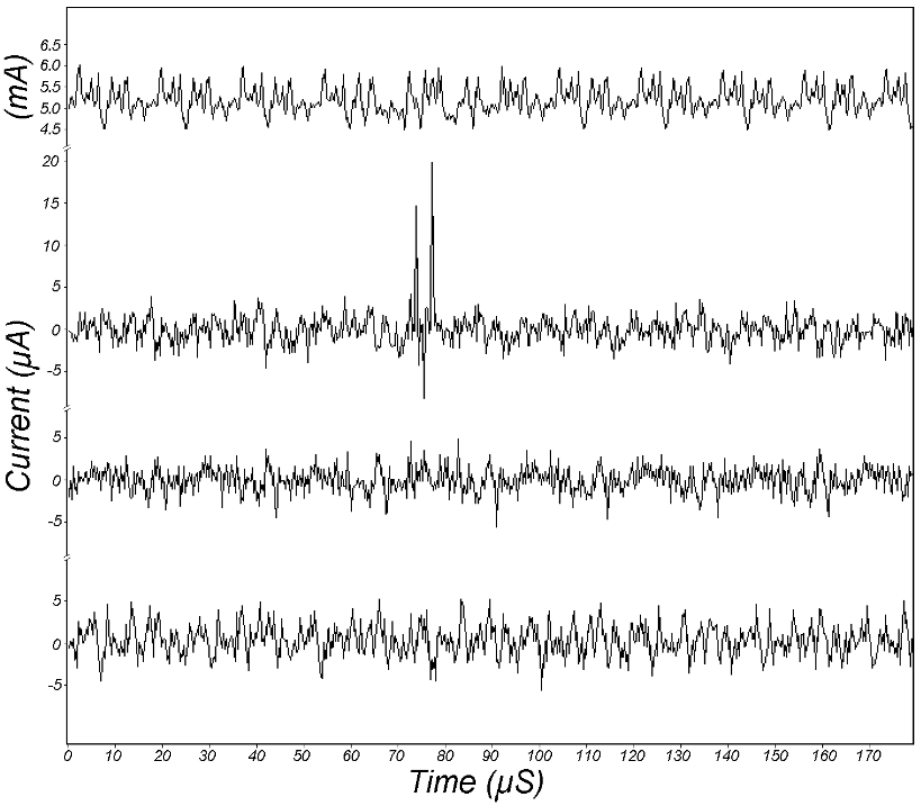


Figure 2.9: Example of DPA

are correct we can expect to see a high PCC value. If the guess is incorrect our PCC should be significantly lower. Using this we can find the true subkey value by finding the guess, which results in the highest PCC value. The leakage model often uses the hamming weight (or hamming distance) of an intermediate variable. The hamming distance is a measure of the difference between two bit-strings. The hamming weight is a measure of the number of ones in a bit-string. The definitions for hamming distance and hamming weight can be found in definition 2.2.1 and definition 2.2.2 respectively. Say we have a collection of m power traces $\mathbf{T}_i$. Every trace has an associated intermediate variable x_i. Furthermore, we have a leakage model $L(x, k)$ for the intermediate variable x and secret key guess k. We can use the following equation to estimate the PCC at time point j:

$$\rho_T(k)[j] = \frac{m \sum_{i=1}^{m} \mathbf{T}_i[j] L(x_i, k) - \sum_{i=1}^{m} \mathbf{T}_i[j] \sum_{i=1}^{m} L(x_i, k)}{\sqrt{m \sum_{i=1}^{m} \mathbf{T}_i^2[j] - (\sum_{i=1}^{m} \mathbf{T}_i[j])^2} \sqrt{m \sum_{i=1}^{m} L(x_i, k)^2 - (\sum_{i=1}^{m} L(x_i, k))^2}} \tag{2.4}$$

To find the most likely guess of k we can take the absolute maximum of $\rho_T(k)$ across all time points for every guess k. The k guess, which corresponds to the highest absolute maximum will be the most likely value for the secret key. In practice, CPA has shown itself to be an effective SCA technique.

Definition 2.2.1 (Hamming Distance) *The Hamming Distance $HD(a, b)$ between two equal lengths bit-strings a and b is the number of positions at which the bits differ.*

Definition 2.2.2 (Hamming Weight) *The Hamming Weight $HW(a)$ of a bit-string a is the Hamming Distance between a and the equal length zero bit-string: $HW(a) = HD(a, 0)$.*

The previous two techniques fall into the category of model-based side-channel analysis. Multiple power traces are required to find a correlation in the measurement. An alternative technique is to use profiled side-channel analysis. In profile-based techniques, a model is built using power trace measurements from a clone device. The advantage of profiled techniques is that in some cases they only require a single trace in order to extract secret information. Yet to do so, often millions of traces are required in advance to build the reference model. Furthermore, access to a clone device is required.

Template Attacks In 2003 Chari introduced a novel attack method named template attacks. Template attacks first make a template for every possible key (or subkey). This step is called profiling. To make a template many traces are required. For every key, we need a significant amount of traces to make sure that our templates are statistically sound. Commonly multivariate Gaussian distributions are used as templates. These distributions have an associated probability density function that indicates how likely it is to measure a specific value. Many traces for every key are measured and used to estimate the corresponding multivariate Gaussian distribution. If we are then given a trace for which we would like to know the secret key, then using the probability density functions it is possible to find the most likely distribution. The key corresponding to this distribution will be the most likely key for the given trace. Many adaptations of template attacks exist.

Please note that this was not an exhaustive list of all non-invasive side-channel analysis techniques. Only the most common SCA techniques have been covered. In the last decades, the field of side-channel analysis has been growing. There is constant development of side-channel analysis techniques. One can also incorporate machine learning for more efficient side-channel analysis or construct attacks that are a hybrid of the techniques listed above. In general, most other analysis techniques have their foundations based on one or more of the four attack techniques listed above.

2.2.3 Countermeasures

Side-channel attacks can form a serious threat to the digital domain. Countermeasures need to be developed in order to combat side-channel attacks. The aim of a countermeasure is to make it significantly more difficult to do SCA. Multiple different strategies exist to make SCA infeasible or impossible. Generally, two strategies mitigation strategies exist. The first approach involves the reduction and elimination of valuable leakage. This can be achieved by introducing more noise in the power consumption, for instance. The second approach is to remove the relation between secret information and the leakage. Countermeasures can be implemented both in software and hardware.

Noise generators Noise generators aim to increase the signal-to-noise ratio. Since noise is normally distributed, the average of many traces will have a constant noise factor. One can, therefore artificially increase the signal-to-noise ratio by capturing and averaging many traces. Noise generators should aim to make the number of required captures infeasible.

De-correlation Both DPA and CPA try to obtain information by comparing different traces. These attacks require that the traces are correlated in some way. For example, by executing the same operation at a specific time step. De-correlation tries to remove this correlation. A common method of de-correlation is by removing the synchronization between traces. One method of achieving this is through the insertion of random delays, for instance.

Masking Masking is an effective, but resource-costly countermeasure technique [33]. In masking the inputs are split into different shares. These shares are statistically independent of the original inputs. The algorithm is now executed on the different statistically independent shares. Finally, the shares can be combined again into a single result. This result should be equivalent to the result that would have been obtained with an unmasked implementation. Masking works, because it removes the correlation between the actual input values and the leakage of intermediate computations. Multiple different masking techniques exist depending on the computations that are being done. Different techniques are necessary, one technique might only work for a specific subset of operations. One possible technique is boolean masking works. Boolean masking works in the following manner. Say we have a bit x and would like to split this bit into two shares. We can compute $x_r = x \oplus r$, where r is a random bit value. Since r is random x_r will be statistically independent from x. We can now operate function f on x_r and r separately. One requirement is that f

is linear with respect to the $\oplus$ operator. So $\forall_{a,b} : f(a) \oplus f(b) = f(a \oplus b)$. Using this method one can compute $f(x)$ by computing $f(x_r) \oplus f(r)$. With the key property that leakage from $f(x_r)$ and $f(r)$ is statistically independent of x. Besides boolean masking, there are also other masking techniques. Another common masking technique is arithmetic masking [34].

2.3 RISC-V

The ISA is an abstract model of how a processor should execute machine code. It is up to the developer to implement the ISA on actual hardware. This abstract model allows different processors to run the same machine code provided they adhere to the ISA. Different processors can be developed with different requirements in mind. RISC-V [35] is an open standard instruction set architecture specification following the reduced instruction set computer paradigm. The open-access format and customizability of RISC-V have attracted many interested parties and researchers. In the future, it is projected that there will be an increasing number of IoT and embedded devices utilizing the RISC-V ISA.

2.4 Related Work

After the publication of the Kyber algorithm [18], multiple studies have been conducted on the SCA properties of different implementations. Researchers identified multiple safety-critical operations, which could potentially be leveraged by an attacker. A range of different attack techniques have been proposed. From SPA [12] to profiled machine-learning based attacks [36] and even fault injection attacks [37], [38]. Generally, one can categorize the attacks on Kyber into three categories. Firstly, there is the message recovery attack. In a message recovery attack, the adversary tries to obtain the message (also known as plaintext) through the use of side-channel analysis. The advantage of message recovery attacks is that often no knowledge of the secret key is required. Message recovery attacks are often done in the message encoding or decoding step. Secondly, there are attacks that aim at recovering the stored secret-key. Often in a IoT setting a secret-key is permanently stored on a device. This makes it possible to recover the message after the secret-key is known. Thirdly, there are attacks that aim at attacking the IND-CCA notion of security.

Profiled Attacks The first SCA attack applicable on Kyber was already developed before Kyber's publication. Primas et al. targeted the NTT operation, which was

commonly used in most lattice-based cryptography algorithms [39], [40]. Their attack technique was a profiled form of SCA known as soft-analytical side-channel analysis (SASCA). An advantage of their attack is that only a single trace is required. Thus enabling attacks on applications of Kyber with ephemeral key setting. Other profile-based SCA attacks were developed that targeted different parts of Kyber such as the NTT, but also message encoding/decoding and modular operations [36], [41]–[43]. Profiled attacks tend to require a very high number of traces in advance and access to a clone device. Because of this, it might be preferable under certain circumstances to do a non-profiled SCA attack. Profiled attacks often use machine learning to train a model that is able to perform predictions on the underlying data using side-channel information.

Non-profiled Attacks Multiple non-profiled non-invasive SCA attacks have also been developed for Kyber [11], [13], [14], [44], [45]. Among these, Xu et al [12] showed that is possible to perform a SPA attack that utilized the electromagnetic (EM) side-channel [12]. Most of these attacks targeted correlations between the power consumption and intermediate variables which consisted of a combination of the ciphertext and secret-key. Five of the listed attacks use CPA to exploit these correlations to obtain the secret-key. These attacks tend to focus on critical operations that involve the secret-key in some manner. One example of this is the multiplication of the secret-key with part of the ciphertext.

Plaintext-checking Oracle Attacks A totally different SCA attack focuses on attacking the IND-CCA notion of security. These attacks try to attack the FO-transform, which aims to provide IND-CCA security. Multiple of these attacks has been proposed in literature [46]–[50]. These attacks use SCA to instantiate a plaintext-checking oracle. In the context of a PKE algorithm, a plaintext-checking oracle is given a message and ciphertext as input. The plaintext-checking oracle is able to verify that decryption of the ciphertext by the PKE algorithm will result in the provided message. The existence of a plaintext-checking oracle will break the IND-CCA security guarantee. Furthermore, using this oracle it is possible to perform a key-recovery plaintext-checking attack. This shows that it is also of the utmost importance to protect the FO-transform against SCA attacks.

Fault-injection Attacks All three categories above are forms of non-invasive SCA attacks. They do not physically interact with the hardware that is running the algorithm. There are also studies that research Kyber's resistance to invasive side-channel attacks. One common form of invasive side-channel attack is fault-injections. Here an attacker tries to make hardware behave in an unexpected way by physically

disturbing it. Examples of physical disturbances are glitching the clock or sending EM pulses to the device. Hermelink et al suggested a fault-injection attack that required a single bit-flip [37]. Doing a single bit-flip using a laser requires very expensive and specialized equipment. To this extent, Delvaux et al improved their attack by relaxing the attack constraints [38]. They also showed that the attack works in practice by attacking a masked implementation of Kyber with the use of clock glitching.

Of the previously mentioned different SCA attacks most of them focus on the *pqm4* [29] implementation of Kyber. The *pqm4* implementation is specifically optimized for the ARM Cortex-M4 processor, which is widely available. That being said, the *pqm4* implementation is not the only implementation that has been studied. Some attacks also target *pqclean* [51], which is a clean reference implementation that can be used as a reference for other implementations. Some attacks target implementations that contain software and hardware countermeasures. The previously listed results show that implementations of Kyber very much need to take SCA into account. Unprotected implementations have a high chance of being compromised due to the susceptibility to SCA. To make Kyber more robust it is paramount that countermeasures are implemented to prevent possible attacks.

Countermeasures Multiple countermeasures have been proposed for Kyber [52]–[57]. Possible countermeasures range from shuffling, randomization, and clock delays to masking. In related works, masking seems to be a popular choice for Kyber. This is because it gives good protection and can be implemented both in software and hardware. There is often a trade-off between the performance and security of countermeasures. Masking for example tends to decrease performance or increase resource utilization by a linear factor. When we consider IoT devices resources tend to be scarce, but high-security guarantees against SCA are very much required. The perfect countermeasures would require few resources, but provide high-security guarantees. Unfortunately, constructing such a countermeasure is not trivial. And future research is very much required to guarantee the robustness of applications utilizing Kyber as a KEM.

Most of this thesis is based on the previous work by Xu et al [12]. Their work showed that it is possible to attack both *pqclean* and *pqm4* implementations of Kyber using SPA of the EM side-channel. They found that it was almost trivial to break the *pqclean* implementation by visual inspection. This leakage however did not translate to the *pqm4* implementation. To break that implementation they moved to a different part of the Kyber algorithm and altered their strategy. The fact that it was possible

to break the *pqclean* through SPA alone makes one wonder about the robustness of the algorithm with respect to SCA. It would be interesting to see whether this behaviour is also visible on a Kyber implementation for a RISC-V microprocessor. Xu et al's work provide a nice foundation for a possible comparison between RISC-V and ARM implementations.

Chapter 3

Simple Power Analysis on Kyber

Xu et al were able to do a SPA attack on the *PQCLEAN* [51] implementation of Kyber using thresholding [12]. For specific ciphertexts, different key coefficients had different visual characteristics in the power trace. Their adversary model for their attack has the following requirements:

- Adversary is able to obtain EM traces of decryption
- Adversary is able to provide chosen ciphertexts
- Kyber *PQCLEAN* implementation is used
- Implementation uses static key setting
- Adversary can use visual inspection to identify secret-keys.

Xu's attack methodology was shown to be effective on an STM32F407. In the proof of concept, EM traces were captured at 2.5GHz and downsampled to 500MHz. Only four traces were required in order to capture a significant portion of the stored secret-key. In all cases, at least 99.6% of the secret-key coefficients were recovered for Kyber512.

The main observation was that the leakage from `fqmul` in the final step of the inverse NTT leaked the secret-key coefficients. Depending on the chosen ciphertext subkeys could be categorized into different partitions. For example, for one specific ciphertext, one could categorize use the leakage to categorize a subkey into the partitions of $\{-2, -1\}$, $\{0\}$, $\{1, 2\}$. For the first partition, this means that the subkey either is equal to -2 or -1. To be able to differentiate between -2 and -1 another trace is captured using a different chosen ciphertext. In the case of Kyber512, two traces reveals half of the stored secret-key coefficients. Another two captures are then required to recover the full-secret key.

This chapter investigates the same power side-channel leakage characteristics on a comparable ARM Cortex MCU (STM32F303) and a RISC-V based MCU (FE310-G002). Rather than analyzing EM leakages, the focus of this thesis is on the power side-channel. The EM and power side-channel leakage share a strong relationship, as they both arise from changes in electrical currents within a device. The EM side-channel can be considered more powerful than the power side-channel. This is because the EM side-channel corresponds to many different types of emanations that are caused by different components. Contrary to EM, the power side-channel only gives a net overview of the power consumption. That being said, obtaining useful EM measurements can pose challenges because of noise and spatial factors.

Section 3.1 introduces the experimental setup that has been used to perform the various experiments. Section 3.2 describes the first SPA study and corresponding results.

3.1 Experimental Setup

3.1.1 Hardware

In order to perform SCA one has to be able to execute an algorithm and obtain the corresponding side-channels measurements. The ChipWhisperer is a platform that facilitates performing power SCA and glitching attacks. To obtain the power consumption of a device a shunt resistor is inserted in serial with the power rail at the supply voltage or ground level. The voltage drop across the resistor is directly related to the power consumption of the device under test (DUT). All power traces of the different experiments were obtained using the ChipWhisperer-Lite capture board. ChipWhisperer provides a Python library, which allows for easy interfacing with the ChipWhisperer-Lite over universal serial bus (USB). In our experiments, a Raspberry Pi was used to interface with the ChipWhisperer. The ChipWhisperer-Lite utilizes a low-noise amplifier together with a 105MS/s 10-bit Analog-to-Digital Converter to obtain power measurements. The clock signal of the DUT is provided by the ChipWhisperer. This way power samples are synchronized with the operations that are being executed on the DUT. Sampling the power consumption can be done at a rate four times faster than the clock rate. This increases the time resolution of the power traces. In total, the ChipWhisperer-Lite has a sample buffer size of 24573 samples. This is a relatively small buffer size for time-consuming cryptographic operations. As a result of the small buffer, in some cases, more captures were required to capture the full procedure. The actual capturing can be initiated by

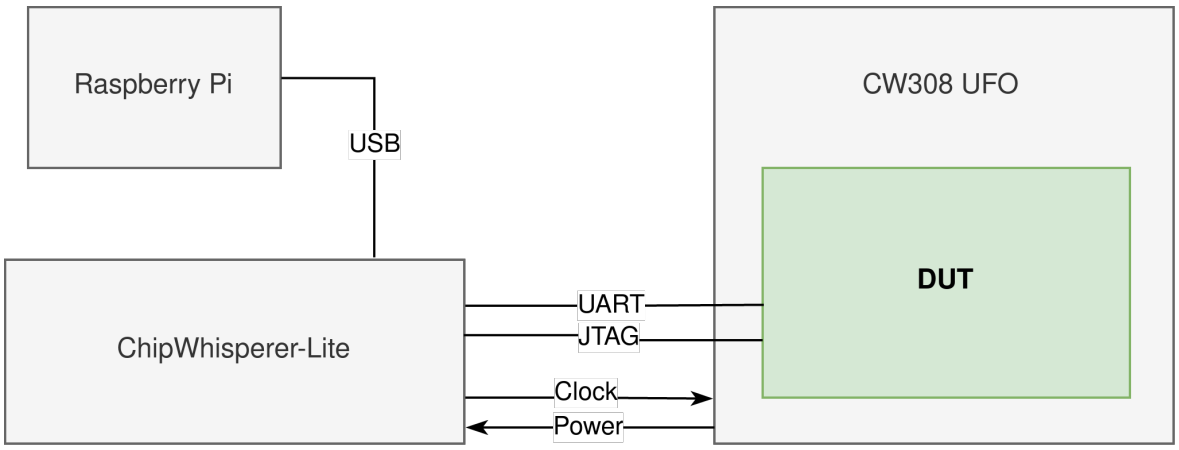


Figure 3.1: Experimental Setup

pulling dedicated trigger general purpose input-output (GPIO) pins to logical high on the target device. ChipWhisperer provides various target boards, which can be used as DUTs. The ChipWhisperer UFO (CW308) is a carrier board that is able to support various target boards. It has onboard voltage regulators and makes it possible to interface between the different target boards. Experimentation was done on two different targets. An ARM-based and a RISC-V based MCU. For a general overview of the setup see figure 3.1. Figure 3.2 and figure 3.3 show the ChipWhisperer-Lite and CW308 UFO respectively.

STM32F303 The STM32F303 is the first of the two targets used in the experiments. It is an ARM Cortex-M4 MCU from the popular STM32F series. The Cortex-M series dedicates itself to a wide range of embedded applications. The STM32F303 processor contains several hardware peripherals that allow for signal-processing applications. Another important feature is the floating-point unit (FPU), which allows efficient operations on floating-point numbers. The target is provided a clock signal of 7.37MHz and sampling is done at four times the clock rate. The processor also contains 256KB FLASH and 40KB SRAM memory. ChipWhisperer provides a dedicated STM32F303 target for the ChipWhisperer UFO. A neat feature is that the processor can be programmed over serial by the ChipWhisperer-Lite via a serial bootloader.

FE310-G002 The second target that has been used for our experiments, is the FE310-G002. ChipWhisperer provides a dedicated UFO version that allows for easy interfacing with the ChipWhisperer platform. RISC-V is an upcoming ISA. Currently, the availability of RISC-V boards is limited. One of the options is the FE310-G002. The FE310-G002 is one of the ASIC RISC-V boards available on the market. The

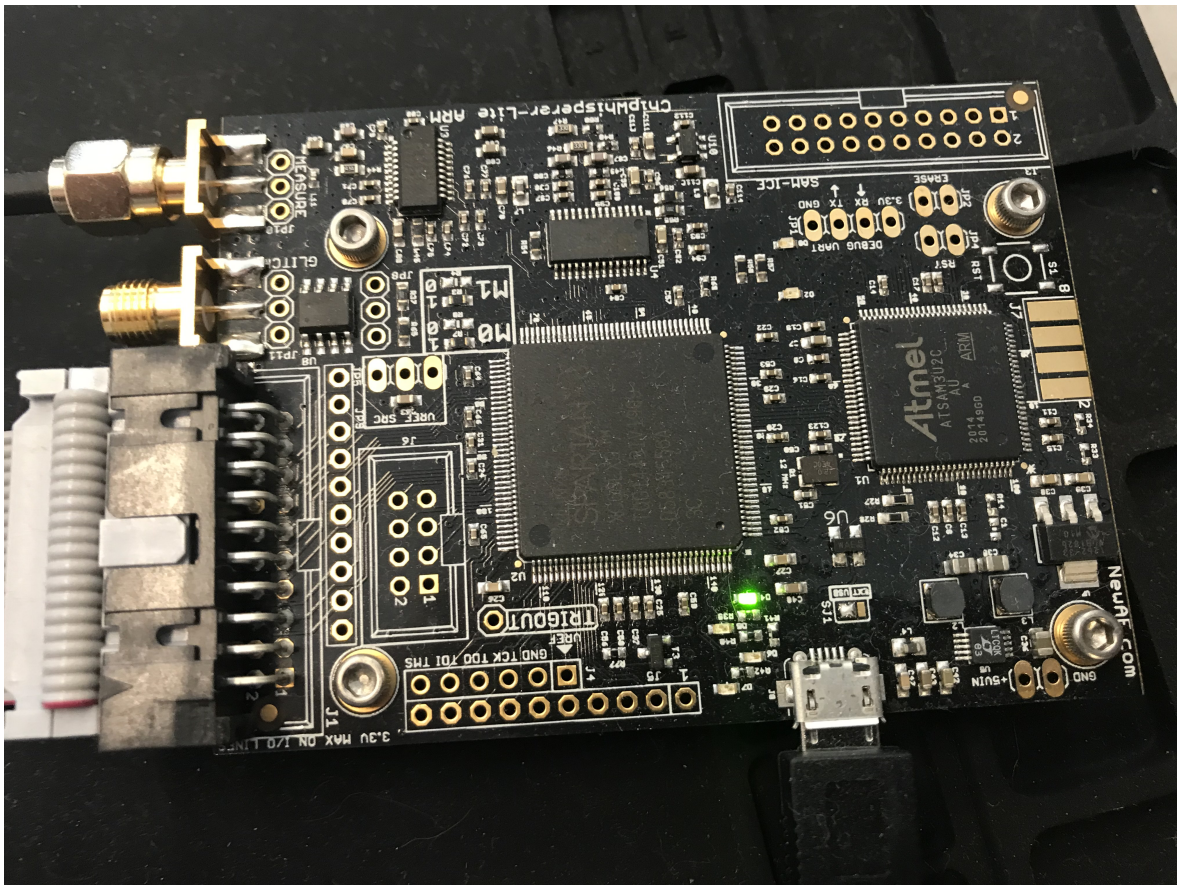


Figure 3.2: ChipWhisperer-Lite

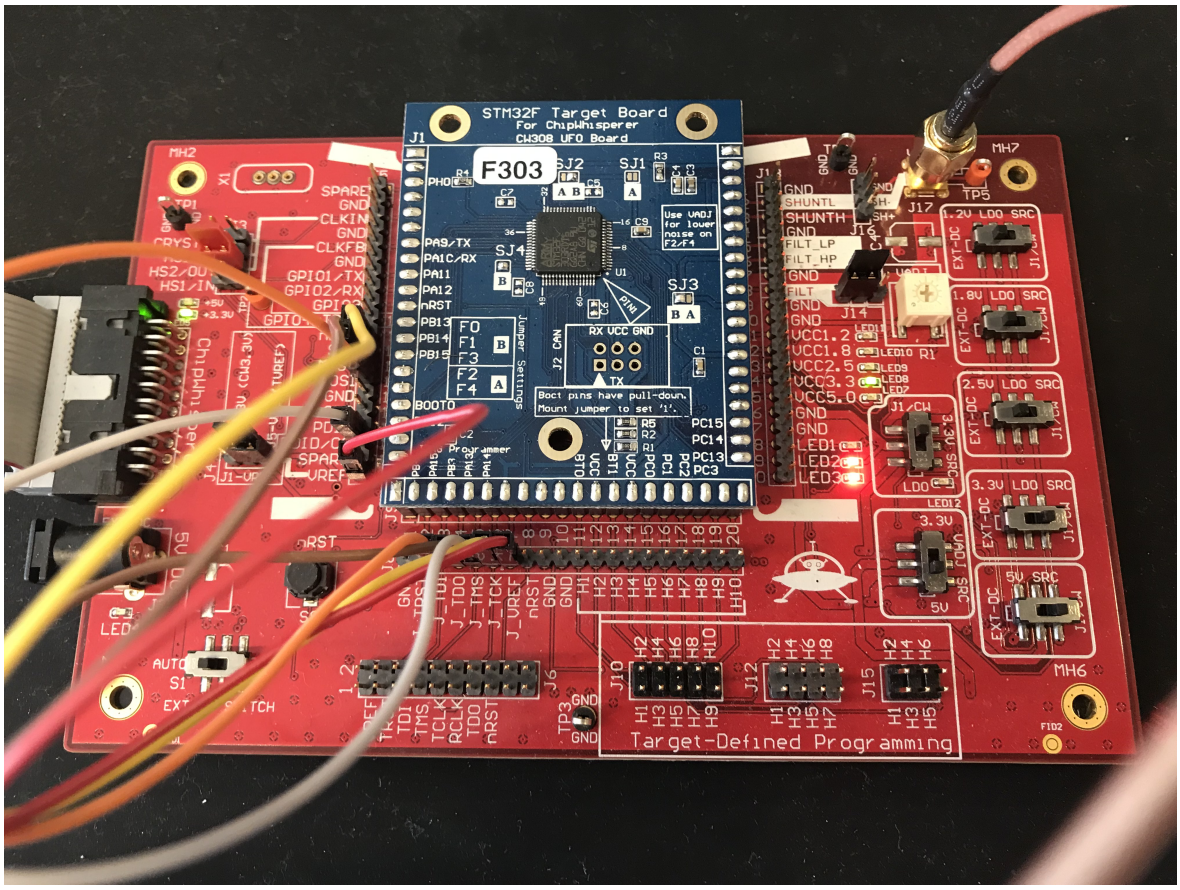


Figure 3.3: CW308 UFO with STM32F303 target

FE310-G002 is designed as a system-on-a-chip (SoC) and incorporates several hardware peripherals. As well as common serial interfaces (SPI, I2C, UART). The board is provided a clock signal of 7.37MHz. The target also has hardware support for multiplication and division. Contrary to the STM32F303, the FE310-G002 requires multiple cycles to perform multiplication operations. Additionally, it lacks a dedicated FPU. The FE310-G002 has 32MB of FLASH memory and 16KB of dedicated SRAM memory. This is a limited amount of RAM and memory usage must be managed carefully. The FLASH memory of the FE310-G002 can be programmed through JTAG. The ChipWhisperer-Lite offers a special MPSSE mode. This mode allows for direct communication with the FE310-G002 through OpenOCD. In our case, this mode can be used to communicate directly between the Raspberry Pi and FE310-G002 with OpenOCD. The Raspberry Pi uses OpenOCD to program the FE310-G002 directly with the ChipWhisperer-Lite acting as a bus.

3.1.2 Software

All experiments were done using the *PQCLEAN* library. The *PQCLEAN* library is a collection of clean reference implementations for various PQC schemes. Kyber is also included in the *PQCLEAN* library. There is also a PQC library that is specifically optimized for the ARM Cortex-M4 family, namely *PQM4* [29]. For all experiments, the *PQCLEAN* library has been used. This has been done to prevent significant software changes between experiments involving the STM32F303 and FE310-G002. That being said, the software between STM32F303 and FE310-G002 will differ significantly. Not only do the two targets have different hardware capabilities, but they also have different ISAs. Software for the ARM processor is compiled using `gcc-arm-none-eabi (15:9-2019-q4-0ubuntu1) 9.2.1`. The RISC-V software is compiled using
`riscv64-unknown-elf-gcc (SiFive GCC 8.2.0-2019.05.3)`. ChipWhisperer provides a communication protocol named SimpleSerial. This makes it possible to communicate with the target while it is running. The memory constraints of the targets make it impossible to fully fit Kyber in memory. In these cases, data is prepared externally on the Raspberry Pi and sent to the target with the SimpleSerial protocol. The critical section for which power traces should be captured can then be executed locally on the target using the prepared data.

3.2 Leakage of ordered key coefficients

This experiment investigates whether secret-key coefficients can be visually identified from a power trace using SPA. Xu et al [12] observed significant visual dif-

ferences for secret-key coefficients in the range $\{-2, 2\}$ when using specific ciphertexts. They targeted a specific section of the inverse NTT shown in algorithm 7 Algorithm 7 shows the final step of the inverse NTT after which $\mathbf{s^T} \cdot \mathbf{u}$ has been computed. Before the loop $r[j]$ contains all coefficients of $\mathbf{s^T} \cdot \mathbf{u}$ times a factor $128 \cdot R^{-1}$. To correct for the factor, we need to do modular multiplication of all coefficients by $128^{-1} \cdot R$. This is done using the $fqmul(a, b)$ function, which does a multiplication of a and b followed by Montgomery reduction (see: 2.1.2). Since $fqmul$ also introduces another factor R^{-1}, we need to multiply with $128^{-1} \cdot R^2$ instead.

Algorithm 7 Critical Section

1: $f \leftarrow 128^{-1} \cdot R^2$
2: $trigger\_high()$
3: **for** $j \in \{0, 255\}$ **do**
4: $\quad r[j] \leftarrow fqmul(r[j], f)$
5: **end for**
6: $trigger\_low()$

3.2.1 Chosen Ciphertexts

Under normal circumstances, leakage from the $fqmul$ function in every iteration of the loop corresponds to a combination of secret-key coefficients.

To illustrate this we will consider two polynomials:

$$u = u_0 + u_1 \cdot X$$
$$s = s_0 + s_{255} \cdot X^{255}$$

Then:

$$\begin{aligned} u \cdot s &= u_0 s_0 + u_1 s_0 \cdot X + u_0 s_{255} \cdot X^{255} + u_1 s_{255} \cdot X^{256} \\ u \cdot s / \mod (X^{256} + 1) &= (u_0 s_0 - u_1 s_{255}) + u_1 s_0 \cdot X + u_0 s_{255} \cdot X^{255} \end{aligned}$$

Now the first coefficient of the polynomial multiplication contains information about two secret-key coefficients. In this case, the very first iteration of the loop in the critical section would leak information corresponding to two secret-key coefficients. In the worst case, the first coefficient could even contain information about all 256 other secret-key coefficients. In that case, a SCA attack would be impractical, because there are just too many possible combinations. Ideally, we would like to find a way such that every iteration only leaks information corresponding to a single subkey. We cannot control the secret-key polynomial s. What we can control however is

the ciphertext polynomial u. By appropriately selecting the ciphertext it is possible to make it such that every iteration leaks information corresponding to a single subkey. We can do this by choosing $u = u_0 + 0 \cdot X + \cdots + 0 \cdot X^{255}$. Now the result will be equal to $u \cdot s = u_0 s_0 + u_0 s_{255} \cdot X^{255}$. Now the first iteration will leak information corresponding to $u_0 s_0$. We have successfully leaked information corresponding to only a single subkey.

For now, we have only considered polynomials. In Kyber we are dealing with polynomial vectors. Fortunately, the above example can be easily extended to deal with polynomial vectors. In the case of Kyber512, we have two-dimensional vectors. The variable $r[j]$ will contain the dot product of the ciphertext vector and the secret-key vector. The elements will be multiplied using polynomial multiplication followed by modular reduction. Say we have two vectors s and $\mathbf{u}$. The dot product of the two vectors will be equal to $\mathbf{s^T} \cdot \mathbf{u} = \mathbf{u_0} \cdot \mathbf{s_0} + \mathbf{u_1} \cdot \mathbf{s_1}$. If we choose $\mathbf{u_0} = u$ and $\mathbf{u_1} = 0$, then $\mathbf{s^T} \cdot \mathbf{u} = u \cdot \mathbf{s_0} + 0 \cdot \mathbf{s_1} = u \cdot \mathbf{s_0}$. In that case, $r[j]$ will contain information about the first half of the secret-key. To get the second half we can repeat the process but with $\mathbf{u_0}$ and $\mathbf{u_1}$ reversed. This technique can also be applied to the other versions of Kyber.

Compression and Decompression To reduce the ciphertext size, Kyber makes use of a lossy compression technique. The ciphertext will be compressed before it is transmitted. Then during decryption, the ciphertext will be decompressed. The compression technique discards some least significant bits for every coefficient. This can be done because the least significant bits have a very small influence on the correctness probability of the key encapsulation procedure. When choosing ciphertexts it is important to take compression and decompression into account. As the decompressed ciphertext may be different from the original ciphertext. Choosing from a special subset of coefficient values is a simple method of guaranteeing that the original ciphertext is the same as the decompressed ciphertext. If we consider the crafted ciphertext described previously, then of the possible 3329 ciphertexts, there are 1024 possible ciphertexts satisfying the condition $ct = Decompress(Compress(ct))$. The code which generates this subset is listed in appendix A. All other experiments described in this thesis use this method to select appropriate ciphertexts.

3.2.2 Hypothesis

Say we choose the ciphertext using the technique described above. We indicate the value of the first coefficient of $\mathbf{u_0}$ by u (all other coefficients are zero). Furthermore,

we partition the secret-key coefficients to be:

$$\mathbf{s_0}[i] = \begin{cases} -2, & \text{for } i \in \{0, 49\} \\ -1, & \text{for } i \in \{50, 99\} \\ 0, & \text{for } i \in \{100, 155\} \\ 1, & \text{for } i \in \{156, 205\} \\ 2, & \text{for } i \in \{206, 255\} \end{cases}$$

Considering Xu's results we could hypothesize that there should be clear visual differences between the different secret-key partitions depending on the value chosen for u. Xu also observed that the observed leakage heavily depended on the values used for u.

3.2.3 Experimental Procedure

Both the ARM and RISC-V processors have been programmed with the critical section and SimpleSerial protocol. The data is prepared externally on the Raspberry Pi, such that it contains $r[j]$ prior to the loop. This data is transferred to the target. The target will then execute the critical section and trigger the capture process. This is done for all possible values of u. On the STM32F3 and FE310, a single loop iteration takes 18 and 30 instructions respectively. In the case of the FE310 two captures are required to obtain the full critical section. This is because of the ChipWhisperer-Lite's limited buffer size. These two captures are combined into a single trace after measurements.

3.2.4 Results

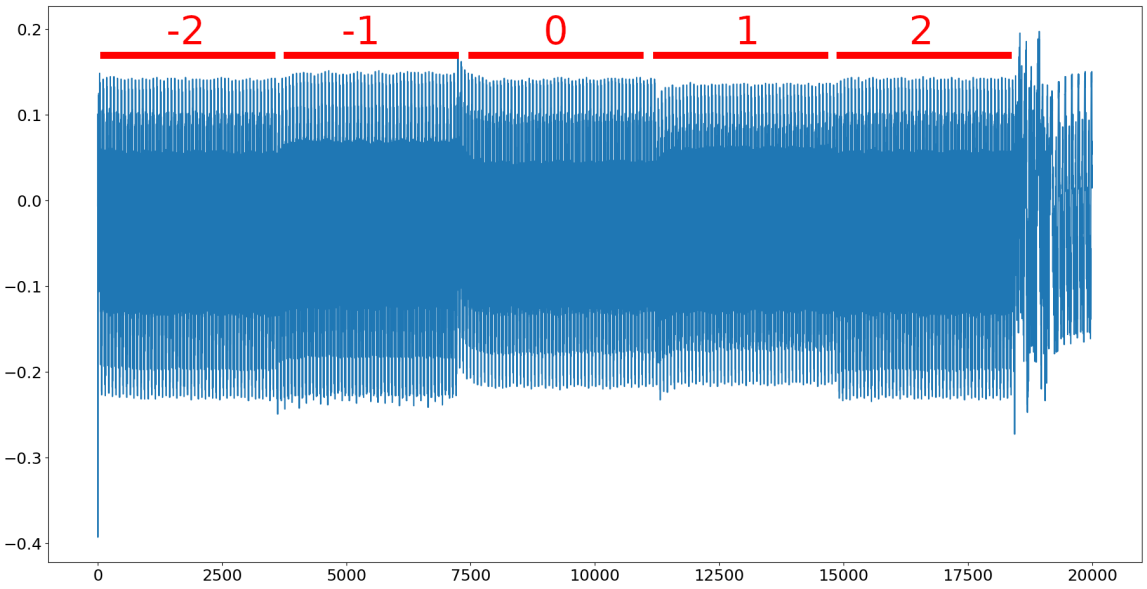
Figure 3.4 and 3.6 show power traces of the critical section shown in algorithm 7 with $u = 55$. The first figure belongs to the ARM MCU and the second figure belongs to the RISC-V MCU. The secret key coefficient partitions have been labelled in red above the trace belonging to STM32F303. The secret-key coefficients have been partitioned in the same way on the FE310-G002. The loop on the STM32F3 takes approximately 18500 samples. The loop on the FE310 takes about 30800 samples. Note that the traces also contain some samples from after the critical section. In the graph, low values on the y-axis indicate samples with high power consumption. This has to do with the way that the measurement probe and shunt resistor have been placed. When the power consumption is high, lower voltages will be observed.

There is a stark visual difference between the two MCUs. The different key partitions are visible in the STM32F303. There is also some surprising behaviour when

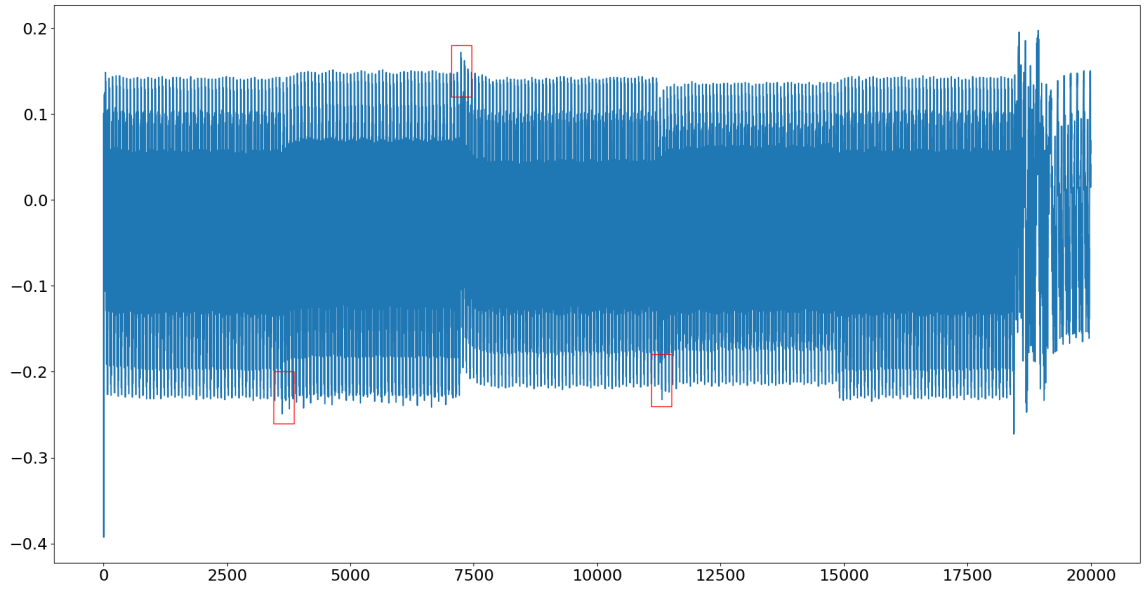
key transitions occur. When a key transition occurs there is a spike and some decaying behaviour. These have been highlighted in figure 3.4b. Contrary to the STM32F303, in the FE310 the trace is fairly constant. The partitions that were visible in the STM32F303 cannot be seen.
A simple post-processing technique we can do is to look at the rate of change between different samples. We can do this by applying the finite difference method. The new samples s' are defined as $s'[i] = s[i] - s[i-1]$ for $i = 1...n$, where n is the number of samples. Figure 3.5 shows the resulting post-processed trace. The different key partitions are even more visually discernible.

3.2.5 Implication of results

Firstly, there are different partitions visible on the STM32F3 trace. Yet it is not immediately clear whether this difference is significant enough to construct an attack. Also, there are spikes visible between certain transitions. These are especially prevalent in the transitions from -2 to -1 and -1 to 0. If these spikes become more prevalent when the secret-key is irregular, then recovering the secret-key with SPA techniques might not be trivial. There is also some discharge (possibly capacitive) after certain transitions. Most notably between -1 to 0 and 0 to 1. Given that there is a clear visual distinction between the partitions, there is very significant side-channel leakage in the power trace. The finite-difference method looks at the rate of change between the different samples. Specific data-sensitive instructions might require a large surge of power. This can possibly be an explanation of why the finite-difference method results in more visually discernible partitions. This simple analysis suggests that it might be possible to perform key recovery by simply looking at the Even if SPA might fail, other SCA techniques might be strong enough to form a serious threat to the security of the cryptographic scheme.
The FE310-G002 did not exhibit clear visual leakage, this does not guarantee that no leakage is present. Stronger SCA can be explored to show whether sensitive side-channel leakage is present. In the next chapter CPA is performed on the critical section.



(a) STM32F303



(b) Spikes and decaying behaviour after key transitions

Figure 3.4: Simple Power Analysis (STM32F303)

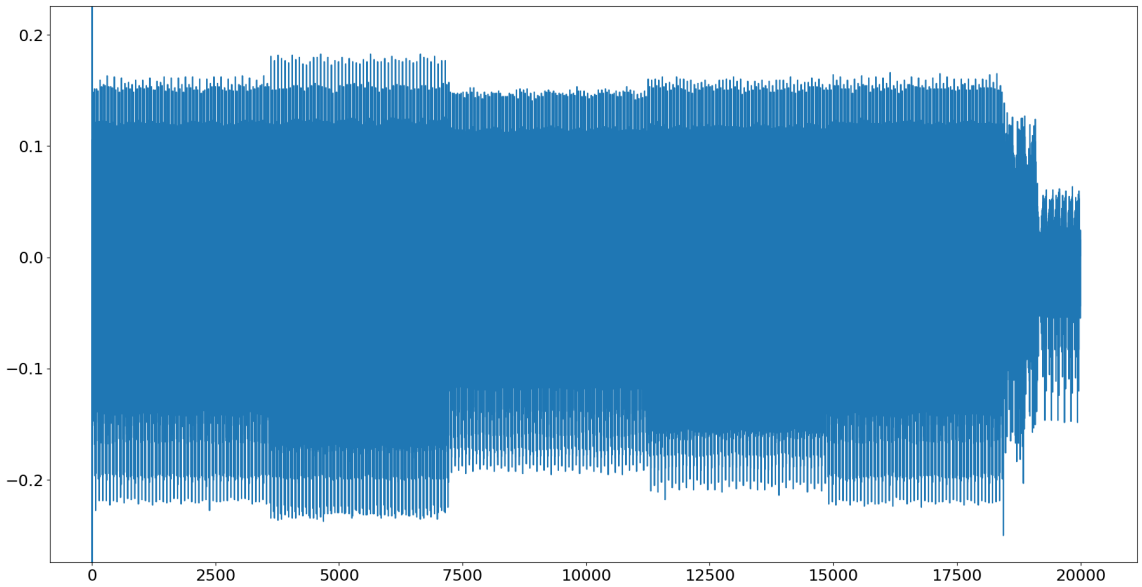


Figure 3.5: Finite Difference of Power Trance (STM32F303)

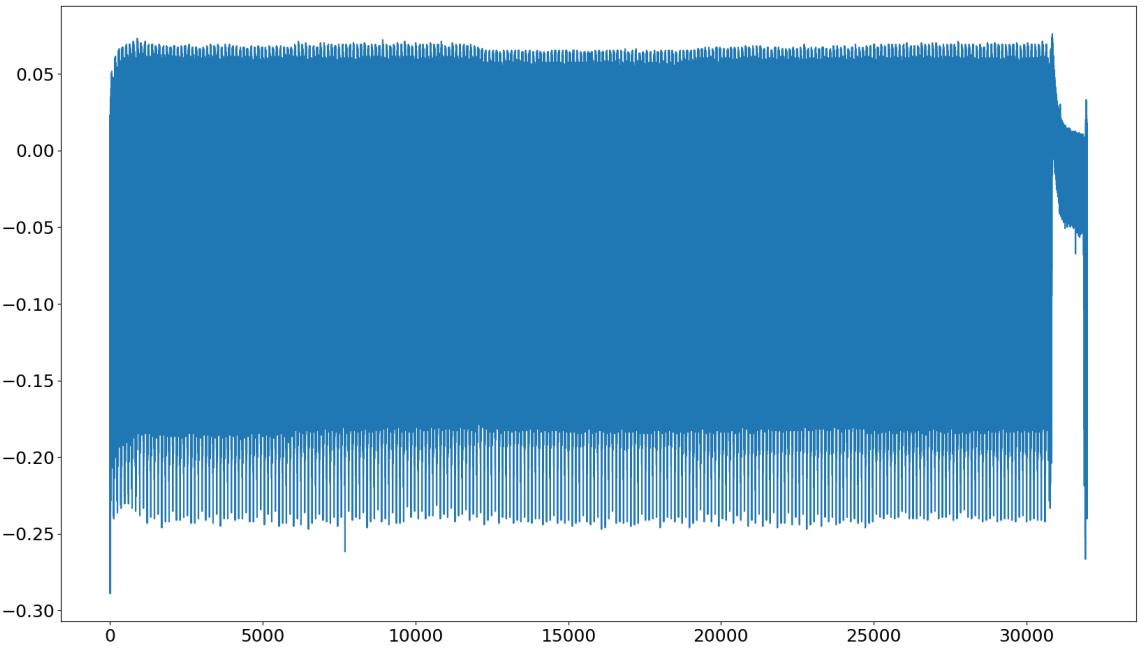


Figure 3.6: Simple Power Analysis (FE310-G002)

Chapter 4

Correlation Power Analysis

In this chapter a novel correlation power analysis (CPA) attack on the *pqclean* implementation of Kyber is proposed. Additionally, the attack is experimentally evaluated on both the STM32F303 and FE310-G002 MCUs. The attack targets the same critical section as described by algorithm 7.

4.1 Identifying Leakage

Firstly, we want to identify whether there is a relationship between some intermediate variable and power consumption. In the critical section, $fqmul$ is the main operation which might be leaking information. Experiment 1 has shown for sure that there is visible leakage present on the ARM MCU. The RISC-V trace did not exhibit any visible leakage. That being said, a lack of visual indication does not guarantee that there is no leakage. As a matter of fact, in most cases, the leakage will not be visibly present. In this experiment, we will test for leakage in the $fqmul$ function. One possible intermediate variable which could be related to the power consumption is the output of the $fqmul$ function. If this is the case, then this could be possibly leveraged to do an attack, since the $fqmul$ operates on sensitive data. The experiment described in this section tries to identify leakage in the $fqmul$ operation for both the ARM and RISC-V MCU.

4.1.1 Experimental Procedure

The same hardware setup as described in section 3.1.1 is used for this experiment. Furthermore, very similar firmware has been used. The only adaptation is that $fqmul(a, b)$ can be invoked individually and not within the context of a loop.

Firstly, multiple captures will be done for different inputs to $fqmul(a, b)$. We fix input

b to be equal to 3328. This is done to make sure that the multiplications result in non-trivial reductions. For a we choose any value in the range of $\{-3328, 3338\}$. Using these inputs it is possible to generate every element in the multiplicative group of integers under modulo 3329. Furthermore, every possible hamming weight (0-16) for the output can be obtained using these inputs.

Secondly, the different traces of the captures will be inspected to see whether there is a relation between leakage and the hamming weight of the output of $fqmul$.

Thirdly, the PCCs of the hamming weight and power leakage will be plotted. If there is a correlation, one can expect to see values close to 1 or -1.

4.1.2 Results

The execution of the $fqmul$ operation took 112 and 144 cycles on the ARM and RISC-V MCU respectively. Figure 4.1 shows a single $fqmul$ call for both the ARM and RISC-V MCU. Figure 4.2 is a plot of power traces belonging to different values of the intermediate variable. The darker the colour of the plot, the higher the hamming weight of the intermediate variable. Figure 4.2b is a close-up of one of the instructions with the most notable leakage. The window of this close-up is indicated by the red rectangle on 4.2a. What can be seen from the close-up is that the traces form a clear gradient. From top to bottom, we see that the traces correspond to increasingly larger hamming weights. This is expected because lower trace values correspond to higher power consumption. The ARM and RISC-VMCU had minimum PCC values of -0.9749 and -0.9152. This suggests that ARM MCU might be slightly more vulnerable to SCA.

Figure 4.3 and 4.4 plot the traces next to the PCC values for both the ARM and RISC-V MCU. The PCC values correlate the power leakage at a time point with the hamming weight of $fqmul$'s output. The PCC value is a normalized form of the covariance. Extreme values indicate a very strong positive or negative linear relation. For both MCU we see a strong negative relation between trace values and the hamming weight. This is in line with the visual inspection of 4.2b.

Conclusion Both results indicate that there is a strong correlation between the hamming weight of the output of $fqmul$ and the power leakage. The RISC-V MCU exhibits a slightly stronger correlation between power consumption and output than the ARM MCU. This strongly indicates that visual inspection alone does not give a

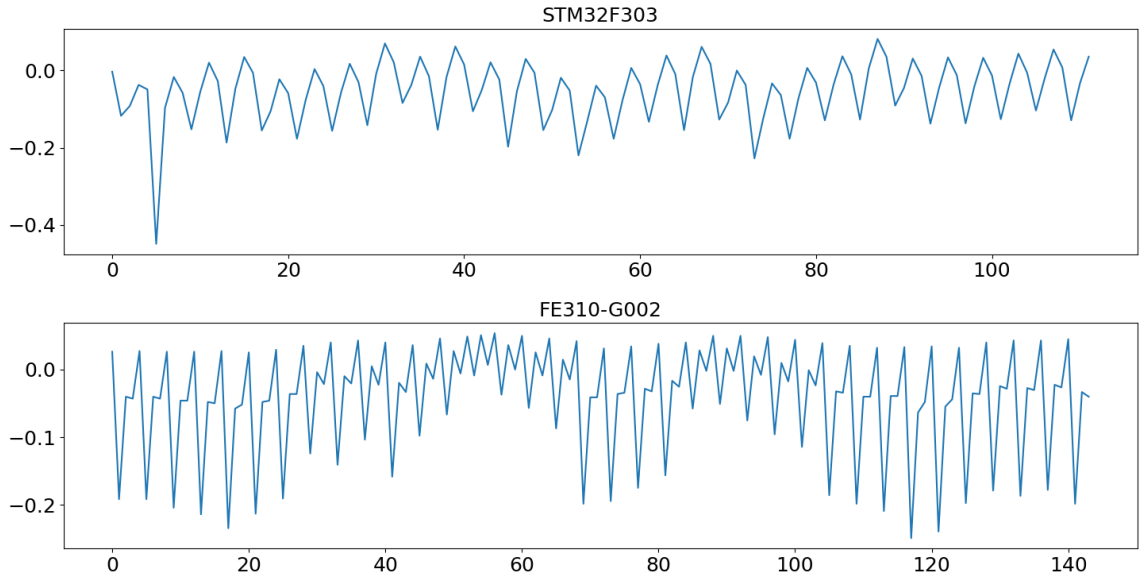
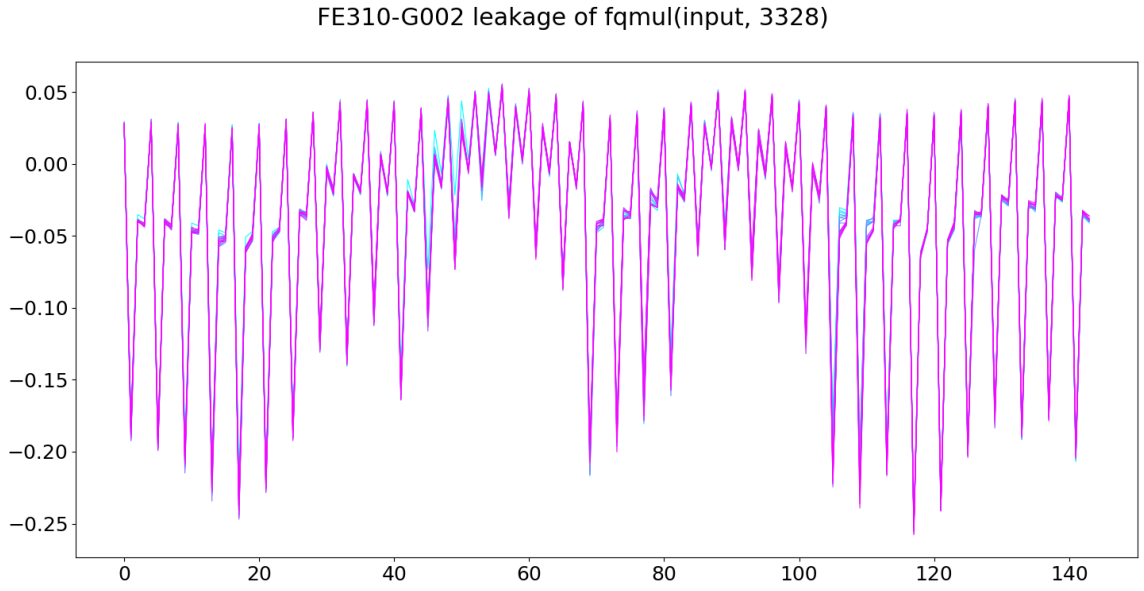


Figure 4.1: $fqmul$ call on ARM and RISC-V MCU

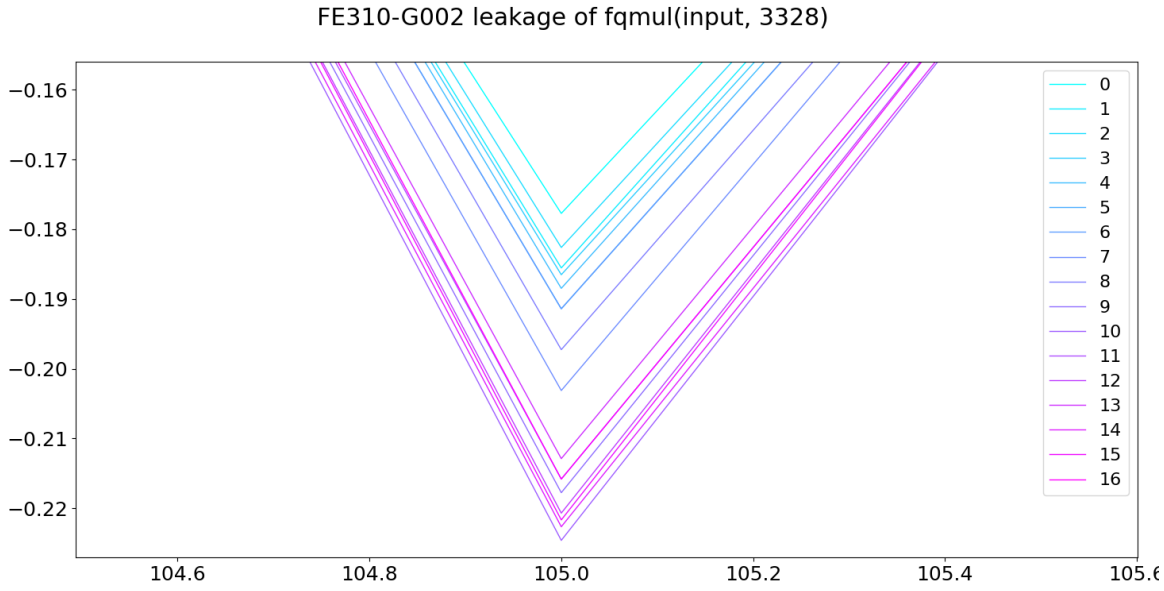
good indication of leakage.
Leakage has been identified and this indicates a weakness in the Kyber implementation. The presence of a weakness does not guarantee that the weakness is also exploitable. However, since $fqmul$ operates on sensitive values in the critical section there is a high likelihood that secret information can be recovered. It is a good entry point for further experimentation. The next section (4.2) proposes and experimentally validates a possible CPA attack, which exploits the discovered leakage.

4.2 Correlation Power Analysis Attack

The results of experiment 3 suggest that there is a high correlation between the output of $fqmul$ and the power consumption. This correlation can be exploited by a potential adversary through the use of a CPA attack. CPA attack is a form of a non-profiled SCA attack. In a CPA attack the power side-channel leakage is correlated with a theoretical model of the power consumption. See section 2.2.2 for a more in-depth explanation of the workings of CPA. In this section a CPA attack is proposed that utilizes the identified leakage to recover the full secret-key. The first part of this section contains a description of the attack procedure. The second part describes the experiment done to verify the effectiveness of the attack. Furthermore, the assumptions that are made for the attack are listed in the adversary model.



(a) Full $fqmul$ procedure



(b) Close-up

Figure 4.2: Leakage for different output values of $fqmul$ (FE310-G002)

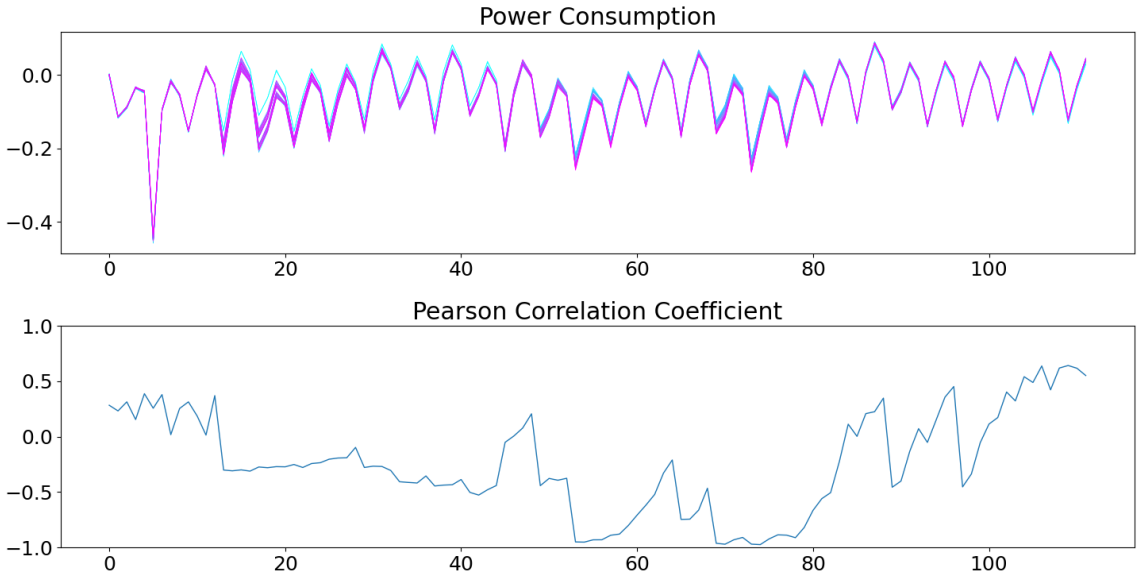


Figure 4.3: Pearson Correlation Coefficient STM32F303

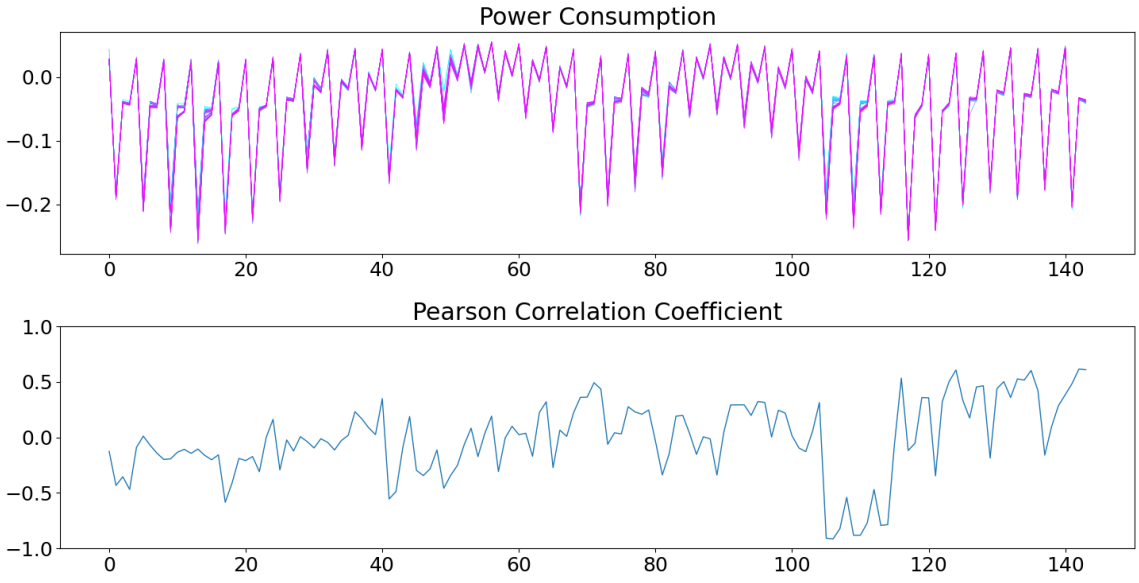


Figure 4.4: Pearson Correlation Coefficient FE310-G002

Adversary Model

- Adversary can obtain power traces of the decryption procedure.
- Adversary can provide chosen ciphertexts.
- Adversary can initiate the decryption procedure.
- Adversary can identify the start of the critical section.
- Target uses PQLCEAN implementation of Kyber.
- Target uses static key setting.

4.2.1 The Leakage Model

A leakage model is required to perform a CPA attack. The leakage model gives allows us to correlate the estimated theoretical power consumption to the actual power consumption. If the leakage model is correct then one can expect to see extreme PCC values. In section 4.1 was shown that $HW(fqmul(a, b))$ is a reasonable leakage model for the critical section. Under normal circumstances, however, we are not able to see the inspect intermediate variables and thus we do not know the a or b that is being used. In the critical section described in algorithm 7 we do know the value $b = 128^{-1} \cdot R^2$. The other input a is unknown and is a combination of the ciphertext and the secret-key. However as seen in section 3.2.1 we can choose ciphertexts, such that a is a combination of u and a single coefficient from the secret-key sk_i (also referred to as a subkey). Given that the result of the computation will be equal to $u \cdot sk_i \mod 3329$ and that the leakage corresponds to the hamming-weight of the output of the $fqmul$ function, one might think to use the following leakage function:

$$L_{example}(u, sk_i) = HW(u \cdot sk_i \mod 3329)$$

This, however, does not work in practice due to the implementation of the $fqmul$ function. The function $fqmul$ has an output range of $\{-3328, 3328\}$. This means that $fqmul$ can output two different values corresponding to a single element of the integers modulo 3329. For example, 5 and -3324 both represent the same element in $\mathbb{Z}_{3329}$. They can be considered to be mathematically equivalent. All elements of $\mathbb{Z}_{3329}$ can be represented by a positive and negative value in the range of $\{-3328, 3328\}$ (with the exception of zero). We will refer to these values as the positive and negative remainder. These values will be denoted by x^+ for the positive remainder and x^- for the negative remainder. For example, if we have $x \equiv 5 \mod 3329$, then $x^+ = 5$ and $x^- = -3324$. From a mathematical perspective, these values are equivalent, but when dealing with hardware it is important to make this distinction. The bit

representations of 5 and -3324 are very different (especially in 2's complement representation). As a result, the leakage characteristics of 5 and -3324 will also differ significantly. We cannot use the leakage function $L_{example}$, because we do not know whether $u \cdot sk_i \mod 3329$ will be the positive or negative remainder. The leakage model would be inaccurately guessing what the hamming weight would be. It would only be correct 50% of the time. This has been verified through experimentation.

One approach to solving this issue is by investigating whether it would be possible to predict whether the remainder will be positive or negative. The $fqmul$ function has the useful property that it is deterministic. If one knows the input, then it is possible to determine the output. By predicting whether the input to $fqmul$ is positive or negative, one can accurately determine the output. In the critical section, we are looking at the last operation done in the inverse NTT. Before this many more $fqmul$ calls have been executed. To transform polynomials to the NTT domain, but also to multiply two polynomials in the NTT domain. Throughout this complicated process, variables get multiplied by ciphertext coefficients and secret-key coefficients. This results in the sign of the input being dependent on both the ciphertext and secret-key. As a matter of fact, the sign of $r[j]$ for a single entry depends on all other secret-key coefficients. This search space is too large and therefore this approach does not work in practice.

A second approach could be to investigate which of the two options is more likely to occur. Maybe there is a significant probabilistic difference between the input being a positive or negative remainder. Experimentation of the $fqmul$ function led to the following observation. The condition that $fqmul(x^+, 128^{-1} \cdot R^2) = fqmul(x^-, 128^{-1} \cdot R^2)$ holds 98.08% of the time for $x \in \{-3328, 3328\}$. In other words, most of the time the inputs x^+ and x^- will both result in the same output.

With this observation in mind, the following leakage model is proposed:

$$L(u, sk_i) = HW(fqmul(u \cdot sk_i \cdot 128 \cdot R^{-1} \mod {}^+3329, 128^{-1} \cdot R^2)) \tag{4.1}$$

Where u is the ciphertext value as described in section 3.2.1. The guess for the i'th subkey is represented by sk_i. And the function $x \mod {}^+3329$ will output the positive remainder of $x \mod 3329$. Experimentation showed that the leakage model $L_{example}(u, sk_i)$ is correct 50.05% of the time, whereas $L_(u, sk_i)$ has a success rate of 99.08%. This leakage model can be used to recover the full secret-key one subkey at a time.

4.2.2 Dealing with the case: $sk_i = 0$

There is an issue related to a special case where the subkey is equal to zero. CPA computes the linear correlation between two distributions. It checks whether an increase in one value tends to increase of the other value. One requirement for this is that we have two distributions with at least a little bit of variance. But if the subkey is equal to zero, then $L(u, 0) = HW(0) = 0$. Regardless of our input ciphertext u. The output will always be zero and there is no variance in the output of our leakage function. Under ideal circumstances (no noise), the power consumption would also have zero variance. In practice, however, there will be some variance in the power measurements. In this special case, the PCC is undefined. There is no correlation to be made because there is no variation in the leakage model values. We, therefore cannot perform CPA for the case, where $sk_i = 0$. We will refer to this case as the zero case. There are multiple ways of tackling this problem.

If the subkey is zero, then you would expect that all other PCC values are low because there is no correlation to be made for the other subkey guesses. Hence one method of solving this is giving zero guesses a fixed PCC value α. This value needs to be chosen carefully such that in all other cases, it is lower than the PCC value of the correct key guess and the highest otherwise.

Another method could be to use the variance of the power consumption. In the zero case, the output of $fqmul$ is always zero. In other cases, the output varies depending on the ciphertext. If the variance in the power consumption is very low between traces, then most likely it belongs to the zero case. Note that this only applies to the time window of the trace related to the output of $fqmul$. To do this, one has to find a threshold for the variance which will be used to identify zero cases from the other cases. Also, a correct time window needs to be found. Figure 4.5 proves this assumption to be true. The blue plots correspond to the standard deviation of traces belonging to subkeys which are non-zero. The red plots belong to zero subkeys. It is evident from the different plots that between $t = 10$ and $t = 40$ zero subkeys tend to have a significantly lower standard deviation. This shows that the standard deviation can be a useful metric for identifying zero-valued subkeys.

Both methods have their shortcomings when it comes to non-profiled SCA analysis. They require fine-tuning of certain parameters to be effective. This might be difficult for an attacker to do when a clone device is not available. An attacker however will be able to capture traces belonging to the situation where the output of $fqmul$ is equal to zero regardless of the subkey. This can be done by setting all coefficients of the ciphertext to zero. Now the attacker has access to power traces

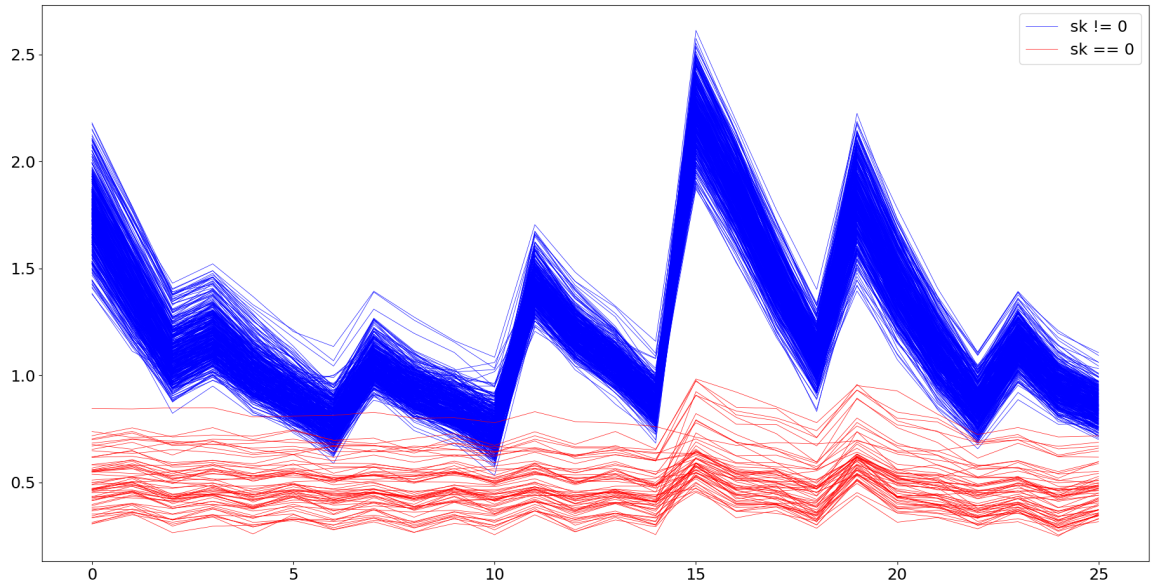


Figure 4.5: STD of traces belonging to different subkeys

knowing that the output of $fqmul$ equals zero. The underlying assumption is that the power samples related to the output of $fqmul$ will very closely resemble the power samples of the zero case. This is because the zero case also has a $fqmul$ output of zero. These captures can in turn be utilized for finding appropriate values for the different parameters. This way, an attacker can obtain a collection of traces belonging to the distribution where the output of $fqmul$ equals zero. This leads to a third method, where the attacker uses a collection of obtained power traces where $fqmul$'s output is zero to identify zero cases. A possible statistic that could be used for this is the student's t-test. Again this requires careful selection of a time window related to the $fqmul$ output. Applying these techniques does not alter the attack's nature. The attack will still fall in the category of non-profiled SCA attacks.

4.2.3 Attack Procedure

There is an offline and online component to the CPA attack. During the online component, an adversary will capture multiple power traces of the decryption procedure. During the offline component, an adversary will analyze the captured power traces in order to find the most likely key value.

Preparation (offline) During this phase, an adversary will craft several chosen ciphertexts. Appropriate selection of ciphertexts is important. Choosing appropriate ciphertexts is described in section 3.2.1. Note that the ciphertexts determine whether the leakage corresponds to the first half or second half of the ciphertext (in

the case of Kyber512). Therefore, it is a good idea to use ciphertexts which result in both halves being equally represented.

Capturing power traces (online) An adversary has to obtain several power traces of the decryption procedure. Every power trace has an associated ciphertext. This phase often requires physical access to the device. Furthermore, the adversary has to be able to initiate the decryption procedure on demand. Generally, a larger amount of captures will result in a higher success rate.

Performing Correlation Power Analysis (offline) In the final phase the adversary will perform CPA to recover the full secret-key. The adversary will have to go through a couple of steps to find the most likely subkey guess.

Initially, a trace contains leakage corresponding to 256 $fqmul$ calls. Every trace needs to be windowed, such that each window corresponds to a single $fqmul$ call. Depending on the chosen ciphertext, every i'th window corresponds to the i'th or $(i + 256)$'th subkey.

After windowing has been performed, PCC values can be computed for different subkey guesses. This will be done using the leakage function given by equation 4.1. Zero explained before, subkey guesses for zero cannot be given a PCC value and need to be handled differently.

After all the values have been computed, the most likely secret-key guesses can be determined. Normally this is done by taking the guess that belongs to the highest absolute PCC values.

4.2.4 Experiment Procedure

Setup:

Both the hardware and software setup from experiment 1 were used. These are discussed in section 3.1.1 and section 3.1.2 respectively.

Capturing:

Power captures of the decryption procedure have been done for both the RISC-V and ARM MCU. In total 200 captures were done for the RISC-V MCU. 100 for each half of the secret-key. For the ARM MCU 400 captures were done for every half, resulting in a total of 800. For every capture, a random ciphertext was chosen as

described in 3.2.1. The secret-key was fixed with every subkey being one of seven possible values in the range $\{-3, 3\}$. The clock rate was fixed at 7.37MHz. Sampling was synchronized with the clock and done at four times the clock rate. The critical section requires approximately 4625 and 7700 clock cycles to be fully executed on the ARM and RISC-V processor respectively. The captures were initiated by setting a dedicated GPIO pin to high. A single power trace of the critical section on the RISC-V MCU is shown in figure 4.6.

Correlation Power Analysis:

Windowing To window the traces a naive method was used. A real attacker would probably like to use a more sophisticated technique. However, for the purposes of this thesis, this method proved itself more than sufficient.

Traces were windowed by using a reference trace of a single $fqmul$ call. The idea is to compare the reference trace to the 256 $fqmul$ calls. If a section of the full trace is very comparable to the reference trace, then this most likely corresponds to a single $fqmul$ call. By sliding the reference trace over the full trace one can hopefully find points at which the overlap of the traces is very comparable. This has been done by computing the sum of absolute difference (SAD) at every point while sliding the reference trace over the full trace. Local minima of the SAD graph will then correspond to the start of a single $fqmul$ call. Figure 4.7a shows a close-up of the critical section shown in figure 4.6. This plot has been used to identify the windows of the loop iterations. Figure 4.7b shows a closeup of the resulting SAD plot for six $fqmul$ calls. There are six low peaks clearly indicating the location of the $fqmul$ calls.

These local minima were identified by using an adaptive signal thresholding algorithm. The locations of these peaks correspond to the location of a single $fqmul$ call. Using this information it is possible to subdivide the traces, such that each subdivision corresponds to a single $fqmul$ call and thus a single loop iteration. Figure 4.7c shows a closeup of a trace that has been windowed. The red lines indicate the different subdivisions.

Dealing with $sk_i = 0$ Only the first two methods described in section 4.2.2 have been examined. Of these, the most robust method seemed to be inserting zero subkey guesses with a fixed PCC value of $\alpha = 0.75$. Figure 4.8 shows the partial guessing entropy (PGE) of the CPA attack on the RISC-V MCU when using different α values. There is quite a large window for α values where the attack is 100% successful. Indicating that the choice of α does not have to be very accurate. Using the variance was not robust as the desired threshold differed greatly depending on

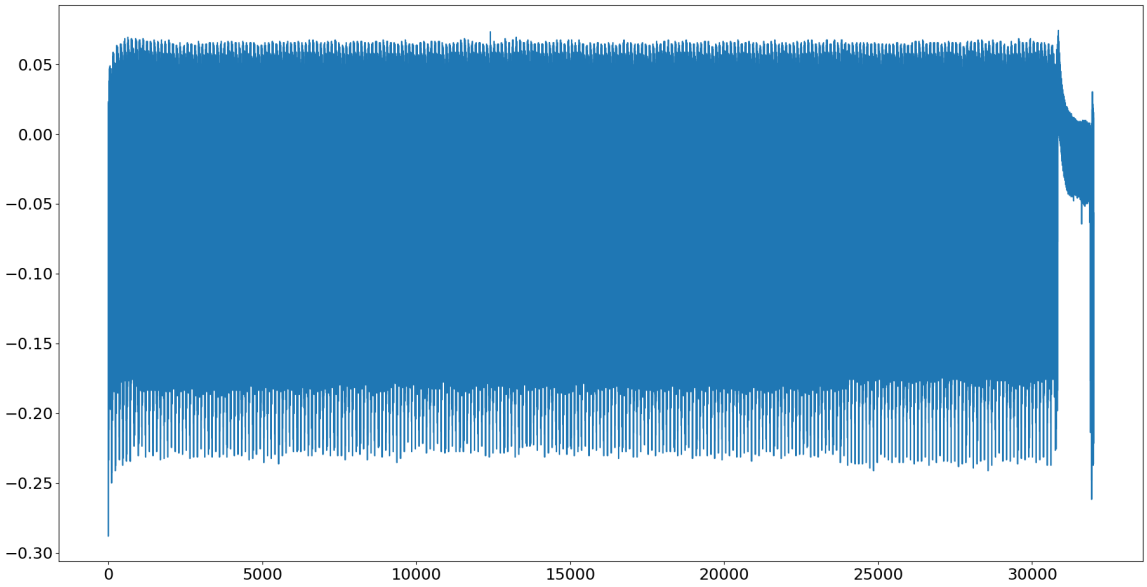


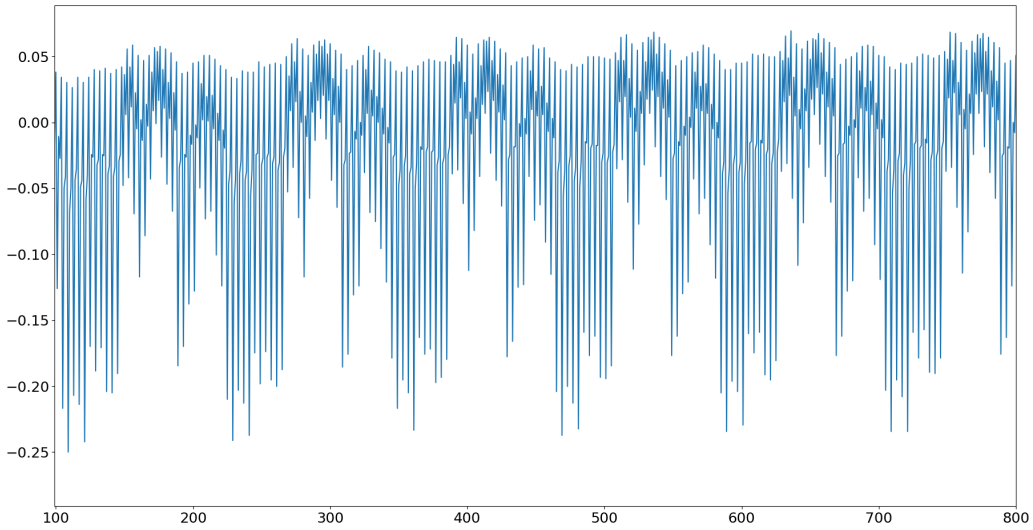
Figure 4.6: Critical section (FE310-G002)

the number of traces that were analyzed. Due to time constraints, the third method has not been experimentally verified.

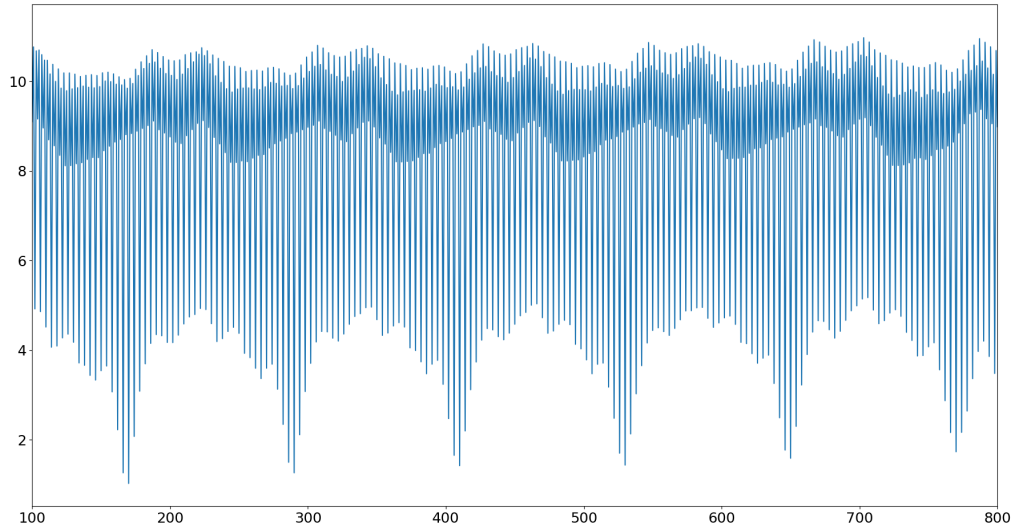
Computation of Pearson correlation coefficients All offline analysis was done through Python with the use of the scientific computing package Numpy [58]. Estimates for the power consumption for a specific key guess were made using the leakage model in equation 4.1. The $fqmul$ function in the leakage model was simulated using a CFFI to the PQCLEAN library. PCC values were calculated using equation 2.4. To allow for efficient computation, intermediate sums were recomputed upon the addition of a new trace. This way PCC values can be updated dynamically when a new trace is obtained. This also allows for online analysis where an attacker computes PCC values during the capturing process. Normally the highest absolute PCC value is used to select the most likely subkey. In this case, this did not result in a high success rate. The reason for this is discussed in section 4.2.5. Instead the minimum PCC value is used to select the most likely subkey. This is a natural choice because a strong negative correlation was found in section 4.1 between the trace and hamming weight.

4.2.5 Results

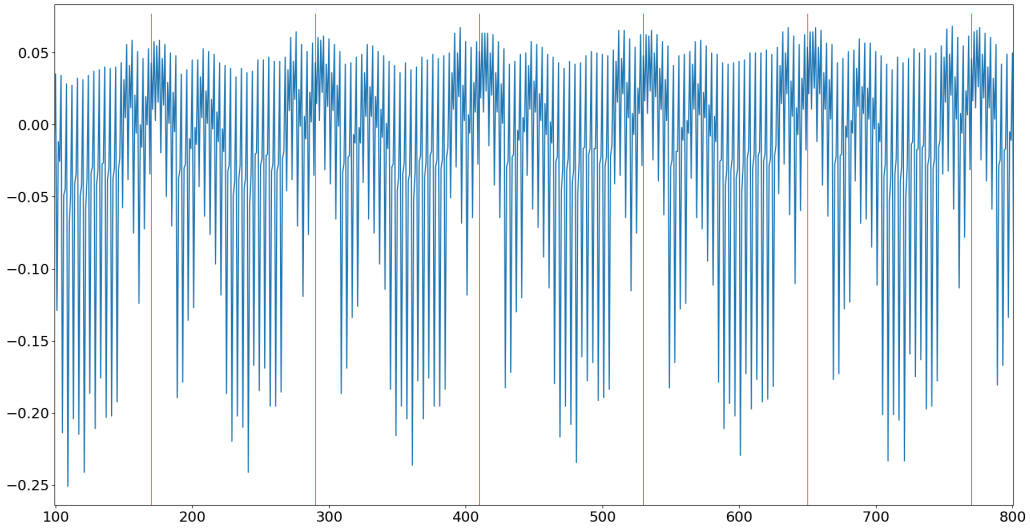
The CPA attack was able to do full key recovery for both the MCUs. A common metric used to indicate the effectiveness of an attack is to use the partial guessing entropy (PGE). For every subkey, the attack creates a subkey guessing vector. This



(a) Critical section



(b) SAD plot



(c) Critical section subdivided

Figure 4.7: Windowing Process

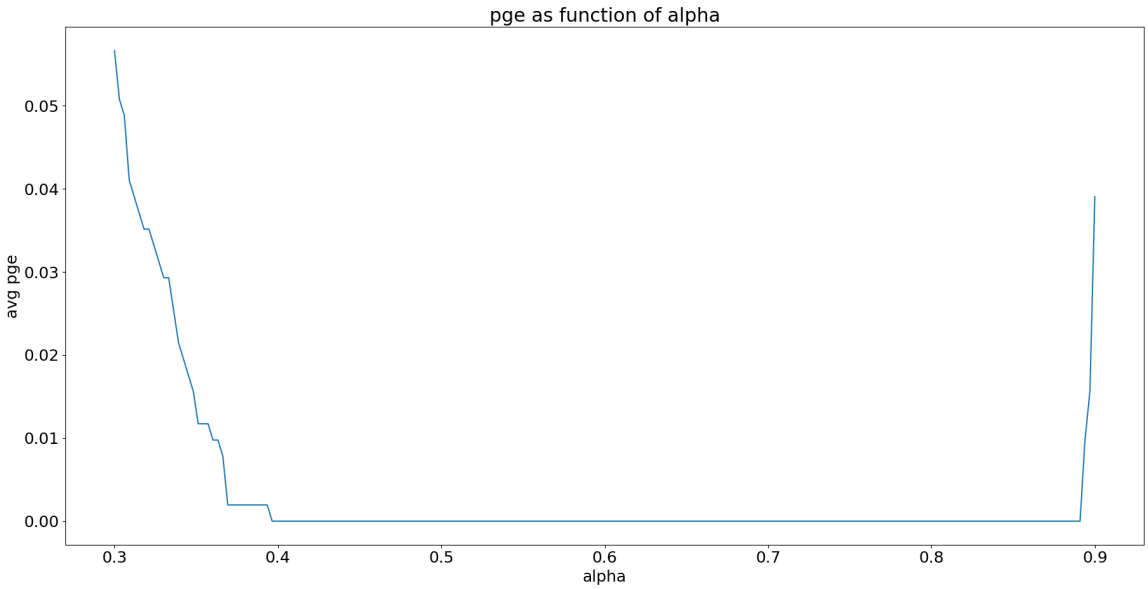
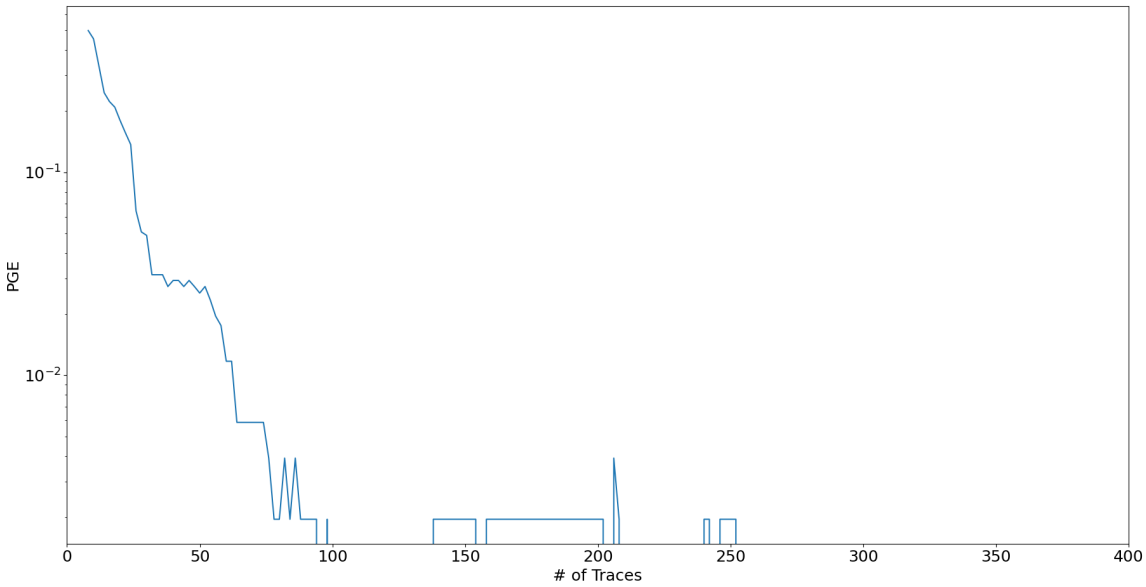


Figure 4.8: PGE as function of α

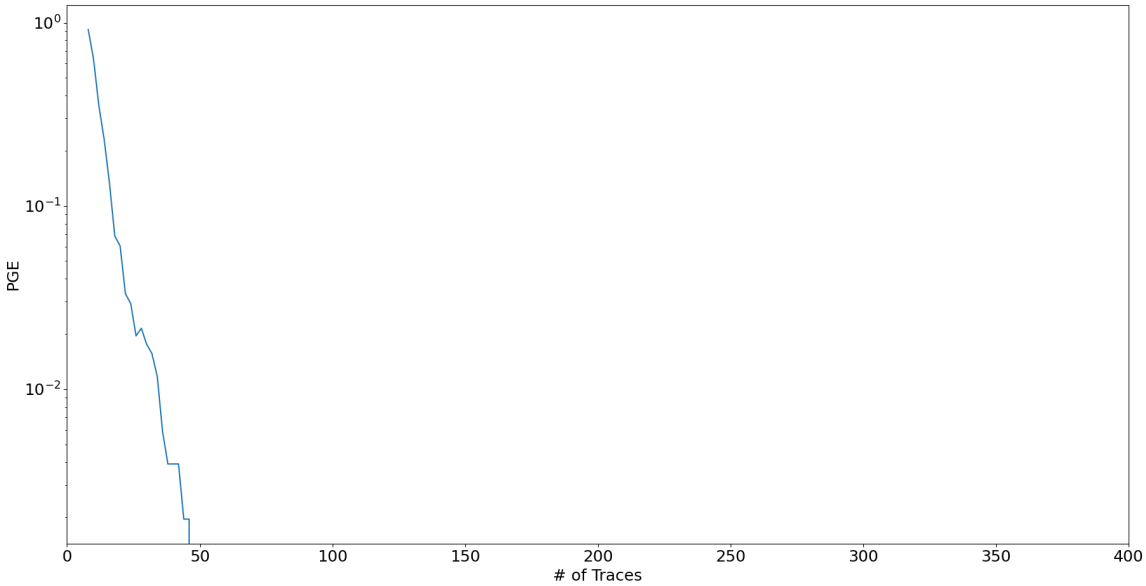
vector is a list of most likely, to least likely subkey values. The PGE is the distance from the first guess to the true subkey value. If the first guess is correct we will have a PGE of 0. To compute the PGE in the context of an attack, we take the average PGE of all subkeys. Figure 4.9 shows the PGE w.r.t. the number of traces that have been analyzed. The shown graph is from a single trial. The mean PGE has been plotted on a logarithmic scale. The full secret-key is found after 96 and 48 traces for the STM32F303 and FE310 respectively. The STM32F303 finds the true key after 96 traces but requires 254 traces before the key guess converges to the true key. The key was recovered within 7 seconds for the STM32F303 MCU and 4 seconds for the FE310 MCU.

Selection Function

Normally for SCA, the maximum absolute PCC value is used to identify the most likely subkeys. This however does not work as effectively for the discussed CPA attack. Recall that with our chosen ciphertext attack the function computes $output = fqmul(v, 1441)$. Where v is some intermediate variable. There is a relation between the hamming weight of $output$ when the input to $fqmul$ is v or $-v$. This relation is shown in figure 4.10a. Given our intermediate variable v, we take the hamming weight of $hw_+ = HW(fqmul(v, 1441))$ and $hw_- = HW(fqmul(-v, 1441))$. The count that the pair (hw_+, hw_-) is observed is plotted in figure 4.10a. Pairs that are observed more often are indicated with darker colours. As can be seen, the distribution is not random. The hamming weight of v and $-v$ seems to be sym-



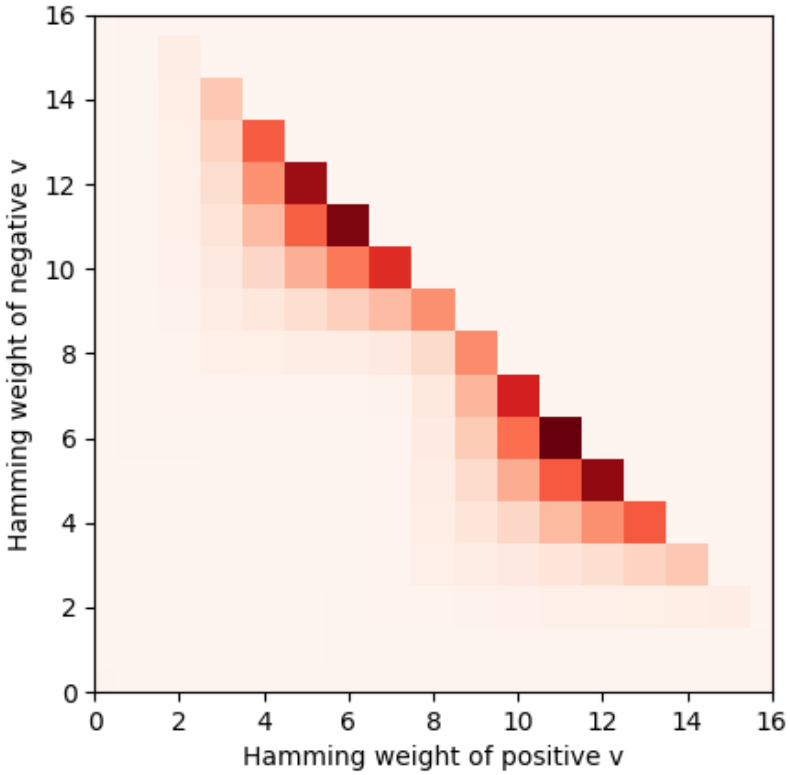
(a) STM32F303



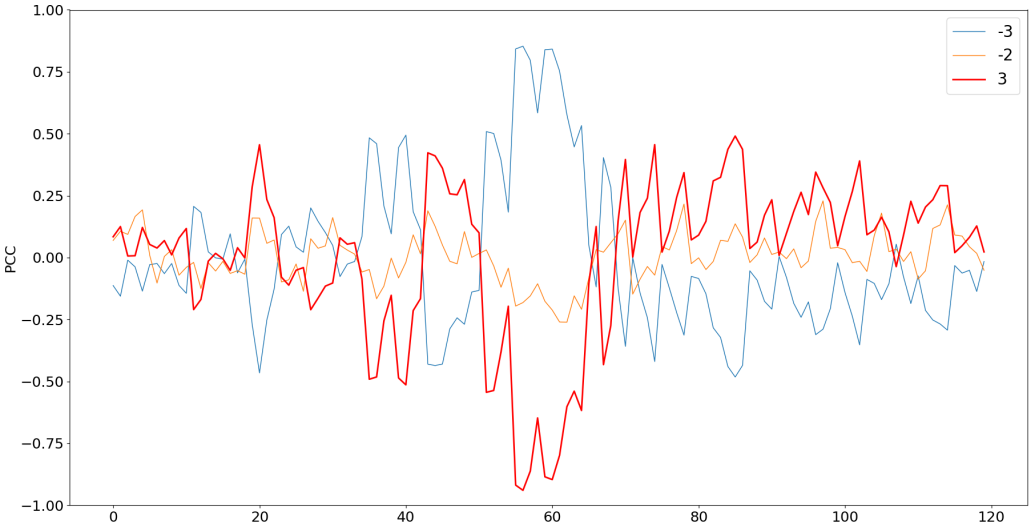
(b) FE310

Figure 4.9: Attack Result

metrically located around the mean. If we find that we have a negative correlation between hw_+ and the power consumption, then we will often also find a positive correlation for hw_-. It is quite common for our leakage model to have v and $-v$ for two guesses. In our leakage model $v = sk \cdot u \cdot 128 \cdot R^{-1}$. Say the correct subkey value is $sk = 3$, then $v = 3u \cdot 128 \cdot R^{-1}$. Since this value is correct, we will find a negative correlation between the hamming weight of v and the power consumption. However, during the CPA attack, we will also try for the subkey value -3, which results in the intermediate variable $-3u \cdot sk \cdot 128 \cdot R^{-1} = -v$. As a result of the hamming weight distribution, we will then find a positive correlation. See figure 4.10b, which has the true subkey as 3. We see that we get approximately horizontally symmetric PCC plots for the true subkey value 3 and incorrect subkey -3. With 3 and -3 resulting in a negative and positive correlation respectively. If we take the highest absolute value, then there is a big chance that we will incorrectly select 3 as the most likely subkey guess. Because of this reason, we take the absolute of the minimum PCC value as our most likely subkey guess.



(a) Hamming weight relation



(b) PCC plots for different subkey guesses (true subkey = 3)

Figure 4.10: Hamming weight distribution causes symmetric PCC plots

Chapter 5

Countermeasures against the proposed attack

Both chapter 3 and chapter 4 showed that clean Kyber implementations can be easily compromised by SCA attacks. It is important that countermeasures are developed that aim to prevent the feasibility of these SCA attacks. In this chapter mitigation strategies are discussed that prevent the proposed attack.

The first section discusses how malicious ciphertexts can be prevented. This would break the attack because the attack relies on leakage that corresponds to specific ciphertexts.

The second section discusses two low-cost 'hiding' software-based countermeasures are proposed that break the proposed CPA attack described in chapter 4. A 'hiding' countermeasure aims at hiding the leakage from an attacker. There are two main ways that one can go about hiding the leakage. The first method tries to hide leakage in the amplitude dimension. This is generally an attempt at decreasing the signal-to-noise (SNR) ratio of the traces. The second method alters the leakage with respect to the time domain. Both proposed countermeasures fall into the time category. This makes it significantly harder to correlate leakage across different traces.

The first countermeasure adds random delays, which make it harder for an attacker to align sensitive operations for CPA analysis. The second countermeasure performs random shuffling. By shuffling the order of the loop iterations, it is a lot more difficult to identify which $fqmul$ invocation belongs to which subkey.

In this thesis, only software-based countermeasures are proposed. An advantage of focusing on countermeasures that rely on software is that they tend to be more flexible. Possible issues or vulnerabilities can be patched by updating

firmware. This would be significantly harder to realize using hardware.

5.1 Preventing Malicious Ciphertexts

The proposed attack relies heavily on being able to provide chosen ciphertexts and initiate decryption. Choosing appropriate ciphertexts makes it possible to get leakage corresponding to single subkeys. If leakage were to contain information about a combination of subkeys, then the search space would increase exponentially. This would effectively, render the attack infeasible.

The success of the proposed attack heavily relies on the ability to provide chosen ciphertexts, as the selection of appropriate ciphertexts allows for obtaining leakage corresponding to individual subkeys. However, if the leakage of a single $fqmul$ operation were to correspond to a combination of subkeys, the search space would increase exponentially, rendering the attack effectively infeasible.

To illustrate this, consider the polynomial multiplication of a secret-key $s = s_0 + s_1X + \cdots + s_{255}X^{255}$ with a ciphertext u. Note for illustration purposes we only consider polynomials and not vectors of polynomials. If $u = u_0$, then $s \cdot u = s_0u_0 + s_1u_0X + \cdots + s_{255}u_0X^{255}$. In this case, every coefficient of the result contains information about a single subkey. So we only have to consider 7 possible subkey values per coefficient. However, if $u = u_0 + u_1X$, then $s \cdot u = s_0u_0 - s_{255}u_1 + (s_0u_1 + s_1u_0)X + \cdots + (s_{255}u_0 + s_{254}u_1)X^{255}$. In this case, every coefficient of the result contains information about two subkeys. Then the number of options that need to be considered is $7 \cdot 7 = 49$. This number will increase exponentially for the number of non-zero coefficients in u.

One effective method of rendering the attack infeasible would be to disallow an attacker to provide chosen ciphertexts altogether. In practical settings, however, this is difficult to realize. If an attacker has access to a physical device, then it is likely that the attacker is also able to perform man-in-the-middle attacks and thus provide maliciously crafted ciphertexts. Maybe it is possible to prevent the decryption of malicious ciphertexts by including additional cryptography that verifies the integrity and authenticity of ciphertexts. This however adds additional complexity to the design of an implementation. Furthermore, it is not immediately obvious how such a mechanism can be integrated into Kyber.

A more direct approach is to reduce the ciphertext space slightly by disallowing ciphertexts that would result in leakage corresponding to single subkeys or small

combinations of subkeys. One possible method to achieve this is by limiting the number of zero coefficients allowed in the ciphertext polynomials. Introducing this requirement for ciphertexts introduces a trade-off between security and ciphertext space. The practical implementation of this method is not immediately obvious. One possible approach could involve repeating the encapsulation procedure until a ciphertext is generated that satisfies the desired security requirement. However, such an implementation falls outside of the scope of this thesis and will be left as future work.

5.2 Software Countermeasures

5.2.1 Countermeasure 1: Random Delays

One of the main requirements for the attack proposed in chapter 4 is that an attacker can correlate the leakage of instructions belonging to different traces. In the critical section, the leakage at a time point corresponds to the same instruction for all obtained traces. This makes it trivial for an attacker to correlate the power consumption across different traces. By inserting random delays it is significantly harder to correlate traces in the time dimension as the synchronization between traces is broken. One method of adding random delays is the insertion of dummy instructions. Dummy instructions do not alter the result of the output in any way. They will however induce leakage at random points in time. Now an attacker has the additional challenge of aligning traces such that critical instructions are executed at the same point in time. A requirement for implementing random delays is that the software has access to some form of (pseudo) randomization. Embedded systems that perform cryptographic operations often offer some method of acquiring (pseudo) random numbers. Implementation of random delays should also make sure that the frequency and length of delays are chosen appropriately. For example, long delays that only appear occasionally are easy to detect [59]. Improved floating means [60] is one technique that has been proposed to generate delays with an appropriate length depending on the desired amount. Naturally, the addition of delays will increase the execution time of the algorithm. The performance overhead might increase significantly depending on the frequency and length of these delays. The induced performance overhead however is quite small compared to other suggested methods such as masking.

As a proof of concept, random delays have been simulated by inserting dummy instructions into the assembly of the critical section. Figure 5.1 shows the complete decryption procedure with and without delays. In this example, there was a

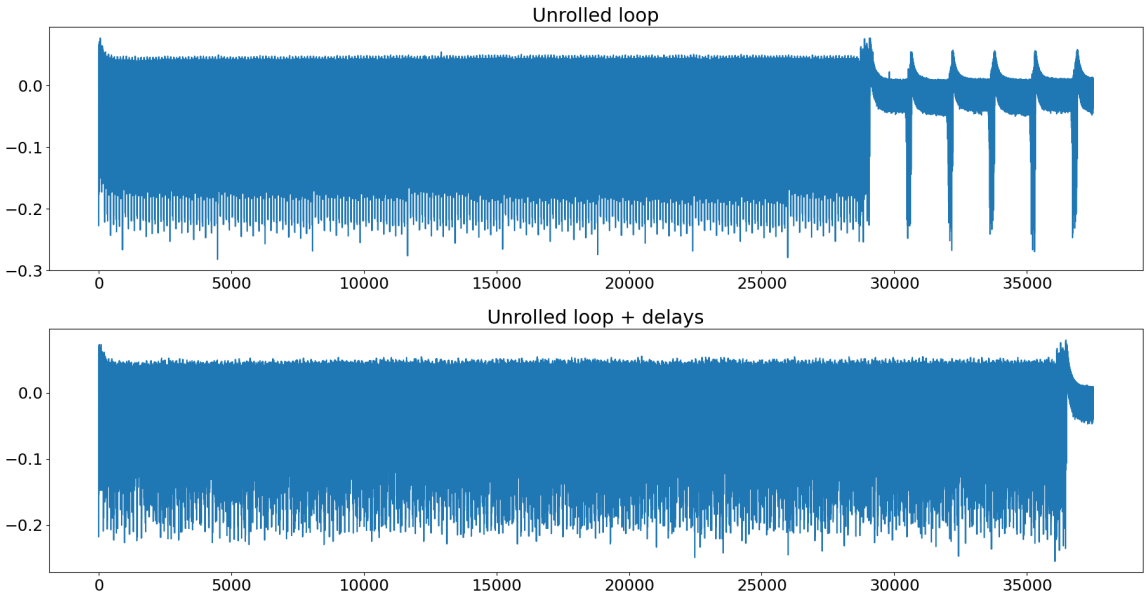


Figure 5.1: Traces without and with random delays

50% chance of delays being added after every non-delay instruction. The number of dummy instructions added was taken uniformly from $\{0, 4\}$. Figure 5.2 shows a closeup of the decryption procedure. What is noticeable, is that the trace without delays seems significantly more periodic. It is easier to pick out the repeating pattern that is a result of the loop iterations. This pattern is also present in the trace containing the delays, but it is significantly harder to see. The pattern is hidden by noise in the time dimension. This example has relatively few delays. When more delays are added the pattern becomes increasingly harder to find.

5.2.2 Countermeasure 2: Shuffling

The second proposed zero-overhead software-based countermeasure is random shuffling. In the critical section, the coefficients in $r[j]$ are operated on in order from $j = 0 \ldots 255$. When an appropriate ciphertext is chosen, leakage corresponding to the j'th or $(j + 256)$'th subkey is present. These leakages will present themselves in the same order that the loop is executed. Execution of the critical section will always leak information about the subkeys in a predictable order. Random shuffling of the loop order for every execution prevents this predictable behaviour. Now an attacker has the difficult task of identifying to which subkey a loop iteration belongs. This countermeasure also requires access to randomization but is otherwise easy to implement. Furthermore, this countermeasure can be implemented side-by-side with random delays without needing to make any additional adjustments. This countermeasure has also been simulated by altering the assembly of the critical section.

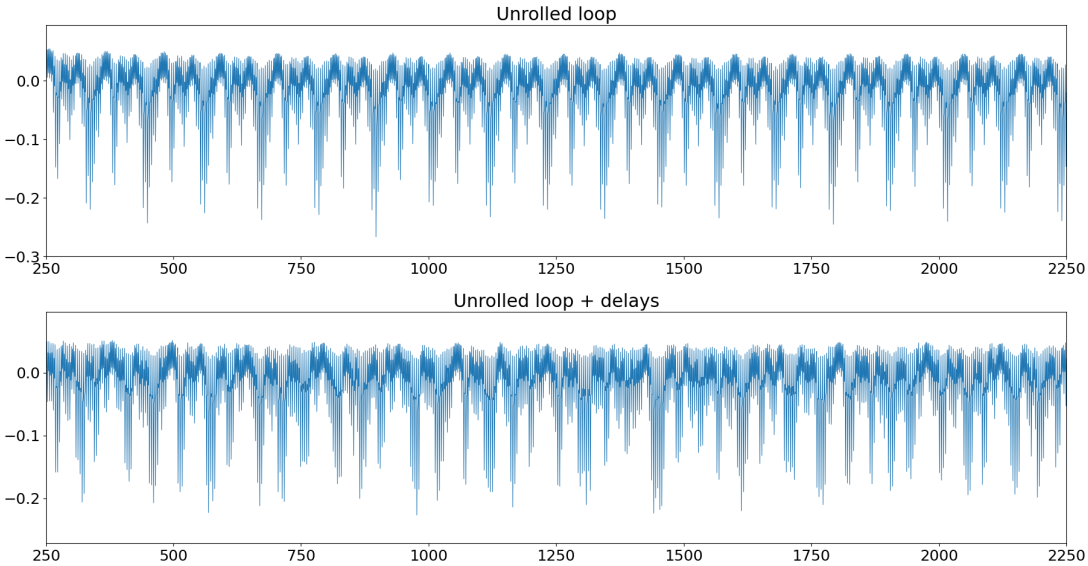


Figure 5.2: Traces without and with random delays (close-up)

The inclusion of this countermeasure alone breaks the proposed CPA attack.

5.3 Considering SCA during Implementation Development

The previously mentioned countermeasures are not the only possible countermeasures. Other powerful methods also exist, such as masking or hardware-based techniques. Random delays and shuffling have been suggested, because of their flexibility and low-performance overhead. Software-based countermeasures can be more easily patched than hardware, increasing flexibility. Masking is a very powerful countermeasure technique but comes at a high cost of performance. In the setting of the previous attack, the presence of random delays and shuffling easily prevents the proposed CPA attack. Both countermeasures do not remove secret information from the traces. They only aim at creating additional challenges for a possible adversary. After the inclusion of random delays, an adversary must find a way to synchronize different traces. After the inclusion of shuffling, an adversary must find a way of identifying the index of subkeys. These countermeasures are effective for non-profiled SCA attacks. When dealing with stronger profiled single-trace SCA attacks, these countermeasures might not prove sufficient anymore.

The findings of the previous chapters show that it is of paramount importance that every implementation of Kyber chooses appropriate countermeasures against

SCA. Various aspects can be considered when making design choices to strengthen against SCA. These areas include the cryptographic algorithm design, software implementations and underlying hardware components. The selection of appropriate countermeasures relies not only on the capabilities and limitations of the hardware and acceptable overhead but also on the security guarantees that the application needs to provide. It is important to acknowledge that there is no perfect countermeasure that can fully eliminate the risk of SCA and therefore it is the responsibility of designers to carefully choose the most suitable countermeasure for their specific scenario.

An intriguing prospect is the potential for RISC-V to enable the development of effective countermeasures. The open collaborative nature of RISC-V might result in the development of novel countermeasures that integrate tightly with the processor core. Currently, PQC instruction set extensions are being discussed. It will be interesting to see how SCA considerations might influence this discussion. Another prospect of RISC-V is the design of SCA-resilient hardware accelerators that integrate tightly with the ISA. Generally, hardware implementations are considered more difficult to attack compared to software implementations [43]. Hardware accelerators that can safely perform security-sensitive cryptographic operations might prove a valuable asset for implementing PQC.

Chapter 6

Discussion

In this chapter, the previous research results are analyzed and discussed. A summary of the found results and the results are interpreted. Furthermore, the implications of the results are discussed in light of the research questions. Finally, the limitations of the performed research are discussed.

6.1 Summary

This study performed SCA on the PQCLEAN implementation of Kyber on both an ARM and RISC-V based MCU.

In chapter 3, SPA was performed to identify potential leaks of sensitive information. It was found that there was indeed leakage present on the ARM MCU. Secret-key partitions could be identified visually. It was unclear, however, whether the observed leakage could be exploited through SPA techniques. Contrary to the ARM case, On the RISC-V MCU this behaviour was not present.

In chapter 4, a more sophisticated SCA technique was applied, namely CPA. CPA was performed on the critical section at the end of the inverse-NTT operation. The results demonstrated the critical section leaked side-channel information on both MCUs. A strong correlation was found between the power consumption and the output of a function called $fqmul$. Exploiting this weakness. a chosen-ciphertext CPA attack was constructed. The developed attack was able to recover the full secret-key in 96 and 48 traces for the ARM and RISC-V MCU respectively.

Chapter 5 discusses possible low-cost software-based countermeasures that could be integrated in order to mitigate the attack developed in the previous chapter. Both countermeasures proved to be effective at preventing the attack.

6.2 Interpretation of Results

One of the research objectives was to see how vulnerable RISC-V Kyber implementations are to SCA. This research objective has been partly addressed by focusing on a RISC-V Kyber implementation of PQCLEAN. It is evident that without any countermeasures, Kyber is highly vulnerable to SCA attacks. The discernible difference between secret-key values in the power trace belonging to the ARM MCU by visual inspection alone indicates significant information leakage through the power side-channel. The CPA attack demonstrated that an attacker can easily recover the secret-key of a decryption device with very few traces. Normally in terms of SCA attacks millions of traces are considered, but the proposed attack requires less than a hundred traces, emphasizing the susceptibility of Kyber implementations to SCA attacks.

The target that was considered in this attack did not contain countermeasures. An implementation that has countermeasures integrated will be significantly harder to attack. The main thing to take away from the results is that any implementation of Kyber (especially for IoT or safety-critical applications) need to consider the SCA characteristics. As possible SCA attacks can form a profound risk to the security of the implementation.

One fundamental aspect of the Kyber algorithm, which makes it vulnerable to SCA is the small subkey space. For every subkey, there are only seven possible guesses available. If an adversary has access to leakage corresponding to a single subkey, then he has a very small key space to explore. This is fundamental to the algorithm and will always be the case. One possible method of counteracting this is by preventing the exposure of isolated subkeys, but this might be difficult to realize in practice.

The SCA characteristics were analyzed on MCUs based on both the ARM and RISC-V ISAs. Contrary to the ARM case, the visual inspection did not reveal significant leakage on the RISC-V MCU. The reason for this differing behaviour is difficult to determine without further testing. It could be a result of different ISAs, underlying hardware components, or differences in executed instructions. The STM32F303 for example has a single-cycle multiplication unit. Whereas the FE310-G002 has a multi-cycle multiplication unit. Despite the lack of visual leakage in the RISC-V case, leakage was still present, as demonstrated by chapter 4. The CPA attack required about half the number of traces to recover the secret-key for the RISC-V MCU compared to the ARM MCU.

The research also highlights that SCA characteristics observed in one MCU implementation can be applicable to a different MCU with a completely different architecture. The proposed attack worked for both MCUs without requiring any adaptations. This is not entirely surprising since instructions dealing with memory, for example, might still be executed on similar hardware structures.

Another research objective was to see which countermeasures could be implemented on a RISC-V Kyber implementation. This study demonstrated that the addition of relatively simple, low-cost software-based countermeasures can significantly increase the difficulty of SCA attacks. The countermeasures can be considered low-cost because they do not induce a large performance overhead. Also, an algorithmic countermeasure was suggested aimed at preventing the occurrence of leakage that corresponds to single subkeys. It is important to acknowledge that the presence of countermeasures does not guarantee the complete mitigation of attacks. The ever-evolving landscape of side-channel attack techniques suggests the existence of sophisticated attacks that may render these countermeasures less effective.

Similar to the broader field of cyber security, SCA finds itself caught in an arms race. As countermeasures are developed to increase the difficulty of SCA, new attack strategies are devised. This dynamic underscores the need for ongoing research, robust countermeasure development, and robust evaluation of the resilience of cryptographic implementations against SCA attacks. Maintaining a proactive approach is essential to the future of cryptographic implementations.

6.3 Limitations

In this section, the limitations of the research are discussed.

Firstly, it is important to consider that the targeted PQCLEAN library is meant to serve as a **clean reference implementation**. In real-world scenarios, a more realistic target will most likely be optimized for a specific platform and may incorporate several countermeasures. Furthermore, the power traces obtained in this study came from an environment optimized for SCA. When dealing with a real-world target, the power traces might contain significantly more noise. The decrease in SNR can make SCA more difficult. However, it is worth noting that a decrease in SNR can be counteracted by obtaining more traces, as in most cases, noise can be assumed

to follow the normal distribution.

While this research successfully demonstrates the leakage in MCU implementations of Kyber, it **does not identify which instructions are responsible for the sensitive side-channel leakage**. A clear understanding of the root cause of the identified leakage is lacking. One possible hypothesis is that memory instruction contributes the most significant amount of information leakage. The argument is, that data movement over buses results in the highest amount of power being consumed. Further investigation is needed to determine the precise factors contributing to the observed side-channel leakage.

The analysis of Kyber was not extensive. Only a small section of the Kyber algorithm has been analyzed. The critical section described in algorithm 7 is not the only section of the algorithm that can potentially leak critical information. Montgomery reduction is also performed in other parts of the Kyber algorithm. These could also be potentially exploited as the same leakage will be present. Then there are also the other operations performed in Kyber which have not been analyzed. These operations can also induce possibly exploitable side-channel leakage. Introducing avenues for future research. Furthermore, only two RISC-V Kyber implementations were investigated. An unprotected implementation and an implementation with countermeasures. We therefore cannot make any concrete statements about the SCA characteristics of other RISC-V implementations. To answer **Rq 1** more comprehensively, other RISC-V Kyber implementations should also be studied.

The investigation into the exact number of traces required for the attack has not been thoroughly explored in this study. Typically, an attack's effectiveness is measured with many trials. This process entails capturing a large volume of power traces and performing a significant amount of analysis. This requires a significant amount of time. In order to provide more precise quantification of the attack's effectiveness, further analysis is required. given this limitation, the detailed discussion regarding the disparity in the required number of traces between the ARM and RISC-V MCUs has not been covered. By conducting additional research and analysis, it would be possible to delve into the underlying factors contributing to this discrepancy. It would be valuable knowledge to know why certain architectures or implementations cause more side-channel leakage.

In conclusion, while this research sheds light on the vulnerabilities and leakage present implementations of Kyber, it is important to recognize the limitations of the study. There are aspects where the study can be improved upon, such as acquir-

ing a deeper understanding of the root causes for leakage and performing a more elaborate analysis of the full algorithm.

Chapter 7

Conclusions and recommendations

7.1 Conclusions

To ensure secure future-proof cryptography, it is important to consider the SCA characteristics of its implementations. One of the most promising PQC KEM is Kyber. However, as highlighted in chapters 3 and 4, the decryption procedure of Kyber can be susceptible to sensitive side-channel leakage. Power side-channel leakage was identified in the Montgomery reduction procedure on a clean reference implementation of Kyber on a RISC-V MCU.

A novel chosen-ciphertext CPA has been proposed that exploits the identified side-channel leakage. This attack has been experimentally proven to be successful on an ARM and RISC-V MCU. In both cases, less than 100 traces were required for full secret-key recovery on Kyber512. These findings highlight the SCA susceptibility of unprotected Kyber implementations. Leakage from a RISC-V MCU did not significantly differ from leakage taken from an ARM MCU.

In response to these findings, chapter 5 proposes two low-cost software-based countermeasures to prevent the previously introduced attack on Kyber. The suggested countermeasures implement random delays and shuffling in order to destroy the synchronization of traces. The countermeasures are easy to implement and significantly increase the difficulty of performing the proposed attack. The countermeasures have been implemented both on an ARM and RISC-V-based MCU. The addition of these countermeasures is effective at preventing the proposed CPA attack.

Overall, these results show the importance of considering SCA when designing PQC implementations. Furthermore, it shows the necessity of including appropriate countermeasures to counteract side-channel attacks.

7.2 Recommendations

In this section recommendations are given on how to improve this research. Also, suggestions are given for future research directions.

Improving the conducted research

This study was quite limited in its scope. It focused on a small part of the Kyber algorithm. One way to make the research more extensive is to perform SCA on other parts of the algorithm. This could be done effectively by using SCA metrics that have been designed to indicate side-channel leakage, such as NICV or TVLA. Furthermore, a clean reference implementation is not a realistic target for SCA attacks. To make the threat scenario more realistic one could investigate the leakage of RISC-V optimized implementations. The effectiveness of the proposed countermeasures was only experimentally verified. Their effectiveness could be better supported with the use of the previously mentioned metrics.

Future research suggestions

There are multiple avenues for future research on the SCA resilience of RISC-V implementations of Kyber.

SCA resilience of RISC-V PQC accelerators Several hardware accelerators leveraging RISC-V have been proposed in literature (some with SCA in mind) [55], [61]–[63]. These accelerators should also be analyzed with respect to side-channel analysis.

New mitigation strategies Another direction could be to look for new mitigation strategies against side-channel attacks on Kyber for RISC-V. One can wonder if the RISC-V ISA offers any advantages in regards to preventing side-channel leakage compared to other closed ISAs. This is a relevant question that might aid in protecting PQC candidates from side-channel attacks.

Other side-channel attacks In this thesis only the power side-channel was considered and exploited with a non-profiled attack. Many other possible side-channel attacks can be performed. Different types of side-channels can be considered. Different attack techniques such as non-profiled to profiled, but also non-invasive to

invasive. Future research could investigate how other attacks could possibly exploit side-channels on RISC-V Kyber implementations.

SCA of other PQC candidates The future is unpredictable. There is no guarantee that Kyber will be the replacement for RSA in the future. Taking this into account it also makes sense to perform SCA on RISC-V implementations for the other PQC candidates.

7.3 Contributions

During this study, two contributions were made to SCA platforms.

Firstly, a bug fix was provided to the ChipWhisperer platform. The HAL that ChipWhisperer provided did not initialize the UART correctly, leading to baud rate errors. The thread discussing this issue discussion can be found here.

Secondly, a SCA environment was developed for the CAES research group. This environment provides a sandbox that aids in performing power SCA on the FE310-G002 MCU. Furthermore, the environment comes with a tutorial that hopefully will quickly familiarize future researchers with SCA on the FE310-G002 using the sandbox environment.

Bibliography

[1] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, pp. 120–126, 2 1978. [Online]. Available: https://dl-acm-org.ezproxy2.utwente.nl/doi/10.1145/359340.359342

[2] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *https://doi.org/10.1137/S0097539795293172*, vol. 26, pp. 1484–1509, 7 2006. [Online]. Available: https://epubs.siam.org/doi/10.1137/S0097539795293172

[3] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-dilithium: A lattice-based digital signature scheme,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, pp. 238–268, 2 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839

[4] D. J. Bernstein, R. Niederhagen, A. Hülsing, J. Rijneveld, S. Kölbl, and P. Schwabe, “The sphincs+ signature framework,” *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 2129–2146, 11 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3319535.3363229

[5] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, “Falcon.” [Online]. Available: https://falcon-sign.info/

[6] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-kyber algorithm specifications and supporting documentation (version 3.01),” 2021.

[7] M. Randolph and W. Diehl, “Power side-channel attack analysis: A review of 20 years of study for the layman,” pp. 1–33, 6 2020, a good survey about SCA using power analysis.

[8] “Iot connected devices by vertical 2030 — statista,” statistic on number of IoT devices¡br/¿. [Online]. Available: https://www.statista.com/statistics/1194682/iot-connected-devices-vertically/

[9] A. Ukil, J. Sen, and S. Koilakonda, “Embedded security for internet of things,” *Proceedings - 2011 2nd National Conference on Emerging Trends and Applications in Computer Science, NCETACS-2011*, pp. 50–55, 2011.

[10] F. Regazzoni, “Physical attacks and beyond,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10532 LNCS, pp. 3–13, 2017.

[11] A. Karlov and N. L. de Guertechin, “Power analysis attack on kyber,” *Cryptology ePrint Archive*, 2021.

[12] Z. Xu, O. Pemberton, S. S. Roy, D. Oswald, W. Yao, and Z. Zheng, “Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber,” *IEEE Transactions on Computers*, vol. 71, pp. 2163–2176, 9 2022.

[13] Y. Yang, Z. Wang, J. Ye, J. Fan, S. Chen, H. Li, X. Li, and Y. Cao, “Chosen ciphertext correlation power analysis on kyber,” *Integration*, vol. 91, pp. 10–22, 7 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167926023000378

[14] R. C. Rodriguez, F. Bruguier, E. Valea, and P. Benoit, “Correlation electromagnetic analysis on an fpga implementation of crystals-kyber,” *Cryptology ePrint Archive*, 2022.

[15] S. Bhasin, J. L. Danger, S. Guilley, and Z. Najm, “Nicv: Normalized inter-class variance for detection of side-channel leakage,” *IEEE International Symposium on Electromagnetic Compatibility*, vol. 2014-December, pp. 310–313, 12 2014.

[16] T. Schneider and A. Moradi, “Leakage assessment methodology a clear roadmap for side-channel evaluations,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9293, pp. 495–513, 2015. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-48324-4_25

[17] PapagiannopoulosKostas, GlamočaninOgnjen, AzouaouiMelissa, RosDorian, RegazzoniFrancesco, and StojilovićMirjana, “The side-channel metrics cheat sheet,” *ACM Computing Surveys*, vol. 1, 2 2023. [Online]. Available: https://dl-acm-org.ezproxy2.utwente.nl/doi/10.1145/3565571

[18] D. Hofheinz, K. Hövelmanns, and E. Kiltz, “Crystals - kyber: A cca-secure module-lattice-based kem,” *Proceedings - 3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*, pp. 353–367, 7 2018.

[19] O. Regev, “The learning with errors problem,” *Proceedings of the Annual IEEE Conference on Computational Complexity*, pp. 191–204, 2010.

[20] ——, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM (JACM)*, vol. 56, 9 2009. [Online]. Available: https://dl-acm-org.ezproxy2.utwente.nl/doi/10.1145/1568318.1568324

[21] P. L. Montgomery, “Modular multiplication without trial division,” *Mathematics of Computation*, vol. 44, pp. 519–521, 1985. [Online]. Available: https://www.ams.org/mcom/1985-44-170/S0025-5718-1985-0777282-X/

[22] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of Computer and System Sciences*, vol. 28, pp. 270–299, 4 1984.

[23] C. Rackoff and D. R. Simon, “Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 576 LNCS, pp. 433–444, 1992. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-46766-1_35

[24] E. Fujisaki and T. Okamoto, “Secure integration of asymmetric and symmetric encryption schemes,” *Journal of Cryptology 2011 26:1*, vol. 26, pp. 80–101, 12 2011. [Online]. Available: https://link.springer.com/article/10.1007/s00145-011-9114-1

[25] D. Hofheinz, K. Hövelmanns, and E. Kiltz, “A modular analysis of the fujisaki-okamoto transformation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10677 LNCS, pp. 341–371, 2017.

[26] A. Sprenkels, “The kyber/dilithium ntt.” [Online]. Available: https://electricdusk.com/ntt.html

[27] “Number-theoretic transform (integer dft).” [Online]. Available: https://www.nayuki.io/page/number-theoretic-transform-integer-dft

[28] M. J. Kannwischer, P. Schwabe, D. Stebila, and T. Wiggers, “Improving software quality in cryptography standardization projects,” in *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops,*

Genoa, Italy, June 6-10, 2022. Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 19–30. [Online]. Available: https://eprint.iacr.org/2022/337

[29] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking nist pqc on arm cortex-m4," *Cryptology ePrint Archive*, 2019. [Online]. Available: https://libopencm3.org/

[30] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1109, pp. 104–113, 1996.

[31] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1666, pp. 388–397, 1999.

[32] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3156, pp. 16–29, 2004.

[33] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1666, pp. 398–412, 1999. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-48405-1_26

[34] J. S. Coron and L. Goubin, "On boolean and arithmetic masking against differential power analysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1965 LNCS, pp. 231–237, 2000. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-44499-8_18

[35] "Risc-v international." [Online]. Available: https://riscv.org/

[36] B. Y. Sim, J. Kwon, J. Lee, I. J. Kim, T. H. Lee, J. Han, H. Yoon, J. Cho, and D. G. Han, "Single-trace attacks on message encoding in lattice-based kems," *IEEE Access*, vol. 8, pp. 183 175–183 191, 2020.

[37] J. Hermelink, P. Pessl, and T. Pöppelmann, "Fault-enabled chosen-ciphertext attacks on kyber," *Cryptology ePrint Archive*, 2021.

[38] J. Delvaux, "Roulette: A diverse family of feasible fault attacks on masked kyber," *IACR Transactions on Cryptographic Hardware and Embedded*

Systems, vol. 2022, pp. 637–660, 8 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9835

[39] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10529 LNCS, pp. 513–533, 2017. [Online]. Available: https://link-springer-com.ezproxy2.utwente.nl/chapter/10.1007/978-3-319-66787-4_25

[40] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11774 LNCS, pp. 130–149, 2019.

[41] M. Hamburg, J. Hermelink, R. Primas, S. Samardjiska, T. Schamberger, S. Streit, E. Strieder, and C. van Vredendaal, "Chosen ciphertext k-trace attacks on masked cca2 secure kyber," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, pp. 88–113, 8 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9061

[42] J. Mu, Y. Zhao, Z. Wang, J. Ye, J. Fan, S. Chen, H. Li, X. Li, and Y. Cao, "A voltage template attack on the modular polynomial subtraction in kyber," *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, vol. 2022-January, pp. 672–677, 2022.

[43] R. Wang, K. Ngo, and E. Dubrova, "Making biased dl models work: Message and key recovery attacks on saber using amplitude-modulated em emanations," *Cryptology ePrint Archive*, 2022.

[44] C. Mujdei, A. Beckers, J. M. B. Mera, A. Karmakar, L. Wouters, and I. Verbauwhede, "Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication," *Cryptology ePrint Archive*, 2022.

[45] H. Ma, S. Pan, Y. Gao, J. He, Y. Zhao, and Y. Jin, "Vulnerable pqc against side channel analysis - a case study on kyber," pp. 1–6, 1 2023.

[46] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based pke and kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, pp. 307–335, 2020.

[47] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "Drop by drop you break the rock - exploiting generic vulnerabilities in lattice-based pke/kems using em-based physical attacks," *Cryptology ePrint Archive*, 2020.

[48] Y. Tanaka, R. Ueno, K. Xagawa, A. Ito, J. Takahashi, and N. Homma, "Multiple-valued plaintext-checking side-channel attacks on post-quantum kems," *Cryptology ePrint Archive*, 2022.

[49] R. Ueno, K. Xagawa, Y. Tanaka, A. Ito, J. Takahashi, and N. Homma, "Curse of re-encryption: A generic power/em analysis on post-quantum kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, pp. 296–322, 11 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9298

[50] G. Rajendran, P. Ravi, J.-P. D'Anvers, S. Bhasin, and A. 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*, vol. 2023, pp. 418–446, 3 2023. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/10289

[51] M. J. Kannwischer, P. Schwabe, D. Stebila, and T. Wiggers, "Improving software quality in cryptography standardization projects," *Cryptology ePrint Archive*, 2022.

[52] P. Ravi, R. Poussier, S. Bhasin, and A. 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," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12586 LNCS, pp. 123–146, 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-66626-2_7

[53] A. Jati, N. Gupta, A. Chattopadhyay, and S. K. Sanadhya, "A configurable crystals-kyber hardware implementation with side-channel protection," *Cryptology ePrint Archive*, 2021.

[54] T. Kamucheka, A. Nelson, D. Andrews, and M. Huang, "A masked pure-hardware implementation of kyber cryptographic algorithm," *FPT 2022 - 21st International Conference on Field-Programmable Technology, Proceedings*, 2022.

[55] T. Fritzmann, M. V. Beirendonck, D. B. Roy, P. Karl, T. Schamberger, I. Verbauwhede, and G. Sigl, "Masked accelerators and instruction set extensions for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, pp. 414–460,

2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9303

[56] D. Heinz, M. J. Kannwischer, G. Land, T. Pöppelmann, P. Schwabe, and D. Sprenkels, "First-order masked kyber on arm cortex-m4," *Cryptology ePrint Archive*, 2022. [Online]. Available: https://github.com/

[57] J. S. Coron, F. Gérard, S. Montoya, and R. Zeitoun, "High-order table-based conversion algorithms and masking lattice-based encryption," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, pp. 1–40, 2 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9479

[58] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[59] J. S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5747 LNCS, pp. 156–170, 2009. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-04138-9_12

[60] ——, "Analysis and improvement of the random delay countermeasure of ches 2009," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6225 LNCS, pp. 95–109, 2010. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15031-9_7

[61] T. Fritzmann, G. Sigl, and J. Sepúlveda, "Risq-v: Tightly coupled risc-v accelerators for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, pp. 239–280, 8 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8683

[62] G. Xin, J. Han, T. Yin, Y. Zhou, J. Yang, X. Cheng, and X. Zeng, "Vpqc: A domain-specific vector processor for post-quantum cryptography based on risc-v architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, pp. 2672–2684, 8 2020.

[63] J. Lee, W. Kim, S. Kim, and J. H. Kim, “Post-quantum cryptography coprocessor for risc-v cpu core,” *2022 International Conference on Electronics, Information, and Communication, ICEIC 2022*, 2022.

Appendix A

Computing 'compression resistant' ciphertext coefficient values.

```
1 import kyber512 as kyber
2
3 ffi = kyber.ffi
4 polyvec = ffi.new('polyvec *')
5 r = ffi.new('uint8_t[KYBER_POLYVECCOMPRESSEDBYTES]')
6
7 bijections = list()
8
9 for a in range(-3328,3329):
10     polyvec.vec[0].coeffs[0] = a
11     kyber.PQCLEAN_polyvec_compress(r, polyvec)
12     kyber.PQCLEAN_polyvec_decompress(polyvec, r)
13     if polyvec.vec[0].coeffs[0] == a:
14         bijections.append(a)
15
16 print(bijections)
```