A Practical Template Attack on CRYSTALS-Dilithium

Alexandre Berzati[1], Andersson Calle Viera[1,2], Maya Chartouny[1,3], Steven Madec[1], Damien Vergnaud[2] and David Vigilant[1]

[1] Thales DIS, France
{alexandre.berzati,andersson.calle-viera,maya.saab-chartouni,steven.madec,david.vigilant}@thalesgroup.com
[2] Sorbonne Université, CNRS, Inria, LIP6, F-75005 Paris, France
[3] Université Paris-Saclay, UVSQ, CNRS, Laboratoire de mathématiques de Versailles, 78000, Versailles, France

Abstract. This paper presents a new profiling side-channel attack on the signature scheme CRYSTALS-Dilithium, which has been selected by the NIST as the new primary standard for quantum-safe digital signatures. This algorithm has a constant-time implementation with consideration for side-channel resilience. However, it does not protect against attacks that exploit intermediate data leakage. We exploit such a leakage on a vector generated during the signing process and whose costly protection by masking is a matter of debate. We design a template attack that enables us to efficiently predict whether a given coefficient in one coordinate of this vector is zero or not. Once this value has been completely reconstructed, one can recover, using linear algebra methods, part of the secret key that is sufficient to produce universal forgeries. While our paper deeply discusses the theoretical attack path, it also demonstrates the validity of the assumption regarding the required leakage model, from practical experiments with the reference implementation on an ARM Cortex-M4.

Keywords: Dilithium · Digital signature · Lattice-based cryptography · Post-quantum cryptography · Side-channel attacks · Template Attacks · Integer Linear Programming · Learning with Errors

1 Introduction

Public-key cryptography relies on well-defined mathematical assumptions to enable secure communication over unsecured channels. For instance, RSA [RSA78] security relies upon the difficulty of factoring large integers, and the Diffie-Hellman key exchange [DH76] is based on the hardness of the discrete logarithm. However, assuming a sufficiently powerful quantum computer would be available, Shor has proposed a quantum algorithm [Sho94] that can break those cryptosystems in polynomial time.

Shor's algorithm highlights the need for new cryptographic schemes, and *post-quantum cryptography* (PQC) is an emerging field of cryptography focused on developing cryptographic algorithms that are secure against quantum (and classical) computers. The National Institute of Standards and Technology (NIST) has initiated a process to standardize new quantum-resistant public key cryptographic algorithms. Three PQC signature algorithms were selected in 2022: CRYSTALS-Dilithium [LDK+22], Falcon [SBN+21], and SPHINCS+ [BHK+19]. In this paper, we focus on the Dilithium signature algorithm, which should be used by default in most of the use cases, according to the NIST [AASA+22]. CRYSTALS-Dilithium security relies on the hardness of finding short vectors in lattices.

Its design is inspired by Schnorr signatures [Sch91] and based on the "Fiat-Shamir with Aborts" approach [Lyu09].

Side-channel attacks exploit the physical implementation of cryptographic systems, such as the power consumption or timing information, to infer secrets, making them an important consideration in cryptography. One of the advantages of Dilithium is its simple constant-time implementation, as one can see in the reference implementation [DKL+22] which was designed with particular consideration for side-channel resilience. Still, it does not claim protection against more powerful attacks exploiting intermediate data leakage, such as differential power analysis (DPA) [KJJ99]. It was shown in [MGTF19] that some data leakages exist, and some may be exploitable (see also [MGTF19, RJH+18, MUTS22, ABC+22, LZS+21, QLZ+23] and references therein).

Our Contributions. In this paper, we introduce a new profiling-based side-channel attack on Dilithium.

Dilithium is a signature scheme based on the Fiat-Shamir heuristic and its security relies on the hardness of the Module Learning with Errors and Module Short Integer Solutions problems. Given a public matrix $\mathbf{A}$ of elements in the ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ for integers n and q, the signing algorithm samples a vector $\mathbf{y}$ using an `ExpandMask` function and computes $\mathbf{w} := \mathbf{A}\,\mathbf{y}$. The algorithm extracts high-order and low-order bits out of each coefficient of $\mathbf{w}$ using the `Decompose` function that outputs the `HighBits` and `LowBits` vectors denoted respectively $\mathbf{w}_1$ and $\mathbf{w}_0$. Similarly to Schnorr signature [Sch91], a challenge c is derived from the message being signed and $\mathbf{w}_1$ by using some hash function (modelized as a random oracle in the security analysis) and the signing algorithm eventually computes $\mathbf{z}$ as $\mathbf{y} + c\,s_1$ (where s_1 is part of the signer's secret key) and return the pair $(\mathbf{z}, c)$ as the signature (if it satisfies some security properties). The vector $\mathbf{w}_1$ can be recovered publicly from the signature $(\mathbf{z}, c)$ but the vector $\mathbf{w}_0$ cannot and may actually reveal information on the signer's secret key.

Many works were devoted to present secure implementation of Dilithium signatures against side-channel attacks [MGTF19, RJH+18, MUTS22, ABC+22]. Similarly to the group-based setting or Schnorr/ECDSA signatures, it was advocated that the nonce $\mathbf{y}$ should be protected in order not to leak information on the signing key. Indeed, Mazougui, Ulitzsch, Tibouchi, and Seifert [MUTS22] recently presented an end-to-end profiled side-channel attack on Dilithium targeting the nonce $\mathbf{y}$. They used machine-learning assisted profiling to predict whether a given coefficient in one coordinate of $\mathbf{y}$ is zero or not. Using this information, they can then infer information on the signer's secret key. The importance of the vector $\mathbf{w}_0$ seems more controversial. In [MGTF19], Migliore, Gérard, Tibouchi, and Fouque presented a masked implementation of Dilithium where $\mathbf{w}_0$ was not protected and mentioned that the `Decompose` function is "*by far the most complex operations regarding masking*" in Dilithium. *A contrario*, Azouaoui, Bronchain, Cassiers, Hoffmann, Kuzovkova, Renes, Schneider, Schönauer, Standaert and van Vredendaal have recommended to protect $\mathbf{w}_0$ in [ABC+22].

In this work, we present an efficient (equivalent) key-recovery attack on Dilithium exploiting a leakage on the vector $\mathbf{w}_0$. We design a template attack that enables us to predict whether a given coefficient in one coordinate of $\mathbf{w}_0$ is zero or not. Since the size of coefficients in $\mathbf{w}_0$ are smaller than those in $\mathbf{y}$, this event is more likely than the one used by Mazougui *et al.*, and thus the attack requires theoretically a smaller number of signatures than the one presented in [MUTS22]. The theoretical attack described in Section 3 relies on the simple observation that from a signature where one coordinate of $\mathbf{w}_0$ is null, one can obtain the value of the corresponding coordinate of another secret information used in the signing algorithm. Once this value has been completely reconstructed, one can recover s_1 using linear algebra and s_1 is sufficient to sign arbitrary messages.

To prove the feasibility of our attack in practice for embedded systems, we consider the

C cryptographic library PQClean [KSSW] and use the ChipWhisperer-Lite 32-bit with an STM32f3 micro-controller. With a corpus of 700 000 messages to build our template, we consider three different leakage models on $\mathbf{w}_0$ and provide a comprehensive analysis in Section 4. The construction of our dataset needs less than a day to collect enough representatives for each leakage model plus one more day to perform the traces acquisition on our target. Obviously, in a realistic attack setting, detecting whether some coordinate of $\mathbf{w}_0$ using noisy power traces is not error-free and we provide a technique to filter potential values from signatures and achieve manageable false-positive (0.067%) and false-negative (0.1174%) rates.

2 Background

In this section, we will provide essential background information for a better understanding of the attack.

2.1 Notations

Let us note by $\mathbb{Z}_q$ the ring of integers modulo q and by $\mathbb{Z}_q[X] = (\mathbb{Z}/q\mathbb{Z})[X]$ the set of polynomials with integer coefficients modulo q. We define by $R = \mathbb{Z}[X]\Big/(X^n+1)$ the ring of polynomials with integer coefficients, reduced by the cyclotomic polynomial X^n+1 and $R_q = \mathbb{Z}_q[X]\Big/(X^n+1)$ the ring of polynomials with integer coefficients modulo q, reduced by the cyclotomic polynomial X^n+1.

In the following, all polynomial operations are performed in R_q except if specified otherwise.

For an even (resp. odd) positive integer α, we define $r_0 = r \mod^{\pm} \alpha$ to be the unique element r' in the range $-\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$ (resp. $-\frac{\alpha-1}{2} < r' \leq \frac{\alpha-1}{2}$) such that $r \equiv r \mod \alpha$. For $\alpha \in \mathbb{N}$, we define $r' = r \mod^{+} \alpha$ to be the unique element r' in the range $0 \leq r' < \alpha$.

For an element $w \in \mathbb{Z}_q$, we define $\|w\|_\infty$ as $|w \mod^{\pm} q|$. For an element $w \in R$, i.e., $w = \mathbf{w}_0 + w_1 X + \ldots + w_{n-1}X^{n-1}$, we define $\|w\|_\infty$ as $\max_i \|w\|_\infty$ and we define $\|w\| = \sqrt{\|\mathbf{w}_0\|_\infty^2 + \|w_1\|_\infty^2 + \ldots + \|w_{n-1}\|_\infty^2}$.

Let S_η be all the elements $w \in R$ such that $\|w\|_\infty \leq \eta$ and $\tilde{S}_\eta$ the set $\{w \mod^{\pm} 2\eta : w \in R\}$.

We will denote by $\texttt{Rot}_j(c)$, the rotated coefficient vector of a polynomial c, rotated by j times in an anti-cyclic fashion.

2.2 Dilithium

Dilithium is a post-quantum digital signature proposed by the "Cryptographic Suite for Algebraic Lattices" (CRYSTALS) team. In 2022, NIST selected it as the new primary standard for quantum-safe digital signatures. When writing this paper, the latest specification version is 3.1. It is based on the Fiat-Shamir with aborts principle [Lyu09], and its security can be reduced to the hardness of solving two lattices problems over module lattices, module learning with errors (M-LWE) [LDK+22], and SelfTargetMSIS [KLS18]. It is designed with four main ideas in mind: simple to implement securely, conservative with parameters, minimizing the size of the public key and the size of the signature, and finally, easy to vary security. The essential functions of Dilithium scheme consist of `KeyGen` to generate the key, `Sign` to produce the signature of a message, and `Verify` to verify the signature.

Key Generation. The key generation algorithm is described in Algorithm 1. It generates a matrix $\mathbf{A}$ from a seed ρ via some function `ExpandA` (see [LDK+22]). Afterward, the algorithm samples random secret key vectors s_1 and s_2. Each coefficient of these vectors is an element of R_q with small coefficients of size at most η. Finally, the second part of the public key is computed as $\mathbf{t} = \mathbf{A}\, s_1 + s_2$.

Algorithm 1 KeyGen

1 $\zeta \leftarrow \{0,1\}^{256}$
2 $(\rho, \rho', K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} := \mathrm{H}(\zeta)$ ▷ H instantiated as SHAKE-256
3 $\mathbf{A} \in R_q^{k \times l} := \texttt{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT Representation as $\hat{\mathbf{A}}$
4 $(s_1,\, s_2) \in S_\eta^l \times S_\eta^k := \texttt{ExpandS}(\rho')$
5 $\mathbf{t} := \mathbf{A}\, s_1 + s_2$ ▷ Compute $\mathbf{A}s_1$ as $\texttt{NTT}^{-1}(\hat{\mathbf{A}} \cdot \texttt{NTT}(s_1))$
6 $(\mathbf{t}_1,\, \mathbf{t}_0) := \texttt{Power2Round}_q(t,\, d)$
7 $tr \in \{0,1\}^{256} := H(\rho \,||\, \mathbf{t}_1)$
8 **return** $pk = (\rho,\, \mathbf{t}_1)$, $sk = (\rho, K, tr, s_1, s_2, \mathbf{t}_0)$

Signature. The signing algorithm is described in Algorithm 2. It is constructed by a rejection sampling loop, generating a new signature until it satisfies some security properties. First, the algorithm generates a masking vector of polynomials $\mathbf{y}$ with coefficients less than γ_1. The signer then computes $\mathbf{w} = \mathbf{A}\mathbf{y}$ and sets $\mathbf{w}_1$ to be the high-order bits of the coefficients in this vector. The challenge c is then created as the hash of the message and $\mathbf{w}_1$. The output c is a polynomial in R_q with exactly τ ± 1's and the rest 0's. The potential signature is computed, using s_1, as $\mathbf{z} = \mathbf{y} + c\, s_1$. To ensure that the signature does not leak information about the key and for correctness purposes, lines 13 and 17 execute some bound checks. If one of these verification fails, a new signature candidate is generated.

Signature verification. The verification algorithm is described in Algorithm 3. The verifier computes the high-order bits of $\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 \cdot 2^d$, and accepts if all the coefficients of $\mathbf{z}$ are less than $\gamma_1 - \beta$ provided that c is the hash of the message and $\mathbf{w}_1'$.

Details on Reference Implementation. We can find several differences between the specification of Dilithium [LDK+22] and the various implementations, such as the reference implementation [DKL+22] or the one from the PQClean Library [KSSW]. In this paragraph, we point out a specific detail that will be of interest for the attack. In Algorithm 2 line 12, we compute $\mathbf{r}_0 = \texttt{LowBits}_q(\mathbf{w} - c\, s_2,\, 2\,\gamma_2)$ and then we check whether $\|\mathbf{r}_0\| < \gamma_2 - \beta$. However, in the reference implementation and to optimize the computation, we check whether $\|\mathbf{w}_0 - c\, s_2\| < \gamma_2 - \beta$. If this inequality holds then $\|\mathbf{w}_0 - c\, s_2\|$ is the low part of $\|\mathbf{w} - c\, s_2\|$ according to Lemma 3 in [LDK+22].
Note that $\mathbf{w}_0$ was already computed in the Algorithm 2 line 8, but we did not keep it in memory.

In the following, and if not stated otherwise, we will refer to the PQClean implementation of Dilithium which is slightly equivalent to the reference implementation.

Decompose. To break up an element in $\mathbb{Z}_q$ into their "high-bits" and "low-bits", we can use the function `Decompose`. In fact, we divide r by α to get $r = r_1\, \alpha + r_0$ where $r_0 = r \mod^{\pm} \alpha$ and $r_1 = (r - r_0)/\alpha$.

We give in Figure 1 the C code of the PQCLean `Decompose` function that is implemented in the same way in most of the online libraries available.

Algorithm 2 Sign

$\mathbf{A} \in R_q^{k \times l} := \texttt{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT Representation as $\hat{\mathbf{A}}$
$\mu \in \{0,1\}^{512} := \mathrm{H}(tr \,||\, M)$
$\kappa := 0$, $(\mathbf{z}, \mathbf{h}) := \perp$
$\rho' \in \{0,1\}^{512} := \mathrm{H}(K \,||\, \mu)$ (or $\rho' \leftarrow \{0,1\}^{512}$ for randomized signing)
while $(\mathbf{z}, \mathbf{h}) = \perp$ **do** ▷ Pre-compute $\hat{s}_1 := \texttt{NTT}(s_1)$, $\hat{s}_2 := \texttt{NTT}(s_2)$ and $\hat{\mathbf{t}}_0 := \texttt{NTT}(\mathbf{t}_0)$
  $\mathbf{y} \in \tilde{S}^l_{\gamma_1} := \texttt{ExpandMask}(\rho', \kappa)$
  $\mathbf{w} := \mathbf{A}\,\mathbf{y}$ ▷ $\mathbf{w} := \texttt{NTT}^{-1}(\hat{\mathbf{A}} \cdot \texttt{NTT}(\mathbf{y}))$
  $\mathbf{w}_1 := \texttt{HighBits}_q(\mathbf{w}, 2\,\gamma_2)$
  $\tilde{c} \in \{0,1\}^{256} := \mathrm{H}(\mu \,||\, \mathbf{w}_1)$
  $c \in B_\tau := \texttt{SampleInBall}(\tilde{c})$ ▷ Store c in NTT representation as $\hat{c} = \texttt{NTT}(c)$
  $\mathbf{z} := \mathbf{y} + c\,s_1$ ▷ Compute cs_1 as $\texttt{NTT}^{-1}(\hat{c} \cdot \hat{s}_1)$
  $\mathbf{r}_0 := \texttt{LowBits}_q(\mathbf{w} - cs_2, 2\,\gamma_2)$ ▷ Compute cs_2 as $\texttt{NTT}^{-1}(\hat{c} \cdot \hat{s}_2)$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(\mathbf{z}, \mathbf{h}) := \perp$
  else
    $\mathbf{h} := \texttt{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - cs_2 + c\,\mathbf{t}_0, 2\,\gamma_2)$ ▷ Compute $c\,\mathbf{t}_0$ as $\texttt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{t}}_0)$
    if $||c\,\mathbf{t}_0||_\infty \geq \gamma_2$ or $|\mathbf{h}|_{\mathbf{h}_j=1} > \omega$ **then**
      $(\mathbf{z}, \mathbf{h}) := \perp$
  $\kappa := \kappa + l$
return $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Algorithm 3 Verify

$\mathbf{A} \in R_q^{k \times l} := \texttt{ExpandA}(\rho)$
$\mu \in \{0,1\}^{512} := \mathrm{H}(\mathrm{H}(\rho \,||\, \mathbf{t}_1) \,||\, M)$
$c := \texttt{SampleInBall}(\tilde{c})$
$\mathbf{w}'_1 := \texttt{UseHint}_q(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$
return $[\![||\mathbf{z}||_\infty < \gamma_1 - \beta]\!]$ and $[\![\tilde{c} = \mathrm{H}(\mu \,||\, \mathbf{w}'_1)]\!]$ and $[\![|\mathbf{h}|_{\mathbf{h}_j=1} \leq \omega]\!]$

2.3 Side-Channel Attacks on Dilithium

Concrete security was not the only priority for the NIST PQC competition as for the first time, an emphasis is also put on security against Side Channel attacks. One of the paper's objectives is to state if some parts of Dilithium implementation require specific treatment concerning Side Channel Attacks (SCA). In fact, this section first recalls Template attack principle, which we will use to demonstrate the practicability of our new attack. The remaining of this section will give an overview of the physical attacks already published.

Template Attacks. Side-Channel Attacks have proven to be extremely effective as a practical way to attack cryptographic algorithm implementations. In particular, when open devices are available, with the possibility to set and modify the key value, profiling attacks such as templates can be very powerful. First introduced by Chari, Rao and Rohatgi [CRR03], this attack is divided into four steps. The first step, using a copy of the target device, records a large number of power traces using many different keys and inputs. The second step is creating a template of the device operation by selecting points of

```
int32_t PQCLEAN_DILITHIUM2_CLEAN_decompose(int32_t *a0, int32_t a) {
  int32_t a1;
  a1  = (a + 127) >> 7;
  a1  = (a1 * 11275 + (1 << 23)) >> 24;
  a1 ^= ((43 - a1) >> 31) & a1;

  *a0  = a - a1 * 2 * GAMMA2;
  *a0 -= (((Q - 1) / 2 - *a0) >> 31) & Q;return a1;
  return a1;
}
```

Figure 1: C Code of the `Decompose` function.

interest that are supposed to contain a considerable proportion of the leakage information. Third, the attacker records a small number of power traces using multiple plaintexts on the victim's device. Finally, the template is applied to the attack traces. For each possible subkey, the attacker determines which value is most likely to be the correct one and repeats the same step until the entire key is recovered.

This paper describes the practical realization of a template attack. This is the most effective way to establish the attack on an open platform with an open implementation, but other SCA methods could be applied in practice. Moreover, the efficiency of the template attack is better when the clone (hardware and software) is similar to the target. Since we are working on a public implementation and a public evaluation platform, this also shows that it is the best method to use.

State of the art. Several practical Fault Injection Attacks (FIA) leading to the key recovery have already been published. Indeed Bruinderink and Pessl [BP18] showed the applicability of differential fault attacks on Dilithium using multiple paths. The attacker can produce a correct signature and a faulty signature. By faulting somewhere in the generation of the challenge c, without changing the value of y, the attacker can produce a different variable z under the same number of rejection rounds. From the correct signature, we have $z = y + c\,s_1$, and from the faulty signature, we have $z' = y + c'\,s_1$. Hence, $z - z' = (c - c')\,s_1$ and therefore we can compute $s_1 = (c - c')^{-1}(z - z')$.

Furthermore, Bruinderink and Pessl also showed, as in other papers [RJH+18] [MGTF19], that it is possible to forge signatures with only the extracted portion of the secret key s_1.

Regarding SCA, previous works also have shown that some leakages during the signature could be exploited to infer the whole Dilithium key. For example, Fournaris, Dimopoulos and Koufopavlou [FDK20] demonstrated a correlation power analysis of the polynomial multiplication $c\,s_1$. The analysis focuses on the polynomial multiplication operation during the sample rejection loop, regardless of the technique used (schoolbook polynomial multiplication, sparse matrix polynomial multiplication, or Number Theoretic Transform (NTT) representation).

Also, Kim, Lee, Han, Sim and Han [KLH+20] proposed a machine learning-based profiling attack on unprotected and masked versions of Dilithium-2. They targeted the information gained through load save and reduction operation in Power Consumption traces. They showed that to obtain all the secret key parts in the algorithm, only two of s_1, s_2, $\mathbf{t}_0$ are needed. Then by solving a simple equation, the attacker can find the last one. To validate their attack, the authors targeted the s_1 value and made experiments in a simulated setting where a uniform noise is added to the trace supposing 8-bit Hamming weight leakage and linear regression model. However, it was not implemented on a 32-bit platform.

More recently, Mazougui *et al.* [MUTS22] showed an end-to-end profiled side-channel attack on Dilithium targeting the nonce y. They identified a leak in the generation of the

coefficient of y by using machine learning to distinguish whether the coefficient is zero or not. Their attack extracts information about the vector y from a single trace. Given coefficient $y_i = 0$ then $z_i = 0 + (c\,s_1)_i$. Since z and c are known at the end of the signature, collecting many of these equations allows the attacker to recover all s_1 coefficients using linear algebra.

3 A new profiling attack on Dilithium

This section presents a new theoretical path of profiling attacks on Dilithium. The prerequisite is a sufficiently strong leakage of the value $\mathbf{w}_0$. To our knowledge, this is the first complete attack exploiting a leakage on $\mathbf{w}_0$, even if Azouaoui *et al.* already pointed out that protecting $\mathbf{w}_0$ would be a good recommendation [ABC+22]. For the attack's success, we propose some methods to minimize the error during the matching phase so that we introduce as few errors as possible in our equations. We then show how the system of equations can be solved using the Least Squares Method (LSM). If needed anyway, we propose an error management technique, which computes the correct secret key. Finally, we highlight the number of signatures required to perform this attack and multiple vulnerable operations in the reference implementation that may render the attack possible.

3.1 Main Idea

Our goal is to recover the secret key s_1 because knowing s_1 is sufficient to sign arbitrary messages [RJH+18], [MGTF19] and [BP18]. The attack consists in collecting signatures for which one coefficient of $\mathbf{w}_0$ is known (from profiling attacks) to be equal to 0. As discussed at the end of Section 2.2, $\mathbf{w}_0$ value is computed in the reference implementation without any particular protection.

Since $\mathbf{z} = \mathbf{y} + c\,s_1$ and $\mathbf{t} = \mathbf{A}\,s_1 + s_2$ we have

$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t} = \mathbf{A}\,\mathbf{y} - c\,s_2. \tag{1}$$

By replacing $\mathbf{w} = \mathbf{A}\,\mathbf{y}$ and $\mathbf{t} = t_1\,2^d + \mathbf{t}_0$ in Equation (1), we get

$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - c\,\mathbf{t}_0 = \mathbf{w} - c\,s_2,$$

and by rewriting this equation, we obtain

$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d = \mathbf{w} + c\,(\mathbf{t}_0 - s_2).$$

Since we get from the `Decompose` function that $\mathbf{w} = \mathbf{w}_1\,2\,\gamma_2 + \mathbf{w}_0$ and if we suppose that for some (i, j), $(\mathbf{w}_0)_{i,j} = 0$, then we have

$$\left(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d\right)_{i,j} = \left(\mathbf{w}_1\,2\,\gamma_2 + c\,(\mathbf{t}_0 - s_2)\right)_{i,j}. \tag{2}$$

Given that the elements $\mathbf{A}$, $\mathbf{z}$, c, $\mathbf{t}_1$, d, γ_2 are public and that $\mathbf{w}_1$ can be computed from the verification part, we can recover $(\mathbf{t}_0 - s_2)_{i,j}$ from Equation (2). We repeat the same step for all i and j such that $0 \leq i < K$ and $0 \leq j < n$ in order to fully recover $\mathbf{t}_0 - s_2$. Finally, the knowledge of $\mathbf{t}_0 - s_2$ allows to find s_1:

$$\mathbf{t} = \mathbf{A}\,s_1 + s_2 = \mathbf{t}_1\,2^d + \mathbf{t}_0,$$

hence,

$$\mathbf{A}\,s_1 = \mathbf{t}_1\,2^d + (\mathbf{t}_0 - s_2). \tag{3}$$

However, $\mathbf{A}$ is not square in all the Dilithium versions, but if we multiply the Equation (3) by $\mathbf{A}^t$, we get that $(\mathbf{A}^t\,\mathbf{A})$ is square and is invertible with very high probability [ABC+22]. Thus, we get

$$s_1 = (\mathbf{A}^t\,\mathbf{A})^{-1}\,\mathbf{A}^t\,\left(\mathbf{t}_1\,2^d - (s_2 - \mathbf{t}_0)\right).$$

Knowing s_1 suffices to sign arbitrary messages [RJH+18].

Remark. Equation (2) holds if $\mathbf{w}_0 = 0$. The attacker thus needs to distinguish signatures for which $\mathbf{w}_0 = 0$ from those where $\mathbf{w}_0 \neq 0$. We will denote by `Classifier` this procedure.

3.2 Candidates Filtering

In practice, there might be some noise when measuring a signal. This may lead to a wrong prediction that $(\mathbf{w}_0)_{i,j}$ is equal to 0, with introductions of errors in our system of equations. In order to overcome this, we need to remove the values where $(\mathbf{w}_0)_{i,j} \neq 0$ as early as possible to reduce the number of wrong equations. Hence, we put a filter on the value of $\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - \mathbf{w}_1\,2\,\gamma_2$.

Under the assumption that $(\mathbf{w}_0)_{i,j} = 0$ for some (i, j) and from Equation (2) we have

$$\left(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - \mathbf{w}_1\,2\,\gamma_2\right)_{i,j} = \left(c\,(\mathbf{t}_0 - s_2)\right)_{i,j}.$$

Since $(s_2)_{i,j} << (\mathbf{t}_0)_{i,j}$,

$$\left(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - \mathbf{w}_1\,2\,\gamma_2\right)_{i,j} \approx (c\,\mathbf{t}_0)_{i,j}.$$

According to the specification of Dilithium [LDK+22], we can assume that the low order bits are uniformly distributed, i.e., $\mathbf{t}_0 \sim \mathcal{U}([-2^{d-1} + 1,\ 2^{d-1}])$. Therefore, since c is a vector with exactly τ coefficients in $\{-1,\ 1\}$ and the rest equal to 0, by the central limit theorem, $c\,\mathbf{t}_0$ is normally distributed with mean 0 and standard deviation σ, i.e.,

$$\begin{cases} c\,\mathbf{t}_0 & \sim \mathcal{N}(0, \sigma^2) \\ \sigma^2 & = \dfrac{(2^{d-1} - (2^{d-1} + 1) + 1)^2 - 1}{12}\,\tau = \dfrac{2^{2d} - 1}{12}\,\tau \end{cases}$$

Hence, the probability that $|(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - \mathbf{w}_1\,2\,\gamma_2)_{i,j}|$ is lower or equal to $2\,\sigma$ is equal to 0.954.
Therefore, if we assume that $(\mathbf{w}_0)_{i,j} = 0$ then,

$$|(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - \mathbf{w}_1\,2\,\gamma_2)_{i,j}| \leq 2\,\sigma. \tag{4}$$

For instance, in Dilithium-2, we have $2\,\sigma = 2\sqrt{\dfrac{2^{26} - 1}{12}\,39} \approx 29\,537$. Experimentally, we can discard approximately 70% of the $N := n \times K = 1024$ values where we might not have zero.

Using this filter, we can remove equations where we actually have values of $(\mathbf{w}_0)_{i,j} = 0$. However, in case of the presence of noise, it might be preferable to perform more signatures rather than introducing errors in the system of equations.

3.3 Resolution - Least Square Method

Let us write $e = (\mathbf{w}_0)_{i,j}$. If our classifying process manages to detect zero values with enough precision, then this coefficient vanishes, and we get Equation (2). On the other hand, if our classifier is wrong and tells us that the value of $(\mathbf{w}_0)_{i,j} = 0$, but in fact, we do not have a zero, then we will get samples of the form

$$\left(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d\right)_{i,j} = \left(\mathbf{w}_1\,2\,\gamma_2 + c\,(\mathbf{t}_0 - s_2)\right)_{i,j} + e,$$

where e is the error. Note that for most of our equation $e = 0$. This equation can be rewritten as

$$\left(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^d - \mathbf{w}_1\,2\,\gamma_2\right)_{i,j} = \left(c\,(\mathbf{t}_0 - s_2)\right)_{i,j} + e. \tag{5}$$

Let L be the left-hand side of Equation (5) for all $(i,\ j)$, and let $\mathbf{C}$ be the matrix of $c_{i,j}$ for all $(i,\ j)$. Since $\|\mathbf{C}\,\mathbf{t}_0\| \leq \tau 2^{d-1}$, $\|\mathbf{C}\,s_2\| \leq \beta$ and $\|e\| \leq 2\,\gamma_2$, then $\|\mathbf{C}\,(\mathbf{t}_0 - s_2) + e\| < q$,

and thus we do not have a modular reduction. Therefore, this problem can be viewed as an LWE without modular reduction [BDE+18]. We use some statistical techniques to find an approximate solution of $(\mathbf{t}_0 - s_2)$ from the noisy linear system formed by equation of the form (5). Using the Least Squares Method described by Bootle *et al.* in [BDE+18], we get a candidate solution

$$(\mathbf{t}_0 \tilde{-} s_2) = (\tilde{C}^T \tilde{C})^{-1} \tilde{C}^T L,$$

where $\tilde{C}^T$ is the matrix where for each vector i, each line j is comprised of the rotated corresponding challenge $c_{i,j}$.
If $\|(\mathbf{t}_0 - s_2) - (\mathbf{t}_0 \tilde{-} s_2)\|_\infty < \frac{1}{2}$, then $\lceil (\mathbf{t}_0 \tilde{-} s_2) \rfloor = (\mathbf{t}_0 - s_2)$. We get that most of the coefficients of $(\mathbf{t}_0 \tilde{-} s_2)$ are correct and the rest are wrong by ± 1, so either rounding up or down should yield the correct solution. Let us denote by `LSM` this procedure.

3.4 Error Management

It is important to notice that in most cases, the error term in our equations is equal to zero under the assumption that our classifier is correct in most instances. This means the solution may result from a "noisy" set of equations, i.e., with a small number of incorrect equations. Several techniques that can recover the correct value from the given perturbed set of equations are described hereafter.

LWE with side information. With the "noisy" equations, our system looks like the LWE problem. Dachman-Soled, Ducas, Gong and Rossi [DDGR20] proposed a framework to integrate SCA "hints" to solve a LWE instance. In order to use hints from side channel information, the learning with errors with side information instance is first transformed into a distorted bounded distance decoding (DBDD) instance. This transformation allows us to track the distribution of the secret vector. By providing hints, we can potentially modify this distribution to make it easier to find the solution vector. If we give enough hints, the secret vector recovery becomes feasible using lattice reduction attacks.

ILP. One can also see the problem of finding a given polynomial from this set of noisy equations as identifying the polynomial that maximizes the number of equations. In other words, we want to find the polynomial that fits most equations in our system. To this end, Marzougui *et al.* [MUTS22] formulated an Integer Linear Program (ILP) to solve this problem. Using the solution candidate obtained from the least-squares method, the authors use the big M [BJS11] method to factor in the noisy equations. Ultimately, the solution obtained from the ILP must correspond to the correct value.

Majority judgment. If we want the probability, of finding all $n \times K$ coefficients 3.5, to be equal to 1 then we will necessarily have some couples (i, j) where $(\mathbf{w}_0)_{i,j}$ will be equal to 0 several times. Moreover, our distinguisher is incorrect on one coefficient at most, by taking two different equations, there is a small chance that it is wrong on the same coefficient. Since we are doing the LSM just before this step, we will only have some rounding problems, i.e., we will be very close to the exact value. By solving several LSM with different equations, we can perform a majority judgment on each coefficient (i, j) to find the correct intermediate value.

3.5 Attack Summary

Pseudo-Algorithm 4 describes all steps to find a solution candidate $(\mathbf{t}_0 - s_2)$.

The number of signatures needed is an important performance indicator of an SCA. The purpose of this subsection is to find out how many signatures are required for our attack to

Algorithm 4 Equivalent Secret Key Recovery

$L = [\,[0] \times N] \times K,\ \tilde{C} = [\,[0] \times N] \times K$
for $m = 0$ to $M - 1$ **do**
  for $i = 0$ to $K - 1$ **do**
    for $j = 0$ to $n - 1$ **do**
      $z, c, h \leftarrow m$
      $val = (\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^d - \mathbf{w}_1 2\gamma_2)_{i,j}$
      if $|val| \leq 2\sqrt{\frac{2^{2d}-1}{12}\tau}$ **then**
        $(\tilde{\mathbf{w}}_0)^m_{i,j} := \texttt{Classifier}(T_{(\tilde{\mathbf{w}}_0)^m_{i,j}})$
        if $(\tilde{\mathbf{w}}_0)^m_{i,j} = 0$ **then**
          $L[i][j] = val$
          $\tilde{C}[i][j] = \texttt{Rot}_j(c)$
for $i = 0$ to $K - 1$ **do**
  $\mathbf{t}_0\_minus\_s_2 = \texttt{LSM}(L[i])$

recover the $N = n \times K$ coordinates, where $(\mathbf{w}_0)_{i,j} = 0$ for all i and j such that $0 \leq i < K$ and $0 \leq j < n$. We know that for a fixed (i, j), $(\mathbf{w}_0)_{i,j} = 0$ with probability $\frac{1}{2\,\gamma_2}$. Hence, one coordinate is not equal to zero with probability $1 - \frac{1}{2\,\gamma_2}$. Over T experiences, a coordinate is not equal to zero with probability $(1 - \frac{1}{2\,\gamma_2})^T$. Hence, a coordinate is equal to zero at least once with probability $1 - (1 - \frac{1}{2\,\gamma_2})^T$. Therefore, over T experiences the n coordinates are equal to zero at least once with probability $\left(1 - \left(1 - \frac{1}{2\,\gamma_2}\right)^T\right)^n$.

Example 1. For Dilithium-2, $\gamma_2 = 95\,232$. In order to recover the $N = 256 \times 4 = 1024$ coordinates where $(\mathbf{w}_0)_{i,j} = 0$, we need 7 129 039 signatures. In this case, we get the probability of recovering all the coefficients equal to 1. Note that in practice, for Dilithium-2 2.5 millions of signatures were needed to recover the $N = 256 \times 4 = 1024$ coordinates where $(\mathbf{w}_0)_{i,j} = 0$ with high probability. Thus this is consistent with our analysis because we can see from the graph in Figure 2 that the probability is 0.9996 when we are around 2.8 million signatures.

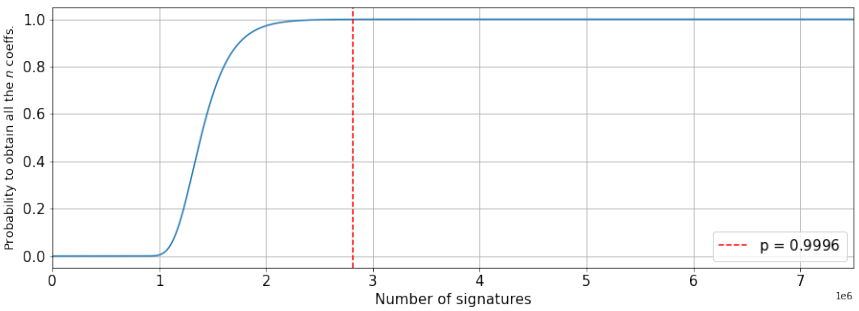

Figure 2: Probability of having recovered all the $N = n \times K$ coordinates for Dilithium-2.

3.6 Potential Leakage Spots

This section discusses potential leaking operations in Dilithium.

Probably the most intuitive location is the `Decompose` function line 8, where $\mathbf{w}$ value is split into its upper and lower parts. This operation is sensitive as the $\mathbf{w}_0$ value is directly used in various locations, especially in the reference implementation in C. Previous work [MGTF19] has already shown that this sub-routine leaks information, but $\mathbf{w}_0$ was not identified as a sensitive value.
In the same way, we performed Welch's T-test, using the fixed vs. random approach on the `Decompose` function (see Figure 3) to localize potential leakages on our implementation target.

Furthermore, the computation of the $\mathbf{r}_0$ line 12 Algorithm 2 value performed in the normal domain by subtracting cs_2 to $\mathbf{w}_0$ is also a good candidate for an attack. If the targeted value is zero, we should have an apparent leakage in Hamming weight (HW). Even though it is a promising operation, this is let for future practical work.

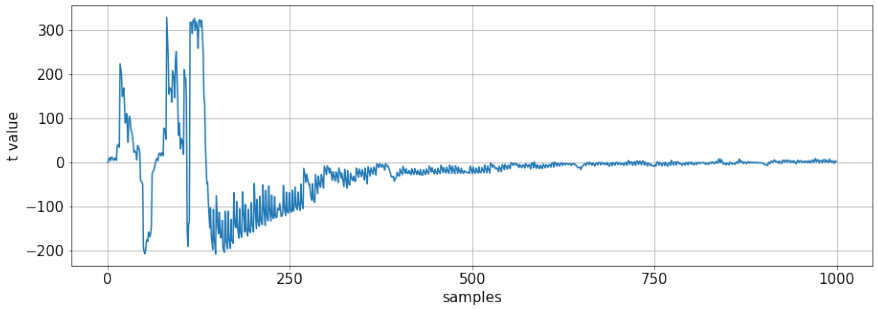

Figure 3: Welch T-test for `Decompose`.

As shown in Figure 3, we can observe clear peaks, which confirm that this operation is sensible.

4 Experimental Results

This section demonstrates the practical feasibility of the theoretical attack when no specific countermeasure is implemented. All side-channel analyses were coded in Python and will soon be available as notebooks. Practical results are shown here only for the first coefficient of the first coordinate of w_0. Since the implementation manipulates each coefficient independently in the same way, the presented methodology and results can be transposed accordingly for all other indexes and coordinates of w_0.

4.1 Setup

For our attack, we used the Post-quantum cryptographic library PQClean [KSSW], which offers "clean" standalone C implementations [KSSW] of most post-quantum schemes included in the NIST round 3 and 4 submissions [nis]. Focusing mainly on the embedded system domain, the ChipWhisperer-Lite 32-bit, with an STM32f3 micro-controller, is often used in related academic research. We chose it as a test bench to deploy the Dilithium-2 signature scheme. We compiled Dilithium-2 using the gcc-arm cross-compiler arm-none-eabi-gcc 10.2.1 (with the options -O3).

To go to the nitty-gritty, we use the ChipWhisperer (CW) "cw" library [OC14] for trace acquisition, and we use an open-source Side Channel python library to process the

side-channel information and perform the desired analysis. We also use the LSM and the rounding subroutines implemented in the "numpy" library [HMvdW+20]. We also used the C reference implementation of Dilithium [DKL+22] running on an Intel Core i7 to determine the number of coefficients we discarded with our filter and the number of coefficients $\mathbf{w}_0 = 0$ not identified.

4.2 Learning Phase

For the learning phase, the Points of Interest (POIs) selected in the previous section were used to generate the profiles according to the leakage model. The goal is to characterize the consumption leakage of the decompose function that was identified in Subsection 3.6 as being crucial in the computation of $\mathbf{w}_0$.

4.2.1 Reverse engineering

```
1 for (i = 0; i < K; ++i) {
2   trigger_high();
3   PQCLEAN_DILITHIUM2_CLEAN_poly_decompose(&v1→vec[i],&v0→vec[i],&v→vec[i]);
4   trigger_low();
5 }
```

Figure 4: PQClean Dilithium2 signature generation code snippet.

As mentioned earlier, using the PQClean library's default implementation, the executed code can be viewed in Figure 4, where the trigger signal is active throughout the whole four `poly_decompose` calls.

The ChipWhisperer Lite has limited space capacity and can only handle a maximum of 24 400 samples, making it challenging to identify essential features in traces or capture longer operation sequences such as an entire Dilithium signature. To overcome this, we first decided to use a Lecroy WavePro HD oscilloscope to locate the chosen leakage spot and determine if we could make the acquisition directly on the ChipWhisperer.

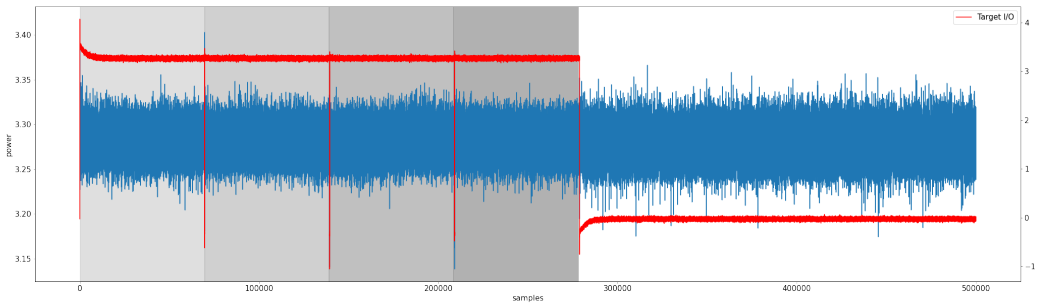

Figure 5: `poly_decompose`(w_i) $0 \leq i < K = 4$.

A single `poly_decompose` call in the Dilithium signature generation trace is presented in Figure 5, where the blue plot represents the power consumption across time. The red plot is the I/O trigger, marking the beginning and end of the `poly_decompose` process. It is easy to observe the different sequential `Decompose` (total of K) executed during a single round of the signature in Dilithium. Let us now focus on a single one.

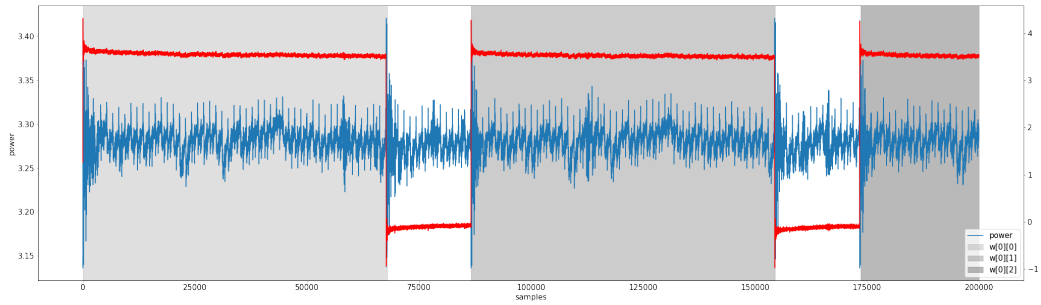

Figure 6: Zoom on `Decompose`(w[0][j]), for j $\in \{0, 1, 2\}$.

In Figure 6, we zoomed on the first three `Decompose` with the oscilloscope. Once again, the red plot is the trigger that allows us to clearly differentiate the location of each `Decompose` calls on the $w_{i,j}$. We can see a repeating pattern.

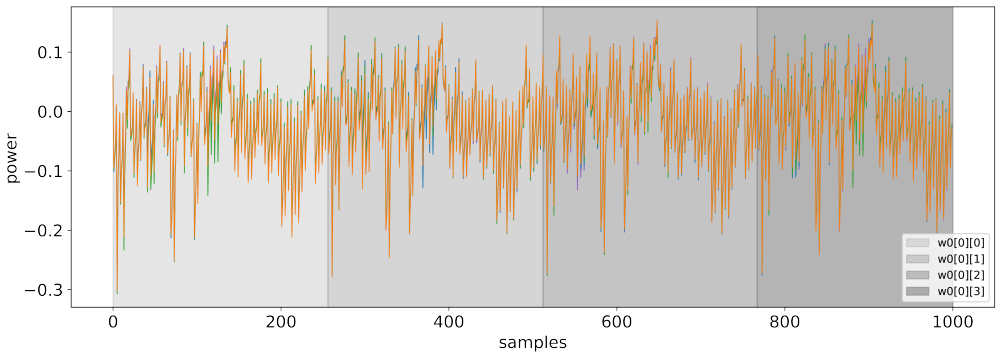

Figure 7: CW `Decompose`(w[0][j]), for j $\in \{0, 1, 2, 3\}$.

In Figure 7, we zoomed on a single decompose and performed the same acquisition directly on the ChipWhisperer. We highlight the four decompositions through the four shades of gray, where each coefficient of the polynomial w_i is decomposed. We can see a repeating pattern that spans for about 260 samples, corresponding to the calls to the `Decompose` function.

Each polynomial of the vector $\mathbf{w}$ is independent as well as each value. Without loss of generality, we can suppose that by creating a template on the first coefficient of $\mathbf{w}_0$, we can apply it to the other coefficients. Moreover, one of the particularities of the Dilithium signature scheme is the rejection sampling. This can increase the complexity of the profiling step of the template. To overcome this difficulty, we decided only to consider the very first decompose of the first round to build our template, even if more rejection rounds were computed.

4.2.2 Leakage models and $\mathbf{w_0}$ leakage detection

Since we are now able to identify the `Decompose` consumption profile, we will now focus on the leakage inferred by $\mathbf{w}_0$ and particularly when its value is equal to 0. To achieve this, we perform a reverse analysis of the intermediate value $\mathbf{w}_0$ for different Models and explain in detail how we constructed the dataset used to perform the corresponding analyses. For this purpose, we built a profiling data set with 700 000 traces collected from the CW.

Selection function. By default, the traces collected from the CW are easy to process as they do not require any synchronization. We know how the `Decompose` function works, and we can identify the inputs corresponding to the fixed outputs, so we choose the identity of $(\mathbf{w}_0)_{i,j}$ as a selection function to determine the intermediate values which will be targeted by our attack.

Distinguishers. As a second step of our analysis, we apply some statistical tools that are widely used when building profiled attacks. We choose to test the ANOVA [YJ21], NICV [BDGN14], SNR [FDLZ14], and CPA [BCO04] to find which one gives the best results in our study case. Based on the previous reverse analysis and the code review of the `poly_decompose` function implementation presented in Figure 1, we identify three relevant leakage models to be applied to our selection function $\mathbf{w}_0$: Model A, B and C.

Leakage model A. With this model, we consider $\mathbf{w}_0$ as four independent bytes and perform a reverse analysis on the value of each byte value separately, which leads to a 256 partitions analysis of each byte value. We know that for all $0 \leq i < K$ and $0 \leq j < n$, $-\gamma_2 < (\mathbf{w}_0)_{i,j} \leq \gamma_2$. In our study case, for Dilithium-2, $\gamma_2 = (q-1)/88 = 95\,232$, which is `00 01 74 00` in hexadecimal, so if $(\mathbf{w}_0)_{i,j} \geq 0$ it results that $(\mathbf{w}_0)_{i,j}$ is bounded by this value. On the other hand if $(\mathbf{w}_0)_{i,j} < 0$ we have that $(\mathbf{w}_0)_{i,j}$ is between `FF FF FF FF` and `FF FE 8C 02`.

While only two classes are possible for the most significant byte (MSB), which is easily distinguishable, there are four possible groups for the second MSB. It can be trickier to distinguish between the cases `00` and `01` or the cases `FF` and `FE`.

As we will be performing the template only on the filtered values, we know that $|(\mathbf{w}_0 - c\,s_2 + c\,\mathbf{t}_0)_{i,j}| < 2\sigma$, but because $|c\,s_2| \leq \beta$ and $|c\,\mathbf{t}_0| < 2\,\sigma$ then it must hold that $|(\mathbf{w}_0)_{i,j}| < 2\,\sigma + \beta + 2\,\sigma$. Now, if $(\mathbf{w}_0)_{i,j} \geq 0$ the value is between `00 00 00 00` and `00 00 E7 0E` and if $(\mathbf{w}_0)_{i,j} < 0$ it is between `FF FF FF FF` and `FF FF 18 F2`. There are only two classes possible now for the two MSBs. Therefore, for Dilithium-2, the template attack can be simplified on the last two bytes to determine if we have a zero value which can potentially speed up the attack. However, this simplification does not hold for both Dilithium-3 and Dilithium-5.

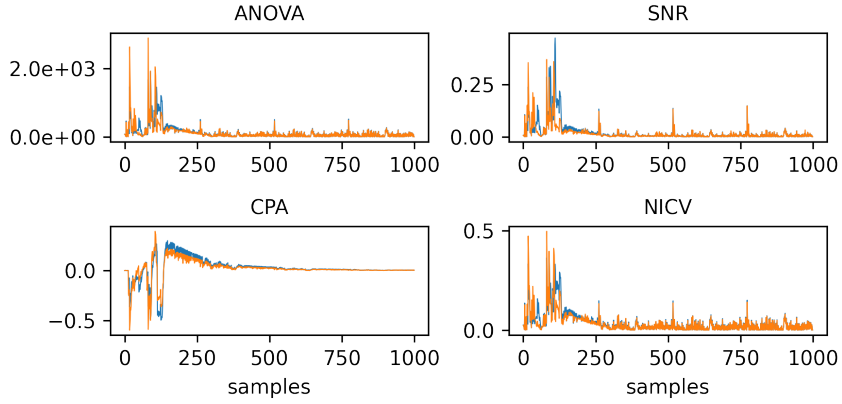

Figure 8: Reverse analysis of the Leakage model A.

In Figure 8, we have plotted the results for the four statistical indicators cited previously. We can observe that they give essentially the same information. Clear leakage peaks can be seen from samples 0 to 250 at roughly the same position. This will be useful later on for selecting the Points of Interest.

Leakage Model B. Here we will consider the same byte decomposition of $\mathbf{w}_0$ from the Model A. The Hamming weight model is used instead of the byte value model, leading to an analysis of 9 partitions instead of 256. To classify $\mathbf{w}_0 = 0$ by value or by Hamming weight amounts to the same, but statistically, in the HW model, values different from 0 will be concentrated in the same classes. For instance, values 1 and 3 will have the same HW of 1. Model B aims to determine whether the information will stand out more.

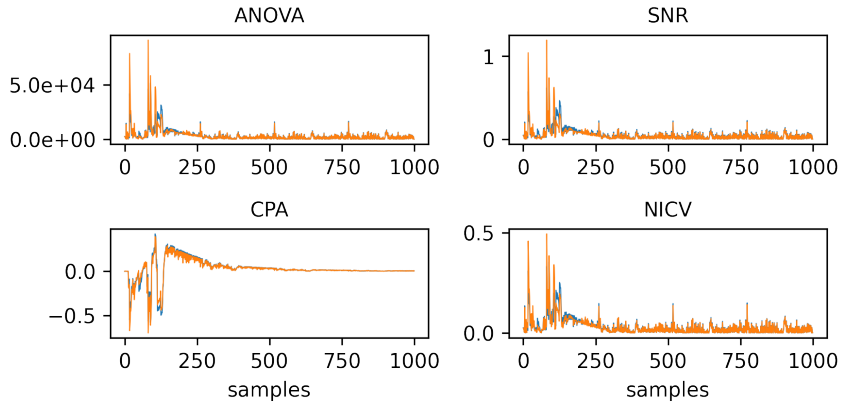

Figure 9: Reverse analysis of the Leakage model B.

We obtain results that are likely the same as the Model A, confirming that the operation we targeted leaks information at the early stage of the power traces of the `Decompose` function.

Leakage Model C. With this model, we consider the Hamming weight of the entire $\mathbf{w}_0$ word, which leads to an analysis of 33 partitions. We consider this model because our CW is 32-bits based.

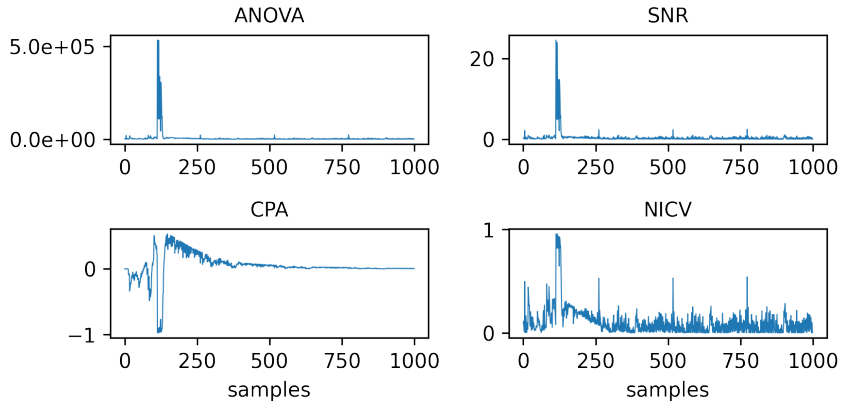

Figure 10: Reverse analysis of the Leakage model C.

Most of the information seems to be located around sample 125 which is still accurate with previous analyses.

Building the profiling data set. The number of traces used to build the templates is large enough regarding the strong leakage of our hardware. However, we must consider

having as many occurrences as possible for each class for the three leakage models. For the leakage model A and B, each byte value or Hamming weight of the byte value has a significant representation when building the data set with random messages. For Model C, we need to compute many signatures offline before finding messages that produce enough of the 33 possible values of $HW(\mathbf{w}_0)$, mainly on the borders of the distribution of this selection function. To do so, we used the C reference implementation, and we coded a script that performs Dilithium signatures for a given security parameter until there is a significant number of representatives for each class. We have set the number of messages per class to 21 000. Then we complete with random messages to have a data set of 700 000 messages. This amounts to a total of approximately 18 hours to collect the necessary messages corresponding to the 33 partitions. Finally, we also need 24 more hours to perform the corresponding acquisitions with the CW.

4.2.3 POIs selection for templates

The templates must be created using a reduced number of samples in each trace due to mathematical computation complexity. To do so, we keep only a few samples empirically from the distinguisher's results. We select the ANOVA distinguisher for this selection because it works with data partitioning, which is very similar to the templates partitioning mechanism and gives good results for all three leakage models.

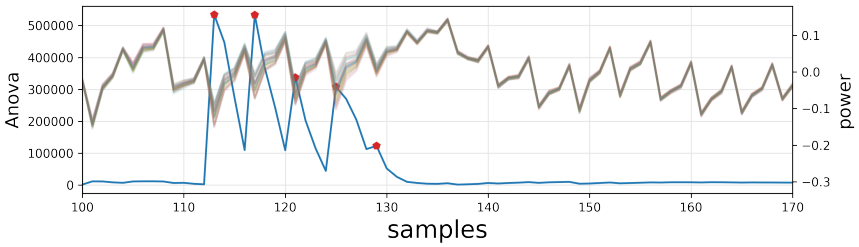

Figure 11: POIs selection for Leakage Model C.

Figure 11 illustrates our selection of the leakage peaks as Points of Interest to keep the best indices in our power traces for the model C. We have also plotted some power traces in the background, where we can see significant variability in the exact locations compared to the rest of the samples between different signature executions.

4.3 Matching Phase

We now validate our templates created using the POIs previously selected. For this purpose, we create a matching data set of 1000 traces collected from the CW in around 2 minutes. The data set is built with a targeted intermediate value fixed to $\mathbf{w}_0 = 0$ for all traces. For our experiments, we do not consider cross-device template matching as we use the same target to build the profiling data set used for matching.

The results of our experiments are shown in Figures 12 to 18. For each leakage model A, B, and C, it appears that the best template score corresponds to our targeted value $\mathbf{w}_0 = 0$. We also observe the evolution of the scores showing that our value is clearly distinguishable from the others right from the first trace prediction.

Divide and conquer $\mathbf{w}_0$ LSBs (model A and model B). The Figures 12 to 15 show that the template matching guesses correctly the first and the second LSB of $\mathbf{w}_0 = 0$, and the results are shown for 10 traces. Furthermore, we can see that the corresponding score for the correct guess is clearly distinguishable from the rest. As stated previously, the score

for the correct guess is already detectable with one trace. Still, surprisingly enough, the score does not seem to improve dramatically with the number of traces added.

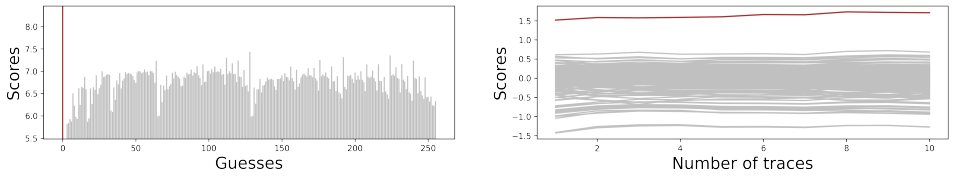

Figure 12: Model A matching value LSB 0.

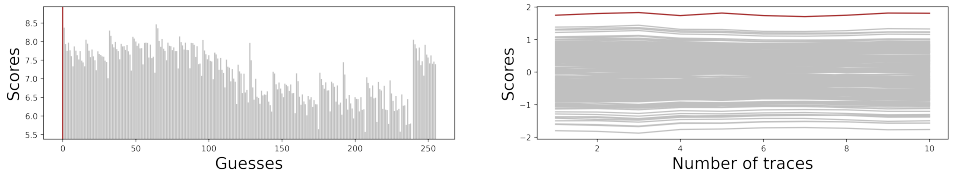

Figure 13: Model A matching value LSB 1.

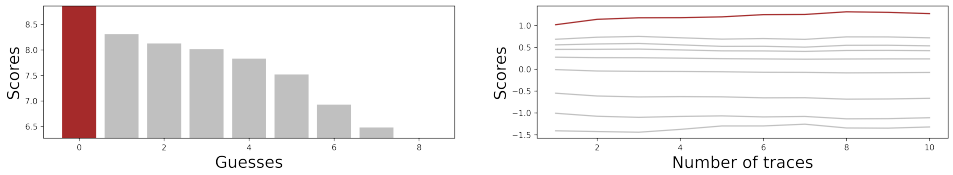

Figure 14: Model B matching HW LSB 0.

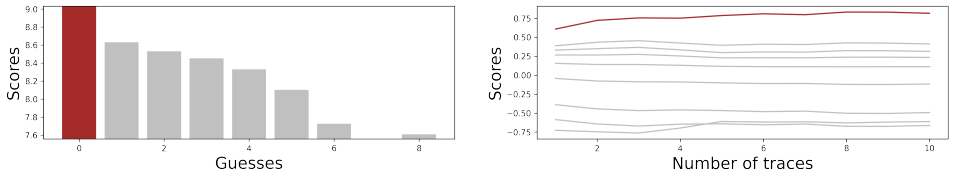

Figure 15: Model B matching HW LSB 1.

Divide and conquer $\mathbf{w_0}$ MSBs (model A and model B). For completeness and to show what a potential matching could be for the versions 3 and 5 of Dilithium, we decided to study also the MSB of the targeted value. From the leakage Model A description, we know that without the filtering step only two templates are required for the first MSB of $\mathbf{w}_0$ and four templates for the second MSB in the case of Dilithium-2. Figures 16 and 17 show that the template matching guesses correctly the first and the second MSB of $\mathbf{w}_0 = 0$.

Even though the two MSBs are correctly matched, we can see that the scores of the second MSB for values 0 and 1 from Model A and B are really close to each other. It could be challenging to distinguish them from a weaker leakage. This raises the importance of the filter step previously to the leakage assessment.

Template matching on word $\mathbf{w_0}$ (model C). The Figure 18 shows a successful matching result for the Hamming weight model applied on the whole word $\mathbf{w}_0$.

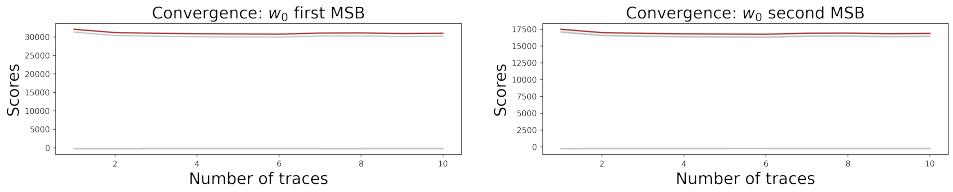

Figure 16: Model A matching HW MSB 0 and 1.

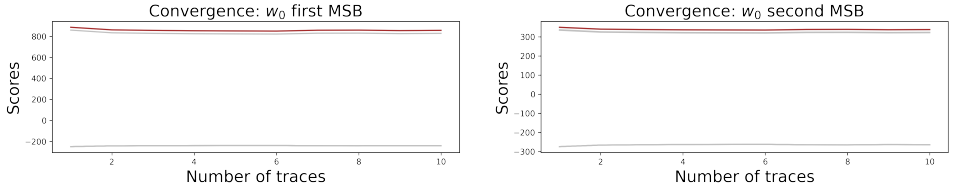

Figure 17: Model B matching HW MSB 0 and 1.

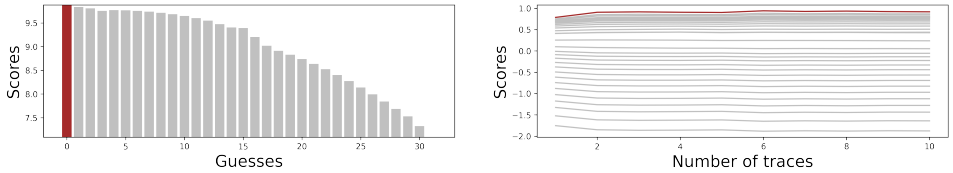

Figure 18: Model C matching value HW 32 bits value.

4.4 Results Interpretation

In our analysis of the three models, we found that Model C required more traces to have a sufficient number of representatives for each class, particularly for HW ≤ 3, between 13 and 19 (because of the difference between positives and negatives values) and ≥ 30. This resulted in a substantially high number of signatures to construct our building data set. There was no significant difference between Models A and B. For both of these Models, it could be challenging to differentiate close values such as `00` and `01` or `FF` and `FE`. To speed up the attack, we have implemented a filter that increases the speed of the attack up to two times when focusing on the last two bytes, but this only applies to Dilithium-2. In our experiments, with over 10 000 000 signatures, the filter discarded approximately 69.3% of the 1024 coefficients, leaving on average only 314 coefficients that could potentially be equal to 0. Additionally, we evaluated the number of true $(\mathbf{w}_0)_{i,j} = 0$ discarded by the filter and found that it was approximately 94.66%, which is close to the 95.4% theoretically expected.
To test the false positives, we decided to perform 100 000 signatures with 10 traces each time to complete the matching. In the end, we found that we have 67 false positives, i.e., when we wrongly predict a coefficient $\mathbf{w}_0 = 0$. We also tested for false negatives with our template. With the same number of traces as for the matching, we evaluated 50 000 messages with fixed intermediate value $\mathbf{w}_0 = 0$. We observe that we have 87 coefficients not detected. Combined with the values rejected by the filter, this amounts to acquiring more signatures before the attack.

4.5 Modularity and further steps

Depending on the setup used, one can focus on a given model according to the corresponding leakage assessment or the version of Dilithium. Also, depending on the equipment at

disposition, we can choose to use the filter or not. With few samples, we do not capture the totality of the decompositions, so we will need to make several acquisitions. For instance, as in our case with the CW, we can at most capture 95 coefficients on the $n \times K$ needed. Therefore, we will need to do at least 11 set of acquisitions per message to capture the entire coefficients. So by selecting which coefficients to focus on beforehand, one can limit the number of acquisitions to perform. The next step in the process, given a set of positions where $\mathbf{w}_0 = 0$ and corresponding signatures, is to use the Algorithm 4 to perform the Least Squares Method (LSM) resolution to find a solution candidate of $\mathbf{t}_0 - s2$. We also have to ensure that the values are signed for the LSM to solve the system correctly. With our implementation in Sage, the whole Algorithm 4 takes a little more than 2 minutes. We use the majority judgment technique to find simply and efficiently the accurate $\mathbf{t}_0 - s2$, as shown previously in 3.4. This process is repeated several times with different equations. For example, during our experiments, we repeated it 3 times, for a duration of 78 seconds each and a majority vote was taken on each position (i, j) to find the correct $\mathbf{t}_0 - s_2$. We did not use the framework presented in [DDGR20] given that no implementation of a solver using the LWE with side information was available. Still, it could be interesting to compare the performances of this solution compared to ours. Finally, we can recover the secret key s_1. To do so, we first convert the matrix A back into the normal domain using the predefined functions in the Dilithium library. Then we solve Equation (3.1). One way to check if we have the right s_1 is to ensure it is in the correct interval $[-\eta, \eta]$.

5 Discussions and Countermeasures

In this paper, we have practically shown that exploiting the leakage of $\mathbf{w}$ during the decomposition can lead to the full recovery of the secret key s_1, which is sufficient to generate valid signatures. In their previous work, Migliore *et al.* [MGTF19] showed a masked version of Dilithium that does not mask the `Decompose` operation in line 8 of Algorithm 2. However, Azouaoui *et al.* recommended that it is an important operation to mask.

In the absence of any countermeasure, implementations using signed values may magnify some exploitable leakage, especially since the signed representation is the most efficient to implement on an ARM Cortex-M4 [GKS20]. For instance, the subtraction $\mathbf{w}_0 - c\,s_2$ with positive coefficients of $\mathbf{w}_0$ and $c\,s_2$ would suddenly generate a negative number when, for these specific coefficients, $\mathbf{w}_0 < c\,s_2$. This strong Hamming distance may give some hints to the attacker to classify very small values of $\mathbf{w}_0$. One way to prevent attacks is to minimize potentially vulnerable operations in the algorithm. The computation $\mathbf{w}_0 - c\,s_2$ is a possible source of leakage, but it can be eliminated using the standard method to compute the $\mathbf{r}_0$ value. However, without further analysis, it's uncertain if this solution will not create new leakage during the calculation of the $\mathbf{w} - c\,s_2$ value or even simply have an overhead to high.

There exist simple known countermeasures. For example, shuffling and masking can easily make this side-channel attack rather impossible in practice when correctly implemented.

Shuffling. Shuffling the manipulation of coefficients for all identified sensitive values $\mathbf{w}_0$ during `Decompose` and during the calculation of $\mathbf{r}_0$ may render the attack very complex. Indeed during the matching phase, the attacker needs to link the observation that one coefficient of $\mathbf{w}_0$ equals zero, with the related indexes i and j for this zero coefficient. This operation is needed to mount the correct system of equations and thus for the attack's success. The calculation of $\mathbf{r}_0$ and `Decompose` do not need to process individual coefficients in any specific order.

Secret sharing. Sharing sensitive variable $\mathbf{w}_0$ in d additive shares, for `Decompose` and for the subtraction $\mathbf{w}_0 - c\,s_2$ would avoid any $d-1$-th order attack. Detecting when a coefficient $(\mathbf{w}_0)_{i,j} = 0$ using leakage from d multiple shares exponentially increases the number of traces needed with increasing masking order. For instance, an implementation of `SecDecompose`, which is an efficient secure implementation of `Decompose`, is already discussed by Azouaoui *et al.* [ABC+22]. The arithmetic sharing must be kept until the complete computation of $\mathbf{r}_0$.

One can notice that with the same setup and under the same hypothesis, the attack path described here could also be exploited through Machine Learning side-channel attack [MPP16] instead of a template attack. On a close platform, like a smart card, other unsupervised SCA, such as correlated power analysis, could be mounted to perform the same attack.

For future work, we want to investigate if there are other sensible operations that we did not identify yet as well as evaluate the leakage of the $\mathbf{w}_0 - c\,s_2$ operation.

References

[AASA+22] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status report on the third round of the nist post-quantum cryptography standardization process, 2022-07-05 2022.

[ABC+22] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Markus Schönauer, Tobias Schneider, François-Xavier Standaert, and Christine van Vredendaal. Leveling Dilithium against leakage: Revisited sensitivity analysis and improved implementations. *IACR Cryptol. ePrint Arch.*, page 1406, 2022.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29, Cambridge, Massachusetts, USA, August 11–13, 2004. Springer, Heidelberg, Germany.

[BDE+18] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 494–524, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[BDGN14] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Nicv: Normalized inter-class variance for detection of side-channel leakage. In *2014 International Symposium on Electromagnetic Compatibility, Tokyo*, pages 310–313, 2014.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2129–2146. ACM Press, November 11–15, 2019.

[BJS11] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[BP18] Leon Groot Bruinderink and Peter Pessl. Differential fault attacks on deterministic lattice signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):21–43, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7267.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[DKL+22] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Gregor Seiler, Peter Schwabe, and Damien Stehlé. PQ-CRYSTALS, Dilithium. https://github.com/pq-crystals/dilithium, 2022. GitHub repository. Accessed: 2022-12-15.

[FDK20] Apostolos P. Fournaris, Charis Dimopoulos, and Odysseas G. Koufopavlou. Profiling dilithium digital signature traces for correlation differential side channel attacks. In Alex Orailoglu, Matthias Jung, and Marc Reichenbach, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation - 20th International Conference, SAMOS 2020, Samos, Greece, July 5-9, 2020, Proceedings*, volume 12471 of *Lecture Notes in Computer Science*, pages 281–294. Springer, 2020.

[FDLZ14] Yunsi Fei, A. Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based fundamental model for side-channel attack analysis. Cryptology ePrint Archive, Report 2014/152, 2014. https://eprint.iacr.org/2014/152.

[GKS20] Denisa O. C. Greconici, Matthias J. Kannwischer, and Daan Sprenkels. Compact Dilithium implementations on cortex-M3 and cortex-M4. Cryptology ePrint Archive, Report 2020/1278, 2020. https://eprint.iacr.org/2020/1278.

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

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

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[KSSW] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. Pqclean. https://github.com/PQClean/PQClean. Accessed: 2022-12-15.

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

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[LZS+21] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based Fiat-Shamir signatures in the presence of randomness leakage. *IEEE Trans. Inf. Forensics Secur.*, 16:1868–1879, 2021.

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking Dilithium - efficient implementation and side-channel evaluation. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 344–362, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. Cryptology ePrint Archive, Report 2016/921, 2016. https://eprint.iacr.org/2016/921.

[MUTS22] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on Dilithium: A small bit-fiddling leak breaks it all. Cryptology ePrint Archive, Report 2022/106, 2022. https://eprint.iacr.org/2022/106.

[nis] National institutes of standards and technology, post-quantum cryptography. https://csrc.nist.gov/Projects/post-quantum-cryptography. Accessed: 2022-12-15.

[OC14] Colin O'Flynn and Zhizhang David Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, 2014.

[QLZ+23] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Jingdian Ming, Chengbin Jin, and Huizhong Li. Practical public template attack attacks on CRYSTALS-Dilithium with randomness leakages. *IEEE Trans. Inf. Forensics Secur.*, 18:1–14, 2023.

[RJH+18] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Side-channel assisted existential forgery attack on Dilithium - A NIST PQC candidate. Cryptology ePrint Archive, Report 2018/821, 2018. https://eprint.iacr.org/2018/821.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SBN+21] Deepraj Soni, Kanad Basu, Mohammed Nabeel, Najwa Aaraj, Marcos Manzano, and Ramesh Karri. *FALCON*, pages 31–41. Springer International Publishing, Cham, 2021.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[YJ21] Wei Yang and Anni Jia. Side-Channel Leakage Detection with One-Way Analysis of Variance. *Security and Communication Networks*, 2021:6614702, March 2021. Publisher: Hindawi.