IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 3, 24 pages.

https://doi.org/10.62056/a0txl8n4e

Check for updates

Non-Profiled Higher-Order Side-Channel Attacks against Lattice-Based Post-Quantum Cryptography

Tolun Tosun[1], Elisabeth Oswald[2] and Erkay Savaş[1]

[1] Sabanci University, Istanbul, Turkiye
[2] University of Klagenfurt, Klagenfurt, Austria

Abstract. In this work, we present methods for conducting higher-order non-profiled side-channel attacks on Lattice-Based Cryptography (LBC). Our analysis covers two scenarios: one where the device leakage is known and follows Hamming weight model, and another where the leakage model is not Hamming weight based and unknown to the attacker. We focus on the Post-Quantum Cryptography (PQC) standards, the Dilithium digital signature (*i.e.* ML-DSA) and the Kyber key encapsulation (*i.e.* ML-KEM) algorithms. For Hamming weight leakage, we develop efficient higher-order Correlation Power Analysis (HOCPA) attacks in which the attacker must compute a function known as the optimal prediction function. We revisit the definition of optimal prediction function and introduce a recursive method for computing it efficiently. Our approach is particularly useful when a closed-form formula is unavailable, as in LBC. Then, we introduce **sin** and **cos** prediction functions, which prove optimal for HOCPA attacks against second and higher-order masking protection. We validate our methods through simulations and real-device experiments on open-source masked implementations of Dilithium and Kyber on an Arm Cortex-M4. On the real device, we achieve full secret-key recovery using only **700** and **2400** traces for second and third-order masked implementations of Dilithium, and **2200** and **14500** traces for second and third-order masked implementations of Kyber, respectively. For the unknown leakage scenarios, we leverage generic Side-Channel Analysis (SCA) distinguishers. A key challenge here is the injectivity of modular multiplications in NTT based polynomial multiplication, typically addressed by bit-dropping in the literature. However, we experimentally show that bit-dropping is largely inefficient against protected implementations of LBC. To overcome this limitation, we present a novel two-step attack to Kyber, combining generic distinguishers and lattice reduction techniques. Our approach decreases the number of predictions from q^2 to q and does not rely on bit-dropping. Our experimental results demonstrate a speed-up of up to **23490**× in attack run-time over the baseline along with improved success rate. In certain scenarios, higher-order attacks become feasible only through the proposed approach, as classical methods are shown to be unsuccessful.

Keywords: Higher-Order Side-Channel Analysis · Correlation Power Analysis · Kyber · Dilithium · Masking · Kruskal-Wallis Test

1 Introduction

Shor's algorithm [Sho94] has established that quantum computing poses a significant threat to classical public-key cryptographic schemes, including RSA and ECC currently widely

E-mail: toluntosun@sabanciuniv.edu (Tolun Tosun), Elisabeth.Oswald@aau.at (Elisabeth Oswald), erkay.savas@sabanciuniv.edu (Erkay Savaş)

Received: 2025-07-08 Accepted: 2025-09-02

deployed in secure communication systems. Consequently, LBC has gained significant attention due to its conjectured resistance to quantum attacks. Among the four initial PQC standards published by NIST, three of them are lattice-based schemes. Among those, Crystals-Kyber [BDK+18] Key Encapsulation Mechanism (KEM) and Crystals-Dilithium [DKL+18] Digital Signature Algorithm (DSA) are based on module lattices (ML), which are standardized under the names ML-KEM and ML-DSA, respectively.

While lattice-based cryptographic algorithms are resistant to quantum attacks, protecting them against SCA attacks is a highly active research area, and potential attack scenarios must be carefully considered. Power based SCA attacks are generally categorized into profiled and non-profiled ones. Profiled attacks assume access to an identical (open) clone of the victim device for power measurement and modeling, whereas non-profiled attacks operate without such a reference. To mitigate power based SCA attacks, masking [CJRR99] is the state-of-the-art countermeasure, and its effectiveness is quantified by the masking order.

The main operation in LBC is the polynomial multiplication, which is efficiently implemented using the Number Theoretic Transform (NTT) algorithm. This operation is also the main target for non-profiled SCA attacks against LBC [CKA+21, MWK+24, TS24, TMS24, SLKG23, RBVB23, QLZ+23, KT23]. This is expected, as polynomial multiplication involves both secret and public data as input operands, which is a necessary condition for non-profiled attacks. When the power leakage characteristic of the victim device is known to be a function of the Hamming weight of processed data, it is effectively exploited by the well-known CPA [BCO04]. However, a function known as the optimal prediction function must be calculated to tackle the masking countermeasure [PRB09]. While this computation is straightforward for symmetric cryptography, additional effort is required for attacking protected implementations of LBC [TMS24].

Although the Hamming weight assumption usually holds for software implementations, the leakage characteristic of the victim device can be more complex for hardware implementations [LBS19, GMPO19]. In such cases, and especially when the leakage model cannot be recovered through a profiling device, it remains unknown to the attacker. A common approach in this scenario is to use so-called generic distinguishers, which perform the profiling on-the-fly. Efficient examples of generic distinguishers are Mutual Information Analysis (MIA) [GBTP08] and Kruskal-Wallis Test (KW) [YOR23]. Application of those in the context of LBC is also challenging and remains relatively underexplored. This difficulty arises primarily from the fact that the modular multiplications in the NTT based polynomial multiplication are injective functions [HRG14]. A known workaround is to use the so-called bit-dropping trick; however, this method incurs information loss and we demonstrate that its efficacy against masked LBC implementations remains uncertain. Among the existing non-profiled attacks against LBC, only [TS24, TMS24, QLZ+23] present attack results against protected implementations, which are limited to first-order masking and also assume Hamming weight leakage. In this work, we explore and develop methodologies that enable higher-order non-profiled SCA attacks against LBC in both Hamming weight leakage and non-Hamming weight leakage scenarios.

Related Work. The most of the published SCA attacks against LBC is profiled [PPM17, BAE+24, DNGW23, BNGD23, WBD24, XPR+21, KAA21]. In the profiled setting, [PPM17] showed that a single measurement from the victim device is sufficient to retrieve the secret input of NTT in certain circumstances. [DNGW23, BNGD23, WBD24, XPR+21, UMTS24] target data conversion primitives between polynomial and binary representations, such as those involving message encoding or decoding. These studies (except [UMTS24]) are essentially chosen-ciphertext attacks (CCA) against Kyber, which require sending specially crafted ciphertexts to the attacked device. Notably, [DNGW23] successfully attacks an open-source fifth-order masked implementa-

tion of Kyber using deep-learning techniques. The authors also discuss that their methodology can be considered as a public profiling attack. In this case, the attack still involves a profiling step, as in traditional profiled attacks. However, the profiling can be performed directly on the victim device, eliminating the need for a separate clone device accessible to the attacker. As previously mentioned, to the best of our knowledge, all non-profiled SCA attacks target the NTT based polynomial multiplication [CKA+21, MWK+24, TS24, TMS24, SLKG23, RBVB23, QLZ+23, KT23]. [TMS24] shows that the signed arithmetic widely employed in efficient implementations of LBC leads to a vulnerability by creating a strong dependency between the Hamming weight and sign of the number. The authors leverage this observation by introducing the absolute value prediction function, which is effective against first-order masking. [QLZ+23, KT23] combine SCA attacks and other cryptoanalysis techniques such as lattice attacks. In particular, these studies post-process the output of the SCA attack using lattice reduction algorithms such as LLL and BKZ, allowing a significant number of incorrect predictions by the SCA to be tolerated and corrected.

Our Contributions. Below is a list of the contributions introduced in this work.

- We introduce a novel and efficient method for computing the optimal prediction functions for higher-order SCA attacks using a recursive approach. Our approach enables the derivation of closed-form expressions for higher-order attacks when possible, while also offering advantages in scenarios where closed-form solutions are infeasible, such as when the attacked device uses unsigned modular arithmetic in LBC.
- We propose novel prediction functions, based on *sin* and *cos* functions, for HOCPA attacks against the arithmetic masking in LBC. These functions are particularly effective for attacking second and higher-order masking when the modular arithmetic in the target implementation is signed.
- We explore application of generic distinguishers, such as MIA and KW, for attacking lattice-based schemes when the leakage model of the victim device is unknown. We present a novel two-step attack on incomplete NTT multiplication (*e.g.* Kyber) by combining generic distinguishers with lattice problems, namely solving instances of the SVP. We show that our approach is particularly favorable against protected implementations.
- Our study reveals that, recovering the *delta*s, the factors between the even and odd-degree coefficients of Kyber's secret polynomials in incomplete NTT domain, are sufficient to recover the entire secret polynomial.
- We provide practical side-channel attack results against higher-order protected implementations. Our results are unique in the literature as these are the first non-profiled third- and fourth-order attacks demonstrated against both Kyber and Dilithium. We also present the first non-profiled attack results in which the victim device's leakage function is unknown to the attacker.

2 Background

2.1 Notation

Uppercase bold letters denote matrices while lowercase bold letters denote vectors. Elements of those are accessed by the lower-index, such as $\mathbf{A}_{i,j}$ and $\mathbf{a}_i$. We use the notation $[\mathbf{a}_i]_{i=0}^{m}$ to denote a vector formed by the elements $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_m$. Uppercase *monospace* letters denote sets. Elements of those are accessed by the lower-index, such as $\mathtt{A}_i$. We use the notation $\{\mathtt{A}_i\}_{i=0}^{m}$ to denote a set containing elements $\mathtt{A}_0, \mathtt{A}_1, \ldots, \mathtt{A}_m$. Lowercase italic letters denote variables, such as x. Uppercase letters denote random variables, such as X. $\leftarrow$

denotes sampling uniformly at random, such as $X \leftarrow \mathtt{X}$. Upper-index with round brackets denotes samples of random variables, such as $X^{(i)}$. Upper-index with curly brackets denotes secret shares of variables used in masking schemes, such as $X = X^{\{0\}} + X^{\{1\}}$. We also use the shorthand notation $X^{\{0:d-1\}} = \{X^{\{0\}}, X^{\{1\}}, .., X^{\{d-1\}}\}$ to refer to all d shares. Polynomials are also denoted by lowercase italic letters, such as $a(x)$. For brevity, we often omit the indeterminate x and simply refer to the polynomial as a. Calligraphic letters denote functions, such as $\mathcal{F}$.

2.2 Lattice-Based Cryptography (LBC)

Crystals-Kyber and Crystals-Dilithium are among the first Post-Quantum Cryptography (PQC) standards selected by NIST, now standardized under the names ML-KEM and ML-DSA, respectively. The main mathematical object is the ring of polynomials $\mathcal{R}_q = \mathbb{Z}_q/(x^n + 1)$, where n is referred to as the ring modulus and q is referred to as the coefficient modulus. In this ring, each element is a polynomial of degree $n - 1$ whose coefficients are in $\mathbb{Z}_q$. The security of Kyber and Dilithium relies on the well known Ring Learning With Errors (R-LWE) problem [Reg09].

Kyber. The secret and public key pair is generated through the R-LWE equation $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $(\mathbf{t}, \mathbf{A}) \in (\mathcal{R}_q^k \times \mathcal{R}_q^{k\times k})$ is public and $\mathbf{s} \in \mathcal{R}_q^k$ is the vector of secret polynomials. The vector of polynomials $\mathbf{e}$ represents the error term and is discarded after the key generation process. The parameter k is depends on the chosen security level. Coefficients of polynomials in $\mathbf{s}$ are sampled from a centered binomial distribution whose output is in $[-\eta, \eta]$ where η is another parameter determined by the desired security level. Specifically, (k, η) takes the values $(2, 3)$, $(3, 2)$, and $(4, 2)$ for the low (Kyber512), medium (Kyber768), and high (Kyber1024) security levels, respectively. For all security levels, $q = 3329$ and $n = 256$.

Dilithium. The key generation in Dilithium is also based on the R-LWE equation $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. In contrast to Kyber, the error term is retained as part of the secret key in Dilithium and is denoted by $\mathbf{s}_2 \in \mathcal{R}_q^k$. On the other hand, $\mathbf{A} \in \mathcal{R}_q^{l\times k}$ is not necessarily a square matrix. Coefficients of the polynomials in $\mathbf{s}_1$ and $\mathbf{s}_2$ are short like Kyber. In this case, these are sampled uniformly at random from $[-\eta, \eta]$. The relaed parameters (k, l, η) is $(4, 4, 2)$, $(6, 5, 4)$, $(8, 7, 2)$ for low (Dilithium2), medium (Dilithium3) and high (Dilithium5) security levels of Dilithium, respectively. For all security levels, $q = 8380417$ and $n = 256$.

2.3 Number Theoretic Transform (NTT)

The Number Theoretic Transform (NTT) is the state-of-the-art approach for performing polynomial multiplication in $\mathcal{R}_q$. For operands $s, c \in \mathcal{R}_q$, the outline for NTT-style multiplication is as follows:

$$s \cdot c = \mathrm{NTT}^{-1}(\mathrm{NTT}(s) \star \mathrm{NTT}(c)) \tag{1}$$

NTT and NTT^{-1} denote the forward and inverse applications of the NTT, respectively. The multiplication in NTT domain is referred to as the *base multiplication*, which is performed element-wise and denoted by $\star$. To simplify notation, we denote NTT-domain representation of polynomials by using *hat*, $\hat{s} = \mathrm{NTT}(s)$. While the NTT and inverse NTT is usually implemented by so-called butterfly circuits in $O(n \log n)$ time, it is also possible to model these as matrix multiplications. For instance, the inverse NTT can be expressed as $s = \mathbf{M}\hat{s}$, where $\mathbf{M} \in \mathbb{Z}_q^{n\times n}$ is considered as the transformation matrix while s and $\hat{s}$ are represented by their coefficient vector.

Dilithium's Complete NTT. For the NTT to be *complete* over $\mathcal{R}_q$, the modulus q must satisfy $q = 1 \pmod{2n}$. The modulus employed by Dilithium leads to a complete NTT. In this case, the element-wise multiplication refers to n independent modular multiplications, $\hat{s}_i \cdot \hat{c}_i \pmod{q}$ where $\hat{s}_i, \hat{c}_i \in \mathbb{Z}_q$ for $0 \leq i < n$.

Kyber's Incomplete NTT. On the other hand, the modulus q employed by Kyber leads to an *incomplete* NTT as $q = 1 \pmod{n}$ for $q = 3329$. Then, each $\hat{s}_i \cdot \hat{c}_i$ corresponds to multiplication of degree-1 polynomials where $\hat{s}_{i,0}, \hat{s}_{i,1}, \hat{c}_{i,0}, \hat{c}_{i,1} \in \mathbb{Z}_q$ for $0 \leq i < n/2$. The base multiplication algorithm for this case is presented in Algorithm 1.

Modular Reduction. Operating over $\mathcal{R}_q$ for the NTT and other operations require performing modular arithmetic in $\mathbb{Z}_q$. In classical cryptography, the elements of $\mathbb{Z}_q$ are represented by the unsigned range $[0, q)$. On the other hand, many implementations of LBC employs signed arithmetic in the central range $[-q/2, q/2]$ [KRSS19]. We evaluate both of these reduction ranges in our study.

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 Number Theoretic Transform (NTT) transformation on $s, c \in \mathcal{R}_q$

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

1: **for** $\forall i \in \{0, \ldots, n/2 - 1\}$ **do** ▷ Compute $\hat{z}_i = \hat{s}_i \cdot \hat{c}_i$
2: $\quad \hat{z}_{i,0} \leftarrow \hat{s}_{i,0} \cdot \hat{c}_{i,0} + \hat{s}_{i,1} \cdot \hat{c}_{i,1} \cdot \zeta_i \pmod{q}$ ▷ δ_i is the twiddle factor
3: $\quad \hat{z}_{i,1} \leftarrow \hat{s}_{i,1} \cdot \hat{c}_{i,0} + \hat{s}_{i,0} \cdot \hat{c}_{i,1} \pmod{q}$
4: **end for**

2.4 Non-Profiled SCA Attack

We assume that the side-channel leakage L is a random variable that is the sum of a data-dependent leakage function $\mathcal{L}(X)$ for the random intermediate variable $X \in \mathtt{X}$ and independent noise following Gaussian distribution $\mathcal{N}(\mu, \sigma)$:

$$L = \mathcal{L}(X) + \mathcal{N}(\mu, \sigma) \tag{2}$$

In the non-profiled SCA attack setting, the attacker has access to ν side-channel leakages $\mathtt{L} = \{L^{(i)}\}_{i=0}^{\nu-1}$ and the corresponding publicly known and varying inputs to the attacked algorithm $\mathtt{C} = \{C^{(i)}\}_{i=0}^{\nu-1}$. The known data are referred to as *traces*. The attacker chooses a target function $\mathcal{G}$ that combines the unknown secret s and C with $X = \mathcal{G}(s, C)$. The attacker places an hypothesis s' for the attacked secret and computes the set of hypothetical intermediates $\mathtt{X}' = \{X'^{(i)}\}_{i=0}^{\nu-1}$ where $X'^{(i)} = \mathcal{G}(s', C^{(i)})$. Notice that we treat X' as a random variable defined as $X' = \mathcal{G}(s', C)$, since it depends on the randomness of C. Then, L is statistically compared to X' over the observations $\mathtt{L}$ and $\mathtt{X}'$. The hypothesis with the best statistical score is the output of the attack. The statistical method used by the attacker is known as *distinguisher* in the SCA literature. Distinguishers studied in this work are explained in Section 2.6.

Application to NTT Multiplication in LBC. We denote the secret polynomial in lattice-based cryptosystems by s and its representation in the NTT domain by $\hat{s}$. This can correspond to any $\mathbf{s}_i$ in Kyber, or to $\mathbf{s}_{1,i}$ or $\mathbf{s}_{2,i}$ in Dilithium, for any i. To perform non-profiled SCA attack on s, a natural choice for the target function is the base multiplication [TS24, TMS24, CKA+21, MWK+24], which allows attacking each $\hat{s}_i$ independently. When the NTT is complete as in Dilithium, this is a set of independent modular multiplications, $\mathcal{G}(\hat{s}_i, \hat{c}_i) = \hat{s}_i \cdot \hat{c}_i \mod q$. For the incomplete NTT case as in Kyber and specific implementations of Dilitihum [AHKS22], the base multiplication is presented

in Algorithm 1, which consists of independent multiplications of degree-1 polynomials. The attacker can target the lower degree-coefficient from these multiplications $\mathcal{G}(\hat{s}_i, \hat{c}_i) = \hat{z}_{i,0}$, the higher degree coefficient $\mathcal{G}(\hat{s}_i, \hat{c}_i) = \hat{z}_{i,1}$, or both $\mathcal{G}(\hat{s}_i, \hat{c}_i) = \hat{z}_{i,0}|\hat{z}_{i,1}$. For any of these target functions, both $\hat{z}_{i,0}$ and $\hat{z}_{i,1}$ depend on both $\hat{s}_{i,0}$ and $\hat{s}_{i,1}$, so these must be predicted together [MWK+24, TS24, TMS24].

2.5 Masking of NTT Multiplication

The masking countermeasure operates as a secret sharing scheme through the cryptographic execution. For $(d-1)$-th order masking, d shares are used. In particular for NTT-based polynomial multiplication, the secret polynomial is masked as $s = s^{\{0\}} + s^{\{1\}} + .. + s^{\{d-1\}}$ where $s^{\{j\}}$ are sampled uniformly at random in $\mathcal{R}_q$ for $0 \leq j < d-1$ and the last share is computed as $s^{\{d-1\}} = s - \sum_{j=0}^{d-2} s^{\{j\}}$. Since shares are distributed additively, this type of masking is known as *arithmetic masking*. We refer to the set of d secret shares by $s^{\{0:d-1\}}$. As a result of masking s, the polynomial multiplication $s \cdot c$ is performed over the shares independently, $s^{\{j\}} \cdot c$ for $0 \leq j < d$, where $s \cdot c = \sum_{j=0}^{d-1} s^{\{j\}} \cdot \hat{c}$. Accordingly, the base multiplication, which is the target function for the SCA attack explained in the previous section, is also applied share-wise, $\hat{s}^{\{j\}} \star \hat{c}$. The output of the chosen target function for individual shares, $X^{\{j\}} = \mathcal{G}(\hat{s}_i^{\{j\}}, \hat{c}_i)$, satisfies $X = \sum_{j=0}^{d-1} X^{\{j\}} \pmod q$ (except for the case $\mathcal{G}(\hat{s}_i, \hat{c}_i) = \hat{z}_{i,0}|\hat{z}_{i,1}$). We frequently refer to the latter equality throughout this paper. Note that power leakages corresponding to any $d-1$ shares, *e.g.* $\{\mathcal{L}(X^{\{i\}})\}_{i=0}^{d-2}$, and X are mutually independent.

2.6 SCA Distinguishers

2.6.1 Correlation Power Analysis (CPA)

CPA [BCO04] is a widely-used side-channel distinguisher, based on Pearson's correlation. The attacker estimates the correlation coefficient between X' and L as follows:

$$\hat{\rho}(\mathcal{L}'(X'), L) = \frac{\hat{\text{cov}}(\mathcal{L}'(\mathtt{X}'), \mathtt{L})}{\hat{\text{std}}(\mathcal{L}'(\mathtt{X}')) \cdot \hat{\text{std}}(\mathtt{L})} \tag{3}$$

CPA requires the attacker to predict the device leakage function, denoted by $\mathcal{L}'$.

2.6.2 Higher-Order CPA (HOCPA)

To perform a CPA attack on a masked implementation, a function that combines the leakage of shares $L_0, L_1, .., L_{d-1}$ must be used. The most frequently used combination function is the mean-free product [PRB09], which is defined as follows for a d-th order attack:

$$\mathcal{C}(L_0, L_1, .., L_{d-1}) = \prod_{i=0}^{d-1} \big(L_i - E[L_i]\big) \tag{4}$$

To simplify the notation, we use $\mathcal{C}(\{L_i\}_{i=0}^{d-1})$ for the above definition. To efficiently perform a HOCPA attack by means of a combination function, an optimal prediction function must be calculated [PRB09]. Specifically, for a d-th order attack:

$$f_{opt}^d(x) = E\left[\mathcal{C}\big(\{\mathcal{L}'(X^{\{i\}})\}_{i=0}^{d-1}\big) \big| X = x\right] \tag{5}$$

where $X = \sum_{i=0}^{d-1} X^{\{i\}}$ thus the expectation is taken over the random shares $X^{\{0:d-2\}}$. The attacker then estimates $\hat{\rho}(f_{opt}^d(X'), \mathcal{C}(\{L_i\}_{i=0}^{d-1}))$. The correlation achieved by the optimal prediction function f_{opt}^d is called *optimal correlation*.

2.6.3 Mutual Information Analysis (MIA)

MIA [GBTP08] is a generic and information-theoretic side-channel distinguisher. Unlike CPA, MIA does not require the attacker to predict a specific device leakage function $\mathcal{L}'$. Let $\mathcal{H}(L)$ denote the entropy of L, and let $\mathcal{H}(L|X')$ denote the conditional entropy for L and X'. The mutual information between L and X' is defined as

$$\mathcal{I}(L, X') = \mathcal{H}(L) - \mathcal{H}(L|X') \tag{6}$$

Verbally, the entropy in L not covered by X', namely $\mathcal{H}(L|X')$, is subtracted from $\mathcal{H}(L)$ to formulate the mutual information in between. In practice, the attacker estimates $\hat{\mathcal{I}}$ over $\mathtt{L}$ and $\mathtt{X}'$. Computing $\mathcal{H}$ requires the probability density functions of its input, X' and L. A common practice to estimate probabilities over observations is to use the histogram method. MIA trivially generalizes to a higher-order attack. For $d = 2$:

$$\mathcal{I}(L_0, L_1, X') = \mathcal{I}(L_0, L_1) - \mathcal{I}(L_0, L_1|X') \tag{7}$$

A drawback of generic distinguishers is that these can't distinguish if the target is an injective function, such as the modular multiplication employed in this study. A solution to address this challenge to apply bit-dropping to the output of the target function so it is no longer injective.

2.6.4 Kruskal-Wallis Test (KW)

KW [YOR23] is another generic distinguisher based on ranked side-channel leakages. Initially, the observed leakages $\mathtt{L}$ are ranked from smallest to largest while the ties are resolved by assigning the average of the ranks that the ties would have received, denoted by $\mathtt{R} = \text{rank}(\mathtt{L})$. The ranked data are partitioned based on hypothetical intermediates $\mathtt{X}'$. The set $\mathtt{R}^x = \{\mathtt{R}_i \mid \mathtt{X}'_i = x, 0 \leq i < \nu\}$ contains the ranks of leakages corresponding the traces where the intermediate $X' = x$.

Then, KW statistic is defined as

$$\mathcal{KW}(\mathtt{X}', \mathtt{R}) = (\nu - 1)\frac{\sum_{x \in \mathtt{X}} \nu_x (\overline{\mathtt{R}}^x - (\nu+1)/2)^2}{\sum_{x \in \mathtt{X}} \sum_{i=0}^{\nu_x} (\mathtt{R}_i^x - (\nu+1)/2)^2} \tag{8}$$

where ν_x denotes the number of elements in the set $\mathtt{R}^x$ and $\overline{\mathtt{R}}^x = (\sum_{i=0}^{\nu_x} \mathtt{R}_i^x)/\nu_x$. To perform a higher-order attack using the KW distinguisher, a combination function such as the mean-free product $\mathcal{C}$ is used to combine the leakages before ranking them.

2.7 Lattice Attacks

Prior work has demonstrated that recovering a subset of the NTT domain secret coefficients $\hat{s}_i$ through the SCA attack is sufficient to efficiently compute the entire normal domain secret polynomial s [KT23, QLZ+23] using lattice attacks. Intuitively, this is due to the fact that the attack in the NTT domain leads to an overdetermined system. While the search space for the NTT domain secret polynomial is q^n, it is significantly smaller in the normal domain due to the small coefficient distributions, as explained in Section 2.2. In particular, it is $(2 \cdot \eta + 1)^n$ for both Kyber an Dilithium, where coefficients are drawn from different distributions (see Section 2.2 for the corresponding values of η).

The idea is to encode the (inverse) NTT transformation as a lattice problem, such as SIS or LWE. In particular, [KT23] consider the inverse NTT transformation $s = \mathbf{M}\hat{s}$. Let $\mathbf{M}^k\hat{s}^k$ and $\mathbf{M}^u\hat{s}^u$ denote the known and unknown parts of this transformation, respectively:

$$s = \left[\mathbf{M}^k \mid \mathbf{M}^u\right]\left[\hat{s}^k \mid \hat{s}^u\right] = \mathbf{M}^k\hat{s}^k + \mathbf{M}^u\hat{s}^u \tag{9}$$

where $\mathbf{M}^k \in \mathbb{Z}_q^{n\times k}$, $\mathbf{M}^u \in \mathbb{Z}_q^{n\times u}$ and $n = k + u$. Given the known vector $\mathbf{v} = -\mathbf{M}^k \hat{s}^k \in \mathbb{Z}_q^n$ and s being short by definition, the following becomes an LWE instance,

$$\mathbf{v} = \mathbf{M}^u \hat{s}^u - s \tag{10}$$

The LWE problem is solved by lattice reduction algorithms, such as BKZ. The number of NTT domain coefficients that is required to compute s depends on the block size used in BKZ, which also increases the run-time of the algorithm For instance, recovering $\Upsilon = 38$ NTT domain coefficients out of 128 is needed for Kyber768 when BKZ-50 is used [KT23][1]. Here, we use Υ to denote the minimum number of correctly recovered NTT domain coefficients. Also note that BKZ-50 refers to applying the BKZ algorithm with a block size of 50. The method in [QLZ+23] differs slightly, as it leverages the forward NTT. For Dilithium3 with BKZ-50, their approach requires $\Upsilon = 79$ out of 256 secret coefficients. Readers may refer to the cited works for a detailed exploration of various trade-offs regarding the block size and run-time.

3 New Prediction Functions for HOCPA

3.1 Recursive Computation of OPF

In this section, we present an efficient approach for the computation of the optimal prediction function $f_{opt}^d(x)$ presented in Equation (5) based on recursion. Consider masking with d shares. In case Equation (5) does not have an explicit formula, as in the case of arithmetic masking, $f_{opt}^d(x)$ can be calculated using a computer in $O(q^d)$. Needless to say, this can be very expensive for higher order masking. For instance, with $d = 4$ and $\log q = 23$ (*e.g.* third-order masking for Dilithium), $q^d \approx 2^{92}$. We present a better solution by a recursive approach. For $d \geq 2$, we apply the following formula:

$$f_{opt}^d(x) = E\big[\mathcal{C}\big(f_{opt}^{\lceil d/2\rceil}(T), f_{opt}^{\lfloor d/2\rfloor}(X-T)\big)\big|X = x\big] \tag{11}$$

where random variables X and T take values in the discrete space $\mathtt{X}$, *e.g.* $\mathbb{Z}_q$, and T is uniformly distributed. As the base case of the recursion, $f_{opt}^1(x)$ is defined to be the predicted device leakage function, $f_{opt}^1(x) = \mathcal{L}'(x)$. This approach takes $O(\log dq^2)$ computational time with dynamic programming. The computation of the optimal prediction function becomes feasible for large d with our recursive approach. The procedure is as follows for the running example: First, $f_{opt}^2(x)$ is calculated in $O(q^2)$ time using $f_{opt}^1(x) = \mathcal{L}'(x)$. Then, $f_{opt}^4(x)$ is calculated in $O(q^2)$ time leveraging the previously calculated $f_{opt}^2(x)$. In total, $O(2q^2) = O(\log 4q^2)$ computation time is spent. Notice that $\log dq^2 \approx 2^{47}$ for this example.

Proof: We begin by reformulating the definition of the optimal prediction function $f_{\mathrm{opt}}^d(x)$ for the mean-free product:

$$f_{\mathrm{opt}}^d(x) = \mathsf{E}\left[\prod_{i=0}^{d-1} \mathsf{L}'(X^{\{i\}})\middle| X = x\right] + \Gamma \tag{12}$$

where Γ accumulates the constant terms arising from $\mathsf{E}[\mathsf{L}'(X^{\{i\}})]$. We omit this term, as it is independent of x and does not influence the correlation.

Next, we prove Equation (11) using induction. The base case $f_{\mathrm{opt}}^1(x)$ corresponds to a prediction function for an unprotected target, which is naturally $\mathsf{L}'(x)$ itself as in the

[1]For Kyber's incomplete NTT, the transform is applied independently to the even and odd degree coefficients. Therefore, we consider a total of 128 coefficients instead of 256, and the process described in this section is applied independently to the even and odd degree coefficients of s.

first-order CPA. For simplicity, consider even $d \geq 2$ and assume that $f_{\text{opt}}^{d/2}(x)$ is correct. Plugging in Equation (11):

$$f_{\text{opt}}^{d}(x) = \mathsf{E}\left[\mathsf{C}\left(\mathsf{E}\left[\prod_{i=0}^{d/2-1} \mathsf{L}'(X^{\{i\}})\middle| X_0 = T\right], \mathsf{E}\left[\prod_{i=d/2}^{d-1} \mathsf{L}'(X^{\{i\}})\middle| X_1 = X - T\right]\right)\middle| X = x\right] \tag{13}$$

where $X = \sum_{i=0}^{d-1} X^{\{i\}}$, $X_0 = \sum_{i=0}^{d/2-1} X^{\{i\}}$ and $X_1 = \sum_{i=d/2}^{d-1} X^{\{i\}}$. $X^{\{0:d-1\}}$ are uniformly random over $\mathtt{X}$. By re-writing the inner expectations as the sum of products,

$$\begin{aligned} f_{\text{opt}}^{d}(x) = \mathsf{E}\Bigg[\frac{1}{q^{d/2-1}}\Bigg(\sum_{x^{\{0\}}\in\mathtt{X}}\sum_{x^{\{1\}}\in\mathtt{X}} .. \sum_{x^{\{d/2-2\}}\in\mathtt{X}} \prod_{i=0}^{d/2-1} \mathsf{L}'(x^{\{i\}})\Bigg)\cdot \\ \frac{1}{q^{d/2-1}}\Bigg(\sum_{x^{\{d/2\}}\in\mathtt{X}}\sum_{x^{\{d/2+1\}}\in\mathtt{X}} .. \sum_{x^{\{d-2\}}\in\mathtt{X}} \prod_{i=d/2}^{d-1} \mathsf{L}'(x^{\{i\}})\Bigg)\Bigg| X = x\Bigg] \end{aligned} \tag{14}$$

where $x^{\{d/2-1\}} = T - \sum_{i=0}^{d/2-2} x^{\{i\}}$ and $x^{\{d-1\}} = X - T - \sum_{i=d/2}^{d-2} x^{\{i\}} = X - \sum_{i=0}^{d-2} x^{\{i\}}$. Combining the products, we have:

$$f_{\text{opt}}^{d}(x) = \frac{1}{q^{d-2}}\mathsf{E}\Bigg[\Bigg(\sum_{x^{\{0\}}\in\mathtt{X}}\sum_{x^{\{1\}}\in\mathtt{X}} .. \sum_{x^{\{d/2-2\}}\in\mathtt{X}}\sum_{x^{\{d/2\}}\in\mathtt{X}} .. \sum_{x^{\{d-2\}}\in\mathtt{X}} \prod_{i=0}^{d-1} \mathsf{L}'(x^{\{i\}})\Bigg)\Bigg| X = x\Bigg] \tag{15}$$

Taking the expectation over T completes the missing summation term over $x^{\{d-2\}}$:

$$f_{\text{opt}}^{d}(x) = \frac{1}{q^{d-1}} \sum_{x^{\{0\}}\in\mathtt{X}}\sum_{x^{\{1\}}\in\mathtt{X}} .. \sum_{x^{\{d-2\}}\in\mathtt{X}} \left(\left(\prod_{i=0}^{d-2} \mathsf{L}'(x^{\{i\}})\right)\cdot \mathsf{L}'\left(x - \sum_{i=0}^{d-2} x^{\{i\}}\right)\right) \tag{16}$$

which is equivalent to the definition of the optimal prediction function given in Equation (5). The equivalence follows from the reformulation given in Equation (12) by omitting the constant term Γ. We also note that the case of odd d can be handled similarly. ■

3.2 Optimal Correlation

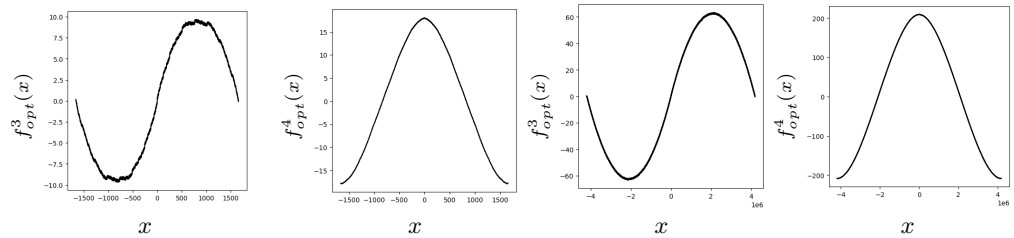
Using the presented recursive computation approach, we computed a set of optimal prediction functions f_{opt}^{d} for the SCA attack targeting NTT multiplication, as outlined in Section 2.4. Recall that the target function is modular multiplication, and the target intermediate X is masked as $X = \sum_{i=0}^{d-1} X^{\{i\}} \pmod q$. Specifically, for this scenario, we computed $f_{opt}^{d}(x)$ as a look-up table for all $x \in \mathbb{Z}_q$. We consider Hamming weight leakage function, denoted by $\mathcal{W}$. The computations of look-up table based $f_{opt}^{d}(x)$ were carried out for $2 \leq d \leq 5$, $\forall q \in \{3329, 8380417\}$ and under both central and non-central modular reduction. Let β denote the machine word size in bits, which is typically 16 or 32 in software implementations. We consider $\beta = 16$ for $q = 3329$ and $\beta = 32$ for $q = 8380417$, consistent with the configurations used in Kyber and Dilithium implementations [KRSS19, HKL+22, CGL+24, BC22]. Then, we computed the optimal correlation using these functions, demonstrated in Table 1. We studied both the noiseless case, where $\sigma = 0$, and scenarios with increasing noise levels, up to $\sigma = 10$.

Table 1: Estimations of $\hat{\rho}(\mathcal{C}(\{\mathcal{W}(X^{\{i\}})\}_{i=0}^{d-1}), f_{opt}^{d}(X))$ for distinct q, β, d, σ and modular reduction strategies (*i.e.* central or not). The estimations are performed with 1 million uniformly random samples for $X^{\{0:d-1\}}$ drawn from the discrete space $\mathtt{X}$. The rows are ordered based on correlation magnitudes. Estimated values are rounded to nearest.

		$q = 3329$, $\beta = 16$						$q = 8380417$, $\beta = 32$					
		σ						σ					
$\mathtt{X}$	d	0	2	4	6	8	10	0	2	4	6	8	10
$[-q/2, q/2]$	1	1.00	0.85	0.63	0.47	0.37	0.31	1.00	0.94	0.81	0.68	0.57	0.48
$[-q/2, q/2]$	2	0.41	0.30	0.16	0.09	0.06	0.04	0.47	0.42	0.31	0.22	0.15	0.11
$[-q/2, q/2]$	3	0.21	0.13	0.05	0.02	0.01	0.01	0.27	0.22	0.14	0.08	0.05	0.03
$[-q/2, q/2]$	4	0.11	0.06	0.02	0.01	0.00	0.00	0.16	0.12	0.07	0.03	0.02	0.01
$[-q/2, q/2]$	5	0.06	0.03	0.01	0.00	0.00	0.00	0.09	0.07	0.03	0.01	0.01	0.00
$[0, q)$	1	1.00	0.63	0.38	0.26	0.20	0.16	1.00	0.77	0.51	0.37	0.29	0.23
$[0, q)$	2	0.17	0.07	0.03	0.01	0.01	0.01	0.12	0.07	0.03	0.01	0.01	0.00
$[0, q)$	3	0.03	0.01	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00
$[0, q)$	4	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$[0, q)$	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

3.3 Explicit Formulas

When the reduction is central, *i.e.* the secret shares $X^{\{0:d-1\}} \in \mathtt{X}$ where $\mathtt{X} = [-q/2, q/2]$, the device leakage corresponds to the Hamming weight, and there is a sufficient margin between β and $\log q$, which is the case for $\{q = 3329, \beta = 16\}$ and $\{q = 8380417, \beta = 32\}$, it is possible to approximate f_{opt}^{d} with an explicit formula, denoted by f_{opt}^{*d}. [TMS24] shows that $f_{opt}^{*2}(x) = |x|$. In this section, we present explicit formulas for higher orders, $d > 2$, and show that these formulas are optimal. These formulas are derived by calculating f_{opt}^{d} as a look-up table using the recursive formula given by Equation (11), as illustrated in the previous section. Figure 1 visualizes the f_{opt}^{d} for $d = 3$ and $d = 4$, providing an intuition for finding closed formulas. By visually inspecting the plots in Figure 1, as well as additional plots for $d > 4$ (not shown), we conjectured closed-form expressions for f_{opt}^{*d}, which follow *sin* and *cos* patterns for odd and even d, respectively. Table 2 presents the closed-form formulas for the estimations f_{opt}^{*d}. A common technique to compute the accuracy of a prediction function is to compute its correlation to the optimal prediction function [PRB09]. Observe that presented formulas are highly accurate for the moduli employed by Kyber and Dilithium. Furthermore, the recurring structural pattern in these expressions facilitated the process of conjecturing the approximating formulas.



(a) $q = 3329$, $\beta = 16$, $d = 3$ (b) $q = 3329$, $\beta = 16$, $d = 4$ (c) $q = 8380417$, $\beta = 32$, $d = 3$ (d) $q = 8380417$, $\beta = 32$, $d = 4$

Figure 1: Visualization of optimal prediction function f_{opt}^{d} for the central reduction range $[-q/2, q/2]$ and distinct q, d, β.

Table 2: Estimations of $\hat{\rho}(f_{opt}^{*d}(X), f_{opt}^{d}(X))$ for distinct q, β, d when the modular reduction is central. The estimations are performed with 1 million uniformly random samples for X.

		q	β	q	β
d	$f_{opt}^{*d}(x)$	3329	16	8380417	32
3	$\sin(2\pi x/q)$	0.997		0.999	
4	$\cos(2\pi x/q)$	0.999		0.999	
5	$-\sin(2\pi x/q)$	0.999		0.999	
6	$-\cos(2\pi x/q)$	0.999		0.999	
7	$\sin(2\pi x/q)$	0.999		0.999	
8	$\cos(2\pi x/q)$	0.999		0.999	

4 A Novel Attack on Incomplete NTT

In this section, we explain a novel two-step attack strategy targeting incomplete NTT-based polynomial multiplication, such as Kyber's NTT or certain optimized implementations of Dilithium [AHKS22]. Recall from Section 2.4 that, when attacking the base multiplication in the incomplete NTT domain, the secret coefficients are predicted in pairs of the form $\{\hat{s}_{i,0}, \hat{s}_{i,1}\}$.

Our attack strategy is as follows. We consider each attacked pair as $\{\hat{s}_{i,0}, \hat{s}_{i,0}\delta_i\}$. Then, we efficiently retrieve each δ_i using generic SCA distinguishers. Once the full vector $\Delta = [\delta_i]_{i=0}^{n/2-1}$ is obtained, we perform a lattice attack to recover the vector $\hat{\mathbf{s}}^{\mathbf{0}} = [\hat{s}_{i,0}]_{i=0}^{n/2-1}$, effectively reconstructing the full secret. We would like to underline that since our attack employs generic distinguishers, the attacker does not need to predict a device leakage function.

4.1 SCA to find Deltas

In this section, we explain how to efficiently find out the relationship between secret coefficients within a pair, *i.e.* δ_i, as defined previously. We re-write the formula for the base multiplication for the higher-order term (see Algorithm 1) as:

$$\hat{z}_{i,1} = \hat{c}_{i,0} \cdot \hat{s}_{i,0} \cdot \delta_i + \hat{c}_{i,1} \cdot \hat{s}_{i,0} = \hat{s}_{i,0} \cdot (\hat{c}_{i,0} \cdot \delta_i + \hat{c}_{i,1}) \pmod{q}$$

As the modular multiplication with a prime modulus is an injective function, $\hat{s}_{i,0}$ can't be distinguished through side-channel attacks with generic distinguishers. In particular, for a fixed value of δ_i, the all the elements in set of hypotheses for $\hat{s}_{i,0}$, which correspond to $\mathbb{Z}_q^*$, get exactly the same score. The attacker can take the advantage of this feature to retrieve δ_i, by considering the following set of hypotheses $\{(1,1),(1,2),..,,(1,q-1)\}$ for $(\hat{s}_{i,0}, \delta_i)$. Note that δ_i can be distinguished for a fixed value of $\hat{s}_{i,0}$ since $\hat{c}_{i,0} \cdot \delta_i + \hat{c}_{i,1} \pmod{q}$ is non-injective. The prediction with the highest score is expected to be $(1, \delta_i)$.

We should underline that the explained attack must be carried out without using a prediction function (which is meaningful when a generic distinguisher is used). For instance, the attack will not work if the distinguisher is MIA and it is used with the Hamming weight prediction function.

Notice that such an approach decreases the number of hypotheses from q^2 to q for the studied incomplete NTT based polynomial multiplication, and without the need for traces with specially chosen ciphertexts.

4.2 Lattice Attack Using Deltas

In this section, we construct a lattice attack using the knowledge of Δ from the previous section to find out $\hat{\mathbf{s}}^{\mathbf{0}}$. We consider the inverse NTT transformation as a matrix multiplication following the approach in [KT23], defined as $\mathbf{s^0} = \mathbf{M}\hat{\mathbf{s}}^{\mathbf{0}}$ for the lower-degree coefficients. Similarly, $\mathbf{s^1} = \mathbf{M}\hat{\mathbf{s}}^{\mathbf{1}}$ where $\hat{\mathbf{s}}^{\mathbf{1}} = [\hat{s}_{i,1}]_{i=0}^{n'-1}$, defined in the same way with $\hat{\mathbf{s}}^{\mathbf{0}}$. Also note that the coefficient vector of s is given by $[\mathbf{s}^0_0, \mathbf{s}^1_0, \mathbf{s}^0_1, \mathbf{s}^1_1, .., \mathbf{s}^0_{n'-1}, \mathbf{s}^1_{n'-1}]$ and $n' = n/2$.

First, we integrate Δ into $\mathbf{M}$ as, $\mathbf{M}_\Delta = \mathbf{M}(\mathbf{I}_{n'}\Delta)$, where $\mathbf{I}_{n'}$ denotes the identity matrix of size n'. Consequently, we have $\mathbf{s}^1 = \mathbf{M}_\Delta\hat{\mathbf{s}}^{\mathbf{0}}$. Based on this modification, we construct the following basis:

$$\mathbf{B} = \begin{bmatrix} \mathbf{M}^T & \mathbf{M}_\Delta^T \\ q\mathbf{I}_{n'} & 0 \\ 0 & q\mathbf{I}_{n'} \end{bmatrix} \tag{17}$$

The lattice $\Lambda(B)$ contains all the linear combinations of the row vectors in $\mathbf{B}$, $\Lambda(B) = \left\{ \sum_{i=0}^{3n'-1} \zeta_i \mathbf{B}_i \mid \zeta_i \in \mathbb{Z} \right\}$. Then, $[\hat{\mathbf{s}}^{\mathbf{0}} \mid \kappa_0 \mid \kappa_1]$ generates the short vector $[\mathbf{s^0} \mid \mathbf{s}^1]$ in $\Lambda(B)$, for some κ_0, κ_1. Therefore, finding $\hat{\mathbf{s}}^{\mathbf{0}}$ is an SVP in $\Lambda(B)$, which can be efficiently solvable for $n = 256$, using the well-known lattice reduction algorithms LLL or BKZ.

The lattice reduction algorithm returns false positives along with the actual $[\mathbf{s^0} \mid \mathbf{s^1}]$. In particular, it returns n' possible solutions. We show that all of these solutions are short and comply with our Δ constraint. Consider the ring $\mathcal{R}_q^{n'} = \mathbb{Z}_q/(x^{n'}+1)$ and let $s^0(x), s^1(x) \in \mathcal{R}_q^{n'}$ denote the polynomials corresponding to the coefficient vectors $\mathbf{s}^0$, $\mathbf{s}^1$, respectively. Indeed these solutions are cyclic rotations of s^0 and s^1 over $\mathcal{R}_q^{n'}$. Rotation in $\mathcal{R}_q^{n'}$ refers to multiplication by the monomial x^α where α denotes the rotation amount. Multiplication by x^α preserves the shortness property of s^0 and s^1. Let $t^0 = s^0 \cdot x^\alpha$ and $t^1 = s^1 \cdot x^\alpha$. Then for $j \in \{0,1\}$, $t_i^j = s_{i-\alpha}^j$ if $i - \alpha \geq 0$ and $t_i^j = -s_{n'-i-\alpha}^j$ otherwise. We can consider the Δ constraint in the NTT domain for $\mathcal{R}_q^{n'}$ as $\hat{\mathbf{s}}^{\mathbf{1}} = \hat{\mathbf{s}}^{\mathbf{0}} \star \Delta$. Taking the inverse NTT of both sides, we get $s^1 = s^0 \cdot \delta$, where $\delta \in \mathcal{R}_q^{n'}$ denote the polynomial corresponding to the inverse NTT of Δ, with coefficients $\delta_i = \mathbf{d}_i$ for $0 \leq i < n'$, and $\mathbf{d} = \mathbf{M}\Delta$. When both sides are multiplied by x^α, we get $x^\alpha \cdot s^1 = x^\alpha \cdot s^0 \cdot \delta$. Therefore, $t^1 = t^0 \cdot \delta$, which shows that t^0 and t^1 also satisfy the Δ constraint thus forming valid yet false positive solutions.

Since there are only valid n' solutions as explained above, the attacker can brute-force to distinguish the actual one. Recall that there exists k secret polynomials in the secret key of Kyber, where k is a parameter based on the target security level, *e.g.* $k = 4$ for Kyber1024. The attacker can efficiently test n' possibilities for each secret polynomial using the public key equation, leading to $(n/2)^k$ possibilities. That is to brute-force 2^{28} possibilities for Kyber1024, and even less for instances of Kyber with lower security levels.

In practice, one can use θ out of n' columns of both $\mathbf{M}^T$ and $\mathbf{M}_\Delta^T$. For instance, when first θ columns are used, $\mathbf{M}^T_{:,0:\theta}$ and $\mathbf{M}^T_{\Delta:,0:\theta}$, the lattice reduction returns vectors of size 2θ instead of $2n'$. Note that these vectors are formed by corresponding elements of the form $[\mathbf{s}^0_{0:\theta} \mid \mathbf{s}^1_{0:\theta}]$. While using a smaller θ reduces the complexity of the lattice reduction, aggressively reducing it causes the results to be incorrect, *i.e.* not slices of $[\mathbf{s}^0 \mid \mathbf{s}^1]$. Also note that since the last $n' - \theta$ elements of both $\mathbf{s}^0$ and $\mathbf{s}^1$ are not retrieved, a second attack is needed. A trivial strategy for this second phase is to target $[\mathbf{s}^0_{n'-\theta:n'} \mid \mathbf{s}^1_{n'-\theta:n'}]$ using $\mathbf{M}^T_{:,n'-\theta:n'}$ and $\mathbf{M}^T_{:,n'-\theta:n'}$.

5 Results

In this section, we present experimental results for the side-channel attacks introduced in Section 3 and Section 4. To support reproducibility, the source code for trace acquisition, attack scripts, and the simulations presented in the previous section are made publicly available in https://github.com/toluntosun21/HighOrderNonProfiledSCALBC.

Attack Setup. For the analysis and attack, we utilized a modified version of the open-source side-channel analysis library, scared[2]. In particular we ported scared to GPU using cupy[3], named scaredcu, and made it publicly accessible at https://github.com/toluntosun21/scaredcu. We run the experiments on a supercomputer with NVIDIA RTX4090 GPU. For the application of lattice attacks, we use the fpylll library[4].

5.1 Attacks on Hamming Weight Leakage

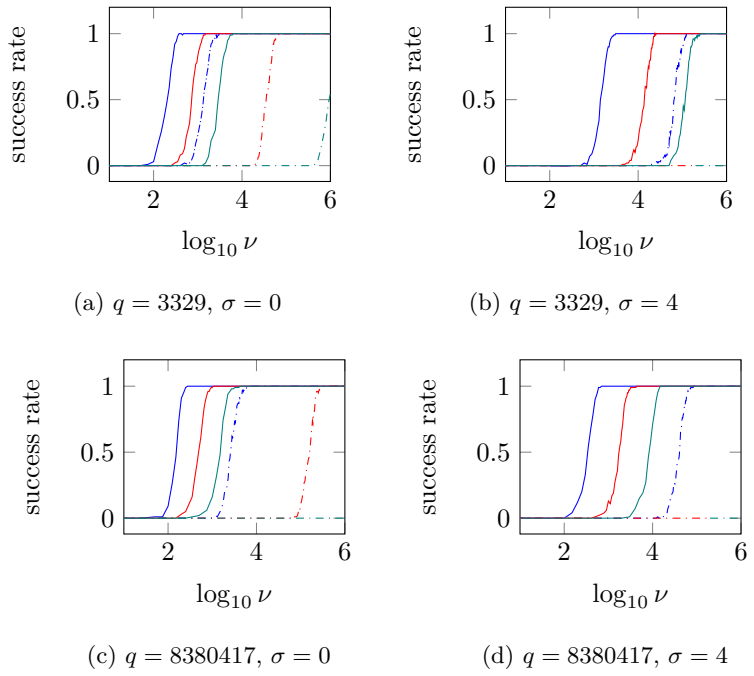
To attack the scenario where the device leakage function is known to be the Hamming weight, we perform the side-channel attacks presented in Section 3. We begin with simulated traces, where power samples are intentionally crafted to reflect Hamming weight leakage. Subsequently, we conduct real-device experiments using publicly available implementations of Kyber and Dilithium in an embedded setting.

(HO)CPA Details. We perform (HO)CPA attacks for $d \in \{1, 2, 3, 4\}$. For attacking the base multiplication with central reduction, we employ the prediction functions with explicit formulas discussed earlier. Specifically, for $d = 2$, we use the absolute value prediction function from [TMS24]. For $d = 3$ and $d = 4$, we apply the *sin* and *cos* prediction functions introduced in Section 3.3. In the case of the non-central reduction, we follow the methodology described in Section 3.1 to compute the prediction functions computationally and recursively. For $d = 1$, we employ the Hamming weight as the prediction function, which is a common practice in the literature, especially for attacking software implementations. For all masked scenarios, we employed the mean-free product combination $\mathcal{C}$. The target intermediate is chosen as the higher-degree coefficient $\hat{z}_{i,1}$ from the output of base multiplication for attacking Kyber's incomplete NTT for $d > 1$. For $d = 1$, we employ $\hat{z}_{i,0}|\hat{z}_{i,1}$. For both Kyber and Dilithium, and when the reduction range is central, we reduce the hypothesis space by half by leveraging the observation that additive inverse of the actual secret also produce peaks in the HOCPA results, as demonstrated in [CKA+21, TS24]. This optimization has a negligible impact on attack accuracy. Consequently, we tested $q/2$ hypotheses for Dilithium and $q^2/2$ hypotheses for Kyber due to incomplete NTT arithmetic.

5.1.1 HOCPA Attacks through Simulations

Generating Simulated Traces. The simulated traces for the base multiplication operation were generated through the following routine. m, ν, q, d, the reduction output range (central or not), the device leakage function $\mathcal{L}$, the noise standard deviation σ are pre-defined parameters. Either $q = 3329$ (Kyber) or $q = 8380417$ (Dilithium). We employed the incomplete NTT arithmetic for $q = 3329$ in the simulations. m denotes the number of subkeys for experiment, *e.g.* number of elements in the vectors. We utilized $m = 100$ in all our experiments. Note that we directly sample vectors of integers and process vectors element-wise, as in the base multiplication NTT domain, but we use a slightly different notation since we do not sample a polynomial and compute its NTT.

[2]https://pypi.org/project/scared/
[3]https://pypi.org/project/cupy/
[4]https://pypi.org/project/fpylll/



(a) $q = 3329$, $\sigma = 0$ (b) $q = 3329$, $\sigma = 4$

(c) $q = 8380417$, $\sigma = 0$ (d) $q = 8380417$, $\sigma = 4$

Figure 2: Success rates of HOCPA attacks on simulated traces with Hamming weight leakage. For each point in the curves, 100 experiments have been performed with random data. Reduction ranges: $[-q/2, q/2]$ (solid), $[0, q)$ (dashdotted). Number of shares: $d = 2$ (blue), $d = 3$ (red), $d = 4$ (green).

1. Sample a random secret vector $\mathbf{s} \leftarrow \mathtt{S}$ uniformly at random, where $\mathtt{S} = \mathbb{Z}_q^m$ for $q = 8380417$ ($\mathtt{S} = \mathbb{Z}_q^{m \times 2}$ for $q = 3329$).
2. Sample a random public vector $\mathbf{c} \leftarrow \mathtt{S}$.
3. Sample a random secret vector of shares $\mathbf{s}^{\{j\}} \leftarrow \mathtt{S}$ for $0 \leq j < d - 1$. Then, set $\mathbf{s}^{\{d-1\}} = \mathbf{s} - \sum_{j=0}^{d-2} \mathbf{s}^{\{j\}} \mod q$.
4. Compute the intermediate value matrix $\mathbf{g} \in \mathbb{Z}_q^{d \times m}$:
   (a) For $q = 8380418$, $\mathbf{g}_i^{\{j\}} = \mathbf{s}_i^{\{j\}} \cdot \mathbf{c}_i \pmod q$ for $0 \leq i < m$ and $0 \leq j < d$.
   (b) For $q = 3329$, $\mathbf{g}_i^{\{j\}} = \mathbf{s}_{i,0}^{\{j\}} \cdot \mathbf{c}_{i,1} + \mathbf{s}_{i,1}^{\{j\}} \cdot \mathbf{c}_{i,0} \pmod q$, performing Line 3 of Algorithm 1, for $0 \leq i < m$ and $0 \leq j < d$.
5. Sample the noise vector $\mathbf{e} \in \mathbb{R}^{dm} \mid \mathbf{e}_i \leftarrow \mathcal{N}(\sigma, 0)$ and compute the leakage vector $\mathbf{l}_i = \mathcal{L}(\mathbf{g}_i^{\{j\}}) + \mathbf{e}_{i+jm}$ for $0 \leq i < m$ and $0 \leq j < d$ where the leakage function is Hamming weight $\mathcal{L}(x) = \mathcal{W}(x)$.
6. Record $\mathbf{s}$, $\mathbf{c}$ and $\mathbf{l}$. Return to Step 2 if generation of ν traces is not yet complete.

Evaluation. We performed a series of HOCPA attacks for $d \in \{2, 3, 4\}$ for both the central and non-central reduction range. Figure 2 demonstrates the results of the HOCPA simulations, in terms of the success rate with respect to ν. The success rate is defined as the number of correctly predicted elements of the secret vector, $\pm\mathbf{s}_i$, divided by m. The results confirm the findings of [TMS24] and extend them to higher orders, *i.e.* $d > 2$, the central reduction is significantly easier to attack. Our findings further suggest that the hardness gap between central and non-central reduction strategies widens at higher orders. Observe that attacking Dilithium's modulus for $d = 4$ under central reduction requires fewer traces than the case for $d = 2$ under the unsigned reduction range. A similar trend is observed for $q = 3329$.

5.1.2 HOCPA Attacks on Real Traces

Target Implementations. We targeted the Kyber implementation of [BC22][5], which is specific to ARM Cortex-M4 and supports masking at arbitrary orders. This implementation inherits the polynomial arithmetic routines from the pqm4 library [KRSS19], which offers the state-of-the-art implementations of PQC algorithms on the Cortex-M4. Those parts are highly optimized and mostly written in assembly. For example, the degree-1 polynomial multiplications in Lines 2-3 of Algorithm 1 are implemented using the `smuad(x)`[6] instruction. The function that implements Algorithm 1 is named `basemul_asm`, which is executed for each share. The modular reduction is performed using Montgomery reduction, which takes 2 clock cycles and outputs in the central range $(-q, q)$. For Dilithium, we targeted the open-source implementation of [CGL+24][7]. The polynomial arithmetic in this implementation is written in C language and compiled for the Cortex-M4, in contrast to the target Kyber implementation. The function that performs the base multiplication is named `poly_pointwise_montgomery`. As the name suggests, it employs Montgomery reductions similar to the to target Kyber implementation. However, these are more costly due to Dilithium's parameters requiring 32-bit arithmetic. For both Kyber and Dilithium, we focus on the medium security levels, *i.e.* Kyber768 and Dilithium3. It is important to note that our methods are applicable to all security levels. Additionally, our analysis focuses on attacking a single secret polynomial s, although the secret key in both Kyber and Dilithium contains a vector of polynomials. The methodology we present can be applied iteratively to target each polynomial in the vector.

Trace Collection. We utilized the NewAE ChipWhisperer CW308 UFO board for power trace acquisition. The victim implementations were executed on an STM32F303 microcontroller, which features an ARM Cortex-M4 core. A trigger signal was provided to the power-collection module to mark the beginning of the base multiplication function corresponding to the first share for both Kyber and Dilithium. As a result, only power traces associated with the base multiplication are recorded. The core of the victim device operates at a frequency of 7.3 MHz, power samples are sampled at a rate of four samples at each clock cycle[8]. It is important to note that the power-collection module shares a reference clock with the victim device, ensuring that the traces are well aligned. We note that the Hamming Weight leakage assumption is also adopted in previous research targeting the STM32F303 on the CW308 UFO board [MWK+24, TMS24, QLZ+23].

Point-of-Interest (PoI) Detection. Figure 3 demonstrates the mean traces collected from the victim implementations for both Kyber and Dilithium. Observe that the iterations of the base multiplications are clearly distinguishable as the same pattern repeats consistently throughout the loop. Each iteration corresponds to 140 samples for Kyber and 96 samples for Dilithium. Note that the target implementation employs loop-unrolling; thus 140 samples actually cover two iterations of Algorithm 1 resulting in 64 total iterations instead of the expected 128. The pattern remains identical across all shares, thereby making the combination of leakage from different shares straightforward. As in [TMS24], we used a constant offset to combine the leakage from different shares. The offset is equal to the total number of power samples that the base multiplication takes, along with additional samples due to loop overhead. We determined this offset through pattern matching. In particular,

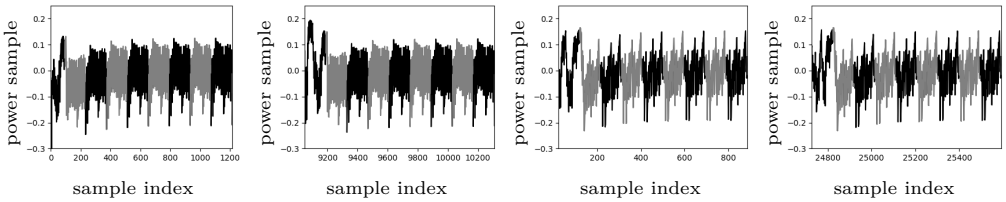
[5] https://github.com/uclcrypto/pqm4_masked/ commit hash: **5fe90ba**

[6] https://developer.arm.com/documentation/ddi0403/d/Application-Level-Architecture/Instruction-Details/Alphabetical-list-of-ARMv7-M-Thumb-instructions/SMUAD–SMUADX

[7] https://github.com/fragerar/tches24_masked_Dilithium/tree/master commit hash: **5b5fd32**

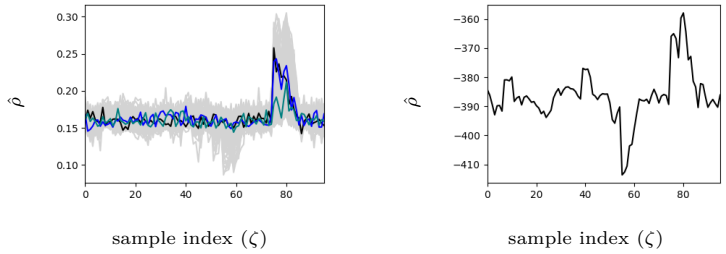
[8] Exceptionally, we decreased the sampling rate to one sample per clock cycle for Dilithium and $d = 4$. This limitation arises from the insufficient buffer size of the employed oscilloscope CW1200, which could not accommodate all samples from the beginning of the base multiplication for the first share to the end of the last share when capturing four samples per clock cycle. To maintain consistency in the presented results, we implicitly multiply the sample indices by four in the affected case.

to attack the secret coefficient $\hat{s}_i$ in Dilithium, we focused on the set of power samples with indexes $\{150 + 96i + 24420j + \zeta\}_{0 \leq j < d}$ as input to the combination function $\mathcal{C}$ for $0 \leq \zeta < 96$. To further refine the focus of the attack to a specific value of ζ, we performed the HOCPA attack for all possible values of ζ and identified the one that maximized the product of the highest correlation scores across all hypotheses and all i. Formally, we computed $\zeta' = \text{argmax}_\zeta \Pi_{i=0}^{n-1} \Psi_\zeta^i$ where $\Psi_\zeta^i = \max_\kappa(\text{Score}_\zeta(\mathtt{H}_\kappa^i))$, $\mathtt{H}^i$ denotes the set of hypotheses for $\hat{s}_i$ and Score_ζ outputs the correlation score for the given hypothesis and ζ. Recall that each $\hat{s}_i$ is attacked independently. ζ' identifies the point of interest (PoI), most strongly associated power sample with the targeted operation, namely $\hat{s}_i \cdot \hat{c}_i$. Figure 4 illustrates the process. The same process is applied to Kyber by adjusting the power sample offsets corresponding to each iteration and the offset between shares. Notice that we leverage the fact that ζ' must remain the same across attacks on each $\hat{s}_i$. The PoI detection mechanism explained in this section does not require profiling.



(a) Kyber, $\hat{s}^{\{0\}} \star \hat{c}$ (b) Kyber, $\hat{s}^{\{1\}} \star \hat{c}$ (c) Dilithium, $\hat{s}^{\{0\}} \star \hat{c}$ (d) Dilithium, $\hat{s}^{\{1\}} \star \hat{c}$

Figure 3: The average power trace corresponding to the execution of the attacked base multiplication functions for both Kyber and Dilithium. The plots demonstrates the first eight iterations of main loops of the base multiplications, focusing on the first two shares (note that sub-figures b and d are not relevant for implementations with $d = 1$). The observed pattern repeats consistently across all iterations and shares.

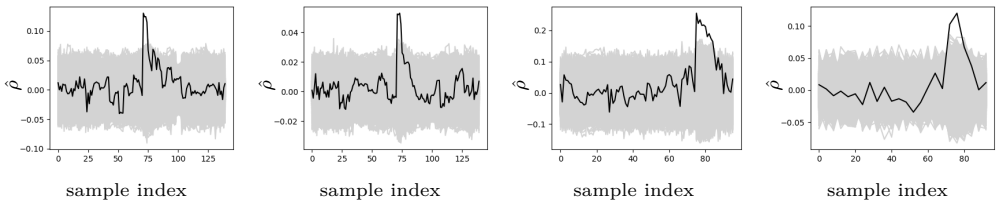


(a) Ψ_ζ^0, Ψ_ζ^1, Ψ_ζ^2 in black, blue and green, Ψ_ζ^i for $3 \leq i < n = 256$ in gray

(b) $\sum_{i=0}^{n-1} \log \Psi_\zeta^i$

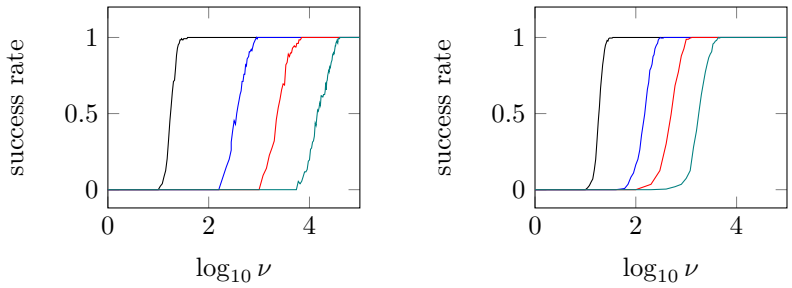
Figure 4: Illustration of PoI identification for Dilithium for $d = 3$ and $\nu = 1000$.

Evaluation of (HO)CPA Attacks. Figure 5 presents the HOCPA results against $\hat{s}_0$ as an example for $d = 3$ and $d = 4$. Observe that the actual secret is clearly distinguishable for each case. The correlation amounts comply with our estimations presented in Table 1. For Dilithium, the peaks are observed for significantly lower number of traces compared to Kyber. Figure 6 presents the success rates of attacks with respect to the available number of traces, measured as the ratio of successfully retrieved NTT domain secret coefficients in our experiments. For Kyber, the (HO)CPAs achieve full success with $\nu = 37$, $\nu = 900$, $\nu = 7000$, $\nu = 40000$ traces for $d = 1$, $d = 2$, $d = 3$ and $d = 4$, respectively. In the case of Dilithium, full success is observed with $\nu = 35$, $\nu = 350$, $\nu = 1300$, $\nu = 4400$ for the



(a) Kyber, $d = 3$, $\nu = 4000$ (b) Kyber, $d = 4$, $\nu = 24000$ (c) Dilithium, $d = 3$, $\nu = 1000$ (d) Dilithium, $d = 4$, $\nu = 4000$

Figure 5: HOCPA against the victim Kyber and Dilithium implementations targeting $\hat{s}_0$ for the given d and ν. The correlation scores corresponding to incorrect hypotheses are shown in gray, score for the correct hypothesis is shown in black.



(a) Kyber (b) Dilithium

Figure 6: Success rates of (HO)CPA attacks against Kyber and Dilithium implementations from [BC22] and [CGL+24], respectively. x-axis denotes $\log_{10} \nu$. The success rate refers to $(\#\text{ correctly predicted } \pm \hat{s}_i / n')$, where n' is 128 for Kyber and 256 for Dilithium. Attacks are performed against different number of shares in the target implementation: $d = 1$ (black), $d = 2$ (blue), $d = 3$ (red), $d = 4$ (teal).

corresponding masking orders. The reported success rates are based on (HO)CPA attacks conducted at the PoI identified using the methodology described in the previous paragraph.

Post-Processing with Lattice Attack. Recall from Section 2.7 that recovering $\Upsilon = 38$ NTT domain coefficients out of 128 is needed for Kyber768 by using the approach of [KT23]. We also adapted the methodology of [KT23] to Dilithium3 and experimentally found out (by performing 100 experiments with uniformly random data) that $\Upsilon = 130$ coefficients out of 256 are sufficient when BKZ-20 is used. The runtime of BKZ in the mentioned scenarios and our test equipment is roughly 30 seconds and 19 seconds for Kyber768 and Dilithium3, respectively. Our approach was to employ a block-size for which the overall post-processing using BKZ finishes in practical time based on the results in the literature and in-depth optimization of this step is out of the scope of our work. A critical aspect in applying the lattice attack is determining which NTT domain coefficients have been correctly predicted, as only these should be included in the process. That is to decide a subset of $\{0, 1, .., n' - 1\}$ of length Υ. If any of the input coefficients is incorrect, the lattice attack fails (the output is larger than η in absolute value) and it must be executed with another subset of coefficients. For this purpose, we make use of the correlation scores of the attacks outputs for each $\hat{s}_i$. Formally, let $\{\hat{s}_0^*, \hat{s}_1^*, .., \hat{s}_{n'-1}^*\}$ denotes the predicted set of NTT domain secret coefficients by the (HO)CPA attacks. Let $\Omega = \{\Omega_0, \Omega_1, .., \Omega_{n'-1}\}$ be the corresponding set of confidence scores, where each $\Omega_i = \text{Score}^i(\hat{s}_i^*)$ reflects the score assigned to $\hat{s}_i^*$ during the attack on $\hat{s}_i$. We then sort Ω in descending order to obtain Ω^*. Starting from the subset of size Υ with the highest-scoring predictions according to

Ω^*, we incrementally consider less likely subsets (as ranked by Ω) until the lattice attack succeeds. Table 3 presents the overall results of the executed (HO)CPA combined with the lattice attack against Kyber and Dilithium. Notice that only **2400** traces are needed in our experiments to break Dilithium with $d = 4$.[9]

Comparison to Literature. While no work performs non-profiled SCA attacks for $d > 2$ against the NTT multiplication of Kyber and Dilithium, the attack in [DNGW23] can be considered an alternative to ours for Kyber, since profiling is done directly on the victim device. Although we acknowledge that the authors do not aim to aggressively minimize the number of traces, they report a total of $\nu = 40K$ traces to break masked Kyber for $d = 3$ and $d = 4$, which is significantly higher than our trace requirements. Moreover, unlike [DNGW23], our approach does not require sending chosen ciphertexts to the victim device. To the best of our knowledge, there is also no work in the literature, beyond the context of LBC, that discusses prediction functions against arithmetic masking for $d > 2$.

Table 3: Number of traces ν and the amount of time required for the lattice attack successfully retrieve s in our experiments. Results are presented for distinct values of d in the target implementations of both Kyber and Dilithium. Note that ν affects the number of known NTT domain coefficients, denoted by $\#\hat{s}_i$, obtained through the SCA attack, which serve as inputs to the lattice attack. We report the minimum ν that allows the subsequent lattice attack finishes in practical time, *i.e.* under a day. #LA denotes the number of executions of the lattice attack (with different subset of NTT domain coefficients) until it succeeds. ν_{CPA} denotes the number of traces the (HO)CPA attack achieved 1.0 success rate in our experiments.

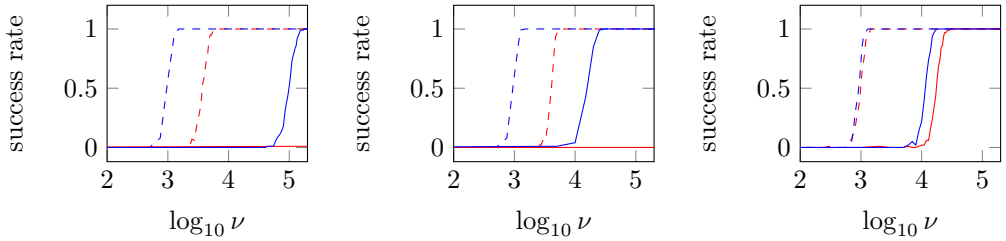
Algorithm	d	ν_{CPA}	ν	$\#\hat{s}_i$	#LA	**Time (m)**
Kyber768	1	37	21	92	4	2
Kyber768	2	900	380	65	700	350
Kyber768	3	7000	2200	59	64	32
Kyber768	4	40000	14500	67	70	35
Dilithium3	1	35	25	243	1	0.31
Dilithium3	2	350	190	201	5	1.58
Dilithium3	3	1300	700	198	1	0.31
Dilithium3	4	4400	2400	198	782	247.6

5.2 Attacks on Unknown Device Leakage

Next, we evaluate the scenario where the device leakage function is unknown to the attacker, executing the attack presented in Section 4. Since -to the best of our knowledge- there is no publicly available implementation where the device leakage is not a function of Hamming weight, we study this case only through simulations.

Simulating Unknown Device Leakage. The simulated traces are generated following the routine presented in Section 5.1. The only exception is that, instead of sticking with

[9] In the case of Dilithium, for attack orders $d = 2$ and $d = 4$, it was not possible to distinguish the secret from its additive inverse because the prediction functions used are symmetric (see Figure 1d). Therefore, we assume that the attacker obtains this sign information from another point in the SCA traces. For Kyber, we did not need to make this assumption.



(a) Randomly weighted bits (b) Strongly non-linear leakage (c) Binary leakage

Figure 7: Success rates of first and second order SCA attacks on simulated traces generated with different leakage functions, $q = 3329$, SNR $= 1$. The attack target is to recover δ_i in incomplete NTT domain. For each point in the curves, 100 experiments have been performed with random data. Distinguishers: KW (blue), MIA (red). Number of shares: $d = 1$ (dashed), $d = 2$ (solid).

$\mathcal{L}(x) = \mathcal{W}(x)$, we also consider the following device leakage functions $\mathcal{L}$ in our experiments from [YOR23]:

1. Randomly weighted bits: $\mathcal{L}(x) = \sum_{i=0}^{\beta-1} w_i x_{[i]}$ with $w_i \leftarrow [-1, 1]$.
2. Strongly non-linear: $\mathcal{L}(x) = \mathcal{S}(x_{[3:0]})$, with $\mathcal{S}$ defined to be the Present S-box.
3. Binary: $\mathcal{L}(x) = \mathcal{S}(x_{[3:0]}) \mod 2$, with $\mathcal{S}$ defined to be the Present S-box.

where $x_{[i]}$ denotes the i-th bit of x and the number of bits in x is denoted by t. $x_{[3:0]}$ denotes the least-significant four bits of x.

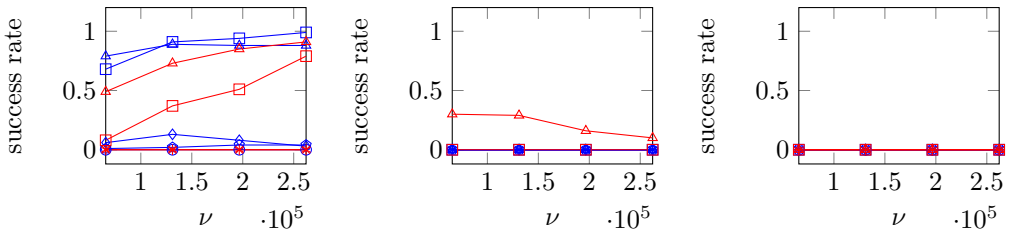
Evaluation of SCA Attack on Deltas. First, we executed the SCA attack on deltas δ_i as explained in Section 4.1. As this approach specifically targets incomplete NTT, we evaluate it only for $q = 3329$. We only considered the classical unsigned modular reduction case but the methodology applies to the central reduction case as well. We selected the σ in the simulated traces based on the desired the signal-to-noise (SNR) ratio for each $\mathcal{L}$. As the described methodology requires using a generic distinguisher, we employ both KW and MIA in our experiments. Figure 7 demonstrates the results for SCA attack on deltas, revealing the effectiveness of our approach in various leakage characteristics. Although we performed the attacks for $d \in \{1, 2\}$, the method generalizes to higher orders given a sufficient number of traces based on our results. KW outperformed MIA in terms of the number of traces required for a successful attack in our experiments. This finding is consistent with the results reported in the KW paper [YOR23], to which we refer the reader for a detailed comparison between KW and MIA.

Application of Lattice Attack using Deltas. Next, we evaluate the lattice reduction approach proposed in Section 4 which aims to recover secret coefficients $\hat{s}_{i,0}$ using deltas δ_i obtained from the aforementioned SCA attacks. To validate the approach, we performed 100 experiments with randomly generated s with short coefficients as explained in Section 2.2, considering $\eta = 2$ *e.g.*, Kyber768. For the lattice reduction, we used the BKZ-50 algorithm, which successfully solved the resulting SVP instances for all instances. Through experimentation, we used $\theta = 96$. Our aim here is not to optimize the attack parameters, but rather to demonstrate the feasibility of recovering $\hat{s}_{i,0}$ via lattice reduction. In our experimental setup, the lattice reduction process took an average of 884 seconds to complete, with a minimum of 264 seconds, a maximum of 4941 seconds, and a standard deviation of 701 seconds.

Table 4: Comparison of attack runtime (in seconds) between the proposed approach and the baseline for attacking the scenario in which the the device leakage function is unknown to the attacker. Results are demonstrated for $d = 2$. Two bins are used to estimate probabilities in the histogram model for MIA.

Distinguisher	KW			MIA		
# Dropped bits	4	6	8	4	6	8
Baseline	23490	12900	860	33060	25740	10010
Proposed		10			88	
Speed-up	2349×	1290×	86×	429×	292×	113×

Comparison with Existing Approach. Recall that our approach reduces the number of hypotheses from q^2 to q compared to the baseline approach, which predicts the pair $\{\hat{s}_{i,0}, \hat{s}_{i,1}\}$ simultaneously (as explained in Section 2.2), and uses bit-dropping. We compare our method to this baseline in terms of attack run-time, demonstrated in Table 4 for the distinguishers KW and MIA. For both distinguishers, the speed-up depends on the number of bits dropped. For instance, when half of the bits are dropped, our approach achieves speed-ups of 1290× and 292× speed-up, for KW and MIA, respectively. Determining which bits to drop remains an open question, and typically several configurations are tested. Due to the long run-times required by the baseline method, we omit an in-depth comparison of success rates. However, Figure 8 provides evidence that (1) the bit-dropping scheme significantly affects the success rate, and (2) in most bit-dropping configurations and under various device leakage assumptions, the baseline exhibits very low and unstable success rates. In a few specific cases, the baseline achieves promising success rates, but even in those cases, our proposed method performs comparably. In fact, the only scenario where the baseline's success rate approaches 1.0 (though never fully reaching it) is under the device leakage model with randomly weighted bits. In particular, the accuracy of the baseline with KW and LSB6 is 0.94 for $\nu = 196608$ while it is 1.0 for proposed approach. This analysis is performed as if the NTT is complete to reduce the run-time, which we believe does not impact the conclusion we get from these results.



(a) Randomly weighted bits (b) Strongly non-linear leakage (c) Binary leakage

Figure 8 Success rates of second-order SCA attacks with the baseline approach on simulated traces generated with different device leakage functions, $q = 3329$, SNR$= 1$, $d = 2$. For each point in the curves, 100 experiments have been performed with random data. Distinguishers: KW (blue), MIA (red). Bit-dropping strategies: LSB4 (Least significant 4 bits) (triangle), LSB6 (square), LSB8 (diamond), MSB4 (asterisk), MSB6 (x), MSB8 (+).

6 Conclusion

In this study, we explored non-profiled SCA attacks against the NTT based polynomial multiplication, which is the core operation in LBC, including the PQC standards Kyber

and Dilithium. We considered both the Hamming weight leakage and unknown device leakage scenarios.

For the Hamming weight leakage, we enabled CPA attacks against second- and higher-order masked implementations of those algorithms, by revisiting the formulation of the optimal prediction function. For certain cases, namely when signed modular arithmetic is employed in the victim device, we provided explicit formulas involving *sin* and *cos* functions. These prediction functions are particularly effective for attacking second and higher-order masked implementations. Otherwise, when unsigned modular arithmetic is used, the optimal prediction function can be efficiently computed via our recursive formula. We performed HOCPA attacks using the proposed prediction functions against open-source masked implementations of Kyber768 and Dilithium3. We recovered the full secret polynomial s with 2200 and 14500 traces against Kyber for second and third order masking, and 700 and 2400 against Dilithium, respectively.

For attacking the unknown device leakage scenario, we presented a novel two-step attack that combines generic distinguishers with lattice reduction algorithms. Our approach is tailored specifically for incomplete NTT as in Kyber. In particular, we showed that recovering deltas δ_i between odd and even degree secret coefficient pairs in incomplete NTT domain is sufficient for full secret polynomial recovery. Our approach is more efficient than existing techniques as it reduces the brute-force space from q^2 to q and avoids the bit-dropping trick that leads to information loss and poor success rate. We observed a speed-up in attack run-time ranging from 86× to 23490× in our experiments, while the success rate of proposed method is significantly better in most cases and comparable in some cases. Performing higher-order attacks to Dilithium under the unknown leakage model remains future work.

Acknowledgments

The work described in this paper has been supported in part by European Union through the Twinning Project 101079319 (acronym enCRYPTON). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

The authors acknowledge Yan Yan for sharing the implementation of KW from the work presented in [YOR23].

References

[AHKS22] Amin Abdulrahman, Vincent Hwang, Matthias J Kannwischer, and Amber Sprenkels. Faster kyber and dilithium on the cortex-m4. In *International Conference on Applied Cryptography and Network Security*, pages 853–871. Springer, 2022. doi:10.1007/978-3-031-09234-3_42.

[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. doi:10.46586/tches.v2024.i2.359-383.

[BC22] Olivier Bronchain and Gaëtan Cassiers. Bitslicing arithmetic/boolean masking conversions for fun and profit: with application to lattice-based kems. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 553–588, 2022. doi:10.46586/tches.v2022.i4.553-588.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6*, pages 16–29. Springer, 2004. doi:10.1007/978-3-540-28632-5_2.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018. doi:10.1109/EuroSP.2018.00032.

[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. doi:10.1007/978-3-031-41181-6_9.

[CGL+24] Jean-Sébastien Coron, François Gérard, Tancrède Lepoint, Matthias Trannoy, and Rina Zeitoun. Improved high-order masked generation of masking vector and rejection sampling in dilithium. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(4):335–354, Sep. 2024. doi:10.46586/tches.v2024.i4.335-354.

[CJRR99] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 398–412. Springer, 1999. doi:10.1007/3-540-48405-1_26.

[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. doi:10.1109/ICCD53106.2021.00094.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018. doi:10.13154/tches.v2018.i1.238-268.

[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. doi:10.1145/3591866.3593072.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis: A generic side-channel distinguisher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 426–442. Springer, 2008. doi:10.1007/978-3-540-85053-3_27.

[GMPO19] Si Gao, Ben Marshall, Dan Page, and Elisabeth Oswald. Share-slicing: Friend or foe? *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):152–174, Nov. 2019. doi:10.13154/tches.v2020.i1.152-174.

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

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough: Deriving optimal distinguishers from communication theory. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, pages 55–74. Springer, 2014. doi:10.1007/978-3-662-44709-3_4.

[KAA21] 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. In *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 35–45. IEEE, 2021. doi:10.1109/HOST49136.2021.9702284.

[KRSS19] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4, 2019. URL: https://eprint.iacr.org/2019/844.

[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. doi:10.1007/978-981-97-1235-9_11.

[LBS19] Itamar Levi, Davide Bellizia, and François-Xavier Standaert. Reducing a masked implementation's effective security order with setup manipulations: And an explanation based on externally-amplified couplings. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):293–317, Feb. 2019. doi:10.13154/tches.v2019.i2.293-317.

[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. doi:10.1145/3569420.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 513–533. Springer, 2017. doi:10.1007/978-3-319-66787-4_25.

[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. doi:10.1109/TC.2009.15.

[QLZ+23] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Mingyao Shao, and Shuo Sun. When ntt meets sis: Efficient side-channel attacks on dilithium and kyber. *Cryptology ePrint Archive*, 2023. URL: https://eprint.iacr.org/2023/1866.

[RBVB23] Rafael Carrera Rodriguez, Florent Bruguier, Emanuele Valea, and Pascal Benoit. Correlation electromagnetic analysis on an fpga implementation of crystals-kyber. In *2023 18th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*, pages 217–220. IEEE, 2023. doi:10.1109/PRIME58259.2023.10161764.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. doi:10.1145/1568318.1568324.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994. doi:10.1109/SFCS.1994.365700.

[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. doi:10.1007/978-3-031-40003-2_25.

[TMS24] Tolun Tosun, Amir Moradi, and Erkay Savas. Exploiting the central reduction in lattice-based cryptography. *IEEE Access*, 2024. doi:10.1109/ACCESS.2024.3494593.

[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. doi:10.1109/TIFS.2024.3359890.

[UMTS24] Vincent Quentin Ulitzsch, Soundes Marzougui, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on dilithium. In Benjamin Smith and Huapeng Wu, editors, *Selected Areas in Cryptography*, pages 3–32, Cham, 2024. Springer International Publishing. doi:10.1007/978-3-031-58411-4_1.

[WBD24] Ruize Wang, Martin Brisfors, and Elena Dubrova. A side-channel attack on a higher-order masked crystals-kyber implementation. In *International Conference on Applied Cryptography and Network Security*, pages 301–324. Springer, 2024. doi:10.1007/978-3-031-54776-8_12.

[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. doi:10.1109/TC.2021.3122997.

[YOR23] Yan Yan, Elisabeth Oswald, and Arnab Roy. Not optimal but efficient: a distinguisher based on the kruskal-wallis test. In *International Conference on Information Security and Cryptology*, pages 240–258. Springer, 2023. doi:10.1007/978-981-97-1235-9_13.