Zero-Value Filtering for Accelerating Non-Profiled Side-Channel Attack on Incomplete NTT based Implementations of Lattice-based Cryptography

Tolun Tosun[1]
Erkay Savas[1]
[1]*Sabanci University Computer Science and Engineering*

***Abstract*—Lattice-based cryptographic schemes such as Crystals-Kyber and Dilithium are post-quantum algorithms selected to be standardized by NIST as they are considered to be secure against quantum computing attacks. The multiplication in polynomial rings is the most time-consuming operation in many lattice-based cryptographic schemes, which is also subject to side-channel attacks. While NTT-based polynomial multiplication is almost a norm in a wide range of implementations, a relatively new method, incomplete-NTT is preferred to accelerate lattice-based cryptography, especially on some computing platforms that feature special instructions. In this paper, we present a novel, efficient and non-profiled power/EM side-channel attack targeting polynomial multiplication based on the incomplete NTT algorithm. We apply the attack on the Crystals-Dilithium signature algorithm and demonstrate that the method accelerates attack run-time when compared to conventional correlation power attacks (CPA). While a conventional CPA tests much larger hypothesis set due to the fact that it needs to predict two coefficients of secret polynomials together, we propose a much faster *zero-value filtering attack* (ZV-FA), which reduces the size of the hypothesis set by targeting the coefficients individually. We also propose an effective and efficient validation and correction technique to estimate and modify the mis-predicted coefficients. Our experimental results show that we can achieve a speed-up of 128.1× over conventional CPA using a total of 13K traces.**

***Index Terms*—post-quantum cryptography; side-channel attack; correlation power analysis; crsytals dilithium;**

I. Introduction

SECURITY of public-key cryptosystems relies on the hardness of well-known mathematical problems such as the discrete logarithm problem for the elliptic curve cryptography (ECC) [1], [2] and the digital signature algorithm (DSA) [3] or the integer factorization problem for RSA [4]. While those hard problems are conjectured to be secure against known cryptanalytic algorithms running on classical computers, it has been shown that Shor's algorithm [5] can solve them in polynomial time on a large-scale quantum computer.

To address the quantum threat, the National Institute of Standards and Technology (NIST) has announced the standardization process for post-quantum public-key cryptographic algorithms (PQC) in 2016. The standardization process covers quantum-resistant digital signature schemes, and public-key encryption and key-establishment algorithms. Currently, the contest is at the fourth round with already standardized algorithms. Lattice-based schemes, based on various hard lattice problems, facilitate the construction of quantum resilient public-key cryptography with a promising level of efficiency. Among the winners, the lattice-based digital signature algorithm Crystals-Dilithium [6] is based on the Module-LWE [7] and Module-SIS (MSIS) problems, while Crystals-Kyber [8] is MLWE based KEM. As Kyber and Dilithium are members of the same family, Crystals, they have several building blocks in common.

In cryptoanalysis, side-channel attacks (SCA) are the ones that target the weaknesses in implementations rather than algorithm specifications, by collecting side information such as running time or power consumption that can leak sensitive (intermediate) information during the execution of the targeted cryptographic operation. Side-channel attacks are considered as one of the main threats, particularly for embedded devices because of the simplicity of side-information collection. The Correlation Power Analysis (CPA) is proposed in [9], which models the power consumption of the device under test and measures the correlation of the model with real-world data to test secret value hypotheses. The power leakage of the device/implementation is often modeled with the Hamming Weight (HW)/Hamming Distance (HD) of/between the intermediate data. The Electromagnetic (EM) side-channels [10] are similar to power analysis as any attack suit designed for the power leakage can be practiced with EM leakage while it can supply more precise information about the sensitive intermediate data. Masking is one of the the most promising countermeasures against power/EM attacks, which randomizes the intermediate data with secret sharing so that characteristics of sensitive data are not reflected on power consumption.

Needless to say that the side-channel security of post-quantum public cryptography is essential as well since post-quantum algorithms are intended to replace the existing public-key standards in the near future and the usage of public-key cryptography in embedded devices will be potentially more extensive. For example, a secure firmware update on an embedded device relies on the security of the employed digital signature while embedded devices are open to timing, power/EM attacks by nature. With increasing interest, several attacks and countermeasures have been proposed for PQC candidates in the literature.

Polynomial multiplication is the core operation for practical constructions of lattice-based cryptography, which are based

Attack	Class	Algorithm	Implementation	Target Operation
this work	Non-profiled	Dilithium	ARM M4	polynomial multiplication
[20]	Non-profiled	Dilithium	Reference C	polynomial multiplication
[21]	Non-Profiled	Dilithium	HW	polynomial multiplication
[19]	Non-profiled	Kyber	ARM M4	polynomial multiplication
[16]	Profiled	Dilithium	Reference C	NTT
[17]	Profiled	Dilithium	Reference C	bit-unpacking
[21]	Profiled	Dilithium	HW	decoding / NTT
[22]	Profiled	Dilithium	Reference C / ARM M4	small polynomial sampling
[23]	Profiled	Dilithium	Reference C	decompose
[24]	Profiled	Kyber	Reference C / ARM M4	NTT
[18]	Profiled	R-LWE Encryption	ARM M4 (masked) [25]	NTT

TABLE I: Related Side-Channel Attacks from the Literature

on ring-learning with errors (R-LWE) problem [11]. Most implementations utilize number theoretic transform (NTT) for efficient polynomial arithmetic [12]. A technique referred as *incomplete NTT* is introduced to handle rings of special structures as well as for efficiency [13]–[15] in implementations of various lattice-based cryptography algorithms. Our study targets the incomplete NTT operation specifically.

Table I summarizes the related side-channel attacks against lattice-based schemes from the literature. Among the attack types, the profiled class forms the majority, where we require a device identical to the one targeted by the attacker, who tries to characterize the leakage when executing a cryptographic algorithm with a known secret key. In other words, the attacker needs to have more capability in a profiled attack compared to non-profiled class. Machine learning based approaches are gaining popularity in the design of profiled attacks for lattice-based cryptography [16], [17]. In addition, Primas et al. [18] present a notable study by combining the side-channel leakage of NTT computation with belief propagation algorithm to conduct a single-trace profiled attack.

As for the non-profiled class, the polynomial multiplication is the most attractive target operation [19]–[21]. Steffen et al. [21] conduct attack on a hardware implementation of Dilithium. Mujdei et al. [19] attack ARM M4 implementation of Kyber [15], which has a very similar NTT implementation with Dilithium, based on [14]. The authors of [19] show that the secret coefficients must be predicted in pairs due to the fact that the incomplete NTT algorithm is used in polynomial multiplication of the targeted implementation. Chen et al. [20] target the reference implementation of Dilithium, concentrating on improving the runtime performance of CPA, as conventional approach requires brute-force effort over 22-bit secrets based on the coefficient modulus length of Dilithium.

In our study, we present a more efficient non-profiled attack on incomplete NTT implementation, which facilitates that the coefficients of the secret polynomials can be predicted individually. To show its efficacy, we use a very recent and fast implementation of Dilithium on ARM M4 [15]. We present several non-profiled side-channel attacks targeting the multiplication in the incomplete NTT domain and finally develop a much more efficient approach exploiting the zero-valued coefficients of the challenge polynomials of the Dilithium scheme.

A. Main Contributions

We can list our contributions as follows:

- We present a novel non-profiled power/EM attack against incomplete NTT based implementations of polynomial multiplication in Lattice-based Cryptography, referred to as Zero-Value Filtering Attack (ZV-FA). Our approach is efficient as it decreases the number of hypotheses significantly by introducing a filtering technique based on zero-value coefficients in the known input/output polynomials of the operation targeted by the CPA attack.
- We present an efficient validation technique for estimating and correcting mis-predicted values for attacking to secret polynomials with short coefficients, utilizing the inverse NTT operation. The method not only ensures full accuracy on the estimated secret polynomials, but also accelerate the attack run time.
- We implement the ZV-FA with the validation technique on the M4 specific implementation of Dilithium [14]. Our experiments demonstrate that a moderate increase in the number of traces can decrease the attack run-time significantly. It is experimentally shown through EM side-channel that, a speed-up of two orders of magnitude in attack run-time can be achieved over a conventional CPA targeting the polynomial multiplication.

II. Notation

Matrices are represented by bold uppercase letters, such as $\mathbf{A}$, while vectors are represented by bold lowercase letters, such as $\mathbf{b}$. Sets are denoted by uppercase blackboard bold letters, such as $\mathbb{A}$. Polynomials are denoted by lowercase italic letters, such as f. Depending on the context, polynomials may be represented together with their indeterminate, such as $f(x)$. Subscripts together with square brackets are used to denote element(s) of matrices and vectors, such as $\mathbf{A}_{[i,j]}$ and $\mathbf{s}_{[i]}$; elements of sets, such as $\mathbb{A}_{[i]}$; coefficients of polynomials, such as $f_{[i]}$. The notation $\mathbf{A}_{[:,j]}$ denotes the j-th column vector of $\mathbf{A}$.

Modular reductions are performed in a centered manner. Specifically, given an integer k and a modulus q, the operation $k' = k \pmod^{\pm} q$ maps k to a unique integer k' in the range of $[-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ for odd q. The set of integers modulo q with centered reduction is denoted by $\mathbb{Z}_q$. On the other hand, the notation $\mathbb{Z}_q^+$ is used to denote the set of integers modulo q using the positive range, namely $[0, q-1]$. The ring of polynomials $\mathbb{Z}_q[x]/(X^n+1)$, where elements are polynomials of maximum degree of $n-1$, whose coefficients are modulo-q reduced, is denoted by R_q.

The dimensionality of matrices and vectors is shown in the superscript. For example, $R_q^{k \times l}$ represents a matrix of

dimensions $k \times l$, whose elements are in R_q. Similarly, superscript is used for the matrices and vectors to express their dimensionality, such as $\mathbf{A}^{N\times M}$ and $\mathbf{b}^N$. The cardinality of the set $\mathbb{A}$ is denoted by $|\mathbb{A}|$. The matrix-vector multiplication of the operands $\mathbf{A}$ and $\mathbf{b}$ is denoted by $\mathbf{Ab}$. Similarly, multiplication of the vector $\mathbf{b}$ with scalar a is denoted by $a\mathbf{b}$. Polynomial multiplication is denoted by the standard symbol $\cdot$, while element-wise multiplication of two vectors is denoted by $\odot$. In some cases, the symbol $\cdot$ is used to represent integer multiplication to support the narrative. The symbol $\times$ denotes Cartesian product between sets, such as $\mathbb{A} \times \mathbb{B}$.

The notation (also referred as infinity norm) $||s||_\infty$ is used to represent the maximum coefficient of the polynomial s in absolute value, whose elements are reduced in centered manner. Similarly, $||\mathbf{s}||_\infty$ is the maximum of the maximum absolute values of coefficients of the polynomials in the vector $\mathbf{s}$. The set S_η consists of polynomials $w \in R_q$ with $||w||_\infty \leq \eta$, where η is a (relatively small) positive integer, referred to as the set of *short polynomials*. Similarly, $\widetilde{S}_\eta$ is another set of short polynomials whose coefficients fall within the range of $[-\eta+1, \eta]$. Another subset of R_q is denoted by B_τ, which consists of polynomials with exactly τ coefficients that are either -1 or 1, ant the rest is zero. $\{0,1\}^N$ denotes the set of N-bit strings. The operator $\leftarrow$ denotes uniformly random sampling from the set on the right-hand side, such as $\rho \leftarrow \{0,1\}^N$. The symbol $\perp$ represents an invalid character.

III. Correlation Power Analysis (CPA)

CPA [9] is a well-known side-channel attack that exploits the dependency between the data processed by a cryptographic implementation and its power consumption. CPA assumes that a secret key is stored in a targeted device, and while it is not possible to directly access the key, the attacker can perform a cryptographic operation (e.g., encryption, decryption, signature generation) that involves the secret key by querying the device. Embedded devices such as IoT chips, which sign sensor data before transmission, are often targeted by CPA.

CPA can be summarized in the following steps:

- The attacker performs N experiments, i.e. queries the victim device, and records the power consumption of the device. M points are sampled in time at each power measurement. Power samples are stored in the matrix $\mathbf{T}^{N\times M}$ while $\mathbf{p}^N$ is the vector of known variables. $\mathbf{T}$ and $\mathbf{p}$ are referred to as *traces*.
- A point of interest (PoI) is selected by the adversary. The PoI should be a function of a known variable that changes at each experiment and the attacked secret that remains the same for all experiments.
- A set of predictions (i.e, hypotheses) is prepared, denoted by $\mathbb{K}$. Then, intermediate value matrix $\mathbf{V}^{N\times S}$ is computed w.r.t. each hypothesis; i.e., $\mathbf{V}_{[i,j]}$ is the value of the PoI computed using $\mathbf{p}_{[i]}$ and $\mathbb{K}_{[j]}$.
- $\mathbf{V}$ is mapped to the hypothetical power consumption matrix, $\mathbf{H}^{N\times S}$ by applying a power model. Most frequently used power models are Hamming Weight (HW) and Hamming Distance (HD). In other words, $\mathbf{H}_{[i,j]} = \mathrm{HW}(\mathbf{V}_{[i,j]})$, in case HW is chosen.

	NIST Security Level	2	3	5
Parameter	Meaning			
d	dropped bits from $\mathbf{t}$	13	13	13
τ	number of ± 1 in c	39	49	60
γ_1	coefficient range of $\mathbf{y}$	2^{17}	2^{19}	2^{19}
γ_2		$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
k,l	dimensions of $\mathbf{A}$	(4,4)	(6,5)	(8,7)
η	coefficient range of $\mathbf{s}_i$	2	4	2
ω	max. # 1's in $\mathbf{h}$	80	55	75

TABLE II: Dilithium parameter set

- The Pearson correlation coefficient between each $\mathbf{T}_{[:,i]}$ and $\mathbf{H}_{[:,j]}$ for $0 \leq i < M, 0 \leq\ j < |\mathbb{K}|$ is approximated by the following:

$$\hat{\rho}_{i,j} = \frac{\hat{\mathrm{cov}}(\mathbf{T}_{[:,i]}, \mathbf{H}_{[:,j]})}{\hat{\mathrm{std}}(\mathbf{T}_{[:,i]})\hat{\mathrm{std}}(\mathbf{H}_{[:,j]})} \tag{1}$$

  where $\hat{\mathrm{std}}$ and $\hat{\mathrm{cov}}$ stand for sample standard deviation and sample covariance functions, respectively. The score for the j^{th} prediction is equal to $\max_i(|\hat{\rho}_{i,j}|)$. The prediction with highest score, namely $\mathrm{argmax}_j(\max_i(|\hat{\rho}_{i,j}|))$, is the output of the attack. CPA is performed exactly the same for the EM side-channel. Overall complexity of the attack is $\Theta(NM|\mathbb{K}|)$.

IV. Dilithium

Crystals: Dilithium [6] is a lattice-based post-quantum digital signature scheme based on the hardness of MLWE and MSIS problems. It consists of three main algorithms: key generation (Algorithm 1), signature generation (Algorithm 2) and signature verification. It operates over the ring of polynomials R_q with dimension $n = 256$ and $q = 8380417 = 2^{23} - 2^{13} + 1$. The rest of the parameter set used by Dilithium can be found in Table II. The NIST security levels 2, 3, 5 are equivalent to the SHA-256 collision search, the AES-192 key search and the AES-256 key search, respectively.

The key generation operation creates a MLWE instance by the equation $\mathbf{t} := \mathbf{As}_1 + \mathbf{s}_2$ as seen in line 5 of Algorithm 1. The public matrix $\mathbf{A}$ and the vectors of short polynomials $\mathbf{s}_1$, $\mathbf{s}_2$ are generated pseudo-randomly from the 256-bit seed ζ, using the helper functions **H**, **ExpandA** and **ExpandS**, respectively. While **H** is instantiated as SHAKE-256, **ExpandS** and **ExpandA** utilizes SHAKE-256 and SHAKE-128, respectively. The public matrix $\mathbf{A}$ is represented by the 256-bit seed ρ in the public key. Another optimization in public key size is to store $\mathbf{t}_1$ which is the upper bits of $\mathbf{t}$, generated by the function $\mathbf{Power2Round}_q$.

Algorithm 1 Dilithium.KeyGeneration

1: $\zeta \leftarrow \{0,1\}^{256}$
2: $(\rho, \rho', K) \leftarrow \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} := \mathrm{H}(\zeta)$
3: $\mathbf{A} \in R_q^{k\times l} := \mathrm{ExpandA}(\rho)$
4: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k := \mathrm{ExpandS}(\rho')$
5: $\mathbf{t} := \mathbf{As}_1 + \mathbf{s}_2$
6: $(\mathbf{t}_1, \mathbf{t}_0) := \mathrm{Power2Round}_q(\mathbf{t}, d)$
7: $tr \in \{0,1\}^{256} := \mathrm{H}(\rho \mid\mid \mathbf{t}_1)$
8: **return** $(pk = (\rho, \mathbf{t}_1),\ sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

The signature generation of Dilithium, given in Algorithm 2, applies the rejection sampling idea. Most of the signature procedure is implemented in a loop, which iterates until a valid and secure signature is found. At each iteration, the pseudo-random numbers are refreshed by incrementing the nonce κ. The parameters are chosen such that expected number of iterations are small, around 4. Inside the signature loop, a vector of masking polynomials $\mathbf{y}$ is generated deterministically from the seed ρ' and the nonce κ using the helper function ExpandMask. $\mathbf{w}_1$ is the high-order bits of $\mathbf{w} = \mathbf{Ay}$. The function SampleInBall and the hash function in line 9 map μ and $\mathbf{w}_1$ to a challenge polynomial in B_τ that has exactly τ coefficients that are either -1 or 1 and the rest are 0, denoted by c. Both ExpandMask and SampleInBall utilizes SHAKE-256. The candidate signature is $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$. The first check in line 13 of Algorithm 2 is executed to assess the security of the signature and while the second concerns both security and correctness. The function MakeHint returns the positions where high bits of $\mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0$ and $\mathbf{w} - c\mathbf{s}_2$ are different. This is needed due to the fact that the verification is performed using $\mathbf{t}_1$ instead of $\mathbf{t}$. At line 16, hint related correctness checks are performed.

Algorithm 2 Dilithium.Sign(*sk*, *M*)

1: $\mathbf{A} \in R_q^{k \times l} :=$ ExpandA(ρ)
2: $\mu \in \{0,1\}^{512} :=$ H($tr \mid\mid M$)
3: $\kappa := 0, (\mathbf{z}, \mathbf{h}) := \perp$
4: $\rho' \in \{0,1\}^{512} :=$ H($K \mid\mid \mu$)
5: **while** $(\mathbf{z}, \mathbf{h}) := \perp$ **do**
6: $\quad \mathbf{y} \in \widetilde{S}_{\gamma_1}^l :=$ ExpandMask(ρ', κ)
7: $\quad \mathbf{w} := \mathbf{Ay}$
8: $\quad \mathbf{w}_1 :=$ HighBits$_q(\mathbf{w}, 2\gamma_2)$
9: $\quad \widetilde{c} \in \{0,1\}^{256} :=$ H($\mu \mid\mid \mathbf{w}_1$)
10: $\quad c \in B_\tau :=$ SampleInBall($\widetilde{c}$)
11: $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
12: $\quad \mathbf{r}_0 :=$ LowBits$_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$
13: $\quad$ **if** $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) := \perp$
14: $\quad$ **else**
15: $\quad\quad \mathbf{h} :=$ MakeHint$_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
16: $\quad\quad$ **if** $||c\mathbf{t}_0||_\infty \geq \gamma_2$ or # 1's in $\mathbf{h}$ is greater than ω **then** $(\mathbf{z}, \mathbf{h}) := \perp$
17: $\quad\quad$ **end if**
18: $\quad$ **end if**
19: $\quad \kappa := \kappa + l$
20: **end while**
21: **return** $\sigma = (\mathbf{z}, \mathbf{h}, \widetilde{c})$

V. Number Theoretic Transform (NTT)

Number Theoretic Transform (NTT) is a special form of Fast-Fourier Transform (FFT) that operates over $\mathbb{Z}_q$ instead of complex numbers $\mathbb{C}$. NTT allows efficient multiplication of polynomials over R_q. Representing a polynomial $a(x) \in R_q$ in the NTT domain can be viewed as an application of Chinese Remainder Theorem (CRT). Polynomial multiplication is achieved by element-wise multiplying the NTT representations of the operands:

$$a(x) \cdot b(x) = \text{NTT}^{-1}(\text{NTT}(a(x)) \odot \text{NTT}(b(x))) \quad (2)$$

Most lattice-based crypto systems, including Dilithium, operates over the ring R_q. NTT in R_q requires $q \equiv 1 \pmod{2n}$, which ensures a primitive $2n$-th root of unity ζ_{2n} exists in $\mathbb{Z}_q$ for which $\zeta_{2n}^n = -1 \mod q$, referred to as *negacyclic NTT*. Therefore, $(x^n + 1)$ is factored to $\prod_{i=0}^{n-1}(x - \zeta_{2n}^{2i+1})$. NTT computes the following isomorphism:

$$a(x) \approx \text{NTT}(a(x)) : \mathbb{Z}_q[x]/(x^n+1) \rightarrow \prod_{i=0}^{n-1} \mathbb{Z}_q[x]/(x-\zeta_{2n}^{2i+1})$$

Here, the i-th element of NTT$(a(x))$ is the remainder from dividing $a(x)$ to $(x - \zeta_{2n}^{2i+1})$. NTT can be computed by evaluating the polynomial at the powers of ζ_{2n}. Let $\mathbf{a} = \text{NTT}(a(x)) \in \mathbb{Z}_q^n$. Then, $\mathbf{a}_{[i]} = a(\zeta_{2n}^{2i+1})$ for $i < n$. However, this computation requires $\Theta(n^2)$ steps. In case n is a power of 2, NTT can be computed efficiently by splitting the polynomial to half of its size in recursive manner until linear degree is reached. The transformation at each layer can be efficiently implemented using *Cooley-Tukey* (CT) butterfly circuit [26]. For degree-$n/2^i$ polynomial $a(x) = a'(x) + a''(x) \cdot x^{n/2^{i+1}}$, the CT butterfly is defined by the map

$$a'(x)+a''(x)\cdot x^{n/2^{i+1}} \rightarrow (a'(x)-\delta\cdot a''(x), a'(x)+\delta\cdot a''(x)).$$

where δ is an odd power of ζ_{2n}, called the *twiddle factor*. In this manner, full NTT computation requires $\log n$ layers. Most applications use *Gentleman-Sande* (GS) butterfly for computing the inverse NTT [27], although it is not necessarily required. Using CT or GS when n is a power of 2, computing NTT / inverse NTT takes $\Theta(n \log n)$ steps.

A. Incomplete NTT

For efficiency NTT can be computed for $k < \log n$ layers so polynomials are recursively splitted to degree-$n/2^k$ polynomial components, denoted by $\text{NTT}_k(a(x))$. The pre-requisite for NTT_k is to have $q \equiv 1 \pmod{\frac{n}{2^{\log n - k - 1}}}$ for the negacyclic NTT [28].

Let $\mathbf{a} = \text{NTT}_k(a(x))$ and $\mathbf{b} = \text{NTT}_k(b(x))$. $\mathbf{a}$ and $\mathbf{b}$ are 2^k-dimensional vectors and $\mathbf{a}_{[i]}, \mathbf{b}_{[i]} \in \mathbb{Z}_q[x]/(x^{n/2^k} - \delta)$ for $i < 2^k$, where δ is some power of ζ_{2n}. Then, $a(x) \cdot b(x)$ can be computed through $\mathbf{a} \odot \mathbf{b}$ as in Equation 2. In this case, element-wise multiplication refers to multiplication of polynomials of degree $n/2^k - 1$, which is mostly implemented by the school-book algorithm, and there are 2^k such multiplications. For instance, when $k = \log(n/2)$, we perform $n/2$ multiplications of degree-1 polynomials.

B. Asymmetric Multiplication

It is shown that, pre-computation based on one of the operands accelerates the incomplete NTT multiplication in certain circumstances [29]. Consider the vector of polynomials $\mathbf{a}$ and $\mathbf{b}$ from the previous section for the case $k = (\log n/2)$.

We know that, $\mathbf{a}_{[i]}(x), \mathbf{b}_{[i]}(x) \in \mathbb{Z}_q[x]/(x^2 - \delta_i)$ for $i < n/2$, where and $x^2 = \delta_i$ for the mentioned ring. Then, we have

$$\mathbf{a}_{[i]}(x) \cdot \mathbf{b}_{[i]}(x) = (\mathbf{a}_{[i,0]} \cdot \mathbf{b}_{[i,0]} + \delta_i \cdot \mathbf{a}_{[i,1]} \cdot \mathbf{b}_{[i,1]}) + (\mathbf{a}_{[i,0]} \cdot \mathbf{b}_{[i,1]} + \mathbf{a}_{[i,1]} \cdot \mathbf{b}_{[i,0]}) \cdot x. \quad (3)$$

The terms $\delta_i \cdot \mathbf{a}_{[i,1]}$ are computed regardless of the values of $\mathbf{b}_{[i,1]}$, or vice-versa. Therefore, the terms $\delta_i \cdot \mathbf{a}_{[i,1]}$ can be pre-computed and retrieved from memory, leading to a method referred to as asymmetric multiplication as explained in the subsequent section. Notice that, this is beneficial only in case $\mathbf{a}$ is involved in more than one multiplication. For instance, it makes sense to use asymmetric multiplication with c in Algorithm 2 since it is involved in multiple multiplications. As explained in the subsequent section, ARM M4 core facilitates the multiplications of two coefficients with only one instruction.

VI. Proposed Side-Channel Attacks

In this section, we first show a straightforward baseline attack and then give the details of more efficient zero-value filtering attack.

A. Target Implementation and Point of Interest (PoI)

We target the Dilithium implementation of the pqm4 library [15], which is based on the work [14]. The aim of the attack is to recover the secret key *sk* so that the attacker can forge signatures. The implementation is specialized for the ARM M4 core, mostly written in Assembly.

A radical design decision made by the authors is that, NTT is *incomplete* as it is computed for 7 layers instead of $8 = \log_2 256$. Consequently, multiplication in this domain is achieved by 128 2×2 school-book multiplications of degree-1 polynomials, with the application of asymmetric multiplication. Both the Forward NTT and the Inverse NTT are carried out by CT butterflies. The Montgomery reduction algorithm is used to perform modular reductions; however, reductions in modular addition and subtractions during the butterflies are omitted for efficiency reasons, which is known as *lazy reduction* in the literature [30]. Related parts of the victim implementation is discussed in more detail in the following section.

Secondly, a carrier modulus is used for NTT representation of short polynomials in the vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ as well as the challenge polynomial c, referred to as *small NTT*. The small NTT operates with the prime $q' = 257$ for Dilithium2 and Dilithium5 while $q' = 769$ is chosen for Dilithium3. The rationale behind the small NTT is that, coefficients of $c\mathbf{s}_1$ and $c\mathbf{s}_2$ does not exceed $\tau\eta$ in absolute value. Recall that coefficient range of the polynomials in $\mathbf{s}_1$ and $\mathbf{s}_2$ is $[-\eta, \eta]$ while c has exactly τ coefficients equal to ± 1 and the rest of the coefficients are 0.

The targeted operation is the product $c\mathbf{s}_1$, which is performed in Line-11 of Algorithm 2. This is a natural choice for a non-profiled attack since it is where the $\mathbf{s}_1$ part of the secret key sk is processed along with the challenge polynomial c. Note that c is computed from $\widetilde{c}$, which is among the outputs of the signature and alternates with respect to the choice of the message M. Each polynomial in the vector $\mathbf{s}_1$ is multiplied by c; a multiplication in the ring R, which is performed in the NTT domain by the Assembly function `small_asymmetric_mul_asm` (See Source Code-1).

```
.macro montgomery q, qinv, a, tmp
  smulbt \tmp, \a, \qinv
  smlabb \tmp, \q, \tmp, \a
.endm

    movw r14, #769
    movt r14, #767
    .equ width, 4
    add.w r12, r0, #256*2
    _asymmetric_mul_16_loop:
    ldr.w r7, [r1, #width]
    ldr.w r4, [r1], #2*width
    ldr.w r8, [r2, #width]
    ldr.w r5, [r2], #2*width
    ldr.w r9, [r3, #width]
    ldr.w r6, [r3], #2*width

    smuad r10, r4, r6
    montgomery r14, r14, r10, r6
    smuadx r11, r4, r5
    montgomery r14, r14, r11, r10

    pkhtb r10, r10, r6, asr#16

    str.w r10, [r0], #width

    smuad r10, r7, r9
    montgomery r14, r14, r10, r6
    smuadx r11, r7, r8
    montgomery r14, r14, r11, r10

    pkhtb r10, r10, r6, asr#16
    str.w r10, [r0], #width

    cmp.w r0, r12
    bne.w _asymmetric_mul_16_loop
```

Source Code 1: Implementation of asymmetric multiplication by the function `small_asymmetric_mul_asm`

To compute $c\mathbf{s}_1$, `small_asymmetric_mul_asm` is invoked ℓ times. During the z^{th} call to the function `small_asymmetric_mul_asm`, the input register `r1` stores the address of $\text{NTT}_7(\mathbf{s}_{1[z]})$ for $0 \leq z < \ell$, which is an 128-dimensional vector of degree-1 polynomials. For the rest of the paper, we will denote any of these vectors by $\mathbf{s}^*$ to simplify notation since our work is identical for all z. The i^{th} polynomial in $\mathbf{s}^*$ is written as $\mathbf{s}^*_{[i]}(x) = \mathbf{s}^*_{[i,0]} + \mathbf{s}^*_{[i,1]}x$, for $0 \leq i < 128$. On the other hand, `r2` stores the address of $\text{NTT}_7(c)$, denoted by $\mathbf{c}$. Similarly, $\mathbf{c}_{[i]}(x) = \mathbf{c}_{[i,0]} + \mathbf{c}_{[i,1]}x$. Finally, `r3` stores the address of $\mathbf{c}'$, pre-computed from $\mathbf{c}$, for which the pre-computation methodology explained in Section V-B is applied. Precisely, $\mathbf{c}'_{[i]}(x) = \mathbf{c}_{[i,0]} + \delta_i \mathbf{c}_{[i,1]}x$, where δ_i are the twiddle factors. Pseudo-code for the function `small_asymmetric_mul_asm` is provided in Algorithm 3. Formally, $\mathbf{r} = \mathbf{s}^* \odot \mathbf{c}$ is computed. For the rest of the paper, the function $\text{NTT}(\cdot)$ refers to the explained incomplete NTT, *i.e.* $\text{NTT}_7(c)$, to simplify the notation.

The coefficients $\mathbf{s}^*_{[i,0]}, \mathbf{s}^*_{[i,1]}, \mathbf{c}_{[i,0]}, \mathbf{c}_{[i,1]} \in \mathbb{Z}_{q'}$ are represented by 16-bit signed integers in the memory. The 2×2 school-book multiplications needed for the degree-1 polynomials are

Algorithm 3 Asymmetric Multiplication Pseudo-Code($\mathbf{s}^* =$ `r1`, $\mathbf{c} =$ `r2`, $\mathbf{c}' =$ `r3`)

1: **for** $i \leftarrow 0$ until 128 **do**
2: $T_0 \leftarrow \mathbf{s}^*_{[i,0]} \cdot \mathbf{c}'_{[i,0]}$, $T_1 \leftarrow \mathbf{s}^*_{[i,1]} \cdot \mathbf{c}'_{[i,1]}$, $\mathbf{r}_{[i,0]} \leftarrow$ Montgomery$(T_0 + T_1)$
3: $T_2 \leftarrow \mathbf{s}^*_{[i,0]} \cdot \mathbf{c}_{[i,1]}$, $T_3 \leftarrow \mathbf{s}^*_{[i,1]} \cdot \mathbf{c}_{[i,0]}$, $\mathbf{r}_{[i,1]} \leftarrow$ Montgomery$(T_2 + T_3)$
4: $T_4 \leftarrow \mathbf{s}^*_{[i+1,0]} \cdot \mathbf{c}'_{[i+1,0]}$, $T_5 \leftarrow \mathbf{s}^*_{[i+1,1]} \cdot \mathbf{c}'_{[i+1,1]}$, $\mathbf{r}_{[i+1,0]} \leftarrow$ Montgomery$(T_4 + T_5)$
5: $T_6 \leftarrow \mathbf{s}^*_{[i+1,0]} \cdot \mathbf{c}_{[i+1,1]}$, $T_7 \leftarrow \mathbf{s}^*_{[i+1,1]} \cdot \mathbf{c}_{[i+1,0]}$, $\mathbf{r}_{[i+1,1]} \leftarrow$ Montgomery$(T_6 + T_7)$
6: $i \leftarrow i + 2$
7: **end for**
8: **return** `r4` $= \mathbf{r}$

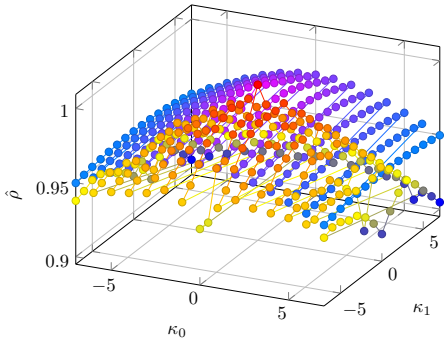

Fig. 1: Average (cold) and Minimum (hot) of the Pearson correlation coefficients, $\hat{\rho}$, computed between $\mathbf{H}_{:,\{\alpha_0,\alpha_1\}}$ and $\mathbf{H}_{:,\{\beta_0,\beta_1\}}$ for $\{\beta_0,\beta_1\} \in \psi_{\alpha_0,\alpha_1}$

achieved by `smuad` and `smuadx` instructions[1] followed by Montgomery reductions. The `smuad(x)` instructions consider their 32-bit inputs as two 16-bit halves, which store the two coefficients of degree-1 polynomials in the (incomplete) NTT domain. The `smuad` instruction multiplies the lower and upper halves of its operands concurrently and sums the results of the two multiplications. The `smuadx` instruction functions very similarly, except it multiples upper-halves of the operands by lower-halves. Lines 2 and 4 of Algorithm 3 illustrate the behaviour of `smuad` instruction while lines 3 and 5 demonstrate the behaviour of `smuadx`. As a more precise example, line 2 of Algorithm 3 is computed in a single step except for the Montgomery reduction by the `smuad` instruction at line 18 of Source Code 1. Notice that, `smuad` is used in the computation of the constant terms in the school-book multiplication while `smuadx` is used in computation of the degree-1 coefficients.

The results of Montgomery reductions for the lower and higher degree coefficients are combined in a single 32-bit register by `pkhtb` instructions at lines 23 and 32 of Source Code 1, followed by memory writes via the `str.w` instructions at lines 25 and 33. We consider the operands of the store instructions as the PoI, assuming a memory operation leads to a power leakage with a greater signal-to-noise-ratio (SNR) compared to the register updates. We use the HW model for the hypothetical power consumption computation.

B. The Baseline Attack

As each output coefficient $\mathbf{r}_{[i,j]}$, written to the memory by the `str.w` instruction, and chosen as the PoI, depends on both $\mathbf{s}^*_{[i,0]}$ and $\mathbf{s}^*_{[i,1]}$, the *baseline scheme* is formed as conventional CPA that predicts $\{\mathbf{s}^*_{[i,0]}, \mathbf{s}^*_{[i,1]}\}$, simultaneously. The crucial question is, within what range must each coefficient be predicted? Recall that, Montgomery modular reductions are omitted from modular addition and subtraction steps of the CT butterfly operations for efficiency during the NTT computations. Consequently, when the function `small_asymmetric_mul_asm` is called, the coefficients $\mathbf{s}^*_{[i,j]}$ can be larger than the modulus q'; precisely, they are in the range $[-7q'-\eta, +7q'+\eta]$, namely $\mathbb{Z}_{14q'+2\eta}$, despite $\mathbf{s}^*_{[i,j]} \in \mathbb{Z}_{q'}$ by definition as implied by [14], [20], [30]. A prediction in $\mathbb{Z}_{14q'+2\eta} \times \mathbb{Z}_{14q'+2\eta}$, which results in a approximately 26-bit search space for the pair $\{\mathbf{s}^*_{[i,0]}, \mathbf{s}^*_{[i,1]}\}$, is excessive. Fortunately, it is sufficient to predict in the set of residues [19], [20], $\mathbb{Z}_{q'}$ due to two main factors: 1) A trivial mathematical fact is that the coefficients are indeed in $\mathbb{Z}_{q'}$, which is sufficient to compute the inverse NTT to obtain the coefficients of the secret polynomial. 2) For the selected PoI function, the integers in $\mathbb{Z}_{14q'+2\eta}$ that are in the same congruence class modulo q', mostly result in the same output. Therefore, the PoI that are calculated for integers of the same congruence class are correlated with each other. Note that, the PoI is the output of the signed Montgomery reduction presented in [30], whose output is in $\mathbb{Z}_{2q'}$. We set an experiment to demonstrate this observation. Let $\alpha_0, \alpha_1 \in \mathbb{Z}_{q'} \times \mathbb{Z}_{q'}$ be a pair of predictions for $\{\mathbf{s}^*_{[i,0]}, \mathbf{s}^*_{[i,1]}\}$ for a value of i. Consider the following set of predictions:

$$\psi_{\alpha_0,\alpha_1} = \{\beta_0 = \alpha_0 \pm \kappa_0 \cdot q'\}_{\kappa_0} \times \{\beta_1 = \alpha_1 \pm \kappa_1 \cdot q'\}_{\kappa_1}$$
$$\text{for } 0 \leq \kappa_0, \kappa_1 \leq 7 \ , \ \beta_0, \beta_1 \in \mathbb{Z}_{14q'+2\eta}.$$

We run 10^5 experiments with uniformly random α_0, α_1 to assess our observation. Let $\mathbf{H}_{:,\{\alpha_0,\alpha_1\}}$ denotes the hypothetical power consumption vector computed w.r.t. $\{\alpha_0,\alpha_1\}$ based on the chosen PoI with a set of uniformly random $\mathbf{c}_{[i]}$ of size $N = 10$K and Hamming Weight (HW) as the power model. Similarly, $\mathbf{H}_{:,\{\beta_0,\beta_1\}}$ denotes the hypothetical power consumption w.r.t. each one of the pairs $\{\beta_0,\beta_1\} \in \psi_{\alpha_0,\alpha_1}$. Figure 1 shows the average and minimum of the Pearson correlation coefficients computed between $\mathbf{H}_{:,\{\alpha_0,\alpha_1\}}$ and each $\mathbf{H}_{:,\{\beta_0,\beta_1\}}$. The variables $\{\beta_0,\beta_1\}$ are enumerated by the factors κ_0, κ_1 from the above definition of ψ_{α_0,α_1} in the figure. The experiments show that, even the distant elements of the congruence class, such as those with $\kappa_0, \kappa_1 = 7$, are in correlation with $\{\alpha_0,\alpha_1\}$, in terms of the hypothetical power consumption.

We emphasize yet another significant point. The distribution of the coefficients $\mathbf{s}^*_{[i,j]}$ is not uniform in $\mathbb{Z}_{14q'+2\eta}$, as observed by [19] within the side-channel analysis of M4 specific implementations of PQC KEMs. We set another experiment to

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

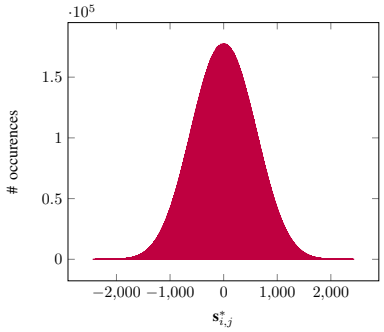

Fig. 2: Distribution of the integer values taken by $\mathbf{s}^*_{[i,j]}$ as a result of the NTT implementation's lazy reductions, where $\mathbf{s}_1$ is generated at uniformly random within 10^6 experiments.

further investigate the statistics of the values taken by $\mathbf{s}^*_{[i,j]}$, in practice. We uniformly randomly generated 10^6 samples for $\mathbf{s}_1$, for which, Figure 2 demonstrates that the distribution of $\mathbf{s}^*_{[i,j]}$ follows a bell shape. The maximum of the observed values is 2412 while the minimum is -2421, showing even $\kappa_0, \kappa_1 \geq 4$ is very rare. Consequently we can conclude that values in congruence class with higher correlation is more likely.

The findings of the experiments show that it is sufficient to perform the search for $\{\mathbf{s}^*_{[i,0]}, \mathbf{s}^*_{[i,1]}\}$ by including just one representative of the congruence class, precisely in $\mathbb{Z}_{q'} \times \mathbb{Z}_{q'}$, leading to q'^2 hypotheses. q'^2 is approximately 16.01-bit for Dilithium2 and Dilithium5, while it is approximately 19.17-bit for Dilithium3. The computational complexity of the baseline scheme is, therefore, $\Theta(q'^2(n/2))$. Clearly, it is an accurate, yet inefficient attacking scheme in terms of the attack run-time. Therefore, we seek more efficient methods to accelerate the attack time in the next section.

C. Boosting the Baseline Attack

Using negative correlation, it is possible to further narrow down the hypothesis set presented in the preceding section. Note that, the Hamming weights of an integer and its additive inverse in 2's complement notation are correlated. Consider the following enumeration of the set of hypotheses for $\{\mathbf{s}^*_{[i,0]}, \mathbf{s}^*_{[i,1]}\}$, ignoring 0s for the sake of simplicity.

$$\begin{aligned}\mathbb{K} = \mathbb{Z}_{q'} \times \mathbb{Z}_{q'} &= \{-\mathbb{Z}^+_{\lfloor q'/2 \rfloor}, \mathbb{Z}^+_{\lfloor q'/2 \rfloor}\} \times \{-\mathbb{Z}^+_{\lfloor q'/2 \rfloor}, \mathbb{Z}^+_{\lfloor q'/2 \rfloor}\} \\ &= \{\mathbb{K}_{0,0}, \mathbb{K}_{0,1}\} \times \{\mathbb{K}_{1,0}, \mathbb{K}_{1,1}\} \end{aligned} \tag{4}$$

We can consider dropping exactly one of $\mathbb{K}_{0,0}, \mathbb{K}_{0,1}, \mathbb{K}_{1,0}, \mathbb{K}_{1,1}$ from the search space. Without loss of generality, let $\mathbb{K}'$ be the reduced set defined as $\{\mathbb{K}_{0,0}, \mathbb{K}_{0,1}\} \times \{\mathbb{K}_{1,1}\}$. For any $\{\alpha_0, \alpha_1\} \in \mathbb{K}$, it is guaranteed that either $\{\alpha_0, \alpha_1\}$ or its additive inverse $\{-\alpha_0, -\alpha_1\}$ is included in $\mathbb{K}'$, which has a straightforward explanation. We can re-write $\mathbb{K}$ by distributing its terms as $\{\mathbb{K}_{0,0} \times \mathbb{K}_{1,0}, \mathbb{K}_{0,0} \times \mathbb{K}_{1,1}, \mathbb{K}_{0,1} \times \mathbb{K}_{1,1}, \mathbb{K}_{0,0} \times \mathbb{K}_{1,1}\}$. On the other hand, $\mathbb{K}'$ can be arranged as $\{\mathbb{K}_{0,0} \times \mathbb{K}_{1,1}, \mathbb{K}_{0,1} \times \mathbb{K}_{1,1}\}$ and let $\mathbb{K}''$ be the set obtained by inverting the prediction for $\mathbf{s}^*_{i,1}$ in $\mathbb{K}'$, namely $\mathbb{K}'' = \{\mathbb{K}_{0,0} \times -\mathbb{K}_{1,1}, \mathbb{K}_{0,1} \times -\mathbb{K}_{1,1}\}$. Clearly, $\mathbb{K} = \{\mathbb{K}', \mathbb{K}''\}$ since $\mathbb{K}_{1,0} = -\mathbb{K}_{1,1}$, proving the argument.

Showing the existence of the secret itself or its additive inverse in the reduced search space $\mathbb{K}'$ is not sufficient to be able to carry out a correlation attack using $\mathbb{K}'$. We should also note that, $\mathbf{V}_{:,\{-\alpha_0,-\alpha_1\}} = -\mathbf{V}_{:,\{\alpha_0,\alpha_1\}}$, for any $\{\alpha_0, \alpha_1\} \in \mathbb{K}$; recalling that $\mathbf{V}_{:,\{\alpha_0,\alpha_1\}}$ is the intermediate value vector computed w.r.t. $\{\alpha_0, \alpha_1\}$. The statistical properties of Hamming Weight suggest that $\mathbf{H}_{:,\{-\alpha_0,-\alpha_1\}}$ correlates with $\mathbf{H}_{:,\{\alpha_0,\alpha_1\}}$. Lastly, we need a distinguisher to tell the difference between the actual key and its additive inverse as the attacker can get either one of them. If the sign of the peak on correlation scores is positive we conclude that the actual key is found. Otherwise, additive inverse of the hypothesis is computed as the output of the attack. We would like to note that, the actual device leakage inversely correlates with the HW of intermediate data in some cases such as the data-bus is pre-charged with all 1's. Then, the behaviour of the explained distinguisher is reversed.

In summary, using the Boosted baseline scheme, denoted by baseline$^+$, the attack complexity drops by 1-bit to $\Theta(q'(q'/2)(n/2))$. The same approach is employed in [20] for attacking the reference implementation of Dilithium, which utilizes 8-layer NTT so it does not require simultaneous prediction of pairs.

D. Decreasing the Number of Hypotheses: Zero-Value Attack

A more practical scheme in terms of the attack run-time can be constructed by attacking to the coefficients $\mathbf{s}^*_{[i,0]}$ and $\mathbf{s}^*_{[i,1]}$ individually, referred here as *Zero-Value (ZV) Attack*. To achieve this, we need to eliminate the effect of one of the secret coefficients from the Montgomery reduction step during the asymmetric multiplication, whose output constitutes the chosen PoI. This can be accomplished by including only the traces to the attack that contain zeros in their coefficients, which multiply one of the secret coefficients. Consider line 2 of Algorithm 3 to develop intuition to the proposed method. Assume $\mathbf{c}'_{[i,1]} = 0$ for some $0 \leq i < n/2$, then T_1 becomes 0 and $\mathbf{r}_{[i,0]} = \text{Montgomery}(T_0 = \mathbf{s}^*_{[i,0]} \cdot \mathbf{c}'_{[i,0]})$. With sufficient number of traces meeting the condition $\mathbf{c}'_{[i,1]} = 0$, predictions on $\mathbf{s}^*_{[i,0]}$ can be made independently from $\mathbf{s}^*_{[i,1]}$ for the specific value of i. In general, traces with $\mathbf{c}'_{[i,1]} = 0$ ($\mathbf{c}'_{[i,0]} = 0$) or traces with $\mathbf{c}_{[i,0]} = 0$ ($\mathbf{c}_{[i,1]} = 0$) are used for predicting $\mathbf{s}^*_{[i,0]}$ ($\mathbf{s}^*_{[i,1]}$). These pre-requisites for the ZV attack are referred to as *zero-value conditions*. Table III lists the five attacking scenarios that can be adopted. For instance, to attack $\mathbf{s}^*_{[i,0]}$, we need the condition $\mathbf{c}'_{[i,1]} = 0$ and use $\mathbf{c}'_{[i,0]}$ or $\mathbf{c}_{[i,0]} = 0$ and use $\mathbf{c}_{[i,1]}$. Note also that $\mathbf{c}'_{[i,0]} = \mathbf{c}_{[i,0]}$ by the definition of the asymmetric multiplication. Therefore, an attack can be launched for $\mathbf{s}^*_{[i,1]}$ by using the traces with $\mathbf{c}'_{[i,1]} = 0$, as illustrated in the 5^{th} scenario in the table. As the conditions are identical with attack scenarios 1, 4 and 5, both $\mathbf{s}^*_{[i,0]}$ and $\mathbf{s}^*_{[i,1]}$ will show peaks in the results.

This approach leads to an attack complexity of $\Theta(q'n)$, $\Theta((q'/2)n)$, without and with the negative correlation trick from the previous section. The drawback of this method is the hardness of finding traces meeting the mentioned conditions. Recall that $\mathbf{c}$ and $\mathbf{c}'$ are computed from $\widetilde{c}$, which is among

Attacking Scenario	Target	Condition	Used Meta	Probability
1	$\mathbf{s}^*_{[i,0]}$	$\mathbf{c}'_{[i,1]} = 0$	$\mathbf{c}'_{[i,0]}$	0.00544
2	$\mathbf{s}^*_{[i,0]}$	$\mathbf{c}_{[i,0]} = 0$	$\mathbf{c}_{[i,1]}$	0.00344
3	$\mathbf{s}^*_{[i,1]}$	$\mathbf{c}'_{[i,0]} = 0$	$\mathbf{c}'_{[i,1]}$	0.00344
4	$\mathbf{s}^*_{[i,1]}$	$\mathbf{c}_{[i,1]} = 0$	$\mathbf{c}_{[i,0]}$	0.00344
5	$\mathbf{s}^*_{[i,1]}$	$\mathbf{c}'_{[i,1]} = 0$	$\mathbf{c}'_{[i,0]}$	0.00544

TABLE III: ZV-Attack Scenarios. Probabilities are experimentally computed for zero-value conditions with $N = 10^5$ valid traces

the output of the signature scheme. We mark a trace and the corresponding challenge $\widetilde{c}$ as valid for the attack if at least one coefficient in $\mathbf{c}$ or $\mathbf{c}'$ is 0; namely,

$$\mathbf{c}_{[0,0]} = 0 \vee \mathbf{c}_{[0,1]} = 0 \vee .. \vee \mathbf{c}_{[n/2-1,0]} = 0 \vee \mathbf{c}_{[n/2-1,1]} = 0 \vee \mathbf{c}'_{[0,0]} = 0 \vee \mathbf{c}'_{[0,1]} = 0 \vee .. \vee \mathbf{c}'_{[n/2-1,0]} = 0 \vee \mathbf{c}_{[n/2-1,1]} = 0 \tag{5}$$

The attacker has two options for collecting valid traces: i) Sending random inputs to the victim device and record only the valid traces, whereby the ratio for $\widetilde{c}$ being valid for random input is experimentally found as 0.23. ii) Compute messages M that lead to valid traces. The ability of the attacker to find messages that leads to valid challenges depends on the knowledge over the secret key ingredient K, which can easily be traced on the Dilithium signature generations function, given in Algorithm 2. Note that, to force signatures that result in valid traces, the attacker needs to learn K, which can be achieved by a side-channel attack on Keccak with similar resources [31], which is out of the scope of this paper. Once a sufficient number of valid traces are found, they can be used to attack $c\mathbf{t}_0$ and $c\mathbf{s}_2$ in addition to $c\mathbf{s}_1$ in all dimensions of vectors $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$, since all secrets are multiplied with same challenges.

The individual probabilities for the coefficients $\mathbf{c}'_{[i,j]}$ and $\mathbf{c}_{[i,j]}$ being 0, for a valid $\widetilde{c}$ are another crucial factor of attack performance. Table III lists the probabilities for the aforementioned conditions, which are obtained experimentally running signature generation algorithm with 10^5 different inputs. Note that, each $\mathbf{s}^*_{[i,j]}$ is attacked with the ones ensuring the corresponding zero-value conditions among the collected traces. The listed probabilities suggest that the conditions are not met very often. Intuitively, assuming the SNR in $\mathbf{T}$ requires 500 traces for the attack to converge, then the attacker must perform approximately 145K measurements on the victim's device considering the probabilities of conditions in the scenarios 2, 3, or 4 in Table III. Although the attacking phase of the presented scheme is significantly faster than the baseline by a factor of $q'/2$, it is extremely difficult to collect valid traces to carry out the attack.

E. Decreasing the Number of Traces: Zero-Value Filtering

While the ZV scheme introduced in Section VI-D requires a large number of traces to retrieve exactly the correct key, alternatively having the correct key fall in top-d candidate list is relatively inexpensive, depending on the value of d. Therefore, the ZV attack method can be used as a filtering mechanism for a conventional CPA as in the baseline scheme, forming a two-stage attacking scheme, referred to as *Zero-Value Filtering Attack* (ZV-FA), which is formalized in Figure 3. In the first stage (a.k.a. *filtering stage*), the attacker detects d candidates each for $\mathbf{s}^*_{[i,0]}$ and $\mathbf{s}^*_{[i,1]}$ using the ZV attack scheme, denoted by $\mathbb{K}_0$ and $\mathbb{K}_1$. Then, a set of predictions $\mathbb{K} = \mathbb{K}_0 \times \mathbb{K}_1$ of size d^2 is formed. $\mathbb{K}$ is scored by a usual CPA in the second stage (a.k.a. *scoring stage*). This is equivalent to carrying out the baseline attack with relatively small number of hypotheses, d^2, as opposed to q'^2. $\{\alpha_0, \alpha_1\}$ is the top scoring pair from the second stage, with score λ.

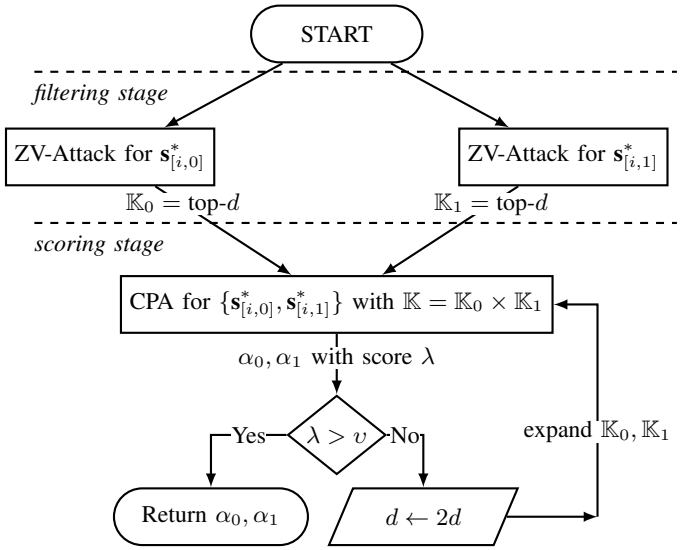

Fig. 3: Flowchart of ZV-FA on $\mathbf{s}^*_{[i]}$

By the filtering stage, this method assumes that the correct key is in the top-d list of predictions of the highest scores for the ZV attack. A threshold mechanism, denoted by υ, validates the assumption through the attack output. The value of d is iteratively increased and $\mathbb{K}_0$ and $\mathbb{K}_1$ are updated accordingly until a prediction scoring greater than υ is found. By increasing d, more candidates are evaluated by the second stage, which increases the probability having the correct key in the top-d list; naturally the second stage takes longer to evaluate more candidates. Needless to say, the evaluated candidates from previous trials are not included during the attack. Although we double d at each time the threshold is not exceeded, a different strategy on increasing d can be applied, as well. The attack becomes identical to the baseline attack for which the threshold is not taken into account if the scores remain below υ until d covers the whole search space.

Compared to the ZV attack scheme, the new ZV-filtering attack is more effective with a significantly smaller number of traces. The number of traces included in the filtering stage is denoted by N_f, while the second stage can be carried out without the zero-value conditions. As a result, it can be carried out with sufficient number of traces to ensure that its output is reliable rather than using the entire set of valid traces, which is excessive for evaluating the score.

F. Using Inverse-NTT to Validate Predictions

The zero-value filtering attack introduced in Section VI-E relies on the assumption that a precise threshold can be found for all attacks on $\mathbf{s}^*_{[i]}$ for $0 \leq i < n/2$, which, however, may not hold in practice as a non-profiled attack is performed in a blind manner. A possible solution to this problem is to use a conservative threshold. However, this approach is computationally expensive, and a conservative threshold can still result in false positives, albeit with a lower probability. Therefore, the attacker needs to verify the found secrets $\mathbf{s}_1$, $\mathbf{s}_2$, and $\mathbf{t}_0$ using the public key and line 5 of Algorithm 1. Note that this verification can only be performed after all the mentioned secrets have been attacked in all vector indices.

A more reasonable strategy for the attacker is to make use of the fact that $\text{NTT}^{-1}(\mathbf{s}^*) \in S_\eta$, is a short polynomial, whose coefficients are in the range $[-\eta, \eta]$. A small mistake in the prediction will diffuse through the inverse NTT computation and ruin the coefficients of the output polynomial, empowering the attacker efficiently validate the attack output. Figure 4 illustrates the flowchart of the ZV-FA from a higher-level perspective with validation using the inverse NTT. Let $\mathbf{a}$ denote the vector of polynomials, a prediction to $\mathbf{s}^*$, formed after completing the individual ZV-FAs to $\mathbf{s}^*_{[i]}$ for all i. Note that, $\mathbf{a}_{[i]}$ is equal to $\{\alpha_0, \alpha_1\}$ in Figure 3. To validate $\mathbf{a}$, $a = \text{NTT}^{-1}(\mathbf{a})$ is computed and the shortness property is sought in the resulting polynomial a. In case the found polynomial is not validated, the mispredicted pair of coefficients in $\mathbf{a}$ is approximated and re-attacked in order to correct it. The approximation is performed by selecting the pair of coefficients with minimum score as computed by ZV-FA. Let $\lambda(\mathbf{a}_{[i]})$ denote the score computed for $\mathbf{a}_{[i]}$ by the application of the ZV-FA to $\mathbf{s}^*_{[i]}$, as can be observed in Figure 3. Then, the index of the minimum scoring pair from $\mathbf{a}$ is found by computing $i' = \text{argmin}_{i'}(\lambda(\mathbf{a}_{[i']}))$ and the baseline$^+$ attack is performed on $\mathbf{s}^*_{[i']}$ to replace $\mathbf{a}_{[i']}$. As the baseline$^+$ attack is a more exhaustive method compared to ZV-FA, a reliable prediction is obtained for the coefficient pair with index i'. Needless to say, until the inverse NTT results in a short polynomial, the process is repeated. We should also not that, once a pair of coefficients in $\mathbf{s}^*$ is corrected by the baseline$^+$, that pair is not included in the subsequent baseline$^+$ attacks as the baseline$^+$ cannot be repeated for the same coefficient pair.

The application of inverse NTT as a reliable and efficient method of verification renders the ZV-FA fault-tolerant. This method ensures the preservation of accuracy regardless of the choice of υ, allowing for the use of lower thresholds that can enhance attack performance. Furthermore, verification can be conducted individually on $\mathbf{s}^*$, *i.e.* any of the polynomials in $\mathbf{s}_1$ or $\mathbf{s}_2$. After obtaining $\mathbf{s}_1$ and $\mathbf{s}_2$, through the side-channel attacks described earlier, the attacker can compute $\mathbf{t}_0$ using line 5 of Algorithm 1.

VII. Results

In this section, we present the results obtained after implementing the above-mentioned attacks on a realistic experimental setting.

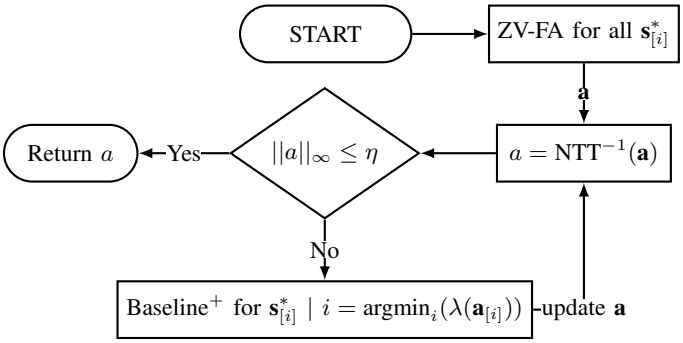

Fig. 4: Application of Inverse NTT Validation to ZV-FA for attacking $\mathbf{s}^*$

A. Experimental Setup

We employ Analog Devices' MAX32520[2] as the victim device to run pqm4's Dilithium signature implementation. The MAX32520 incorporates a 120MHz ARM Cortex-M4F core that can sign random 32B messages in 65.52 ms, on average. For EM trace collection, LeCroy WavePro HD oscilloscope[3] and Langer ICR HH500-6[4] nearfield micro-probe are used. Sampling rate of the oscilloscope is set to 10GS/s, yielding 83.33 samples per clock. We set up a trigger at the beginning of the function `small_asymmetric_mul_asm` to record the relevant time samples because our focus is on the polynomial multiplication. That the attacker would collect time samples throughout the entire multiplication process in a real-world scenario and use pattern detection to locate the mentioned function is beyond the scope of this work. The Scared library[5] is used for analysis and attack, running on a computer equipped with 64GB RAM and AMD Ryzen 9 5900X 12-Core Processor clocked at 3.70GHz. Note that the study requires a Python model that mimics the target processor's behavior for the intended Dilithium implementation, which is developed in-house.

B. Post-processing and Analysis

Our first observation is that, misalignment emerges as time progresses in the trace set. It is clearly seen in Figure 5a that the correlation of $\mathbf{c}$ as well as $\mathbf{c}'$ with EM samples decreases over time, towards greater indexes of both vectors. To cope with the adverse effects of misalignment We performed the following post-processing steps: 1) pattern detection, 2) signal filtering, 3) extraction around peaks. A reference pattern is set by band-pass filtering the first trace between 100MHz - 140MHz and applying moving variance on it. A 50 MHz low-pass filter is applied to the raw traces which are aligned based on the reference pattern. Then, 64 peaks, which correspond to iterations of `small_asymmetric_mul_asm` (see Algorithm 3), are detected and 2480 sequential points after each peak are combined. Figure 5b illustrates the effect of post-processing through meta data correlation. Figure 6 highlights

[2]https://www.analog.com/en/products/max32520.html

[3]https://teledynelecroy.com/oscilloscope/wavepro-hd-oscilloscope

[4]https://www.langer-emv.de/en/product/near-field-microprobes-icr-hh-h-field/26/icr-hh500-6-near-field-microprobe-2-mhz-to-6-ghz/108

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

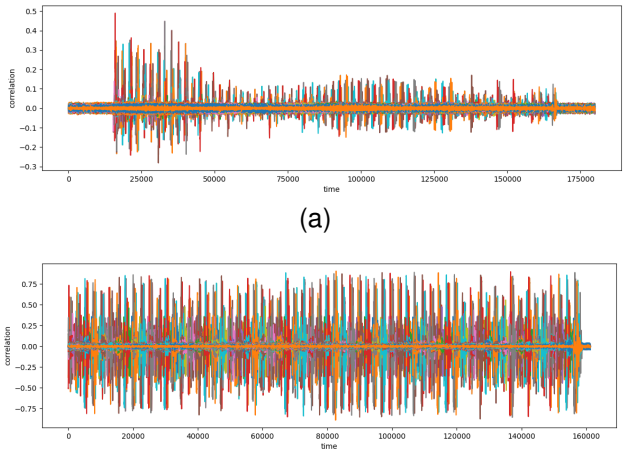

Fig. 5: Reverse meta analysis for **c** and **c**', before (a) and after (b) synchronization.

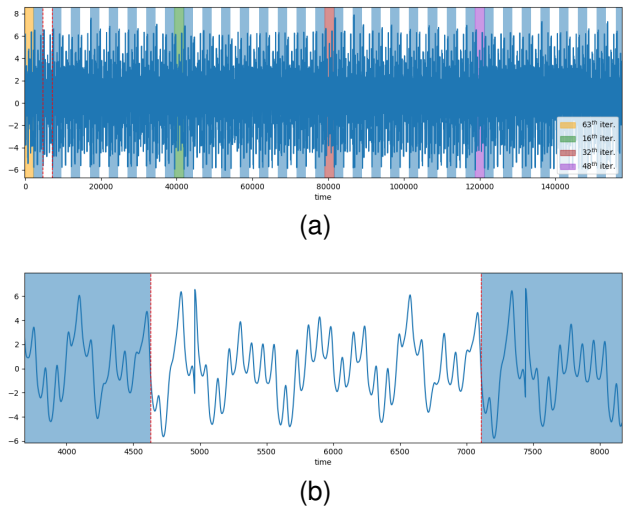

Fig. 6: Iterations of `_asymmetric_mul_16_loop` highlighted over mean trace

the iterations over the average of post-processed traces, conforming with the pre-knowledge on the implementation.

Given the clear visibility of the iterations of `_asymmetric_mul_16_loop` over time samples, it is possible to conduct individual attacks on $\mathbf{s}^*_{[i]}$ during time regions associated with each iteration. Note that, partitioning the attack range over time is critical for the presented performance results of all schemes. As an initial analysis, we have performed the baseline$^+$ on $\mathbf{s}^*_{[0]}$ and $\mathbf{s}^*_{[1]}$. The convergence patterns of the retrieved secrets are presented in Figure 7. It is evident that the convergence behaviors of the two attack indexes are dissimilar. Notably, attacking $\mathbf{s}^*_{[0]}$ targets Line 25 of Source Code 1, while attacking $\mathbf{s}^*_{[1]}$ targets Line 33, resulting in distinctive SNRs. Therefore, we employ different numbers of traces and thresholds for attacks on $\mathbf{s}^*_{[i]}$ depending on the oddness of i.

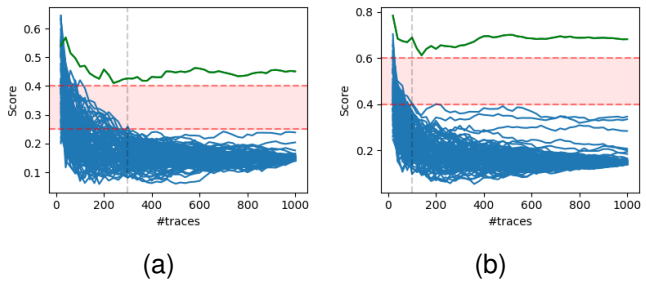

Fig. 7: Key convergence for $\mathbf{s}^*_{[0]}$ (a) and $\mathbf{s}^*_{[1]}$ (b) with the valid threshold ranges

C. Attack and Performance

We start the evaluation by the performance of the baseline and baseline$^+$ schemes. As discussed in the preceding section, the signal-to-noise ratio (SNR) varies for attacking odd and even indices of the secret vector $\mathbf{s}^*$. Accordingly, 300 and 100 traces are employed for attacking even and odd indexes, respectively, based on the key convergence results presented in Figure 7. Clearly, fewer number of traces could indeed work for the initially analysed indexes. However, we need to set a ballpark margin considering noisy indexes. It should be noted that the same number of traces is employed for the scoring stage of the ZV filtering attack, and the focus of this paper is not on reducing N further, but rather on assessing the impact of filtering. The performance of the baseline and baseline$^+$ schemes is reported in Table IV. In terms of accuracy, both schemes exhibit flawless performance and do not pose any concerns. However, in terms of run-time, the performance of the attacks is moderate. As expected, the baseline$^+$ scheme improves the performance of the baseline scheme by a factor of $\mathbf{2}\times$ while preserving accuracy, which supports the correctness of the attack methodology. Nevertheless, even with the baseline$^+$ scheme, retrieving $\mathbf{s}_1$ and $\mathbf{s}_2$ requires approximately 4.5 hours.

For the application of ZV-FA, the attack scenarios 1 and 2 from Table III are employed for attacking the lower degree coefficients of the polynomials, specifically $\mathbf{s}^*_{[i,0]}$ for any $0 \leq i < 128$, while the scenarios 3 and 5 are utilized for the higher degree coefficients $\mathbf{s}^*_{[i,1]}$. It should be noted that scenarios 1 and 5 are not independently executed, as they represent the same attack. The outcomes of different scenarios are combined by multiplying their respective results. Experimental results indicate that the ZV filtering can substantially decrease the attack response time up to two orders of magnitude, dependant upon the number of filtering traces N_f available in the system. The trade-off between N_f and speed-up is illustrated in Figure 8a. It is noteworthy that even a small number of traces can yield a significant improvement in baseline performance. For example, the ZV-FA provides a speed-up of $\mathbf{9}\times$ with $N_f = 5K$, collection of which is feasible. When more valid traces are available in the system, particularly with $N_f = 13K$, ZV-FA achieves a speed-up of $\mathbf{128.1}\times$ over the baseline. We underline that, the achieved speed-up can save approximately **535** minutes (≈ 9 hours) considering the retrieval of whole $\mathbf{s}_1$

Method	N	Runtime($\mathbf{s}^*_{[i]}$)	Runtime($\mathbf{s}^*$)	Runtime($\mathbf{s}_1$)	Runtime($\mathbf{s}_1$,$\mathbf{s}_2$)
Baseline	300,100	≈22.96s	≈49m	≈245m	≈539m
Baseline$^+$	300,100	≈11.48s	≈24.5m	≈122.5m	≈269.6m

TABLE IV: Performance of Baseline and Baseline$^+$ Attacks

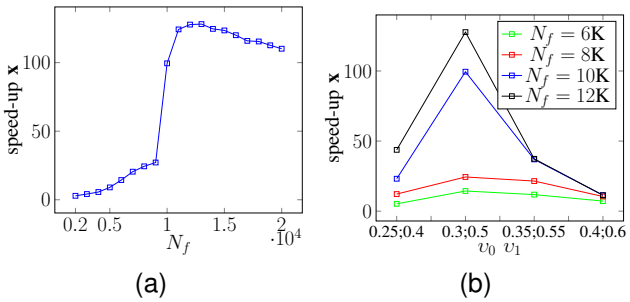

Fig. 8: Speed-up w.r.t. N_f (a) and υ (b)

and $\mathbf{s}_2$ with a trace collection overhead of 14.1 minutes for filtering.

Threshold values for odd and even indexes of $\mathbf{s}^*$ are assigned distinctively, as can be observed in the threshold ranges highlighted in Figure 7. One can also observe from Figure 8b that the best speed-up is reached for the threshold values of $\{\upsilon_0, \upsilon_1\} = \{0.3, 0.5\}$. While attacker may not always know the best threshold values, using other threshold values can also provide significant speedup over the baseline attack. For lower threshold values, false positives deteriorate the performance. On the other hand, the algorithm may not terminate for higher threshold values, as the correct hypothesis may fail to satisfy them. We also note that the scheme's accuracy is preserved for any threshold value thanks to the inverse NTT validation and correction mechanism presented in Section VI-F.

Note also that minimum distances to the corresponding threshold value are taken into consideration instead of absolute scores during the execution of the flowchart in Figure 4 for replacing mispredicted values as we use two distinct threshold values for the elements of $\mathbf{s}^*$ with even and odd indices.

Similarly, the scheme's accurate independently from the number of traces used for filtering N_f, which, however, determines its performance. Recall that, the performance of the ZV-FA scheme strongly depends on the effectiveness of the filtering stage. Figure 9 shows that, even with a moderate values of N_f, a secret pair with score greater than the threshold is discovered within the top-d during the filtering stage for a remarkable portion of the secret coefficients and practical values of d in terms of performance. Particularly for $N_f = 5$K, 84 of 128 (%65) secret pairs are retrieved in top-64, which corresponds to $(769 - 64)/769$ (%92) reduction in the search space from the baseline to scoring stage of ZV-FA.

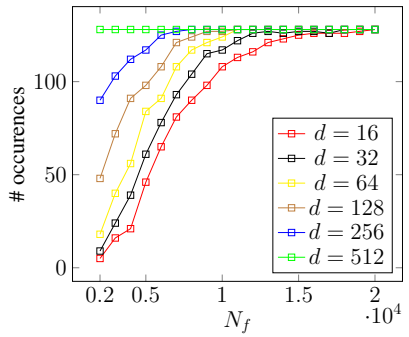

Fig. 9: Number of occurrences for which threshold is exceeded by evaluating the candidates from top-d with respect to d for $\upsilon_0 = 0.3$, $\upsilon_1 = 0.5$

VIII. Conclusion

This paper presents a series of non-profiled side-channel attacks against the incomplete NTT based implementation of lattice-based post-quantum signature algorithm Dilithium [14], [15]. Specifically, the attacks focus on the NTT-based polynomial multiplication $c\mathbf{s}_1$, although they can also be applied to $c\mathbf{s}_2$ without any modification. The target implementation operates with a carrier prime $q' = 769$, which results in a key guess space of q'^2 since two coefficients of the incomplete NTT representation must be predicted together. The baseline and baseline$^+$ schemes are conventional methods that rely on brute-force method in the sets of cardinality q'^2 and $q'^2/2$, respectively. Our study shows that, the cost of a conventional CPA against the studied incomplete NTT based implementation of Dilithium [14] is approximately 3-bit cheaper compared to the attack [20] against Dilithium's traditional implementation utilizing the 22-bit q. To mitigate the search costs, we introduced the zero-value attack, which reduces the size of the set of hypotheses to q' in the brute force attack by taking advantage of multiplication by 0 to eliminate one of the attacked pair of coefficients from the equation. However, this approach requires a significantly higher number of traces. Next, we presented the zero-value filtering attack, which represents a trade-off between the number of traces and attack run-time. With an appropriate number of traces, this attack can achieve a speed-up of two orders of magnitude over the baseline. Finally, we proposed an efficient way of verification of predictions on short polynomials. It makes the proposed scheme accurate independent of parameters such as the threshold, number of filtering traces. Experiments suggest that the ZV-FA is favorable even with moderate parameters.

Although we practiced our approach against Dilithium, it can be generalized to similar incomplete NTT based implementations of lattice-based cryptography, such as the Kyber implementation in [14]. Note that, Kyber also employs 7 layers of incomplete NTT albeit with a different prime, leading $\approx$ 23.4-bit search space for a pair of secret coefficients with exhaustive search [19]. Masking stands as the most promising way of counteracting the presented attacks [32]. As a takeaway, our study evidences that special cases such as zero-values in parts of targeted secrets can still be exploited efficiently, although they are not frequent, by constructing a

filtering mechanism based on them and the accuracy of the attack is preserved if a suitable method of verifying hypothesis can be found.

REFERENCES

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[2] V. S. Miller, *Use of elliptic curves in cryptography*. Springer, 1986.

[3] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology—CRYPTO'89 Proceedings 9*. Springer, 1990, pp. 239–252.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[5] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[6] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium algorithm specifications and supporting documentation (version 3.1)," *NIST Post-Quantum Cryptography Standardization Round*, vol. 3, 2021.

[7] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[8] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: a cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[10] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 29–45.

[11] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *J. ACM*, vol. 60, no. 6, pp. 43:1–43:35, 2013. [Online]. Available: https://doi.org/10.1145/2535925

[12] R. Agarwal and C. Burrus, "Fast convolution using fermat number transforms with applications to digital filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 2, pp. 87–97, 1974.

[13] V. Lyubashevsky and G. Seiler, "NTTRU: truly fast NTRU using NTT," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 3, pp. 180–201, 2019. [Online]. Available: https://doi.org/10.13154/tches.v2019.i3.180-201

[14] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels, "Faster kyber and dilithium on the cortex-m4," in *Applied Cryptography and Network Security - 20th International Conference, ACNS 2022, Rome, Italy, June 20-23, 2022, Proceedings*, ser. Lecture Notes in Computer Science, G. Ateniese and D. Venturi, Eds., vol. 13269. Springer, 2022, pp. 853–871. [Online]. Available: https://doi.org/10.1007/978-3-031-09234-3_42

[15] M. J. Kannwischer, R. Petri, J. Rijneveld, P. Schwabe, and K. Stoffelen, "PQM4: Post-quantum crypto library for the ARM Cortex-M4," https://github.com/mupq/pqm4.

[16] I.-J. Kim, T.-H. Lee, J. Han, B.-Y. Sim, and D.-G. Han, "Novel single-trace ml profiling attacks on nist 3 round candidate dilithium," *Cryptology ePrint Archive*, 2020.

[17] S. Marzougui, V. Ulitzsch, M. Tibouchi, and J.-P. Seifert, "Profiling side-channel attacks on dilithium: A small bit-fiddling leak breaks it all," *Cryptology ePrint Archive*, 2022.

[18] R. Primas, P. Pessl, and S. 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*. Springer, 2017, pp. 513–533.

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

[20] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. 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)*. IEEE, 2021, pp. 583–590.

[21] H. Steffen, G. Land, L. Kogelheide, and T. Güneysu, "Breaking and protecting the crystal: Side-channel analysis of dilithium in hardware," *Cryptology ePrint Archive*, 2022.

[22] E. Karabulut, E. Alkim, and A. 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)*. IEEE, 2021, pp. 35–45.

[23] A. Berzati, A. C. Viera, M. Chartouni, S. Madec, D. Vergnaud, and D. Vigilant, "A practical template attack on crystals-dilithium," *Cryptology ePrint Archive*, 2023.

[24] Z. Xu, O. Pemberton, S. S. Roy, D. Oswald, W. Yao, and Z. Zheng, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber," *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2163–2176, 2021.

[25] O. Reparaz, S. S. Roy, R. De Clercq, F. Vercauteren, and I. Verbauwhede, "Masking ring-lwe," *Journal of Cryptographic Engineering*, vol. 6, no. 2, pp. 139–153, 2016.

[26] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[27] W. M. Gentleman and G. Sande, "Fast fourier transforms: for fun and profit," in *Proceedings of the November 7-10, 1966, fall joint computer conference*, 1966, pp. 563–578.

[28] V. Lyubashevsky and G. Seiler, "Nttru: truly fast ntru using ntt," *Cryptology ePrint Archive*, 2019.

[29] H. Becker, V. Hwang, M. J. Kannwischer, B.-Y. Yang, and S.-Y. Yang, "Neon ntt: faster dilithium, kyber, and saber on cortex-a72 and apple m1," *Cryptology ePrint Archive*, 2021.

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

[31] M. Taha and P. Schaumont, "Differential power analysis of mac-keccak at any key-length," in *International Workshop on Security*. Springer, 2013, pp. 68–82.

[32] V. Migliore, B. Gérard, M. Tibouchi, and P.-A. 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*. Springer, 2019, pp. 344–362.

IX. BIOGRAPHY SECTION

Tolun Tosun received the B.S. and M.S. degrees in computer science and engineering from the Faculty of Natural Science and Engineering, Sabanci University in 2016 and 2019, respectively. He is a Ph.D. student in Sabanci University starting from 2019. His research interests include applied cryptography; side-channel attacks and countermeasures, lattice-based cryptography, applications of (fully) homomorphic encryption, privacy preserving machine learning. He is also working in the industry since 2019 as a cryptography engineer.

Erkay Savas received the BS (1990) and MS (1994) degrees in electrical engineering from the Electronics and Communications Engineering Department at Istanbul Technical University. He completed the PhD degree in the Department of Electrical and Computer Engineering (ECE) at Oregon State University in June 2000. He had worked for various companies and research institutions before he joined Sabanci University in 2002. He has been the dean of Faculty of Engineering and Natural Science, Sabanci University, since July 1, 2020. His research interests include applied cryptography, data and communication security, privacy in biometrics, security and privacy in data mining applications, embedded systems security, and distributed systems. He is a member of IEEE, ACM, the IEEE Computer Society, and the International Association of Cryptologic Research (IACR).