Exploiting the Central Reduction in Lattice-Based Cryptography

Tolun Tosun[1], Amir Moradi[2] and Erkay Savas[1]

[1] Sabanci University, Istanbul, Türkiye
{toluntosun,erkays}@sabanciuniv.edu
[2] Technische Universität Darsmtadt, Darmstadt, Germany
amir.moradi@tu-darmstadt.de

Abstract. This paper questions the side-channel security of **central reduction** technique, which is widely adapted in efficient implementations of Lattice-Based Cryptography (LBC). We show that the central reduction leads to a vulnerability by creating a strong dependency between the power consumption and the sign of sensitive intermediate values. We exploit this dependency by introducing the novel **absolute value** prediction function, which can be employed in higher-order non-profiled multi-query Side-Channel Analysis (SCA) attacks. Our results reveal that – compared to classical reduction algorithms – employing the central reduction scheme leads to a two-orders-of-magnitude decrease in the number of required SCA measurements to exploit secrets of masked implementations. We particularly show that our approach is valid for the prime moduli employed by Kyber and Dilithium, the lattice-based post-quantum algorithms selected by NIST. We practically evaluate our introduced approach by performing second-order non-profiled attacks against an open-source masked implementation of Kyber on an ARM Cortex-M4 micro-processor. In our experiments, we revealed the full secret key of the aforementioned masked implementation with only **250** power traces without any forms of profiling or choosing the ciphertexts.

Keywords: Side-Channel Analysis · Correlation Power Analysis · Post-Quantum Cryptography · Kyber · Dilithium · Plantard · Montgomery · Arithmetic Masking · Centered Reduction

1 Introduction

Shor's algorithm [Sho94] violates the security of traditional public-key cryptography including RSA and ECC through quantum computing. As the development of a larger quantum computer in the number of qubits is being reported each year, the quantum threat gradually becomes a reality. On the other hand, NIST's post-quantum cryptography contest is in the fourth round, with already selected algorithms. Among the winners, the lattice-based algorithms form the majority: Kyber [SAB+22], Dilithium [LDK+22] and Falcon [PFH+22]. Although the post-quantum algorithms can resist quantum computing attacks, special attention should be paid to Side-Channel Analysis (SCA) attacks [KJJ99, BCO04] when these algorithms are implemented in both hardware and software.

The core operation in Lattice-Based Cryptography (LBC) is the polynomial multiplication. For an efficient implementation, the Number Theoretic Transform (NTT) stands out as an excellent approach. NTT is indeed a special form of the Fast Fourier Transform (FFT) that operates on a discrete space. An important building block that significantly impacts the efficiency of the NTT algorithm is the modular reduction of integers, concerning the arithmetic for coefficients of polynomials. Classical techniques such

as the Montgomery reduction [Mon85] and Barrett reduction [Bar87] are already applied to LBC by the existing literature [AHKS22, GKS21, ABCG20, BKS19, Sei18]. The same holds for the relatively new Plantard reduction scheme [Pla21, HZZ+22]. One important distinction regarding the integer reduction in LBC compared to the RSA and ECC is the bit-length of the number to be reduced. LBC requires a reduction of relatively smaller numbers, that usually fit into a single computer word. For instance, Kyber employs a 12-bit coefficient modulus and Dilithium a 23-bit one. Moreover, the signed representation of integers over a modulus instead of the classical unsigned representation is more desired in LBC [HZZ+22, AHKS22, GKS21, ABCG20, BKS19]. That is to make a central reduction (a.k.a *centered* reduction) to the range $[-q/2, q/2]$ instead of $[0, q)$ for an odd modulus q. The Plantard and Montomgery algorithms enable 2-cycle implementation of central reduction on the ARM Cortex-M4 [HZZ+22, GKS21, ABCG20] while Plantard is superior since the output of Montgomery reduction requires an additional subtraction or addition to be correct.

Main Motivation. Overall, the central reduction improves the efficiency of LBC implementations; however, it also creates new possibilities for SCA attacks. A well-known fact regarding the processors realized by CMOS technology is that the power consumption has a relatively strong dependency on the Hamming weight (HW) of the processed data. Respectively, the sign of a number in $[-q/2, q/2]$ becomes the dominant factor influencing its power consumption considering 2's complement to represent negative numbers. Motivated by this, we explore the characteristics of the central reduction in terms of SCA leakage and exploit them, particularly in the presence of a masking countermeasure.

Related Work on Masking LBC. It is well known that masking countermeasures provide a promising way of mitigating EM-/power-based SCA attacks. Accordingly, the existing literature on masking LBC is already quite rich. Some examples applied on Kyber [HKL+22, BGR+21, FBR+22, ÖY23], Dilithium [MGTF19, ABC+23b, CGTZ23], Saber (another promising post-quantum lattice-based Key Encapsulation Mechanism (KEM)) [BDK+21, KDVB+22] and generic lattice-based encryption [RRd+16, BC22, CGMZ23]. Indeed, masking the polynomial arithmetic is considered trivial and is achieved by simply repeating the operations. For instance, a polynomial multiplication $a \cdot b$ can be performed through the random shares a^0 and a^1 individually while satisfy $a = a^0 + a^1$, instead of accessing a in plain. On the other hand, masking the non-linear components of algorithms involves specialized techniques. One simple example is the compression operation in Kyber, which aims to identify the interval each coefficient of a given polynomial a resides. It is easy to see that such an operation cannot be performed independently on the arithmetic shares, as it is the case with polynomial multiplication. A masked implementation of such non-linear operations typically utilizes arithmetic-to-Boolean mask conversion [RRd+16, BGR+21, FBR+22, BC22, KDVB+22, BDK+21, HKL+22] while a Boolean masking scheme ensures $a = a^0 \oplus a^1$. Overall, the existing work regarding masking LBC aims to achieve provable first- or higher-order security by introducing more efficient solutions for masking the non-trivial parts of the algorithms. Particularly, current masked implementations on the ARM Cortex-M4 [HKL+22, BGR+21, ABC+23b, BDK+21, HDR23] inherit the polynomial arithmetic from the well-known pqm4 library [KRSS19a], known for providing state-of-the-art yet unprotected implementations of post-quantum algorithms. To the best of our knowledge, there has been no work questioning the difficulty of possible SCA attacks on the linear parts as long as the implementations are proven to be secure in the desired security order.

Related Work on SCA Attacks against LBC. On the other hand, there exist many SCA attacks in the literature that target implementations of LBC. They can be seen in

two distinct classes: profiled SCA attacks such as [PPM17, KLH+20, XPR+21, BAE+24, BNGD23, DNGW23, MUTS22, KAA22, AAT+21] and non-profiled SCA attacks such as [MWK+24, CKA+21, TS24, SLKG22]. The non-profiled attacks commonly target polynomial multiplication for which a secret polynomial is multiplied with a publicly known polynomial that changes based on the input given to the victim implementation. In [MWK+24], the authors show that the performance of non-profiled attacks against the polynomial multiplication directly depends on the employed multiplication algorithm as well as on the parameters such as the coefficient modulus. While for some instances of LBC, the non-profiled attacks can take a relatively long time to retrieve secret polynomials, acceleration is possible in certain scenarios by collecting more measurements, *e.g.* as shown in [CKA+21, TS24] for distinct implementations. The authors of [TS24] particularly focused on so-called *incomplete* NTT, a special case of NTT that is employed in efficient implementations of both Kyber and Dilithium on the ARM Cortex-M4 [KRSS19a]. Besides, it is shown in [SLKG22] that the polynomial multiplication can be also effectively targeted in the case of hardware implementations. Except [TS24], the aforementioned non-profiled attacks target unprotected implementations, *i.e.* not masked. On the other hand, profiled attacks demand for a stronger adversary model. Particularly, a freely accessible device for profiling that is identical to the victim must be available. Nevertheless, the majority of published SCA attacks on LBC benefit from a profiling phase. To this end, several operations of LBC have been selected as the target of these attacks. Among them, [PPM17, XPR+21, KLH+20] focus on the NTT transformation. The attack presented in [PPM17] is distinctive by revealing *single-trace* vulnerabilities of LBC. It should be emphasized that if the masks during the profiling phase are known (*i.e.* they are also considered in the profiles), single-trace profiling attacks cannot be avoided by masking, since SCA leakage of a single execution is measured, whehre the masks can also be revealed via the profiles. The authors of [BAE+24] specifically focus on the multiplication of polynomials with small coefficients. The works [BNGD23, DNGW23, MUTS22] present attacks on encoding/decoding functions that transform binary input into a polynomial or vice versa. We should highlight that [DNGW23] presents only a message recovery attack that aims to retrieve the decapsulated message rather than the secret key. The target of [KAA22] is the sampling of challenge polynomials, which have a limited number of non-zero coefficients that are also small in magnitude. Table 1 formalizes the above discussion and positions our study among the attacks existing in the literature.

Our Contributions. Below is a list of the contributions we have made in this work.

- To the best of our knowledge, we present the first study in the literature that is particularly developed to effectively exploit the leakage of SCA-protected implementations of LBC without the need for profiling.
- We show that the central reduction techniques that are widely adapted in LBC lead to a source of effectively exploitable SCA leakage. Particularly, information about the sign of arithmetic shares would ease exploiting the leakage and conducting successful key-recovery attacks.
- We show that the employed coefficient modulus as well as the reduction algorithm and the machine word size affect the SCA leakage of masked implementations of LBC, particularly making non-profiled attacks easier to be conducted. For the aforementioned scenarios, we particularly present the so-called optimal correlation and the number of traces required for different noise levels. Our study reveals that SCA attacks on masked implementations with central reduction are significantly more noise-tolerant.
- We introduce a novel prediction function for non-profiled SCA attacks, namely the absolute value prediction function that well predicts the leakage caused by the adaption

Table 1: Qualitative summary of state-of-the-art SCA attacks on implementations of LBC.

Work	**Class**	**Algorithm**	**Implementation**	**Masked**	**Target Function**
this work	Non-Profiled	Dilithium☆, Kyber	Cortex-M4♣	✓✓	poly. mult.
[MWK+24]	Non-Profiled	Kyber, Saber, NTRU[1]	Cortex-M4★	✗	poly. mult.
[CKA+21]	Non-Profiled	Dilithium	Ref. C	✗	poly. mult.
[TS24]	Non-Profiled	Dilithium, Kyber	Cortex-M4★	✓	poly. mult.
[SLKG22]	Non-Profiled	Dilithium	Hardware	✗	poly. mult.
[PPM17]	Profiled	✈	Cortex-M4✧	✓✓	NTT
[KLH+20]	Profiled	Dilithium	Ref. C	✗	NTT
[KLH+20]	Profiled	Dilithium	Ref. C✶	✓	sparse poly. mult.
[XPR+21]	Profiled	Kyber	Ref. C	✗	NTT
[XPR+21]✉	Profiled	Kyber	Cortex-M4★	✗	bin. to poly.
[BAE+24]	Profiled	Dilithium	✤	✗	small poly.mult
[BNGD23]	Profiled	Kyber, Saber	Cortex-M4❧	✓✓	poly. to bin.
[DNGW23]✉	Profiled	Kyber	Cortex-M4❧	✓	bin. to poly.
[MUTS22]	Profiled	Dilithium	Ref. C	✗	bin. to poly.
[KAA22]✉	Profiled	Dilithium, NTRU[1] NTRU Prime[2]	Ref. C Cortex-M4★	✓	small poly. sampling

☆ attacks in the simulation
★ from [KRSS19b]
✧ from [RRd+16]
❧ from [HKL+22]
♣ from [BC22]
✈ generic to NTT applications in LBC
✓✓ presents a novel technique to tackle masking
✶ attacks through an implementation submitted for another project [BAA+19]
✤ not reported
✉ challenge polynomial/message-recovery attack
[1] a post-quantum lattice-based KEM [CDH+20]
[2] a post-quantum lattice-based KEM [BBC+20]

of central reduction in masked implementations of LBC. We show that our introduced prediction function is optimal.

- We practice our approach against a first-order masked implementation of the lattice-based post-quantum KEM Kyber. We experimentally show that only a few hundred traces are required to successfully mount a key-recovery attack.

2 Background

2.1 Notations

The notations we followed in the paper are as follows.

- Vectors are represented by bold lowercase letters such as $\mathbf{x}$, while matrices are represented by bold uppercase letters such as $\mathbf{X}$. Polynomials are represented by lowercase regular letters such as x. Vector-to-vector, matrix-to-vector, and scalar-vector multiplications are denoted by $\cdot$ while element-wise multiplication of vectors or matrices is denoted by $\star$. The i-th element (coefficient) of a vector (polynomial) is denoted by the subscripts, such as $\mathbf{x}_i$ (x_i).

- The central reduction to the range $[-q/2, q/2]$ is explicitly denoted by $\bmod^{\pm} q$, while $\bmod q$ denotes the regular modular reduction to the range $[0, q)$. We also use $\bmod q$ when the output range is not important such as in high-level representation of algorithms, *i.e.* pseudocodes. The set of unsigned integers in $[0, q)$ is denoted by $\mathbb{Z}_q$. Accordingly, ${}^{\pm}\mathbb{Z}_q$ represents the signed representation of integers modulo q, namely the integers in the range $[-q/2, q/2]$. We assume an odd q unless the opposite is explicitly stated.

- Random variables are denoted by uppercase letters such as X. $\mathcal{P}(\cdot)$ denotes the probability function, $E[\cdot]$ the expected value function, and $\mathcal{N}(\mu, \sigma)$ the noise following a Gaussian distribution with mean μ and standard deviation σ.

- Unless otherwise stated, the logarithm is base 2, and $\oplus$ denotes the exclusive OR.

- Shares of variables are represented by superscripts, such as $X = X^0 + X^1$.

- $\mathcal{B}(X)$ denotes the number of bits needed to represent the unsigned integer X. β denotes the machine word size. In this paper, either $\beta = 16$ or $\beta = 32$.

- $\mathcal{W}_\beta(X)$ represents the Hamming weight (HW) of a signed integer X in β-bit 2's complement representation.

- $\mathcal{S}(X) : \mathbb{Z} \to \{0, 1\}$ returns the non-negativeness (sign) of the integer X:

$$\mathcal{S}(X) = \begin{cases} 1, & \text{if } X \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

2.2 Lattice-Based Cryptography (LBC)

Here, we briefly review lattice-based post-quantum algorithms from an SCA attack perspective. We focus on Kyber [SAB+22] and Dilithum [LDK+22], which are among the algorithms selected by NIST at the end of the third round of the post-quantum cryptography standardization process. Afterwards, we discuss the details of the NTT, which is a crucial primitive for efficiently implementing polynomial arithmetic in LBC.

Ring of Polynomials. Most of the lattice-based cryptosystems, including Kyber and Dilithium, operate over the ring of polynomials $\mathcal{R}_q = \mathbb{Z}_q[x]/(X^n + 1)$ that contains polynomials up to degree $n - 1$ while the coefficients are in $\mathbb{Z}_q$. Arithmetic operations in $\mathcal{R}_q$ are the main building block for implementing LBC, and it is the main target of SCA attacks as well.

Kyber. Kyber [SAB+22] is a lattice-based post-quantum KEM, a variant of the LPR encryption scheme [LPR10]. For all security levels, Kyber employs $q = 3329$ and $n = 256$ to instantiate $\mathcal{R}_q$. The key pair is generated by the MLWE [Reg05] equation $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$. The vector of polynomials $\mathbf{e} \in \mathcal{R}_q^k$ is considered as noise and thrown away after the key generation while $\mathbf{s} \in \mathcal{R}_q^k$ forms the secret key. On the other hand, $\mathbf{A} \in \mathcal{R}_q^{k \times k}$ and $\mathbf{t} \in \mathcal{R}_q^k$ are public. The polynomials in both $\mathbf{s}$ and $\mathbf{e}$ are *short*, whose coefficients are sampled from the central binomial distribution $\mathcal{B}_\eta$ with error distribution η. The sensitive operation in a KEM in terms of non-profiled SCA attacks is the decapsulation function where the secret key is involved [MWK+24, TS24]. The decapsulation in an LPR scheme such as Kyber is quite simple: $v - \mathbf{s}^T \cdot \mathbf{u}$, where $\mathbf{u} \in \mathcal{R}_q^k$ and $v \in \mathcal{R}$ together form the ciphertext. Related parameters k and η are chosen depending on the NIST security level as $\{2, 3, 4\}$ and $\{2, 4, 2\}$, respectively.

Dilithium. Dilithium is a lattice-based post-quantum signature following the Fiat-Shamir scheme with aborts approach [Lyu09]. It employs $q = 2^{23} - 2^{13} + 1 = 8380417$ and $n = 256$. The secret-public key pair for Dilithium is generated through the MLWE equation similar to Kyber, *i.e.* $\mathbf{t} = \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2$. Distinctively, both $\mathbf{s}_1$ and $\mathbf{s}_2$ are saved as the secret key while the pseudo-randomly generated matrix $\mathbf{A} \in \mathcal{R}_q^{k \times l}$ and vector $\mathbf{t} \in \mathcal{R}_q^l$ are public as in Kyber. The coefficients of the secret polynomials in $\mathbf{s}_1$ and $\mathbf{s}_2$ are short as well. Specifically, the secret coefficients are sampled uniformly at random in $[-\eta, \eta]$. A natural target for a non-profiled SCA attack on Dilithium is the signature function as it involves the secret key [CKA+21, SLKG23, TS24]. More precisely, the multiplications $c \cdot \mathbf{s}_1$ and $c \cdot \mathbf{s}_2$ are targeted, where – among the outputs of the signature – the challenge polynomial $c \in \mathcal{R}_q$ is public and depends on the input message. The parameters (k, l) are chosen as $\{(4, 4), (6, 5), (8, 7)\}$ with respect to the security level. η is chosen the same as in Kyber.

2.3 Number Theoretic Transform (NTT)

NTT allows efficient multiplication of polynomials in $\mathcal{R}_q$. Given two polynomials $a \in \mathcal{R}_q$ and $b \in \mathcal{R}_q$, the NTT multiplication is performed as follows.

$$\mathrm{NTT}^{-1}\Big(\mathrm{NTT}(a) \star \mathrm{NTT}(b)\Big) \tag{2}$$

To simplify the notation, we denote the NTT transformation for polynomials using 'hat' for the rest of the paper, *i.e.* $\hat{a} = \mathrm{NTT}(a)$. The element-wise multiplication in the NTT domain, $\hat{a} \star \hat{b}$, is known as the base multiplication.

Complete NTT. NTT is considered as an application of the Chinese Remainder Theorem (CRT) to $\mathcal{R}_q$. In case $q \equiv 1 \mod 2n$, a primitive $2n$-th root of unity $\zeta_{2n} \in \mathbb{Z}_q$ exists for which $\zeta_{2n}^n \equiv -1 \mod q$. This setting allows a *complete* NTT over $\mathcal{R}_q$, where $x^n + 1$ can be factored down to the linear factors as $\prod_{i=0}^{n-1}(x - \zeta_{2n}^{2i+1})$. The NTT transformation indeed computes the remainder from the division of its input polynomial by $(x - \zeta_{2n}^{2i+1})$ for each i, resulting in a vector of n elements, *i.e.* in $\mathbb{Z}_q^n$. Consequently, the base multiplication is performed coefficient wise, *i.e.* a modular multiplication for each i.

Implementing NTT Transformation. The *forward* and *backward* NTT transformations can be efficiently implemented in $\log(n)$ steps. Each step is called an NTT *layer*. Indeed, the polynomial is recursively split using so-called *butterfly units* until a linear degree is reached. The forward transformation is usually implemented with *Cooley-Tuckey* (CT) butterflies [CT65] while the backward transformation is commonly realized using *Gentleman-Sande* (GS) butterflies [GS66] although it is not a must. For an input

pair of coefficients a_0 and a_1, the CT butterfly computes the output pair by

$$\hat{a}_0 = a_0 - a_1 \cdot \delta, \qquad \hat{a}_1 = a_0 + a_1 \cdot \delta, \tag{3}$$

where δ is called the *twiddle factor*, a power of ζ_{2n}.

Incomplete NTT. Sometimes, due to performance optimizations or restrictions of the operated ring of polynomials, NTT is not computed for all $\log(n)$ layers [LS19, AHKS22, ABCG20, CHK+21, ACC+22]. This is referred to as *incomplete* NTT. In case the NTT is computed for $m < \log(n)$ layers, $\hat{a}$ for $a \in \mathcal{R}_q$ is a vector with 2^m elements and each element is a polynomial with a degree of $\log(n) - m$. Then, the base multiplication refers to the multiplication of degree-$(\log(n) - m)$ polynomials. For instance, Kyber employs $q = 3329$ and $n = 256$ allowing a 7-layer NTT while $\log(n) = 8$. Therefore, the base multiplication in this setting is achieved by performing 128 individual multiplications of degree-1 polynomials, as demonstrated in Algorithm 1.

Algorithm 1 Base Multiplication for $(\log(n) - 1)$-layer incomplete NTT

Input: $\hat{s}, \hat{c}$; the resulting vectors from $(\log(n) - 1)$-layer forward NTT transformation on $s, c \in \mathcal{R}_q$

Output: $\hat{z} = \hat{s} \star \hat{c}$

1: **for** $\forall i \in \{0, \ldots, n/2\}$ **do** ▷ Compute $\hat{z}_i = \hat{s}_i \cdot \hat{c}_i$

2: $\quad \hat{z}_{i,0} \leftarrow \hat{s}_{i,0} \cdot \hat{c}_{i,0} + \hat{s}_{i,1} \cdot \hat{c}_{i,1} \cdot \delta_i \mod q$ ▷ δ_i is a power of ζ_{2n}

3: $\quad \hat{z}_{i,1} \leftarrow \hat{s}_{i,1} \cdot \hat{c}_{i,0} + \hat{s}_{i,0} \cdot \hat{c}_{i,1} \mod q$

4: **end for**

2.4 Modular Arithmetic

As follows, we briefly review the modular reduction techniques that are adapted in LBC. Compared to ECC or RSA, the modular arithmetic in LBC deals with relatively shorter integers. Additionally, operating with signed integers in modular arithmetic proves to be more efficient in LBC [HZZ+22, AHKS22, GKS21, ABCG20, BKS19]. The main reason for this is due to the fact that it simply eliminates the need for an extra addition for preventing negativeness in the butterfly units (see Equation (3)). Table 2 summarizes state-of-the-art reduction schemes implemented on the ARM Cortex-M4. It can be seen that the smallest latency in terms of the number of clock cycles is achieved by central reduction techniques [ABCG20, AHKS22, HZZ+22], whose output range is centered around 0. As a result, in addition to the NTT transformation, central reduction is also preferred to speed up the base multiplication.

Barrett Reduction. The Barrett reduction was originally proposed in [Bar87]. Its main idea is to subtract a factor of the modulus q from the number to reduce by approximating the division of the number by q through a pre-computed factor and shifting. A signed version of Barrett reduction adapted for LBC is proposed in [Sei18]. The input range of the signed Barrett reduction is $[-\beta/2, \beta/2)$, and the output range is $[0, q]$. A 9-cycle implementation of the signed Barrett reduction for packed integers dedicated to ARM Cortex-M4 is presented in [ABCG20]. Packing of integers refers to storing two $\beta = 16$-bit integers in a $2\,\beta = 32$-bit register, which is then passed to the packed reduction function. Later, a 6-cycle implementation of Barrett reduction for packed integers was reported in [AHKS22], which performs a central reduction with an output range $[-q/2, q/2]$.

Table 2: Summary of state-of-the-art reduction implementations on ARM Cortex-M4.

Scheme	β	**Input Range**	**Output Range**	**Packed**	**Cycles**
Montgomery [ABCG20]	16	$[-q\cdot\beta/2, q\cdot\beta/2)$	$(-q, q)$	✗	2
Montgomery [GKS21]	32	$[-q\cdot\beta/2, q\cdot\beta/2)$	$(-q, q)$	✗	2
Montgomery [ABCG20]	16	$[-q\cdot\beta/2, q\cdot\beta/2)$	$(-q, q)$	✓	8
Barrett [Sei18]✶	16	$[-\beta/2, \beta/2)$	$[0, q]$	✗	3
Barrett [AHKS22]	16	$[-\beta/2, \beta/2)$	$[-q/2, q/2]$	✓	6
Plantard [HZZ+22]	16	$[-q^2 2^{2\alpha'}, q^2 2^{2\alpha'}]$☆	$[-q/2, q/2]$	✗	2
Plantard [HZZ+22]	16	$[-q^2 2^{2\alpha'}, q^2 2^{2\alpha'}]$☆	$[-q/2, q/2]$	✓	5

✶ gives the definition of the algorithm but does not present an implementation on ARM Cortex-M4.
☆ α' is a parameter of Plantard reduction that satisfies $q < 2^{\beta-\alpha'-1}$.

Montgomery Reduction. The Montgomery reduction is first proposed in [Mon85]. Similar to the Barrett reduction, it enables a constant time reduction by eliminating the need for division. A signed version of Montgomery reduction is presented by [Sei18], with an input range $[-q\cdot\beta/2, q\cdot\beta/2)$ and output range $(-q, q)$. While a 3-cycle implementation on ARM Cortex-M4 was initially given by [BKS19] for $\beta = 16$, the state-of-the-art implementation of Montgomery reduction [ABCG20, GKS21] takes 2 cycles for both $\beta = 16$ and $\beta = 32$. Also, an 8-cycle implementation of Montgomery reduction for packed integers was presented in [ABCG20]. We would like to note that a final correction may be required after the signed Montgomery reduction to find the residue in the signed range $[-q/2, q/2]$ which is the ultimate goal. However, we computationally found out that the correction step is not needed most of the time. In particular for $q = 3329$ and $\beta = 16$, only %0.01 of the corresponding input range requires a correction while the output is already in $[-q/2, q/2]$ for the rest. Therefore, we study the output range $[-q/2, q/2]$ for the sake of simplicity for the rest of this paper.

Plantard Reduction. The Plantard reduction [Pla21] is a more recent algorithm compared to its counterparts, Montgomery and Barrett. While the original Plantard reduction operates on unsigned integers, the authors of [HZZ+22] proposed an improved version, which operates on signed integers to be employed in LBC. The output range of the signed version is $[-q/2, q/2]$, the same as the state-of-art Barrett reduction. One advantage of the Plantard reduction is that it enables 2-cycle modular multiplication by a constant, outperforming the 3-cycle Montgomery multiplication. The multiplication by a constant is beneficial for implementing the butterfly units during the NTT transformations (see Equation (3)). On the other hand, the improved Plantard reduction also takes 2 cycles on ARM Cortex-M4, the same as Montgomery. However, Plantard's 2-cycle implementation enables a larger input range and a smaller output range that is desirable. Specifically, the Plantard reduction generates the output in the exact range $[-q/2, q/2]$. In other words, it does not require any final correction. This is a significant improvement over the 2-cycle Montgomery reduction whose output range is $(-q, q)$. As a side note, packed reduction takes 5 cycles.

3 Non-Profiled SCA Attack on NTT Multiplication

In this section, we present the general outline of a non-profiled power SCA attack on an implementation of a polynomial multiplication in the NTT domain.

3.1 Attack Outline

Leakage Model. Let us first define the assumed leakage function of the target device based on the random variable X defined in the space $\mathcal{X}$, constant scaling factor α, and noise sampled from a Gaussian distribution with mean μ and standard deviation σ, which is independent of X as

$$\mathcal{L}(X) = \alpha \cdot \mathcal{W}_\beta(X) + \mathcal{N}(\mu, \sigma). \tag{4}$$

$\mathcal{L}(X)$ is commonly used to simulate SCA leakage of micro-processors in the presence of noise when X is processed.

Adversary Model. Consider the base multiplication $\hat{s} \star \hat{c}$ where $s \in \mathcal{R}_q$ is a secret polynomial, and $c \in \mathcal{R}_q$ is a public polynomial. The goal of an adversary is to reveal $\hat{s}$ through a non-profiled SCA attack where the attacker has access to the power consumption pattern of the underlying device during the computation of $\hat{s} \star \hat{c}$. In a non-profiled attack, the attacker samples the leakage of $\hat{s} \star \hat{c}$ for ν distinct computations where $\hat{c}$ changes for each measurement. The set of samples recorded for each measurement is referred to as a *trace*.

Attack on Complete NTT. As previously mentioned, the base multiplication in the NTT domain is performed element wise. Therefore, each $\hat{s}_i$ can be attacked independently using the knowledge of $\hat{c}_i$. To retrieve $\hat{s}_i$, a set of hypotheses is made. Each hypothesis is tested by evaluating the *target function*, $g(\hat{s}_i, \hat{c}_i)$, thereby statistically comparing the observed leakages (traces). We use the above-defined random variable X as the sensitive output of the target function, $X = g(\hat{s}_i, \hat{c}_i)$, as we consider $\hat{c}_i$ as a random variable changing over measurements. In this study, the *target function* to reveal $\hat{s}_i$, is the multiplication $g(\hat{s}_i, \hat{c}_i) = \hat{s}_i \cdot \hat{c}_i \mod q$ (or $\mod^{\pm} q$ if the reduction is central). Therefore, $\mathcal{X} = \mathbb{Z}_q$ (or $\mathcal{X} = {}^{\pm}\mathbb{Z}_q$), and there are q hypotheses to test. Accordingly, we use $X' = g(\hat{s}'_i, \hat{c}_i)$ to denote the evaluation of the chosen target function for the hypothesis $\hat{s}'_i$. In a Correlation Power Analysis (CPA) attack [BCO04], the output of the *target function* is transformed into *hypothetical leakages* using a *prediction function* with HW function, $\mathcal{W}_\beta(X')$, being the most frequently used prediction function. CPA is based on estimating the correlation, *e.g.* by Pearson correlation coefficient, between measured and hypothetical leakages, $\hat{\rho}(\mathcal{W}_\beta(X'), \mathcal{L}(X))$. With a sufficient number of traces, the attacker expects to obtain $\hat{s}_i = \text{argmax}_{\hat{s}'_i} \left(\hat{\rho}\left(\mathcal{W}_\beta\left(X'\right), \mathcal{L}(X)\right)\right)$. Since the attacker may not know which point in time corresponds to $\mathcal{L}(X)$ in the traces, the procedure is usually repeated for a subset of all samples points.

Attack on Incomplete NTT. For $(\log(n) - 1)$-level incomplete NTT such as in Kyber, q^2 hypotheses should be examined since $\hat{s}_i \cdot \hat{c}_i$ is a multiplication between degree-1 polynomials (see Algorithm 1) [MWK+24, TS24]. In this case, the attacker can use the lower-degree coefficient $g(\hat{s}_i, \hat{c}_i) = \hat{z}_{i,0}$ for $\hat{z}_i = \hat{s}_i \cdot \hat{c}_i$, the higher-degree coefficient $g(\hat{s}_i, \hat{c}_i) = \hat{z}_{i,1}$, or both coefficients $g(\hat{s}_i, \hat{c}_i) = \{\hat{z}_{i,0}, \hat{z}_{i,1}\}$ as the target. If lower- or higher-degree coefficients are used alone, $\mathcal{X} = \mathbb{Z}_q$, similar to when NTT is complete. On the other hand, $\mathcal{X} = \mathbb{Z}_q^2$ if both coefficients are involved in the target function. Since a number and its additive inverse in 2's complement form[1] are correlated, the number of hypotheses can be reduced

[1] Negative integers in 2's complement form are represented by inverting all bits of the corresponding positive integer and adding 1 to the result.

to $q/2$ and $q^2/2$ for complete and incomplete NTT, respectively. In this manner, the attacker learns either the actual secret or its additive inverse.

Application to Kyber and Dilithium. It is important to emphasize that the adversary model as well as the target function explained above can directly be applied to both Kyber and Dilithium. More precisely, performing an attack on the leakage of $\hat{s} \star \hat{c}$ to reveal $\hat{s}$ corresponds to targeting $\mathbf{s}^T \cdot u$ in the case of Kyber and targeting $c \cdot \mathbf{s}_1$ or $c \cdot \mathbf{s}_2$ for Dilithium (see Section 2.2).

3.2 Second-Order Attack

Masking. The most promising way to defeat the above-explained SCA attack is masking [HKL+22, BGR+21, FBR+22, ÖY23, MGTF19, ABC+23b, CGTZ23, RRd+16, BC22, CGMZ23, BDK+21, KDVB+22]. Indeed, masking of the polynomial multiplication is straightforward from an algorithmic perspective, as it can be seen as a linear operation. For a uniformly and randomly generated share s^0, one computes the other share as $s^1 = s - s^0$. Then, the computation $\hat{s} \star \hat{c}$ is performed on the shares as $\hat{s}^0 \star \hat{c}$ and $\hat{s}^1 \star \hat{c}$. Notice that, $\hat{s} \star \hat{c} = \hat{s}^0 \star \hat{c} + \hat{s}^1 \star \hat{c}$. Accordingly, $\hat{s}_i \cdot \hat{c}_i = \hat{s}_i^0 \cdot \hat{c}_i + \hat{s}_i^1 \cdot \hat{c}_i$ for all i. This is referred to as *arithmetic masking*. In particular, the order of masking is defined by the number shares representing the secrets. Here in the given example, first-order masking is applied as two shares are used. Since $\hat{s}_i \cdot \hat{c}_i$ is not computed in plain, the leakage of every single point in SCA traces is expected to be independent of $\hat{s}_i$ and hence independent of the estimated hypothetical leakage, *e.g.* $\mathcal{W}_\beta(X')$. It is noteworthy to mention that an assumed condition for such a claim is that each share s^0 and s^1 should individually follow a uniform distribution.

Combination Function. In order to conduct successful CPA attacks on a first-order masked implementation, the leakage associated to two shares should be combined using a pre-processing function. Since the shares are processed individually (not simultaneously), their associated leakages appear in different time samples. Intuitively for the application studied in this work, the attacker combines the observed leakages associated to $X^0 = g(\hat{s}_i^0, \hat{c}_i)$ and $X^1 = g(\hat{s}_i^1, \hat{c}_i)$ while mean-free product is known as the most efficient pre-processing (combination) function [PRB09, SVO+10]. For leakages of the random shares, $\mathcal{L}(X^0)$ and $\mathcal{L}(X^1)$, the mean-free product is defined as

$$\mathcal{C}(\mathcal{L}(X^0), \mathcal{L}(X^1)) = (\mathcal{L}(X^0) - E[\mathcal{L}(X^0)]) \cdot (\mathcal{L}(X^1) - E[\mathcal{L}(X^1)]) \tag{5}$$

The mean-free product is the most preferred combination function because it effectively reduces noise by removing constant offsets, thereby enhancing the signal-to-noise ratio and emphasizing relevant variations. Needless to say, $E[\mathcal{L}(X^0)]$ and $E[\mathcal{L}(X^1)]$ are approximated by the sample means over the trace set. For sake of completeness, we include the absolute difference combination function [JPS05] in Appendix A, which is also an efficient and frequently used combination function.

Optimal Prediction Function (OPF). [PRB09] shows that an optimal prediction function must be computed in order to efficiently perform a second-order CPA attack by means of a combination function

$$f_{opt}(x) = E\left[\mathcal{C}\left(\mathcal{L}\left(X^0\right), \mathcal{L}\left(X^1\right)\right) \middle| X = x\right]. \tag{6}$$

Recall that $X = X^0 + X^1 \mod q$ (or $\mod {}^{\pm} q$). In simple words, the expected value of the combination function is calculated for the given unmasked value which can be hypothetically computed. A CPA attack performed on a masked implementation by means

of such a pre-processing function is referred to as *Higher-Order CPA (HOCPA)*. Similar to the first-order attack, the attacker estimates the correlation $\hat{\rho}(f_{opt}(X'), \mathcal{C}(\mathcal{L}(X^0), \mathcal{L}(X^1)))$ to rank the hypotheses. The correlation achieved by f_{opt} and the correct hypothesis is referred to as the *optimal correlation*, denoted by $\hat{\rho}_{opt} = \hat{\rho}(f_{opt}(X), \mathcal{C}(\mathcal{L}(X^0), \mathcal{L}(X^1)))$. In this study, we use HOCPA with the mean-free product as the distinguisher and discuss the optimal prediction function and its efficiency for the presented adversary model in different reduction scenarios. However, we replicate some of the experiments discussed in the subsequent sections for the absolute difference combination function.

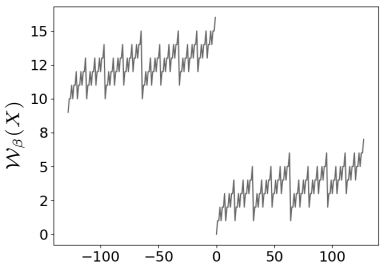
Conditioning on Zero-Values. As the chosen target function is a multiplication for this study, one can fine-tune the optimal prediction function by considering zero-value public data, *i.e.* $\hat{c}_i = 0$. Notice that for the complete NTT, $X = 0$ if and only if $\hat{c}_i = 0$ (assuming $\hat{s}_i \neq 0$), because of the speciality of 0 in multiplication. Then, X^0 and X^1 become 0 as $X^j = \hat{s}_i^j \cdot 0$ and $f_{opt}(0) = E[\mathcal{C}(Y_0, Y_1)|X^0 = 0, X^1 = 0] = \alpha^2 \cdot (E[\mathcal{W}_\beta(X) + \mathcal{N}(\mu, \sigma)])^2$, see Equation (4). When the NTT is incomplete, X can be 0 even though $\hat{c}_i$ is non-zero, because the target X is the addition of two multiplications which can sum up to 0. Recall that X is a function of $\hat{z}_{i,j}$ in this case, which is formulated in Algorithm 1. However, since having $\hat{c}_i = 0$ is less likely for the incomplete NTT ($1/q^2$ for $\hat{c}_{i,0} = \hat{c}_{i,1} = 0$, assuming uniformly random $\hat{c}_i$) we do not concentrate on this case.

4 Distribution of HW of Signed Integers Modulo q

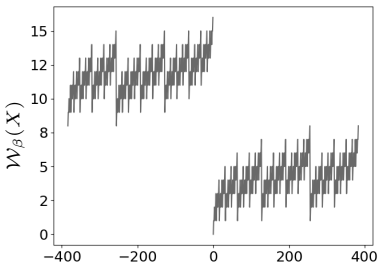
In this section, we study the distribution of HW of the signed representation of integers modulo q, *i.e.* the effect of central reduction on HW. Accordingly, we compare signed and unsigned arithmetic in terms of SCA leakage. We evaluate the primes that are employed in Kyber and Dilithium with $q = 3329$ and $q = 8380417$, respectively, and study the carrier primes that are employed for Dilithium to perform short polynomial arithmetic [AHKS22], namely $q = 257$ for Dilithium2 and Dilithium5, and $q = 769$ for Dilithium3. We should note that masking the short polynomials in their range is possible [ABC+23a]. One important factor for computing the HW of negative integers is the machine word size β which does not have any effect on the HW of positive integers. The machine word size is usually $\beta = 16$ for $q = 257$, $q = 769$, and $q = 3329$ while $\beta = 32$ for $q = 8380417$ in software implementations. Unless otherwise stated, we take these values for β in the studied adversary model. However, we discuss the role of distinct values of β in the SCA leakages.

4.1 HW as a Sign Indicator

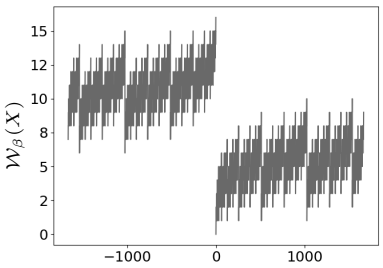
The main observation that led to this study is the clear separation of HW of the non-negative side of ${}^{\pm}\mathbb{Z}_q$, namely $[0, q/2]$ and the negative side $[-q/2, 0)$. As an intuition, consider $q = 257$; the positive interval of ${}^{\pm}\mathbb{Z}_{257}$ corresponds to $[0, 128]$, for which the maximum HW is $\mathcal{W}_\beta(127) = 7$. In other words, the HW of integers $[0, 128]$ lies in $[0, 7]$. On the other hand, the negative side of ${}^{\pm}\mathbb{Z}_{257}$ corresponds to $[-128, 0)$, where the HWs are in the range of $[9, 16]$ assuming 2's complement representation with machine word size $\beta = 16$. Consequently, the HW of a number in ${}^{\pm}\mathbb{Z}_{257}$, reveals its sign immediately. Figure 1 visualizes our observation for all the primes analyzed in this study. Note that there is an overlap between the HW ranges $[-q/2, 0)$ and $[0, q/2]$, for $q = 769$, $q = 3329$, and $q = 8380417$. For instance, the HW of non-negative integers in ${}^{\pm}\mathbb{Z}_{3329}$ are distributed in $[0, 10]$ while that of negative integers are in the range $[6, 16]$. Therefore, there is an overlap for five possible HWs in the interval $[6, 10]$ out of a total of 17 possible values assuming the machine word size $\beta = 16$.



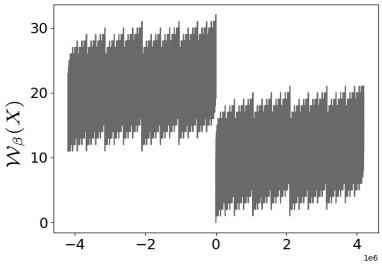
(a) $q = 257$, $\beta = 16$



(b) $q = 769$, $\beta = 16$



(c) $q = 3329$, $\beta = 16$



(d) $q = 8380417$, $\beta = 32$

Figure 1: Distribution of HW of integers in $[-q/2, q/2]$ in 2's complement representation.

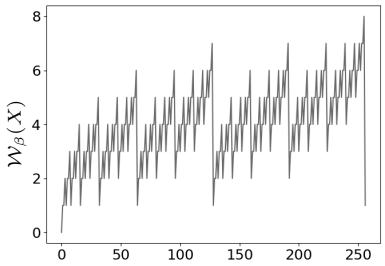


Figure 2: Distribution of HW of integers in $[0, q)$ for $q = 257$.

Based on our observation in Figure 1, we write the following equality for $\mathcal{W}_\beta(X)$ where $X \in {}^{\pm}\mathbb{Z}_q$.

$$\mathcal{W}_\beta(X) = \mathcal{S}(X) \cdot \gamma + (1 - \mathcal{S}(X)) \cdot (\beta - \gamma) + e, \tag{7}$$

for some inner-cluster error term $e \in \mathcal{E}$ where $E[e] = 0$ and $0 < \gamma < \beta/2$. Note that γ stands for the mean of HWs given X is non-negative, *i.e.* $\gamma = E[\mathcal{W}_\beta(X) \mid \mathcal{S}(X) = 1]$. Similarly, let γ^- denote the mean HW given X is negative, $\gamma^- = E[\mathcal{W}_\beta(X) \mid \mathcal{S}(X) = 0]$. Table 3 demonstrates the values of γ and γ^- depending on q. Note that $\gamma^- \approx \beta - \gamma$, which allows us to simplify Equation (7). Naturally, γ^- approaches γ as the so-called gap $\mathcal{M}(q, \beta) = \beta - \log(q)$ decreases. On the other hand, the expected value $E[|e|]$ is not affected by β while it slightly increases as q gets larger. Additionally, whether the integers modulo q are represented in signed or unsigned form has no impact on this expected value.

Table 3: The mean and standard deviation of HW for positive and negative integers with distinct q. γ denotes the mean of HWs for positive integers in ${}^{\pm}\mathbb{Z}_q$, namely $[0, q/2]$. γ^- denotes the mean of HWs for negative integers in ${}^{\pm}\mathbb{Z}_q$, namely $[-q/2, -1]$. γ^* denotes the mean of HWs for $[q/2+1, q-1]$, the set of integers in $\mathbb{Z}_q$ which are considered to be negative in LBC, and e is the error term defined in Equation (7).

q	257	769	3329	8380417
γ	3.48	4.16	5.19	10.99
γ^-	$\beta - 3.5$	$\beta - 4.17$	$\beta - 5.2$	$\beta - 11$
γ^*	4.5	5.16	6.19	11.99
$E[\lvert e \rvert]$	1.33	1.4	1.55	2.34

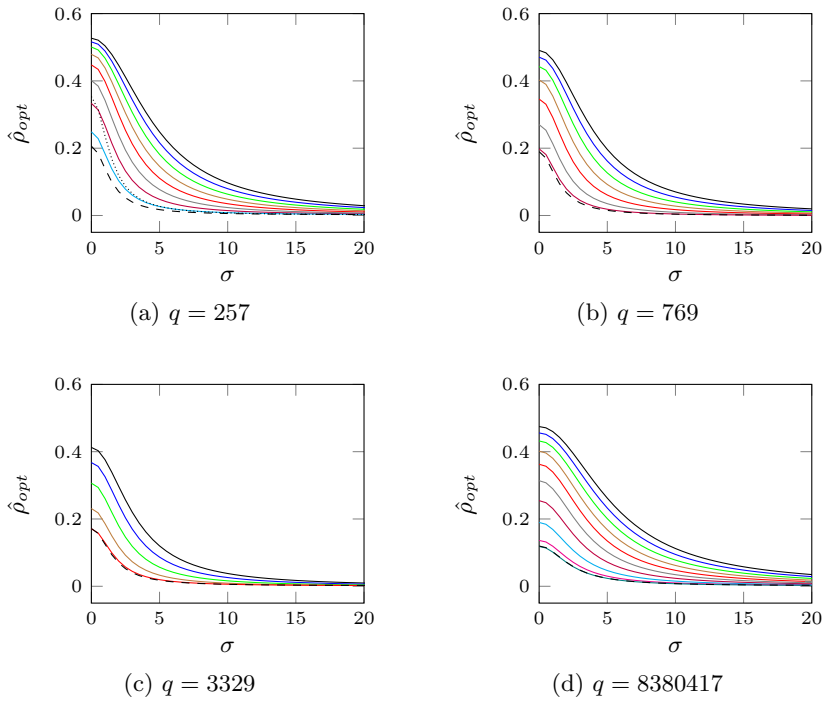
Distribution of HW of Unsigned Integers Modulo q. We would like to note that the argument made in this section does not apply to the unsigned representation of integers modulo q, namely $\mathbb{Z}_q$. It can be seen in Figure 2 that no clear ranges can be identified for unsigned integers when observing their HW. In LBC, the upper half of $\mathbb{Z}_q$ is considered as negative, namely $[-q/2+q, q-1]$. However, Table 3 shows that the mean of HW of negative side, denoted by γ^*, is approximately $\gamma + 1$, independent of β and q.

4.2 Impact of Signed Arithmetic on Optimal Correlation

To formally assess the impact of signed arithmetic on SCA leakages, we compare the optimal correlation achieved by the state-of-the-art combination function, mean-free product, between the cases when the modular reduction is central and when it is non-central. Figure 3 presents the estimated optimal correlation $\hat{\rho}_{opt}$ for distinct prime q and machine word size β and reduction scenarios. It can be seen that $\hat{\rho}_{opt}$ estimated for central reduction achieves more than twice of the one estimated for the non-central reduction, particularly for $\beta = 16$ and $\beta = 32$. Indeed, $\hat{\rho}_{opt}$ is an increasing function of β for a given q when the reduction is central, complying with our initial observation in Table 3. As β decreases, $\hat{\rho}_{opt}$ for the central reduction reaches that of the the non-central case as the limit. More importantly, the correlation shows a strong resistance to noise for the signed case. For instance, when $\sigma = 5$ and $q = 257$ and $\beta = 16$ (such as a masked software implementation of Dilithium), $\hat{\rho}_{opt}$ reaches $\hat{\rho} = 0.24$ while we observe $\hat{\rho} = 0.017$ for the unsigned case. We should refer to Table 2 showing that β increases the input range of the reduction algorithms. However, our analysis shows that it further increases the associated SCA leakages. It also makes sense to compare the optimal correlation achieved in case of Boolean masking with the other results. Similar to what presented in [PRB09], the correlation for Boolean masking reaches $\hat{\rho} = 0.35$ for an 8-bit implementation in a noiseless scenario. Similar to the non-central reduction case, this drops rapidly with the noise, *e.g.* to $\hat{\rho} = 0.02$ for $\sigma = 5$.

We replicated the same analysis for the absolute difference combination function, as detailed in Figure 11 (in Appendix A). Although the estimated optimal correlations are slightly lower across all values of q, β, and σ, the impact of β and σ follows the same pattern observed for the mean-free product.

For this evaluation, we computed f_{opt} as a look-up table for each case. This has a time complexity of $O(q^2)$ to create f_{opt}, which can be costly for large q, such as $q = 8380417$. In the next section, we deal with explicit formulas for f_{opt} dedicated to central reduction.



(a) $q = 257$ (b) $q = 769$

(c) $q = 3329$ (d) $q = 8380417$

Figure 3: Optimal correlation with respect to the noise standard deviation σ for different q and β and reduction strategies. Estimations are performed with 1 million samples uniformly taken for X^0 and X^1.

- ★ Reduction to $[-q/2, q/2]$ for $q = 257$, $q = 769$ and $q = 3329$: $\beta = 16$ (black), $\beta = 15$ (blue), $\beta = 14$ (green), $\beta = 13$ (brown), $\beta = 12$ (red), $\beta = 11$ (gray), $\beta = 10$ (purple), $\beta = 9$ (cyan)
- ★ Reduction to $[-q/2, q/2]$ for $q = 8380417$: $\beta = 32$ (black), $\beta = 31$ (blue), $\beta = 30$ (green), $\beta = 29$ (brown), $\beta = 28$ (red), $\beta = 27$ (gray), $\beta = 26$ (purple), $\beta = 25$ (cyan), $\beta = 24$ (magenta), $\beta = 23$ (teal)
- ★ Reduction to $[0, q)$ (dashed)
- ★ Boolean masking only in (a) for $\beta = 8$ and $q = 256$ (dotted)

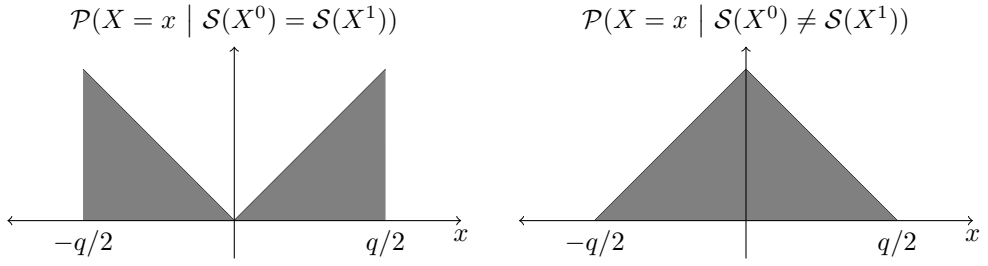
5 Absolute Value Prediction Function

When the target operation is protected using Boolean masking and the underlying circuit is a noisy Hamming weight of intermediates (as in Equation (4)), it is shown by [PRB09] that $\mathcal{W}_\beta$ can be effectively used as the optimal prediction function for HOCPA attacks. However, this is not necessarily the case when masking is arithmetic. To provide an intuition, we estimated $\hat{\rho}(\mathcal{W}_\beta(X), f_{opt}(X))$, as proposed in [PRB09], to measure the accuracy of prediction functions for the studied adversary model, presented in Table 4. The results indicate a significant correlation loss in all scenarios. Hence, in this section, we search for an explicit formula for the optimal prediction function in case of arithmetic masking. Precisely, we show that the absolute value function can be used as the optimal prediction function when targeting arithmetic masking where central reduction is employed.

Table 4: Estimations of $\hat{\rho}(\mathcal{W}_\beta(X), f_{opt}(X))$ for different moduli q and β. The estimations are performed with 1 million uniformly random samples for X while the reduction is central.

β \ q	257	769	3329
16	**−0.741**	**−0.732**	**−0.661**
15	−0.723	−0.706	−0.607
14	−0.698	−0.671	−0.527
13	−0.665	−0.621	−0.405
12	−0.618	−0.545	−0.223
11	−0.548	−0.427	
10	−0.442	−0.241	
9	−0.273		

β \ q	8380417
32	**−0.706**
31	−0.684
30	−0.655
29	−0.618
28	−0.570
27	−0.507
26	−0.424
25	−0.314
24	−0.173
23	−0.003



(a) Case 1:
$(X^0 < 0$ and $X^1 < 0)$ or $(X^0 \geq 0$ and $X^1 \geq 0)$
i.e. $\mathcal{S}(X^0) = \mathcal{S}(X^1)$

(b) Case 2:
$(X^0 \geq 0$ and $X^1 < 0)$ or $(X^0 \geq 0$ and $X^1 < 0)$
i.e. $\mathcal{S}(X^0) \neq \mathcal{S}(X^1)$

Figure 4: Probability distributions of $X = X^0 + X^1 \bmod^{\pm} q$ for $X^0, X^1 \in^{\pm} \mathbb{Z}_q$

Distribution of the Secret Knowing the Sign of Shares. Consider two uniformly random variables $X^0, X^1 \in^{\pm} \mathbb{Z}_q$ and their modular addition $X^0 + X^1 \bmod^{\pm} q$. Given the signs of both variables, Figure 4 demonstrates the probability distribution of $X^0 + X^1 \bmod^{\pm} q$. As depicted in the figure, there are two cases for the distribution given the sign of both random variables. If the sign of X^0 and X^1 are the same, namely $\mathcal{S}(X^0) = \mathcal{S}(X^1)$, then the probability is distributed around $\pm q/2$. Otherwise, it is centered around 0. Indeed, the distributions correspond to the convolution of probability distribution functions. More precisely, one of

$$\mathcal{P}(X^0 = x^0 \mid X^0 < 0), \quad \mathcal{P}(X^0 = x^0 \mid X^0 \geq 0)$$

is convoluted to one of

$$\mathcal{P}(X^1 = x^1 \mid X^1 < 0), \quad \mathcal{P}(X^1 = x^1 \mid X^1 \geq 0).$$

Now suppose that X^0 and X^1 are arithmetic shares representing a secret intermediate variable $X = (X^0, X^1)$. Then, the above discussion shows that information about the sign of the individual shares leads to a strong effect on the distribution of the secret X.

A Model for Mean-Free Product. In the previous section, we showed that the HW of 2's complement representation of an integer in $[-q/2, q/2]$ is a noisy indicator of its sign. Also, recall the leakage in CMOS circuits which is highly relevant to the HW of processed data

(see Equation (4)). Now, let $Y_1 = \mathcal{L}(X^1)$ and $Y_1 = \mathcal{L}(X^1)$ denote the leakage associated to the random shares X^0 and X^1. The mean-free product can be written as follows.

$$\begin{aligned}\mathcal{C}(Y_0, Y_1) = &\Big(\alpha_0 \cdot \mathcal{W}_\beta(X^0) + \mathcal{N}(\mu_0, \sigma_0) - \mathrm{E}\big[\alpha_0 \cdot \mathcal{W}_\beta(X^0) + \mathcal{N}(\mu_0, \sigma_0)\big]\Big) \cdot \\ &\Big(\alpha_1 \cdot \mathcal{W}_\beta(X^1) + \mathcal{N}(\mu_1, \sigma_1) - \mathrm{E}\big[\alpha_1 \cdot \mathcal{W}_\beta(X^1) + \mathcal{N}(\mu_1, \sigma_1)\big]\Big) \end{aligned} \tag{8}$$

For the sake of simplicity, we assume $\alpha = \alpha_0 = \alpha_1$, $\mu = \mu_0 = \mu_1$, and $\sigma = \sigma_0 = \sigma_1$. As X^0 and X^1 are uniformly random signed integers in ${}^{\pm}\mathbb{Z}_q$ and represented by β bits in the computer memory, $E\left[X^0\right] = E\left[X^1\right] = \beta/2$. Then, we can write

$$\begin{aligned}\mathcal{C}(Y_0, Y_1) = &\Big(\alpha \cdot \mathcal{W}_\beta(X^0) + \mathcal{N}(\mu, \sigma) - (\alpha \cdot \beta/2 + \mu)\Big) \cdot \\ &\Big(\alpha \cdot \mathcal{W}_\beta(X^1) + \mathcal{N}(\mu, \sigma) - (\alpha \cdot \beta/2 + \mu)\Big), \end{aligned} \tag{9}$$

and by distributing the terms and as $\mathcal{N}(0, \sigma) = \alpha \cdot \mathcal{N}(0, \sigma/\alpha)$,

$$\mathcal{C}(Y_0, Y_1) = \alpha^2\Big(\mathcal{W}_\beta(X^0) - \beta/2 + \mathcal{N}(0, \sigma/\alpha)\Big) \cdot \Big(\mathcal{W}_\beta(X^1) - \beta/2 + \mathcal{N}(0, \sigma/\alpha)\Big). \tag{10}$$

By plugging Equation (7) into Equation (10) we have

$$\begin{aligned}\mathcal{C}(Y_0, Y_1) = \alpha^2\Big(&\mathcal{S}(X^0) \cdot (\gamma - \beta/2) + \big(1 - \mathcal{S}(X^0)\big) \cdot (\beta/2 - \gamma) + e_0 + \mathcal{N}(0, \sigma/\alpha)\Big) \cdot \\ &\Big(\mathcal{S}(X^1) \cdot (\gamma - \beta/2) + \big(1 - \mathcal{S}(X^1)\big) \cdot (\beta/2 - \gamma) + e_1 + \mathcal{N}(0, \sigma/\alpha)\Big). \end{aligned} \tag{11}$$

Conditional Probability of Sign Equality. As explained above and shown by Figure 4, we conclude that

$$\mathcal{P}\big(X = x \;\big|\; \mathcal{S}(X^0) = \mathcal{S}(X^1)\big) = (2/q) \cdot \big(|x|/(q/2)\big) = |x| \cdot (4/q^2). \tag{12}$$

Based on the dependency of X on $\mathcal{S}(X^0)$ and $\mathcal{S}(X^1)$, we show that the conditional probability $\mathcal{P}\big(\mathcal{S}(X^0) = \mathcal{S}(X^1) \;\big|\; X = x\big)$ is a multiple of $|x|$.

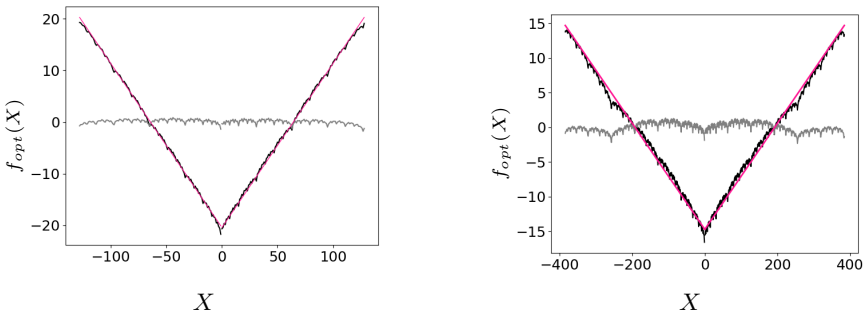
$$\begin{aligned}\mathcal{P}\Big(\mathcal{S}(X^0) = \mathcal{S}(X^1) \;\Big|\; X = x\Big) &= \frac{\mathcal{P}\Big(X = x \;\Big|\; \mathcal{S}(X^0) = \mathcal{S}(X^1)\Big) \cdot \mathcal{P}\Big(\mathcal{S}(X^0) = \mathcal{S}(X^1)\Big)}{\mathcal{P}\Big(X = x\Big)} \\ &= \frac{|x| \cdot (4/q^2) \cdot 1/2}{1/q} = (2/q) \cdot |x| \end{aligned} \tag{13}$$

Estimating the Optimal Prediction Function. We make use of the conditional probability to formally estimate $E\left[\mathcal{C}\left(Y_0, Y_1\right) \middle| X = x\right]$ as

$$\begin{aligned}&E\left[\mathcal{C}\left(Y_0, Y_1\right) \middle| X = x, \mathcal{S}\left(X^0\right) = \mathcal{S}\left(X^1\right)\right] \cdot \mathcal{P}\left(\mathcal{S}\left(X^0\right) = \mathcal{S}\left(X^1\right) \middle| X = x\right) + \\ &E\left[\mathcal{C}\left(Y_0, Y_1\right) \middle| X = x, \mathcal{S}\left(X^0\right) \neq \mathcal{S}\left(X^1\right)\right] \cdot \mathcal{P}\left(\mathcal{S}\left(X^0\right) \neq \mathcal{S}\left(X^1\right) \middle| X = x\right). \end{aligned} \tag{14}$$

Considering the terms including e_0 and e_1 as error, we can write $E\left[\mathcal{C}\left(Y_0, Y_1\right) \middle| X = x\right]$ as

$$\begin{aligned}&(\gamma - \beta/2)^2 \cdot (2/q) \cdot |x| + (\gamma - \beta/2) \cdot (\beta/2 - \gamma) \cdot (1 - 2/q \cdot |x|) + e_{\mathcal{C}} = \\ &(\gamma - \beta/2)^2 \cdot (4/q) \cdot |x| - (\gamma - \beta/2)^2 + e_{\mathcal{C}}, \end{aligned} \tag{15}$$



(a) $q = 257$, $\beta = 16$ (b) $q = 769$, $\beta = 16$

(c) $q = 3329$, $\beta = 16$ (d) $q = 8380417$, $\beta = 32$

Figure 5: Visualization of optimal prediction function for the central reduction range $[-q/2, q/2]$. $f_{opt}(X)$ (black), $e_{\mathcal{C}}$ (gray), $c_0 \cdot |X| + c_1$ (pink).

where

$$\begin{aligned} e_{\mathcal{C}} &= E\left[e_0 \cdot e_1 \mid X = x\right] + 2E\left[e_1 \cdot \mathcal{S}\left(X^0\right)(\gamma - \beta/2) + \left(1 - \mathcal{S}\left(X^0\right)\right)(\beta/2 - \gamma) \mid X = x\right] \\ &= E\left[e_0 \cdot e_1 \mid X = x\right] + 4\,(\gamma - \beta/2) \cdot E\left[e_0 \cdot \mathcal{S}\left(X^1\right) \mid X = x\right]. \end{aligned} \tag{16}$$

Note that $e_{\mathcal{C}}$ can be derived for any valid β when $E\left[e_0 \cdot e_1 \mid X = x\right]$ and $E\left[e_0 \cdot \mathcal{S}(X^1) \mid X = x\right]$ are pre-computed. This approach is beneficial particularly if q is large (*e.g.* 8380417) and the attacker does not know β in advance. Intuitively, $e_{\mathcal{C}}$ is relatively small and its impact decreases as the margin $\mathcal{M}(q, \beta)$ increases. With sufficient $\mathcal{M}(q, \beta)$, Equation (15) is accurately approximated by

$$E\left[\mathcal{C}\left(Y_0, Y_1\right) \mid X = x\right] \approx c_0 \cdot |x| + c_1, \tag{17}$$

where $c_0 = (\gamma - \beta/2)^2 \cdot 4/q$ and $c_1 = -(\gamma - \beta/2)^2$. Figure 5 visualizes these estimations and the corresponding error for two distinct cases of q and β. Since constants c_0 and c_1 do not affect the correlation, $f_{abs}(x) = |x|$ can be used as the optimal prediction function.

On the Accuracy of f_{abs}. Since Equation (16) does not allow to give an exact explicit formula for $e_{\mathcal{C}}$, we proposed approximating $f_{opt}(X)$ using the absolute value function. In order to evaluate the accuracy loss as a result of using f_{abs} instead of Equation (15), we estimated $\hat{\rho}\left(f_{abs}\left(X\right), f_{opt}\left(X\right)\right)$. Table 5 presents the estimations, indicating $\hat{\rho} > 0.99$ for all studied q and β couples mentioned in Section 4. Therefore, we conclude that the absolute value prediction function is highly accurate for the attacks we consider in this work. However, for any other settings (q and β couples), where the absolute value prediction function leads to an undesired accuracy, the $f_{opt}(X)$ function can be computed for all possible values of X as $E\left[\mathcal{C}\left(Y_0, Y_1\right) | X\right]$ (*e.g.* Equation (15)).

Table 5: Estimations of $\hat{\rho}(f_{abs}(X), f_{opt}(X))$ for different moduli q and β. The estimations are performed with 1 million uniformly random samples for X while the reduction is central.

β \ q	257	769	3329
16	**0.999**	**0.997**	**0.992**
15	0.999	0.996	0.984
14	0.998	0.993	0.961
13	0.997	0.987	0.877
12	0.994	0.970	0.546
11	0.985	0.908	
10	0.956	0.635	
9	0.825		

β \ q	8380417
32	**0.998**
31	0.998
30	0.997
29	0.995
28	0.992
27	0.983
26	0.961
25	0.892
24	0.659
23	0.214

The f_{abs} can also be used with the absolute difference combination function [JPS05]. The corresponding results for this configuration are presented in Appendix A.

Alternative Prediction Function. To take the advantage of zero-value public data as explained in Section 3.2, we define the alternative absolute value prediction function as follows.

$$f^*_{abs}(X) = \begin{cases} \big((\beta/2)^2 - c_1\big)/c_0, & \text{if } X = 0 \\ |X| & \text{otherwise} \end{cases} \tag{18}$$

Recall that $E\left[\mathcal{C}\left(Y_0, Y_1\right) \middle| X^0 = 0, X^1 = 0\right] = (\beta/2)^2$.

6 Simulation Results

In this section, we present the result of HOCPA attacks explained in Section 3 making use of simulated traces. In our simulations, we consider different noise levels and various reduction scenarios for each of the studied q. We particularly compare the required number of traces with and without central reduction. Needless to say that f_{abs} and f^*_{abs} are used as the prediction function when the reduction is central while f_{opt} is computed as a look-up table otherwise (as done in Section 4.2).

Simulated Traces. We perform the following routine to generate simulated traces for base multiplication.

1. Generate a secret key vector $\mathbf{s} \in^{\pm} \mathbb{Z}_q^m$ uniformly at random.
2. Generate a public vector $\mathbf{c} \in^{\pm} \mathbb{Z}_q^m$ uniformly at random.
3. Sample $\mathbf{s}^0 \in^{\pm} \mathbb{Z}_q^m$ uniformly at random. Apply first-order masking to $\mathbf{s}$ such that $\mathbf{s} = \mathbf{s}^0 + \mathbf{s}^1 \bmod^{\pm} q$ (or $\bmod q$ depending on the reduction scheme).
4. Compute $\mathbf{s}^0 \star \mathbf{c}$ and $\mathbf{s}^1 \star \mathbf{c}$. For each $i \in \{0, \ldots, m-1\}$ and $j \in \{0, 1\}$, perform a modular multiplication with (or without) central reduction as $\mathbf{z}_i^j \equiv \mathbf{s}_i^j \cdot \mathbf{c}_i \bmod^{\pm} q$ (or $\bmod q$) with m being the number of elements in $\mathbf{s}$ and $\mathbf{c}$. Particularly, the modular reduction is performed in $[-q/2, q/2]$ or $[0, q)$ to simulate the leakage for different reduction scheme presented in Table 2.

5. Compute the HW of each $\mathbf{z}_i^j$ as the simulated leakage ($2\,m$ individual results).

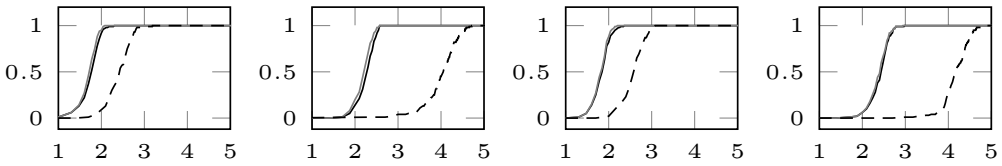
6. Apply Gaussian noise to the simulated leakages as in Equation (4), with $\alpha = 1$, $\mu = 0$, and the given standard deviation σ.

7. Go back to Step 2 until ν traces are generated.

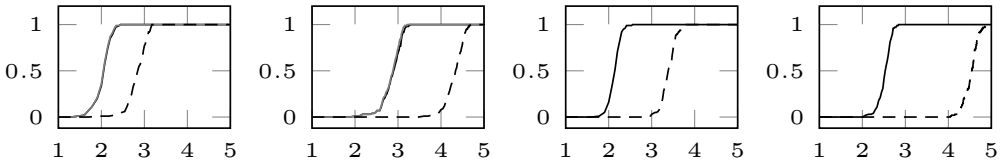
For computing the HWs, 16-bit or 32-bit 2's complement representations are used depending on q, as explained in Section 4. m, ν, σ, q as well as the reduction range (Step 4) are pre-defined parameters. We set $m = 100$ for all simulations. Note that the traces generated by this routine simulate a base multiplication for a *complete* NTT transformation, where the element-wise multiplication is just a modular multiplication. While $q = 8380417$ allows a complete NTT with the ring dimension $n = 256$, other moduli $q = 257$, $q = 769$, and $q = 3329$ do not allow complete NTT but allow incomplete NTT of 7 layers. Recall that the element-wise multiplication is the multiplication of degree-1 polynomials in that case. However, the simulations aim to benchmark our introduced absolute value prediction function and compare leakage of different reduction schemes. Therefore, there is no harm in doing the simulations as if the NTT is complete, which only affects the number of hypotheses. The comparison between the hypothetical and observed leakages is not affected by this behavior. Note also that we use $\mathbf{s}$ and $\mathbf{c}$ here instead of the notation $\hat{s}$ and $\hat{c}$ given in Section 3.

Evaluation. Figure 6 presents the corresponding results, with respect to q, ν, and σ, while the success rate refers to (# correctly predicted $\mathbf{s}_i/m$). It should be noted that the number of traces is displayed on a logarithmic scale. As evident by the results, the implementations with central reduction are significantly more vulnerable to these non-profiled HOCPA attacks in terms of the number of traces. For instance, when $q = 3329$ and $\sigma = 0$, the attack against non-central reduction needs 1500 traces to succeed, which is 6× more than the number of traces needed when the reduction is central. Moreover, the noise σ has a greater impact on the attacks on implementations with non-central reduction compared to the central case. When $q = 3329$ and $\sigma = 4$, the attack on non-central reduction needs 45 k traces to succeed, which is 31× more than what is required in case of central reduction. The difference with respect to the number of traces reaches 123× for $q = 257$ and $\sigma = 4$. We conclude that HOCPA with f_{abs} targeting central reduction remains a major threat in different noise levels conforming with the observation shown in Figure 3. Based on the aforementioned decrease in the number of traces required to attack, employing central reduction in masked LBC might be not the best choice from the SCA perspective. Although central reduction is sometimes preferred for efficiency purposes, non-central reduction can harden higher-order SCA attacks in security-demanding applications.

In general, the attack on central reduction needs relatively small number of traces to succeed. However, the number of traces for a successful attack depends on the margin $\mathcal{M}(q, \beta)$, which is slightly worse for $q = 3329$ compared to the other primes considered here. For $q = 3329$ and $\sigma = 4$, HOCPA with f^*_{abs} requires only 1400 traces to succeed. In a noiseless scenario, where $q = 257$ and $\sigma = 0$, the attack only needs around 110 traces. As previously stated, we consider only two cases $\beta = 16$ and $\beta = 32$. However, we anticipate from Figure 3 that, as β decreases, the number of required traces to attack the central reduction schemes gets closer to that when a non-central reduction scheme is employed. As anticipated, the advantage of using f^*_{abs} over f_{abs} highly depends on the number of times when $\mathbf{c}_i = 0$. For instance when $q = 257$ and $\sigma = 4$, f^*_{abs} leads to %13 reduction in the number of required traces in our experiments. As the chance of observing zero-values decreases when q increases, the advantage of f^*_{abs} decreases as well. For example, when $q = 3329$ and $\sigma = 4$, f^*_{abs} reduces ν by %9 compared to f_{abs}.



(a) $q = 257$, $\sigma = 0$ (b) $q = 257$, $\sigma = 4$ (c) $q = 769$, $\sigma = 0$ (d) $q = 769$, $\sigma = 4$



(e) $q = 3329$, $\sigma = 0$ (f) $q = 3329$, $\sigma = 4$ (g) $q = 8380417$, $\sigma = 0$ (h) $q = 8380417$, $\sigma = 4$

Figure 6: Success rate (in y-axis) of second-order CPA attacks on simulated traces. x-axis denotes $\log_{10} \nu$. For each point in the curves, 100 experiments have been performed with random data.

★ Prediction functions and reduction ranges: f_{abs} with $[-q/2, q/2]$ (solid black), f^*_{abs} with $[-q/2, q/2]$ (solid gray), f_{opt} with $[0, q)$ (dashed).

7 Practical Results: Application to Kyber

In this section, we present the result of applying our proposed approach to perform successful HOCPA attacks on a protected implementation of Kyber. It is important to note that we have previously compared the SCA leakage between central and non-central reduction. Therefore, the motivation of this section is to evaluate the difficulty of conducting successful HOCPA attacks targeting central reduction using real data. Source code of the implementations and the attack scripts as well as the simulations presented in the previous section are publicly available.[2]

Target Implementation. We focus on the ARM Cortex-M4 specific open-source and first-order masked implementation of Kyber from [BC22][3]. The polynomial arithmetic of the implementation is mostly in assembly, ported from the pqm4 project [KRSS19b] and employs the Montgomery reduction that we illustrated in Section 2.4. We also created a second version of the victim implementation by integrating the latest iteration of polynomial arithmetic from pqm4 which employs the Plantard reduction based on [HZZ+22][4]. Hereafter, we denote the untouched target implementation with Montgomery reduction by Ψ_M and the in-house version with Plantard reduction by Ψ_P. In particular, we focus on the function `basemul_asm` which implements the base multiplication $\hat{s} \star \hat{c}$ in the incomplete NTT domain for both Montgomery and Plantard versions. We should note that – to the best of our knowledge – all masked implementations of post-quantum algorithms on the ARM Cortex-M4 that have been reported in the literature are built on top of pqm4 by directly porting the linear operations including polynomial arithmetic [HKL+22, BGR+21, ABC+23b, BDK+21, HDR23]. Therefore, we believe that

[2] https://github.com/toluntosun21/ExploitingCentralReduction
[3] https://github.com/uclcrypto/pqm4_masked/ commit hash: **5fe90ba**
[4] https://github.com/mupq/pqm4 commit hash: **3743a66**

assessing the most recent iteration of pqm4, featuring state-of-the-art polynomial arithmetic, would be beneficial. The open-source Kyber implementation [BC22] employs the Montgomery reduction since the more efficient Plantard reduction did not exist when the polynomial implementation was imported from pqm4. Our experiments are centered around the medium security level, *i.e.* Kyber768, though it does not affect our approach and results.

Setup. We used NewAE ChipWhisperer CW308 UFO board to collect power traces. The victim program was running on a STM32F303, which is equipped with an ARM Cortex-M4. The frequency of the core is set to 7.3 MHz by an external reference clock which is also given to the power-collecting facility (analog-to-digital converter) while 4 power samples are recorded at each clock cycle. We provided a trigger signal for the power-collecting module to indicate the beginning of the function `basemul_asm` for the first share. Hence, only the samples related to the base multiplication were recorded. The attacks have been performed using the scared library[5], with an in-house developed Python model that mimics the intended Kyber implementation. A laptop equipped with an AMD Ryzen™ 7 7840HS[6] 8-core processor and 64 GB RAM was used for running the attack.

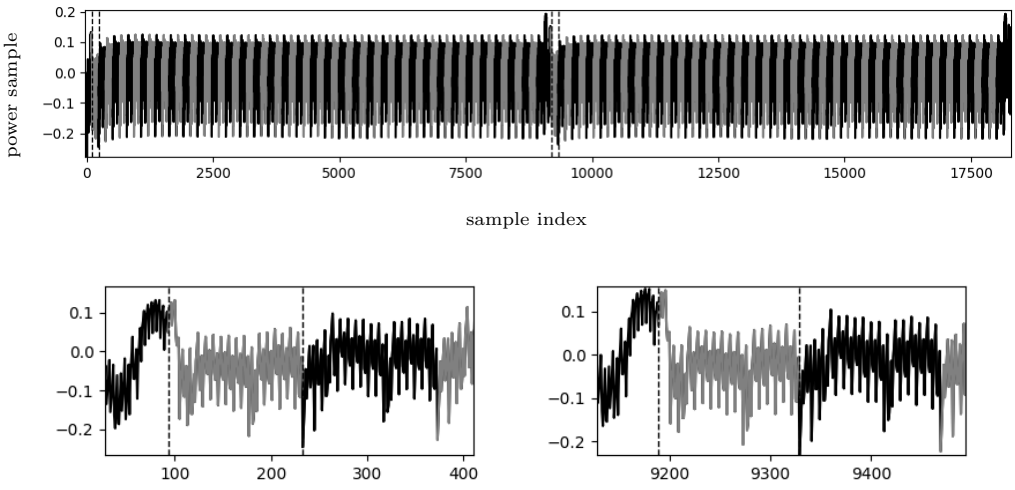
Attack Details. A mean trace over 1000 traces is presented in Figure 7. It should be noted that the iterations of the function `basemul_asm` are visible through the mean trace for both shares. In order to reveal each $\hat{s}_i$, in the corresponding attacks we have only taken into account the relevant part of the power traces based on the iterations. We used a constant offset to combine the leakages associated to two shares (by mean-free product as explained in Section 3.2) based on the pattern observed in Figure 7. It is noteworthy to mention that the same strategy can be easily adapted via educated guesses without prior knowledge of the specific implementation. Recall that $\hat{s}_i$ is a degree-1 polynomial in Kyber, and two coefficients must be predicted together based on the outline presented in Section 3. We tested $q \cdot q/2$ hypotheses ($\approx 2^{22.4}$ as $q = 3329$) with f_{abs} as the prediction function, so that either the actual secret or its additive inverse is found (for both Ψ_M and Ψ_P). The target of the attack is the higher-degree coefficient of each $\hat{s}_i \cdot \hat{c}_i$, precisely $g(\hat{s}_i \cdot \hat{c}_i) = \hat{z}_{i,1}$ computed in Line 2 of Algorithm 1. When f_{abs} is used as the prediction function, an hypothesis and its additive inverse, $\pm\hat{s}_i$, gets the same correlation score due to the nature of absolute value function.[7]

Evaluations. Let us start the evaluations by exemplary presenting the result of the individual attacks on $\hat{s}_0$ for both Ψ_M and Ψ_P in Figure 8. The correlation peaks for the correct hypotheses are observed in the corresponding time samples for the secret coefficient. Observe that the correlation for the correct hypotheses are around 0.3, which can be considered a major correlation for a higher-order attack. Needless to say, the correlation coefficient changes for different values of i. From a better perspective, Figure 9a and Figure 9b present the efficiency of our introduced prediction function f_{abs} in terms of the number of traces needed to succeed. Consistent with the simulation results, f_{abs} is very effective against arithmetic masking with central reduction. In particular, the attacks require 850 and 550 traces to fully recover the secret of the evaluated implementations. The reason why the attack on Ψ_M requires more traces to succeed is that the secret coefficients $\hat{s}_i$ for even values of i lead to lower correlation scores in general compared to the rest of

[5] https://pypi.org/project/scared/

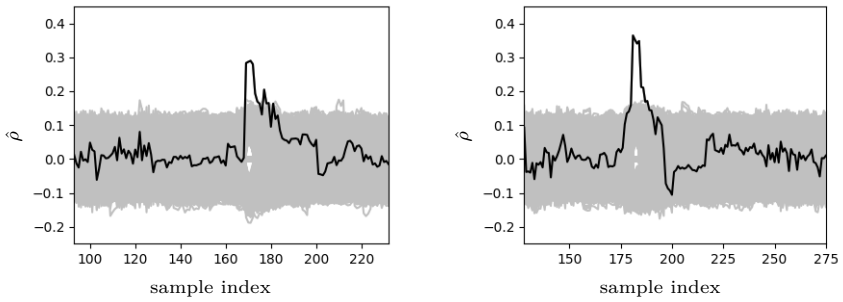
[6] https://www.amd.com/en/products/processors/laptop/ryzen/7000-series/amd-ryzen-7-7840hs.html

[7] One option to distinguish the correct hypothesis from its additive inverse is to re-run the same HOCPA attack on two hypotheses $\pm\hat{s}_i$ using the sign function $\mathcal{S}$, see Equation (1). This intuition is based on the fact that not every bit of the intermediate values equally contributes in the amount of power consumption, *i.e.* not an ideal HW model. This leads f_{opt} to be not fully symmetric with respect to the y axis, see Figure 5.



(a) $\hat{s}_0^0 \cdot \hat{c}_0$ and $\hat{s}_1^0 \cdot \hat{c}_1$ (b) $\hat{s}_0^1 \cdot \hat{c}_0$ and $\hat{s}_1^1 \cdot \hat{c}_1$

Figure 7: The mean power trace associated with the execution of the base multiplication function `basemul_asm` in Ψ_M for both shares, $\hat{s}^0 \star \hat{c}$ and $\hat{s}^1 \star \hat{c}$. The iterations of the function are marked by interleaving black and gray colors. Due to loop unrolling, 64 iterations are observed for each share instead of $n/2 = 128$. Recall that the ring dimension $n = 256$ for Kyber and 7-layer NTT is performed. The first iterations of `basemul_asm` for both shares are marked and zoomed in (a) and (b). The same observation applies to Ψ_P.

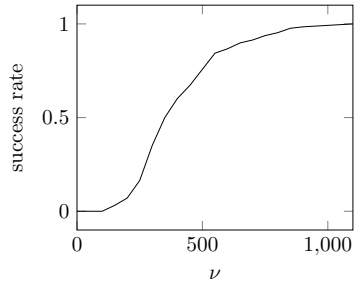


(a) Ψ_M (b) Ψ_P

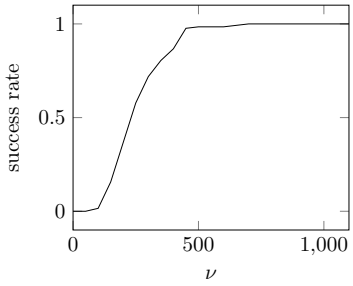
Figure 8: HOCPA with $\nu = 1000$ and f_{abs} targeting $\hat{s}_0$ for both Ψ_M and Ψ_P. The correlation scores of incorrect hypotheses are in gray, and for the correct hypothesis in black.

the attack. While this observation can be micro-architecture and implementation specific, we did not concentrate on improving it as the overall attack still leads to a reasonably low number of traces. We should also remark that the aim of this study is not to compare Ψ_M and Ψ_P since both implementations employ central reduction; rather the goal is to show that the approach generalizes to central reduction techniques.

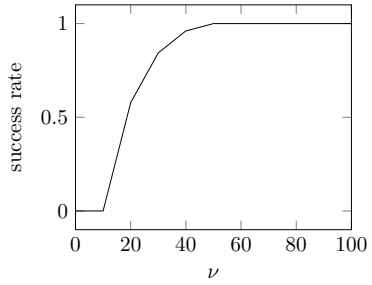
As a reference, we also included the success rate of a classical first-order CPA by the $\mathcal{W}_\beta$ prediction function performed on the same but unprotected implementations in Figure 9c and Figure 9d. In order to keep the consistency, we used the same part of the power traces as those considered in HOCPAs. Distinctively, we used both lower- and higher-degree coefficients from the output of each $\hat{s}_i \cdot \hat{c}_i$ as the target function, namely $g(\hat{s}_i, \hat{c}_i) = \{\hat{z}_{i,0}, \hat{z}_{i,1}\}$ (see Algorithm 1). We should also note that f_{abs} is designed to work



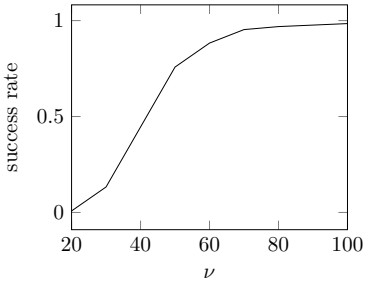
(a) HOCPA on masked Ψ_m with f_{abs}



(b) HOCPA on masked Ψ_p with f_{abs}



(c) CPA on unmasked Ψ_m with $\mathcal{W}_\beta$



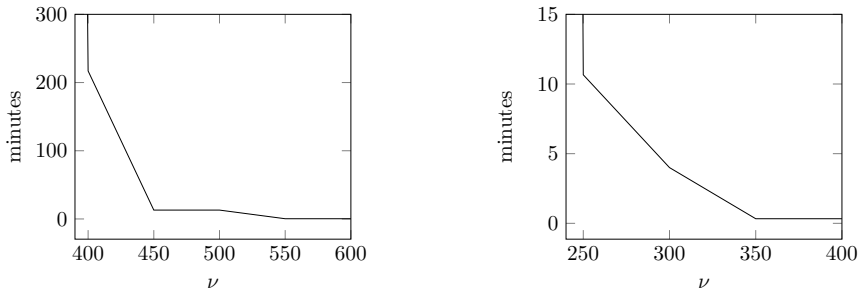
(d) CPA on unmasked Ψ_p with $\mathcal{W}_\beta$

Figure 9: Success rates of the SCA attacks against Kyber. The success rate refers to (# correctly predicted $\hat{s}_i$/128). Retrieving $\pm\hat{s}_i$ is considered as success.

with a single coefficient and we leave construction of a prediction function which takes multiple coefficients as the future work.

Bonus: Combining with Lattice Attack. As given in [KT23], the attacker can take the advantage of the fact that SCA attacks are performed in the NTT domain. Intuitively, there exists 3329^{256} possibilities for the NTT domain secret $\hat{s}$ while this number is 5^{256} for the normal domain secret s, indicating an over-determined system. The authors of [KT23] created an LWE instance from the inverse NTT transformation and showed that by retrieving only 38 out of 128 pairs of NTT domain coefficients, the rest of the coefficients can be revealed by solving the LWE, which is practically solvable. The challenge is to assess which NTT domain coefficients was predicted correctly. We applied a series of lattice attacks until an instance is successfully solved, starting from the subset of predictions to the NTT domain coefficients with highest correlation scores. Each execution of the lattice attack takes 20 seconds, returning success in case of the included subset of coefficients were correctly retrieved by the preceding SCA attack. Figure 10 depicts the relationship between the number of traces and the time needed by the lattice attack to succeed in our experiments. In particular for Ψ_M, the attack succeeds after 643 trials ($\approx$ 3.5 hours) with 400 traces while it succeeds with 250 traces for Ψ_P after 27 trials ($\approx$ 10 minutes)[8]. Indeed, $\nu = 250$ is considered as a very small number to conduct a successful non-profiling higher-order SCA attack on a masked implementation. The lattice attack is implemented using fpylll library[9].

[8]Except for $\nu = 250$ for Ψ_P and $\nu = 400$ for Ψ_M, the timing of lattice attacks are approximations.
[9]https://pypi.org/project/fpylll/



(a) Ψ_M (b) Ψ_P

Figure 10: Time required for the lattice attacks to successfully retrieve the whole secret polynomial s using the predictions for $\hat{s}_i$ which are obtained by HOCPAs.

8 Conclusions and Future Work

In this paper, we investigated the impact of various reduction schemes on SCA leakage of the implementation of modular arithmetics, with a specific focus on the central reduction. State-of-the-art masking LBC, *e.g.* [MGTF19, HKL+22, BGR+21, ABC+23b, BDK+21, HDR23, FBR+22, RRd+16, BC22, CGMZ23], concentrated on developing gadgets to handle non-linear operations and considered the linear part of the algorithms such as the polynomial arithmetic as relatively trivial to mask due to its transparency to arithmetic masking (*i.e.* repeating the operation on each share individually). However, our study reveals that the design decisions such as the reduction technique for the linear parts have a significant impact on the exploitability of the associated SCA leakages and hence on the number of traces required for a successful attack. Our study exposes this fact by showing that a HOCPA attack becomes significantly easier when the masked polynomial multiplication $\hat{s} \star \hat{c}$ is targeted and the modular reduction in the victim's implementation is central. Specifically, the use of signed arithmetic notably increases SCA leakage. We quantify this leakage by studying the optimal correlation and the number of traces required for a successful HOCPA attack.

Our findings reveal that the signed representation of integers modulo q leads to a strong dependency between the sign of an integer and its HW in 2's complement form. We assessed this dependency through simulations involving the parameter sets employed by the post-quantum cryptography winners Kyber and Dilithium. We also efficiently exploited this source of leakage, by introducing the absolute value prediction function. We believe that our work is unique in the literature as it is the only *non-profiled* SCA attack particularly designed to efficiently exploit higher-order leakages. We further have showcased our approach targeting a first-order masked implementation of Kyber. As our attack does not require profiling and is successful with only 250 traces (in our experiments and using our measurement setup), we claim that utilization of the central reduction in masked implementations indeed may ease SCA attacks. To the best of our knowledge, we report the lowest number of traces for a successful non-profiled second-order SCA attack against masked implementations of LBC. We leave the generalization of our introduced absolute value prediction function to higher orders as a work for the future.

As another outcome of our study, it can also be concluded that finding the sample points in power traces associated with the random arithmetic shares is trivial in masked implementations of LBC. Consequently, it is highly recommended to apply shuffling on top of arithmetic masking. Indeed, it is relatively easy to shuffle the base multiplication since $\hat{s}_i \cdot \hat{c}_i$ are performed independently for each i leading to $\frac{n}{2}!$ possible permutations.

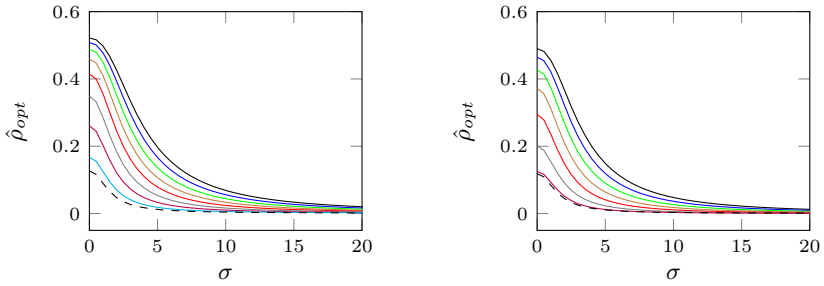
A Absolute Difference Combination Function

Absolute value combination function is defined as follows.

$$\mathcal{C}_{abs}\big(\mathcal{L}(X^0), \mathcal{L}(X^1)\big) = \big|\mathcal{L}(X^0) - \mathcal{L}(X^1)\big| \tag{19}$$

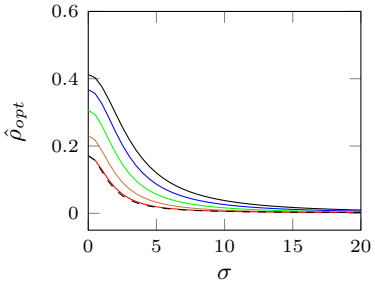
In the following analysis including Figure 11, Figure 12, Table 6, Table 7, and Table 8, for the sake of simplicity we took $\mu_0 = \mu_1 = 0$ regarding the noise terms in $\mathcal{L}(X^0)$ and $\mathcal{L}(X^1)$ (see Equation (4)). Note that, our results hold as long as $\mu_0 = \mu_1$. Results for $q = 8380417$ are not presented due to the computational complexity of experiments. However, they can be anticipated from the results for the other studied primes presented in this section and the ones corresponding to the mean-free product (Figure 3 and Table 5). Estimation of the corresponding optimal prefiction function – so-called $\hat{f}_{opt}(i)$ – is performed with 100 k samples uniformly taken for X^0, X^1, and the noise taken from $\mathcal{N}(0, \sigma)$ for each i.

A.1 Optimal Correlation for $\mathcal{C}_{abs}$



(a) $q = 257$

(b) $q = 769$



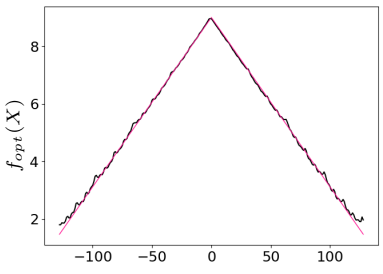
(c) $q = 3329$

Figure 11: Optimal correlation for $\mathcal{C}_{abs}$ with respect to the noise standard deviation σ for different q and β and reduction algorithms. Estimations are performed with 1 million samples uniformly taken for X^0 and X^1.

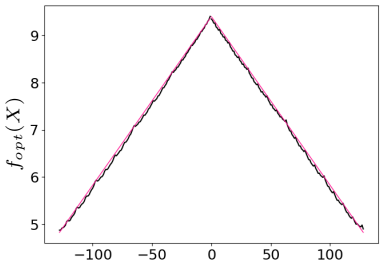
- ★ Reduction to $[-q/2, q/2]$ for $q = 257$, $q = 769$ and $q = 3329$: $\beta = 16$ (black), $\beta = 15$ (blue), $\beta = 14$ (green), $\beta = 13$ (brown), $\beta = 12$ (red), $\beta = 11$ (gray), $\beta = 10$ (purple), $\beta = 9$ (cyan)
- ★ Reduction to $[0, q)$ (dashed)

A.2 Accuracy of Absolute Value Prediction Function for $\mathcal{C}_{abs}$

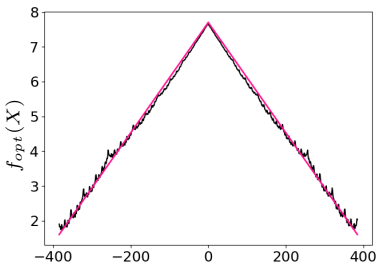
It can be seen in Figure 12 that $\hat{f}_{opt}$ for $\mathcal{C}_{abs}$ is an affine function of f_{abs} with some noise. Table 6, Table 7, and Table 8 present the estimations for $\hat{\rho}\left(f_{abs}\left(X\right), f_{opt}\left(X\right)\right)$ in this configuration. Observe that $|\hat{\rho}| > 0.99$ for the evaluated q and β couples mentioned in Section 4, allowing us to conclude that f_{abs} can be effectively used with $\mathcal{C}_{abs}$.



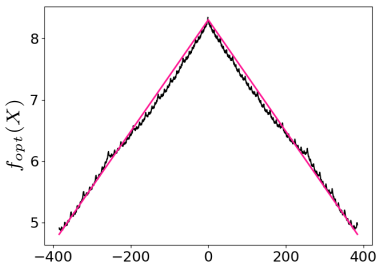
(a) $q = 257$, $\beta = 16$, $\sigma = 0$



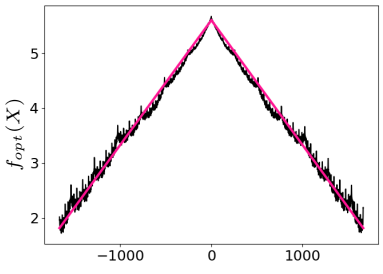
(b) $q = 257$, $\beta = 16$, $\sigma = 4$



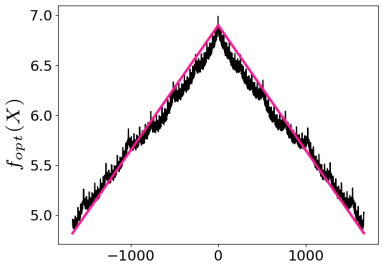
(c) $q = 769$, $\beta = 16$, $\sigma = 0$



(d) $q = 769$, $\beta = 16$, $\sigma = 4$



(e) $q = 3329$, $\beta = 16$, $\sigma = 0$



(f) $q = 3329$, $\beta = 16$, $\sigma = 4$

Figure 12: Visualization of optimal prediction function $\hat{f}_{opt}$ for the central reduction range $[-q/2, q/2]$ with respect to absolute difference combination function. $\hat{f}_{opt}(X)$ (black), $c'_0 \cdot |X| + c'_1$ (pink) for some c'_0 and c'_1.

Table 6: Estimations of $\hat{\rho}(f_{abs}(X), \hat{f}_{opt}(X))$ for $\mathcal{C}_{abs}$, $q = 257$ and different β and σ. The estimations are performed with 1 million uniformly random samples for X while the reduction is central.

β \ σ	0	2	4	6	8	10
16	**−0.999**	**−0.999**	**−0.999**	**−0.999**	**−0.999**	**−0.999**
15	−0.999	−0.999	−0.999	−0.999	−0.999	−0.998
14	−0.998	−0.998	−0.998	−0.998	−0.998	−0.998
13	−0.998	−0.998	−0.997	−0.997	−0.996	−0.996
12	−0.996	−0.996	−0.994	−0.993	−0.993	−0.992
11	−0.992	−0.990	−0.986	−0.984	−0.983	−0.982
10	−0.975	−0.964	−0.957	−0.952	−0.949	−0.947
9	−0.864	−0.834	−0.818	−0.809	−0.801	−0.793

Table 7: Estimations of $\hat{\rho}(f_{abs}(X), \hat{f}_{opt}(X))$ for $\mathcal{C}_{abs}$, $q = 769$ and different β and σ. The estimations are performed with 1 million uniformly random samples for X while the reduction is central.

β \ σ	0	2	4	6	8	10
16	**−0.998**	**−0.998**	**−0.998**	**−0.998**	**−0.998**	**−0.998**
15	−0.997	−0.998	−0.997	−0.997	−0.996	−0.996
14	−0.996	−0.996	−0.995	−0.994	−0.994	−0.994
13	−0.993	−0.992	−0.990	−0.989	−0.988	−0.988
12	−0.985	−0.979	−0.974	−0.972	−0.971	−0.970
11	−0.941	−0.923	−0.912	−0.908	−0.905	−0.902
10	−0.673	−0.645	−0.632	−0.626	−0.620	−0.614

Table 8: Estimations of $\hat{\rho}(f_{abs}(X), \hat{f}_{opt}(X))$ for $\mathcal{C}_{abs}$, $q = 3329$ and different β and σ. The estimations are performed with 1 million uniformly random samples for X while the reduction is central.

β \ σ	0	2	4	6	8	10
16	**−0.996**	**−0.995**	**−0.993**	**−0.993**	**−0.992**	**−0.992**
15	−0.992	−0.989	−0.986	−0.985	−0.984	−0.983
14	−0.978	−0.971	−0.965	−0.962	−0.961	−0.959
13	−0.913	−0.893	−0.882	−0.877	−0.872	−0.869
12	−0.562	−0.551	−0.544	−0.539	−0.534	−0.529

References

[AAT+21] Furkan Aydin, Aydin Aysu, Mohit Tiwari, Andreas Gerstlauer, and Michael Orshansky. Horizontal side-channel vulnerabilities of post-quantum key exchange and encapsulation protocols. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(6):1–22, 2021.

[ABC+23a] Aikata Aikata, Andrea Basso, Gaetan Cassiers, Ahmet Can Mert, and Sujoy Sinha Roy. Kavach: Lightweight masking techniques for polynomial arithmetic in lattice-based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 366–390, 2023.

[ABC+23b] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. Protecting dilithium against leakage revisited sensitivity analysis and improved implementations. *IACR TCHES*, 2023(4):58–79, 2023.

[ABCG20] Erdem Alkim, Yusuf Alper Bilgin, Murat Cenk, and François Gérard. Cortex-m4 optimizations for {R, M} lwe schemes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 336–357, 2020.

[ACC+22] Amin Abdulrahman, Jiun-Peng Chen, Yu-Jia Chen, Vincent Hwang, Matthias J Kannwischer, and Bo-Yin Yang. Multi-moduli ntts for saber on cortex-m3 and cortex-m4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 127–151, 2022.

[AHKS22] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Amber Sprenkels. Faster kyber and dilithium on the cortex-M4. In Giuseppe Ateniese and Daniele Venturi, editors, *ACNS 22*, volume 13269 of *LNCS*, pages 853–871. Springer, Heidelberg, June 2022.

[BAA+19] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[BAE+24] Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes, and Tobias Schneider. Exploiting small-norm polynomial multiplication with physical attacks: Application to crystals-dilithium. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):359–383, 2024.

[Bar87] Paul Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 311–323. Springer, Heidelberg, August 1987.

[BBC+20] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, Chitchanok Chuengsatiansup, Tanja Lange, Adrian Marotzke, Bo-Yuan Peng, Nicola Tuveri, Christine van Vredendaal, and Bo-Yin Yang. NTRU Prime. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[BC22] Olivier Bronchain and Gaëtan Cassiers. Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based KEMs. *IACR TCHES*, 2022(4):553–588, 2022.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, Heidelberg, August 2004.

[BDK+21] Michiel Van Beirendonck, Jan-Pieter D'anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of saber. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 17(2):1–26, 2021.

[BGR+21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR TCHES*, 2021(4):173–214, 2021. https://tches.iacr.org/index.php/TCHES/article/view/9064.

[BKS19] Leon Botros, Matthias J. Kannwischer, and Peter Schwabe. Memory-efficient high-speed implementation of Kyber on cortex-M4. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 209–228. Springer, Heidelberg, July 2019.

[BNGD23] Linus Backlund, Kalle Ngo, Joel Gärtner, and Elena Dubrova. Secret key recovery attack on masked and shuffled implementations of crystals-kyber and saber. In *International Conference on Applied Cryptography and Network Security*, pages 159–177. Springer, 2023.

[CDH+20] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[CGMZ23] Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order polynomial comparison and masking lattice-based encryption. *IACR TCHES*, 2023(1):153–192, 2023.

[CGTZ23] Jean-Sébastien Coron, François Gérard, Matthias Trannoy, and Rina Zeitoun. Improved gadgets for the high-order masking of dilithium. *IACR TCHES*, 2023(4):110–145, 2023.

[CHK+21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. Ntt multiplication for ntt-unfriendly rings: New speed records for saber and ntru on cortex-m4 and avx2. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 159–188, 2021.

[CKA+21] Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma, and Jiwu Jing. An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 583–590. IEEE, 2021.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[DNGW23] Elena Dubrova, Kalle Ngo, Joel Gärtner, and Ruize Wang. Breaking a fifth-order masked implementation of crystals-kyber by copy-paste. In *Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop*, pages 10–20, 2023.

[FBR+22] Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. Masked accelerators and instruction set extensions for post-quantum cryptography. *IACR TCHES*, 2022(1):414–460, 2022.

[GKS21] Denisa O. C. Greconici, Matthias J. Kannwischer, and Amber Sprenkels. Compact dilithium implementations on cortex-M3 and cortex-M4. *IACR TCHES*, 2021(1):1–24, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8725.

[GS66] W Morven Gentleman and Gordon Sande. Fast fourier transforms: for fun and profit. In *Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 563–578, 1966.

[HDR23] Daniel Heinz and Gabi Dreo Rodosek. Fast first-order masked nttru. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 127–148. Springer, 2023.

[HKL+22] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. First-order masked kyber on ARM cortex-M4. Cryptology ePrint Archive, Report 2022/058, 2022. https://eprint.iacr.org/2022/058.

[HZZ+22] Junhao Huang, Jipeng Zhang, Haosong Zhao, Zhe Liu, Ray C. C. Cheung, Çetin Kaya Koç, and Donglong Chen. Improved plantard arithmetic for lattice-based cryptography. *IACR TCHES*, 2022(4):614–636, 2022.

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On second-order differential power analysis. In *Cryptographic Hardware and Embedded Systems–CHES 2005: 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005. Proceedings 7*, pages 293–308. Springer, 2005.

[KAA22] Emre Karabulut, Erdem Alkim, and Aydin Aysu. Single-trace side-channel attacks on ω-small polynomial sampling: With applications to NTRU, NTRU prime, and CRYSTALS-DILITHIUM. Cryptology ePrint Archive, Report 2022/494, 2022. https://eprint.iacr.org/2022/494.

[KDVB+22] Suparna Kundu, Jan-Pieter D'Anvers, Michiel Van Beirendonck, Angshuman Karmakar, and Ingrid Verbauwhede. Higher-order masked saber. In *International Conference on Security and Cryptography for Networks*, pages 93–116. Springer, 2022.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

[KLH+20] Il-Ju Kim, Tae-Ho Lee, Jaeseung Han, Bo-Yeon Sim, and Dong-Guk Han. Novel single-trace ML profiling attacks on NIST 3 round candidate dilithium. Cryptology ePrint Archive, Report 2020/1383, 2020. https://eprint.iacr.org/2020/1383.

[KRSS19a] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking NIST PQC on ARM cortex-M4. Cryptology ePrint Archive, Report 2019/844, 2019. https://eprint.iacr.org/2019/844.

[KRSS19b] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4. 2019.

[KT23] Yen-Ting Kuo and Atsushi Takayasu. A lattice attack on crystals-kyber with correlation power analysis. In *International Conference on Information Security and Cryptology*, pages 202–220. Springer, 2023.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

[LS19] Vadim Lyubashevsky and Gregor Seiler. Nttru: Truly fast ntru using ntt. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 180–201, 2019.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium: Efficient implementation and side-channel evaluation. In *Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17*, pages 344–362. Springer, 2019.

[Mon85] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985.

[MUTS22] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on dilithium: A small bit-fiddling leak breaks it all. *Cryptology ePrint Archive*, 2022.

[MWK+24] Catinca Mujdei, Lennert Wouters, Angshuman Karmakar, Arthur Beckers, Jose Maria Bermudo Mera, and Ingrid Verbauwhede. Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication. *ACM Transactions on Embedded Computing Systems*, 23(2):1–23, 2024.

[ÖY23] Sıla Özeren and Oğuz Yayla. Methods for masking crystals-kyber against side-channel attacks. In *2023 16th International Conference on Information Security and Cryptology (ISCTürkiye)*, pages 1–6. IEEE, 2023.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Pla21] Thomas Plantard. Efficient word size modular arithmetic. *IEEE Transactions on Emerging Topics in Computing*, 9(3):1506–1518, 2021.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 513–533. Springer, Heidelberg, September 2017.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Transactions on computers*, 58(6):799–811, 2009.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[RRd+16] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *Journal of Cryptographic Engineering*, 6(2):139–153, June 2016.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Sei18] Gregor Seiler. Faster avx2 optimized ntt multiplication for ring-lwe lattice cryptography. *Cryptology ePrint Archive*, 2018.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[SLKG22] Hauke Steffen, Georg Land, Lucie Kogelheide, and Tim Güneysu. Breaking and protecting the crystal: Side-channel analysis of Dilithium in hardware. Cryptology ePrint Archive, Report 2022/1410, 2022. https://eprint.iacr.org/2022/1410.

[SLKG23] Hauke Steffen, Georg Land, Lucie Kogelheide, and Tim Güneysu. Breaking and protecting the crystal: Side-channel analysis of dilithium in hardware. In *International Conference on Post-Quantum Cryptography*, pages 688–711. Springer, 2023.

[SVO+10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 112–129. Springer, Heidelberg, December 2010.

[TS24] Tolun Tosun and Erkay Savas. Zero-value filtering for accelerating non-profiled side-channel attack on incomplete ntt based implementations of lattice-based cryptography. *IEEE Transactions on Information Forensics and Security*, 2024.

[XPR+21] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Transactions on Computers*, 71(9):2163–2176, 2021.