Exploiting Small-Norm Polynomial Multiplication with Physical Attacks

Application to CRYSTALS-Dilithium

Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes and Tobias Schneider

NXP Semiconductors
firstname.lastname@nxp.com

Abstract. We present a set of physical attacks against CRYSTALS-Dilithium that accumulate noisy knowledge on secret keys over multiple signatures, finally leading to a full recovery attack. The methodology is composed of two steps. The first step consists of observing or inserting a bias in the posterior distribution of sensitive variables. The second step of an information processing phase which is based on belief propagation, which allows effectively exploiting that bias. The proposed concrete attacks rely on side-channel information, injection of fault attacks, or a combination of the two. Interestingly, the adversary benefits from the knowledge on the released signature, but is not dependent on it. We show that the combination of a physical attack with the binary knowledge of acceptance or rejection of a signature also leads to exploitable information on the secret key. Finally, we demonstrate that this approach is also effective against shuffled implementations of CRYSTALS-Dilithium.

Keywords: Lattice-based Cryptography · Post-Quantum Cryptography · Side-Channel Attacks · Fault Attacks · CRYSTALS-Dilithium

1 Introduction

Over the last years, quantum computing has witnessed significant advances. In turn this has accelerated the research, adoption and standardization of Post-Quantum Cryptography (PQC) schemes: cryptographic schemes that are believed to be secure even when facing an adversary with access to a quantum computer. The selection and standardization of PQC schemes is driven by the National Institute of Standard and Technology (NIST), and as of July 2022 the lattice-based schemes CRYSTALS-Kyber and CRYSTALS-Dilithium have been selected as the primary PQC standards for key establishment and digital signatures, respectively.

In particular, PQC schemes have garnered the interest of the cryptographic community with respect to their efficient and secure embedded implementations. The main security threats and countermeasures thereof investigated are Side-Channel Attacks (SCA) and Fault Attacks (FA). Notably for CRYSTALS-Kyber, additional vulnerabilities are introduced by the use of the Fujisaki-Okamoto (FO) transform [FO99] in the decapsulation process [RRCB20, XPR+22, SCZ+23]. Other works also describe how to target the Number Theoretic Transform (NTT) which is the main arithmetic building block in implementations of lattice-based schemes such as CRYSTALS-Kyber and CRYSTALS-Dilithium [HHP+21]. The protection of CRYSTALS-Kyber against SCA is also a relatively active research area as illustrated by this non-exhaustive list of recent publications [BGR+21, HKL+22, ABH+22, BC22].

This work focuses on CRYSTALS-Dilithium, and compared to CRYSTALS-Kyber or other Key Encapsulation Mechanisms (KEMs), it initially did not receive as much attention. Nonetheless, it has been the target of substantial contributions from the embedded cryptographic community. Notably, the work of Ravi et al. [RJH+18] has shown that it is possible to forge signatures by recovering only half of CRYSTALS-Dilithium's secret key. Marzougui et al. [MUTS22] have demonstrated that leakage of zero values in the masking vector in CRYSTALS-Dilithium leads to key recovery. A similar attack is described in [BVC+23]. In addition, the analysis from Islam et al. [IMS+22], ElGhamrawy et al. [EAB+23] and Ulitzsch et al. [UMB+23] illustrate that there many approaches to recovering CRYSTALS-Dilithium's secret signing key by exploiting the effects of fault injections. When it comes to masked implementation of CRYSTALS-Dilithium hardened against SCAs, very recently, Coron et al. [CGTZ23] revisited the work of Azouaoui et al. [ABC+23], which is in turn a revised masked implementation of the initial masking CRYSTALS-Dilithium proposal by Migliore et al. [MGTF19].

From a high level, all known attacks exploit specific attack vectors or leakages to extract some information which is then used to reduce the hardness of the cryptographic problem CRYSTALS-Dilithium is based on. For instance, in [MUTS22] it is the knowledge that a coefficient of the masking vector $\mathbf{y}$ is equal to zero. In [EAB+23], it is the fault-induced equality of two polynomials in $\mathbf{y}$. Generally, to recover the full secret key most known attacks rely on some algebraic post-processing, such as lattice reduction [DDGR20] or least squares [BDE+18]. However, such techniques are only slightly resilient to the noise inherent to SCAs and FAs.

Contributions. In this work, we propose generalized side-channel and fault attacks against CRYSTALS-Dilithium that scale well with noise and/or precision. The core observation leading to these attacks is that several polynomials are required to follow uniform distribution to ensure black-box security of CRYSTALS-Dilithium. Yet, both side-channel or fault attacks (next denoted together as physical attacks) create a bias on these polynomials, which then lead to exploitable information on the secret key. In short, we summarize our contributions as follows:

- *Generic Attack Framework.* In Section 3, we describe a new generic methodology to accumulate information on the secret key polynomial from noisy knowledge of polynomials. This approach is based on the Belief Propagation (BP) algorithm which is leveraged for Soft Analytical Side-Channel Attacks (SASCA) introduced in [VGS14]. As demonstrated in this work, it enables a large variety of attack scenarios that we systematically study through simulated experiments. For each of these, we evaluate the number of signatures to recover a single key polynomial.

- *Physical Attacks with Accepted Signatures.* In Section 4, we apply the framework to the case where the adversary gets access to released signatures. For side-channel attacks against Dilithium Level-2, we show that for low noise ($\mathtt{SNR} = 100$) only ≈ 4 traces are needed to recover a secret key polynomial. For higher noise levels (e.g., $\mathtt{SNR} = 0.01$), a total of ≈ 700 traces are needed. For fault attacks, between 23 and 2000 faults are needed depending on the fault precision.

- *Physical Attacks without Accepted Signatures.* In Section 5, we study the case of attacks without released signatures. When the signature is not accepted, but the index of the rejected coefficient is leaked (e.g., through an early-abort strategy), $\approx 6 \cdot 10^5$ traces are needed in the low noise settings. Eventually, we target a weakened parameter set and show that the knowledge of whether a full polynomial is rejected (e.g., when no early-abort strategy is implemented), while having no knowledge of which coefficient lead to this rejection, can still lead to key recovery. Fault attacks

are also similarly exploitable, but we do no include these experiments in the paper because of space limitations.

- *Physical Attacks with Shuffled Computation.* In Section 6, we demonstrate that shuffled implementations of `CRYSTALS-Dilithium`, as suggested in [ABC+22], are also vulnerable to physical attacks using our framework. More precisely, for side-channel attacks against Dilithium Level-2, with $\texttt{SNR} = 100$, a total of $\approx 5 \cdot 10^3$ traces are needed to recover the secret key polynomial. For $\texttt{SNR} = 0.5$, $\approx 1.2 \cdot 10^5$ measurements are needed. Similarly results also apply to fault attacks.

Put all together, our attack is a step forward in the direction of the evaluation of hardened implementation of `CRYSTALS-Dilithium` against physical attacks. Concretely, the impact of our contribution on the attack surface and countermeasures for variables within `CRYSTALS-Dilithium` is summarized in Table 1. Again, while attacks on some variables were known (e.g., fault attacks on $\mathbf{c} \circ \mathbf{s_1}$), we extend and generalize these attacks and provide a framework to exploit the leakage that scales efficiently with respect to noise. In Table 1a, we summarize the types of attack that are known in the literature, and what our framework is able to perform. In Table 1b, we summarize the type of countermeasure that should be applied to these variables. In particular, in this paper we conclude that shuffling is not effective for $\mathbf{y}$ and $\mathbf{w_0}$, hence we recommend to use masking for all the sensible variables as hinted in [ABC+23].

Table 1: Summary of sensitivity and countermeasure effectiveness in `CRYSTALS-Dilithium` for valid and rejected signatures.

	$\mathbf{y}$	$\mathbf{w_0}$	$\mathbf{c \circ s_1}$	$\mathbf{c \circ s_2}$
Valid Sig.				
SCA	⋆	⋆	⋆	⋆
FA	⋆	⋆		
Rej. Sig.				
SCA	⋆	⋆	⋆	⋆
FA	⋆	⋆		

(a) Sensitivity: red for effective attack, gray for no effective attack, ⋆ attack included in this work.

	$\mathbf{y}$	$\mathbf{w_0}$	$\mathbf{c \circ s_1}$	$\mathbf{c \circ s_2}$
Valid Sig.				
Masking				
Shuff.				
Rej. Sig.				
Masking				
Shuffling				

(b) Countermeasure: green for effective, red for not effective, gray for no concrete attacks.

2 Background

In this section, we describe the necessary background to understand the contributions of this paper. We start by introducing the general notations used in the paper. We then recall the main step of the signature generation process in `CRYSTALS-Dilithium` and some of its specific details and properties relevant to the remainder of this work. Afterwards, we describe side-channel attacks and the SASCA methodology.

2.1 Notations

We denote by $\mathbb{Z}_q$ the integer ring modulo the prime q, and by $\mathbb{Z}_q[X]/(X^n+1)$ the polynomial ring in X modulo $X^n + 1$. Polynomials in $\mathbb{Z}_q[X]/(X^n + 1)$ are written in bold, e.g., $\mathbf{p}$, with n being the degree of the polynomial. For `CRYSTALS-Dilithium`, $q = 2^{23} - 2^{13} + 1$ and $n = 256$, and we will fix them for the remainder of the paper. The i-th coefficient of a polynomial is denoted $\mathbf{p}_i$. The multiplication between two polynomials $\mathbf{a}$ and $\mathbf{b}$ is written

as $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$ and addition as $\mathbf{c} = \mathbf{a} + \mathbf{b}$. The infinity norm of a polynomial is expressed as $||\mathbf{p}||_\infty$ and is the maximum absolute value of its coefficients. Constants are denoted with Greek letters. A half-open interval containing all elements in the range $\{\alpha, \ldots, \beta - 1\}$ is denoted as $[\![\alpha, \beta[\![$. A closed interval is denoted as $[\![\alpha, \beta]\!]$, where β is included. The variables $x \bmod q$ are in the range $[\![0, q[\![$, variables $x \bmod^{\pm} q$ in the interval $[\![-(q-1)/2, (q-1)/2[\![$. The j-th bit of the i-th coefficient in a polynomial $\mathbf{p}$ is denoted as $\mathbf{p}_i[j]$. We denote random variables with upper case letters, e.g., X, and their realizations with a lower case, e.g., $X = x$. The probability of a realization is given as $\Pr[X = x]$. When clear from the context, we also use the more concise notation $\Pr[x]$. Similarly, the probability conditional to a realization of another random variable is given as $\Pr[X = x|Y = y]$, or $\Pr[x|y]$. The sampling of a random variable x from a set $\mathcal{X}$ is denoted as $x \xleftarrow{\$} \mathcal{X}$.

2.2 Signature Generation and Rejection in `CRYSTALS-Dilithium`

In this section we detail the `CRYSTALS-Dilithium` operations required for the understanding of the remaining sections. More specifically, we describe generation of the signature polynomials $\mathbf{z}$ and $\mathbf{r_0}$. As both are computed and bound checked similarly, we use generic parameters that can apply to both cases. We refer to the `CRYSTALS-Dilithium` specifications for additional details [DLL+17].

In a valid signature, the signature polynomial $\mathbf{z}$ must satisfy

$$\mathbf{z} = ||\mathbf{y} + \mathbf{s} \circ \mathbf{c}||_\infty < \gamma - \beta. \tag{1}$$

During the signing operation, first the polynomial $\mathbf{z} = \mathbf{y} + \mathbf{s} \circ \mathbf{c}$ is computed and bound checked, i.e., the norm of all its coefficients has to be strictly smaller than $\gamma - \beta$. If the bound check does not pass, the signature process is repeated until the aforementioned property is fulfilled (see Section 2.2.2). The same process is applied to the other signature polynomial $\mathbf{r_0}$ with slightly different parameters. Note that there are further rejection checks during the `CRYSTALS-Dilithium` signature generation, e.g., the number of ones in the hint. However, these are not relevant for our attacks and are omitted for simplicity.

In this work, we will discuss the recovery of a single secret key polynomial. Note that `CRYSTALS-Dilithium` is based on MLWE hence the secret key is composed of a small vector of polynomials. As a result, the experiments on a single polynomial can be used to derive the number of measurements and/or faults needed to mount the attack on the full `CRYSTALS-Dilithium` secret key.

2.2.1 Relevant Polynomials Properties

Next, we describe in detail the properties of the polynomials involved in the computation of $\mathbf{z}$ as well as the polynomial operations used. We refer to Table 2 for the Dilithium parameters that are relevant to our attack.

Secret key s. The long term secret key polynomial is denoted as $\mathbf{s}$. This polynomial has a small norm such that $||\mathbf{s}||_\infty \leq \eta$. All the coefficients in this polynomial are independently drawn from a uniform distribution thanks to `ExpandS` during `KeyGen` such that $\mathbf{s}_i \xleftarrow{\$} \{-\eta, \ldots, \eta\}$ for all indexes i. The parameter η depends on the parameter set with $\eta \in \{2, 4\}$. The distribution of the coefficients has a mean $\mu_\mathbf{s} = 0$ and a standard deviation $\sigma_\mathbf{s} = \sqrt{\eta(\eta+1)/3}$.

Challenge c. During `Sign`, a fresh challenge $\mathbf{c}$ is generated deterministically from a random bitstring thanks to `SampleInBall`. The obtained challenge has a special structure. It has exactly τ coefficients that are different from zero and are equal to either 1 or -1. Again, $\tau \in \{39, 49, 60\}$ depending on the parameter set. The polynomial $\mathbf{c}$ is part of the signature,

hence $\mathbf{c}$ is given for valid signatures (when the bound check is passed). This polynomial is considered as non-sensitive in protected implementations [MGTF19, ABC+23].

Mask polynomial $\mathbf{y}$. During `Sign`, a fresh mask polynomial $\mathbf{y}$ is derived for every new signature generation. The coefficients of these polynomials are uniformly distributed on the interval $[\![-\gamma, \gamma[\![$. This polynomial is not part of the valid signature and must remain secret.

Product $\mathbf{x} = \mathbf{s} \circ \mathbf{c}$. The signature generation involves computing the polynomial multiplication between $\mathbf{s}$ and $\mathbf{c}$. Concretely, each coefficient in the resulting polynomial $\mathbf{x}$ is defined as

$$\mathbf{x}_i = \sum_{j=0}^{i} \mathbf{s}_j \mathbf{c}_{i-j} - \sum_{j=i+1}^{n-1} \mathbf{s}_j \mathbf{c}_{n+i-j}. \tag{2}$$

As a result, each $\mathbf{x}_i$ is a weighted sum of secret key coefficients. Since $\mathbf{c}$ only contains exactly τ non-zero coefficients, each $\mathbf{x}_i$ is the weighted sum of a subset of τ secret key coefficients. Hence, $\mathbf{x}$ satisfies the property that $||\mathbf{x}||_\infty \leq \beta$ where $\beta = \tau\eta$. By the central limit theorem, the distribution of $\mathbf{x}_i$ can be estimated by a normal distribution $\approx \mathcal{N}(0, \sqrt{\tau} \cdot \eta(\eta+1)/3)$. However, its distribution can also be explicitly computed by using convolutions of probability tables as detailed in Section 3.

Signature $\mathbf{z} = \mathbf{y} + \mathbf{s} \circ \mathbf{c}$. Thanks to the norm-check performed during signature generation, one ensures that for all valid signatures we have $-\gamma + \beta < ||\mathbf{z}||_\infty < \gamma - \beta$. Putting it all together, the coefficients in $\mathbf{z}$ are given by

$$\mathbf{z}_i = \mathbf{y}_i + \sum_{j=0}^{i} \mathbf{s}_j \mathbf{c}_{i-j} - \sum_{j=i+1}^{n-1} \mathbf{s}_j \mathbf{c}_{n+i-j}. \tag{3}$$

2.2.2 Rejection Probability in a Blackbox Setting

The rejection probability of polynomials can be derived analytically (see [ABD+19, Section 3.4] for details). In the following, R_i stands for the random variable denoting the rejection event. That is, $r_i = 1$ (resp. $r_i = 0$) denotes that the i-th coefficient has been rejected (resp. accepted). Concretely, this probability is given by

$$\Pr[R_i = 1] = 1 - \Pr[R_i = 0] = 1 - \frac{2(\gamma - \beta) - 1}{2\gamma} \tag{4}$$

since for a given value of $\mathbf{x}_i$, exactly $2(\gamma - \beta) - 1$ of the 2γ possible values of $\mathbf{y}_i$ lead to a $\mathbf{z}_i$ contained within the bounds. Similarly, we denote the rejection of a full polynomial with the random variable R. Concretely, the rejection probability is given by

$$\Pr[R = 1] = 1 - \prod_{i=0}^{n} \Pr[R_i = 0] = 1 - \left(\frac{2(\gamma - \beta) - 1}{2\gamma}\right)^n \tag{5}$$

since it requires that all the coefficients are within the bounds.[1] Putting all together, we observe that the rejection probability of polynomials or coefficients is independent of the secret key $\mathbf{s}$ when the adversary only has access to $\mathbf{c}$ and $\mathbf{z}$.

[1] We use the same independence assumption as in the `CRYSTALS-Dilithium` specification [DLL+17, Section 3.3].

Table 2: Relevant `CRYSTALS-Dilithium` parameters for this paper.

NIST Security level	II	III	V
q (modulus)	$2^{23} - 2^{13} + 1$	$2^{23} - 2^{13} + 1$	$2^{23} - 2^{13} + 1$
τ (# of $\pm$ 1's in c)	39	49	60
γ ($\mathbf{y}$ coefficient range)	2^{17}	2^{19}	2^{19}
η (secret key range)	2	4	2
β ($= \tau \cdot \eta$)	78	196	120
average number of signing iterations	4.25	5.1	3.85

Early-Abort. The performance impact of rejecting signatures is relatively significant for `CRYSTALS-Dilithium`, since it requires to restart the signature generation process from quite an early step. Hence, to speed up the `Sign` algorithm, implementations may use an `Early-Abort` strategy. That is, as soon as a coefficient is out of bound, the signature generation is aborted and a fresh $\mathbf{y}$ and $\mathbf{c}$ are sampled. As a result, the execution time of `Sign` leaks which coefficient is rejected. This does not affect the security of `CRYSTALS-Dilithium` in a blackbox setting [DLL+17].

2.3 Side-Channel Attacks

A common type of physical attacks are so-called side-channel attacks, that exploit information obtained through physical leakages such as power consumption or timing to recover information on secret keys [KJJ99, CRR02]. Concretely, the adversary observes the leakage L^j (where j indexes the physical observations, also referred to as traces, of the cryptographic function's different executions) which is a random variable corresponding to the power consumption, electromagnetic radiation or timing of the execution of a cryptographic function (e.g., an AES encryption). From this observation, she can obtain partial information on ephemeral intermediate variables x^j (e.g., an S-box output) given by the conditional probability $\Pr[x|L^j]$. Given some other public information p^j (e.g., a plaintext), she derives information on a long-term secret k (e.g., a secret key) summarized by the conditional probability $\Pr[k|L^j, p^i]$. This estimated probability can be obtained thanks to various tools such as Gaussian templates [CRR02].

However, a single leakage observation may not be enough to completely recover the secret. Hence, the adversary can observe multiple (N) leakage traces and use a maximum likelihood approach to accumulate information on the secret. Precisely, she computes the likelihood of a secret k given the N leakage observations according to

$$\text{Likelihood}[k|L, p] \propto \sum_{j=0}^{N-1} \log \Pr[k|L^j, p^j] \tag{6}$$

and selects the most probable key according to

$$k^* = \max_{k'} \text{Likelihood}[k'|L]. \tag{7}$$

Overall, as the number of traces N increases, the probability that the guessed key k^* is the correct one increases. Hence, the more traces are available, the more successful the attack will be.

2.4 Soft Analytical Side-Channel Attacks (SASCA)

The previously described generic side-channel attack leverages observations on a single ephemeral variable x to recover information on the key. Yet, many variables in cryptographic implementations are dependent on the secret key. In the following, we describe SASCA [VGS14] which is a strategy to exploit simultaneously leakages on various variables.

First, we recall that the relationships between these sensitive variables can be represented with a factor graph. A factor graph is composed of nodes and edges. It contains two types of nodes. The first type is the *variable node* that represents a variable (or any value) within the implementation that can have a given distribution. The second type of node is the *function node*, that represents an operation between variables. These can typically be XOR gates, AND gates, modular additions or modular multiplications. In the factor graph, edges allow connecting variables with function nodes, hence forming a bipartite graph. A side-channel adversary exploiting such a factor graph performs a so-called SASCA that follows the next steps:

1. The adversary defines a factor graph that represents the implementation under attack. It typically contains sub-parts of the secret key (e.g., secret key bytes), public variables (e.g., plaintexts), and ephemeral secret variables (e.g, an S-box output). If the operations are repeated (exploiting multiple traces), parts of the graph can be duplicated.

2. Thanks to access to the leakage L, the adversary can gain partial knowledge on intermediate variables within the factor graph. That is for a variable x, the adversary derives $\Pr[x|L]$ from the leakage L similarly as in standard template attacks (see Section 2.3). She then initializes the variable node x within the factor graph to $\Pr[x|L]$. We denote this distribution as the initial distribution of the variable node and use the notation $\Pr_{ini}[x]$.

3. Once all the information obtained from the leakage is encoded into the factor graph, the adversary can run a Belief Propagation (BP) algorithm. Informally, BP iteratively updates the distributions of intermediate variables thanks to messages received from neighbor nodes, where messages are also distributions. In turn, the messages sent from function to variable nodes and from variable to function nodes are updated. Namely, the messaging passing rule from a variable node v to a function node f is given by
$$m_{v\to f}[x] = \underset{ini}{\Pr}[x] \cdot \prod_{f_n \in \delta_v \setminus f} m_{f_n \to v}[x] \tag{8}$$
where δ_v denotes all the neighbors of v and $\delta_v \setminus f$ denotes the set of all its neighbors excluding f. The message passed from one variable to a function is the product of all the other received messages and the initial distribution. The message passing rule from a function to a variable node is given by
$$m_{f\to v}[x] = \sum_{i \in \mathcal{I}} \zeta(i, x) \cdot \prod_{v_n \in \delta_f \setminus v} m_{v_n \to f}[i_{v_n}] \tag{9}$$
where $\mathcal{I}$ denotes one combination of input/output values and $\zeta(,)$ is the compatibility function of the function node (which is equal to 1 if the combination of input/output values is possible and 0 otherwise).

This adversary is known to be optimal when the factor graph does not contain cycles (i.e., is a tree). When the factor graph contains cycles, then this method becomes heuristic but has been demonstrated to be effective in the context of side-channel attacks [VGS14, HHP+21, BS21].

3 Generic Attack Framework

In this section, we describe the methodology for the attack. We first start with a high-level introduction, followed by the description of the factor graph exploited by the SASCA adversary. Eventually, we discuss optimizations used to compute the propagated messages.

3.1 High Level Description of the Attacks

All the attacks proposed in this work are based on SASCA. These attacks enable to recover each of the secret key polynomials in `CRYSTALS-Dilithium` independently in a divide and conquer fashion, hence we focus on the problem of recovering a single secret key polynomial $\mathbf{s}$. The attack can simply be repeated on each polynomial to recover the full secret key. As illustrated in Figure 1, all the proposed attacks are performed in two phases.

Information Extraction. The first step consists of recovering partial information on each of the coefficients in the polynomial $\mathbf{x}$ with $\mathbf{x} = \mathbf{s} \circ \mathbf{c}$. In each of the considered attacks, the probability of some coefficient $\mathbf{x}_i$ posterior to the physical attack is estimated. These probabilities $\Pr_{ini}[\mathbf{x}_i]$ are the actual ephemeral secret information that the adversary can extract from the `CRYSTALS-Dilithium` signature generation. This can be obtained from side-channel leakages on $\mathbf{y}$ and $\mathbf{x}$, a fault attack biasing the distribution of $\mathbf{y}$ or their combination. This can be used to recover the secret key either together with a valid signature $\mathbf{z}$ or from a rejection event R when the signature is rejected. The information extraction is specific to the adversarial capabilities, and we describe several scenarios in the next sections.

Information Processing. The second step consists of leveraging all the obtained $\Pr_{ini}[\mathbf{x}_i]$ (possibly from several traces) to map them to information on the secret key $\mathbf{s}$ via SASCA. This step is identical for every attack considered in this paper. In the remainder of this section, we describe how information on $\Pr_{ini}[\mathbf{x}]$ can be efficiently accumulated to the secret key $\Pr[\mathbf{s}]$ by virtue of a dedicated factor graph.

3.2 Factor Graph Description

The factor graph is illustrated for a small example of a degree four polynomial in Figure 1. The top variable nodes $\mathbf{s}_i$ denote the coefficients of a secret key polynomial (not to be confused with the polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$ in the description of Dilithium), and the bottom variable nodes are coefficients of the polynomial $\mathbf{x} = \mathbf{s} \circ \mathbf{c}$. These are linked through the weighted sum function nodes $\Sigma_{\mathbf{c},i}$. This function node $\Sigma_{\mathbf{c},i}$ is graphically described in Figure 2 and is the straightforward mapping of Equation 2. Concretely, it implements the expression $\mathbf{x}_i = \sum_j \mathbf{c}'_j \mathbf{s}_j$ where the polynomial $\mathbf{c}' = \mathtt{rot}(\mathbf{c}, i)$ is the rotation of $\mathbf{c}$ defined as $\mathtt{rot}(\mathbf{c}, i) = \mathbf{c} \cdot X^i \bmod X^n + 1$. As depicted by the color in Figure 1, each coefficient in $\mathbf{x}$ is part of an independent sub-graph involving one single $\Sigma_{\mathbf{c},i}$ node.

When the challenge polynomial $\mathbf{c}$ is known, exactly τ of the edges of each independent sub-graph in Figure 2 have to be kept as $\mathbf{c}$ contains only τ coefficients different from zeros. For the non-zero coefficients, the function nodes $\cdot\mathbf{c}'_i$ denote the multiplication by either 1 or -1 depending on the value of $\mathbf{c}'_i$. This function node can be simply implemented as a re-ordering of the propagated messages. For efficiency, we merge the computation on this function node with the subsequent addition as detailed in Section 3.3.

An important remark is that all the polynomials involved in the signature generation have a small norm. As a result, all the intermediate variables are smaller than q meaning that no modular reduction must be done. Therefore, we next discuss the propagation rules and omit the (unnecessary) modular reductions.

3.3 Efficient Propagation Rule Computation

The previously described factor graph could be computed with a generic SASCA tool such as SCALib [CB23]. However, it comes with drawbacks such as execution time and memory consumption. Instead, we describe the propagation rules optimized for the previous factor graph. That is, how both Equation 8 and Equation 9 are efficiently implemented in such a

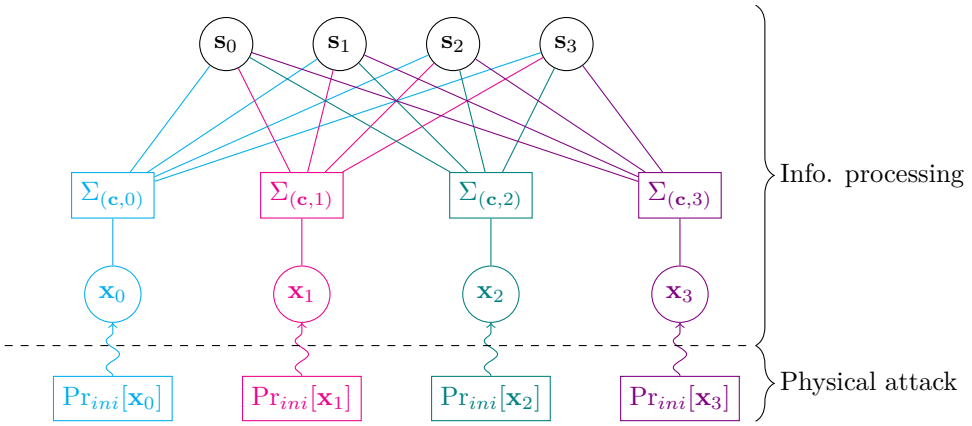


Figure 1: Example factor graph for parameters $N = 1$ trace and polynomial of degree $n = 4$.

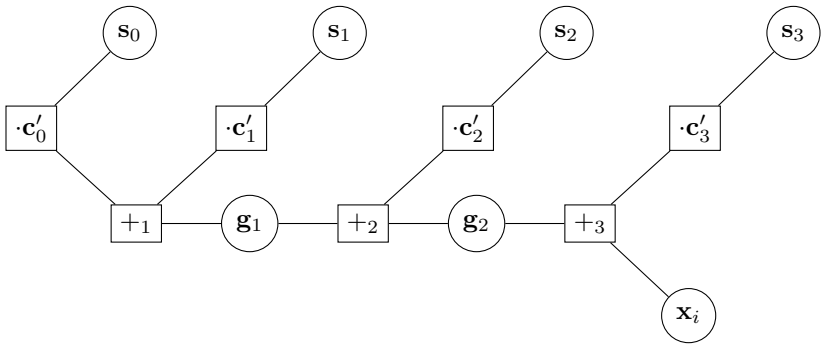


Figure 2: Internal description of $\Sigma_{(\mathbf{c},i)}$ factor node used in Figure 1 and $\mathbf{c}' = \texttt{rot}(\mathbf{c}, i)$

context. To this end, we take a bottom-up approach and first describe the propagation rule for a single $\Sigma_{(\mathbf{c},i)}$ described in Figure 2. Then we discuss how multiple of these can be combined as in Figure 1.

Propagation Rule for $\Sigma_{(\mathbf{c},i)}$. Starting with a single $\Sigma_{(\mathbf{c},i)}$, we detail the notations in the corresponding visual representation (see Figure 2). There, every variable node $\mathbf{g}_i$ stands for the sum $\mathbf{g}_i = \sum_{j=0}^{j=i} \mathbf{c}'_j \cdot \mathbf{s}_j$ where the weights $\mathbf{c}'_i$ are known. We start with the propagation rules around $+_i$, and continue with the propagation on the full factor graph.

The propagation rule that computes the message from the function node $+_i$ to the following variable node $\mathbf{g}_i$ is next denoted as `convadd` and is described in Algorithm 1. In order to compute $m_{+_i \to \mathbf{g}_i}$, the other incoming messages to the function node $+_i$ are needed. Hence `convadd` takes as input the incoming messages $m_{\mathbf{g}_i \to +_i}$ and $m_{\mathbf{s}_i \to +_i}$ together with the known value $\mathbf{c}'_i$ that multiplies $\mathbf{s}_i$. From Equation 9, the outgoing message $m_{+_i \to \mathbf{g}_i}$ is computed by summing over the product of all the other combinations of incoming messages. That is, the algorithm has two nested loops to cover all the input combinations of g' and s'.[2] The corresponding output value o is computed as $o = g' + c' \cdot s'$ and the value of the outgoing message is updated for the value o. Eventually, we note that `convadd` can be used to compute the messages $m_{+_i \to \mathbf{g}_{i-1}}$ by negating $\mathbf{c}'_i$ and adapting the ranges for input values.

[2]Note that the maximum non-zero value of both incoming messages are known. Hence the sum on the possible input combinations can be adapted accordingly.

Algorithm 1 `convadd` $(m_{\mathbf{g}_{i-1}\to+_i}, m_{\mathbf{s}_i\to+_i}, \mathbf{c}_i')$

Input: Input messages $m_{\mathbf{g}_{i-1}\to+_i}$ with $||m_{\mathbf{g}_{i-1}\to+_i}|| = 2(\beta-\eta)+1$ and $m_{\mathbf{s}_i\to+_i}$ with $||m_{\mathbf{s}_i\to+_i}|| = 2\eta+1$. Weight $\mathbf{c}_i' \in [\![-1,1]\!]$.
Output: Message $m_{+_i\to\mathbf{g}_i}$

if $\mathbf{c}_i' = 0$ **then** ▷ Quit early as output message will be the input message
  return $m_{\mathbf{g}_{i-1}\to+_i}$
$m_{+_i\to\mathbf{g}_i} \leftarrow 0^{2\beta+1}$ ▷ Init message with zeros
for $s' \in [\![-\eta,\eta]\!]$ **do**
  for $g' \in [\![-\beta+\eta,\beta-\eta]\!]$ **do**
    $o \leftarrow g' + c' \cdot s'$
    $m_{+_i\to\mathbf{g}_i}[o] \leftarrow m_{+_i\to\mathbf{g}_i}[o] + m_{\mathbf{g}_{i-1}\to+_i}[g'] \cdot m_{\mathbf{s}_i\to+_i}[s']$
return $m_{+_i\to\mathbf{g}_i}$

Next, we describe in Algorithm 2 the propagation rule on the full function node Σ. First, the challenge polynomial $\mathbf{c}$ is rotated with `rot` with the appropriate index in order to obtain $\mathbf{c}$'. Then, all the messages $m_{+_i\to\mathbf{g}_i}$ going from left to right in Figure 2 are computed. To do so, the $m_{\mathbf{g}_0\to+_1}$ is initialized with the incoming message $m_{\mathbf{s}_0\to\Sigma}$ re-ordered according to $\mathbf{c}_0'$. Then, we iterate in ascending order on all the $+_i$. There, we note that $m_{\mathbf{g}_j\to+_{j+1}} \leftarrow m_{+_j\to\mathbf{g}_j}$ as per Equation 8 as there is no other incoming messages to $\mathbf{g}_j$'s. A similar iteration is applied in descending order (right to left on Figure 2) in order to compute messages $m_{+_j\to\mathbf{g}_{j-1}}$. The last step is to compute the messages $m_{\Sigma\to\mathbf{s}_j}$ by computing the propagation rule around $+_j$ with the two already available messages $m_{\mathbf{g}_j\to+_j}$ and $m_{\mathbf{g}_{j-1}\to+_j}$. This is done thanks to `convaddrev` which is slightly adapted `convadd` in order to include the effect of $\mathbf{c}_i'$ on the output variable.[3]

Algorithm 2 Propagation rules for Σ associated to $\mathbf{x}_i$.

Input: All input messages $m_{\mathbf{s}_j\to\Sigma}$ for $j \in [\![0,n[\![$, challenge $\mathbf{c}$, degree of exploited coefficient i and its associated $\Pr_{ini}[\mathbf{x}_i]$.
Output: Generates all the messages $m_{\Sigma\to\mathbf{s}_j}$ for $j \in [\![0,n[\![$.

$\mathbf{c}' \leftarrow \mathtt{rot}(\mathbf{c}, i)$
$m_{\mathbf{g}_0\to+_1} \leftarrow \mathbf{c}_0' \cdot m_{\mathbf{s}_0\to\Sigma}$ ▷ Prop. from $\mathbf{s}_0$ up to $\mathbf{x}_i$
for j from 1 to n **do**
  $m_{+_j\to\mathbf{g}_j} \leftarrow \mathtt{convadd}(m_{\mathbf{g}_{j-1}\to+_j}, m_{\mathbf{s}_j\to\Sigma}, \mathbf{c}_j')$
  $m_{\mathbf{g}_j\to+_{j+1}} \leftarrow m_{+_j\to\mathbf{g}_j}$
$m_{\mathbf{g}_n\to+_{n-1}} \leftarrow \Pr_{ini}[\mathbf{x}_i]$ ▷ Prop. from $\mathbf{x}_i$ up to $\mathbf{s}_0$
for j from $n-1$ to 0 **do**
  $m_{+_j\to\mathbf{g}_{j-1}} \leftarrow \mathtt{convadd}(m_{\mathbf{g}_j\to+_j}, m_{\mathbf{s}_j\to\Sigma}, -\mathbf{c}_j')$
  $m_{\mathbf{g}_{j-1}\to+_{j-1}} \leftarrow m_{+_j\to\mathbf{g}_{j-1}}$
$m_{\Sigma\to\mathbf{s}_0} \leftarrow m_{+_1\to\mathbf{g}_0}$
for j from 1 to n **do**
  $m_{\Sigma\to\mathbf{s}_j} \leftarrow \mathtt{convaddrev}(m_{\mathbf{g}_j\to+_j}, m_{\mathbf{g}_{j-1}\to+_j}, \mathbf{c}_i)$

Propagation Rule for Multiple $\Sigma_{(\mathbf{c},i)}$. In the above, we described the SASCA propagation rule for a single function node $\Sigma_{(\mathbf{c},i)}$. Multiple of these factor nodes can be connected to the variable nodes of the secret key coefficients $\mathbf{s}_i$. This is the case for the factor graph of a

[3] The main difference is that the output message $m_{+_i\to\mathbf{s}_i}$ is computed by using $o = -c'(g_{i-1} - g_i)$.

single polynomial multiplication as described in Figure 1. Concretely, the variable node $\mathbf{s}_i$ receives multiple messages $m_{\Sigma_{(\mathbf{c},j)}\to\mathbf{s}_i}$ from all the $\Sigma_{(\mathbf{c},j)}$ nodes it is connected to ($\mathbf{c}'_i \neq 0$). The messages $m_{\mathbf{s}_i\to\Sigma_{(\mathbf{c},j)}}$ are then computed according to Equation 8 as the product of all other incoming messages to $\mathbf{s}_i$ hence as $m_{\mathbf{s}_i\to\Sigma_{(\mathbf{c},j)}} = \prod_{n\neq j} m_{\Sigma_{(\mathbf{c},j)}\to\mathbf{s}_i}$. This product of a large number of small values can lead to computational errors. Therefore to compute these messages, we first compute and store the sum of log-probabilities of all messages. Then the outgoing messages are computed such as $\log(m_{\mathbf{s}_i\to\Sigma_{(\mathbf{c},j)}}) = \sum_n \log(m_{\Sigma_n\to\mathbf{s}_i}) - \log(m_{\Sigma_j\to\mathbf{s}_i})$. Eventually, the guessed value $\mathbf{s}_i^*$ for a secret key coefficient $\mathbf{s}_i$ is the value maximizing the product of incoming messages similarly to Equation 7. It can be derived from this sum of log-probabilities as

$$\mathbf{s}_i^* = \max_{s'} \sum_n \log(m_{\Sigma_n\to\mathbf{s}_i}[s']). \tag{10}$$

In the above, we only describe the case where a single signature corresponding to a single product $\mathbf{s}\circ\mathbf{c}$ is observed, where one $\Sigma_{(\mathbf{c},i)}$ is added for each of the n coefficients in the output polynomial $\mathbf{x}$. Yet, additional $\Sigma_{(\mathbf{c},i)}$ nodes can also be added to the factor graph by observing N signatures corresponding to different products $\mathbf{s}\circ\mathbf{c}$ for different challenges $\mathbf{c}$ and a constant secret key polynomial $\mathbf{s}$. In such a case, the factor graph contains at most $N\times n$ different nodes $\Sigma_{(\mathbf{c},i)}$. All these $\Sigma_{(\mathbf{c},i)}$ must not necessarily be included in the factor graph either. This can be the case if the associated $\Pr_{ini}[\mathbf{x}_i]$ obtained through the physical attack is known to be secret key independent (e.g., no leakage). In all cases, the above propagation rules remain the same.

3.4 Discussion on the Factor Graph Selection

Knowledge of c. In the above, we assume that the adversary knows exactly $\mathbf{c}$. In the case of released signatures, this knowledge is trivial as it is embedded into the signature. This case is studied in Section 4 and Section 6. When the adversary does not have access to a released signature (see Section 5), $\mathbf{c}$ is not known but can potentially be recovered by other means such as side-channel leakage. Since state-of-the-art hardened implementations of `CRYSTALS-Dilithium` often do not protect the polynomial $\mathbf{c}$ against side-channel attacks [MGTF19, ABC+23] and, in addition, the polynomial is being manipulated as single bits per register, it is assumed to be a relatively easy target for side-channel adversaries. The proposed attack also extends to the setting where only noisy leakage on $\mathbf{c}$ is obtained. In such a case, the multiplication with a weight $\mathbf{c}'$ is simply replaced by a multiplication function node between $\mathbf{s}_i$ and $\mathbf{c}'_i$ making the propagation rule slightly more complex. We leave such a detailed investigation to future work.

Impact of Fast Polynomial Multiplication with NTT. We notice that the factor graph used for this attack implements a school-book polynomial multiplication. However, efficient implementations of `CRYSTALS-Dilithium` usually leverage NTTs to perform polynomial multiplications [AHKS22]. Yet, we stress that the attack methodology is independent of the polynomial multiplication methodology as it is based on the definition of polynomial multiplication itself. The only slight advantage of using NTT-based multiplication is that direct leakage on $\mathbf{x}$ can be avoided as it is not explicitly computed in the standard domain and only in NTT representation (this is not the case in [AHKS22]). Even in this case, the attack is also applicable as direct leakage on $\mathbf{y}$, which cannot be avoided, allows to initialize the factor graph as discussed in the following sections.

Performance Considerations. A performance consideration is that only τ coefficients are different from zero in $\mathbf{c}$ (see Table 2). When the challenge polynomial $\mathbf{c}$ is known, exactly τ of the edges must be kept for each Σ_i. Overall, for each Σ_i node, only the outgoing messages $m_{\Sigma_i\to\mathbf{s}_i}$ and the associated $\Pr_{ini}[\mathbf{x}_i]$ must be stored in memory. This

leads to a total of $\tau(2\eta+1)+2\beta+1$ 64-bit floats that need to be stored. For example, if all the Σ_i are included and 1000 signatures are used for the attack, 0.72 gigabyte is needed for Dilithium-2, 1.7 gigabyte is needed for Dilithium-3 and 1.1 gigabyte is needed for Dilithium-5 to store the full factor graph.

Eventually, we note that the propagation rule for + (`convadd`) can also be implemented by leveraging FFT-based convolutions as proposed in [PPM17]. However, the benefits are not obvious as one of the inputs to the addition is always small ($[\![-\eta,\eta]\!]$). The study of such an approach and the practical benefits it may bring is also left for future work.

4 Physical Attacks with Valid Signatures

In this section we detail the case where the adversary obtains the signature, i.e., both polynomials $\mathbf{z}$ and $\mathbf{c}$. We first describe the leakage and fault models we consider, and then continue with the methodology used to initialize the factor graph described in Section 3 with $\Pr_{ini}[\mathbf{x}_i]$'s. Finally, we describe the results of simulated attacks for both side-channel and fault attacks.

4.1 Leakage and Fault Models

We start by describing the leakage and fault models used for the simulated attacks. We stress that the results presented in this paper are not restricted to these models and also apply to others. In both cases, we assume that the adversary knows exactly the leakage and faults models. We leave the study of unprofiled scenarios to future investigations.

Leakage Model. In this work, we consider leakage on polynomials $\mathbf{x}=\mathbf{c}\circ\mathbf{s}$ and $\mathbf{y}$ described in Section 2.2 in a similar way. For simplicity, we only describe the leakage for $\mathbf{x}$. The leakages under consideration are the sum of a deterministic data-dependent function and Gaussian noise. We assume that the coefficients leak independently. The deterministic component of the leakage function is denoted by

$$\mathcal{L}^{B,\pm}_{\mathbf{x}_i}=\sum_{b\in B}(\mathbf{x}_i \bmod^{\pm} q)[b], \tag{11}$$

which is the sum of bits of $\mathbf{x}_i \bmod^{\pm} q$. The bits involved in the leakage are defined by the list B where each value in B is a bit index. As an example, the leakage $\mathcal{L}^{31,\pm}_{\mathbf{x}_i}$ corresponds to the sign-bit of the coefficient $\mathbf{x}_i$ in 32-bit two's complement representation. In case the data is represented in the interval $[\![-(q-1)/2,(q-1)/2[\![$, the notation $\mathcal{L}^{B,\pm}_{\mathbf{x}_i}$ is used. If the coefficients are represented in the interval $[\![0,q[\![$, the notation $\mathcal{L}^{B,+}_{\mathbf{x}_i}$ is used. From this, the leakage on a polynomial coefficient is a random variable

$$L_{\mathbf{x}_i} \leftarrow \mathcal{L}^{*,*}_{\mathbf{x}_i}+\mathcal{N}(0,\sigma^2_{\texttt{SNR}}) \tag{12}$$

where $\sigma^2_{\texttt{SNR}}$ is the noise variance ensuring the `SNR` for the given deterministic leakage function $\mathcal{L}^{*,*}_{\mathbf{x}_i}$.

Concretely to mount the attack, the adversary first computes the probability of observing a leakage sample $l_{\mathbf{x}_i}$ with standard Gaussian template attacks assuming a given value for $\mathbf{x}_i$. That is

$$\Pr[l_{\mathbf{x}_i}|\mathbf{x}_i,\mathcal{L}^{*,*}_{\mathbf{x}_i},\sigma_{\texttt{SNR}}] \propto \exp\left(-\frac{\left(l_{\mathbf{x}_i}-\mathcal{L}^{*,*}_{\mathbf{x}_i}\right)^2}{2\sigma^2_{\texttt{SNR}}}\right). \tag{13}$$

Second, she computes the value of $\Pr[\mathbf{x}_i|l_{\mathbf{x}_i},\mathcal{L}^{*,*}_{\mathbf{x}_i},\sigma_{\texttt{SNR}}]$ thanks to Bayes's theorem (normalization) over all the possible values $\mathbf{x}_i$ such as

$$\Pr[\mathbf{x}_i|l_{\mathbf{x}_i},\mathcal{L}^{*,*}_{\mathbf{x}_i},\sigma_{\texttt{SNR}}]=\frac{\Pr[l_{\mathbf{x}_i}|\mathbf{x}_i,\mathcal{L}^{*,*}_{\mathbf{x}_i},\sigma_{\texttt{SNR}}]}{\sum_{x'\in\mathcal{X}}\Pr[l_{x'}|x',\mathcal{L}^{*,*}_{x'},\sigma_{\texttt{SNR}}]} \tag{14}$$

Generally, we will assume that the polynomials have a signed representation (e.g., see [AHKS22]) and that the device leaks the hamming weight of intermediates. The latter deterministic leakage component is then denoted as $\mathcal{L}_{\mathbf{x}_i}^{0:31,\pm}$. In the following, the adversary is able to exploit leakage on polynomials $\mathbf{y}$ and/or $\mathbf{x} = \mathbf{s} \circ \mathbf{c}$ depending on the context.

Fault Model. Similarly to side-channel leakage models, the fault attacks detailed in this work apply to various models. In this work, we assume that the adversary is able to insert a fault that will induce a known bias on the bits of a polynomial coefficient. Concretely, the fault adversary can set a bit b to zero with probability α such that:

$$\Pr_{\text{bit}}[b|\lightning_\alpha] = \begin{cases} \alpha & b = 0 \\ 1 - \alpha & b = 1 \end{cases} \tag{15}$$

As a result, the probability of a given faulted coefficient is proportional to

$$\Pr[\mathbf{y}_i|\mathcal{F}_{\mathbf{y}_i,\alpha}^{B,+}] \propto \prod_{b \in B} \Pr_{\text{bit}}[(\mathbf{y}_i \bmod {}^+ q)[b]|\lightning_\alpha], \tag{16}$$

which is the product of the probability on each of its (non-uniform) faulted bits. Similarly, as for Equation 14, the actual $\Pr[\mathbf{y}_i]$ is obtained by normalization. As an example, the fault model $\mathcal{F}_{\mathbf{y}_i,1}^{0:31,\pm}$ sets all the coefficients of $\mathbf{y}_i$ to zeros with probability 1. In the following, we will assume that a single coefficient is faulty in the polynomial $\mathbf{y}$. Yet, multiple coefficients in $\mathbf{y}$ can also be faulted in order to reduce the number of faulted signatures required. We stress that faults can be inserted also with the signed representation of coefficients. In the following, we only consider faults on $\mathbf{y}$, as faults on $\mathbf{x}$ does not seem to be directly exploitable with our framework. We leave the investigation of faults on $\mathbf{x}$ as an open question.

4.2 Initialization of the Factor Graph

The previous equations describe how to derive the probabilities on coefficients in $\mathbf{x}$ and $\mathbf{y}$ from either side-channel leakages or induced faults. In the following, we describe how these probabilities are used to derive the initial probabilities $\Pr_{ini}[\mathbf{x}_i]$ for the SASCA described in Section 3. Concretely, we observe that the knowledge of the released signature polynomial $\mathbf{z}$ enables to directly translate information on $\mathbf{y}$ to information on $\mathbf{x}$ because of the additive relation $\mathbf{z} = \mathbf{x} + \mathbf{y}$. That is, in the case of side-channel leakage on $\mathbf{y}$, the equation

$$\Pr[\mathbf{x}_i|\mathbf{z}_i, l_{\mathbf{y}_i}, \mathcal{L}_{\mathbf{y}_i}^{*,*}, \sigma_{\texttt{SNR}}] \propto \Pr[\mathbf{y}_i = \mathbf{z}_i - \mathbf{x}_i|l_{\mathbf{y}_i}, \mathcal{L}_{\mathbf{y}_i}^{*,*}, \sigma_{\texttt{SNR}}], \tag{17}$$

is used to derive the probability on the corresponding $\mathbf{x}_i$. Similarly, when a fault is introduced on a coefficient $\mathbf{y}_i$, the resulting posterior probability on $\mathbf{x}_i$ is derived thanks to

$$\Pr[\mathbf{x}_i|\mathbf{z}_i, \mathcal{F}_{\mathbf{y}_i,*}^{*,*}] \propto \Pr[\mathbf{y}_i = \mathbf{z}_i - \mathbf{x}_i|\mathcal{F}_{\mathbf{y}_i,*}^{*,*}]. \tag{18}$$

All these probabilities on $\mathbf{x}_i$ can be combined into the initial probability used by SASCA according to

$$\begin{aligned} \Pr_{ini}[\mathbf{x}_i] \propto \Pr[\mathbf{x}_i|\mathbf{z}_i, l_{\mathbf{y}_i}, \mathcal{L}_{\mathbf{y}_i}^{*,*}, \sigma_{\texttt{SNR}}] \\ \cdot \Pr[\mathbf{x}_i|l_{\mathbf{x}_i}, \mathcal{L}_{\mathbf{x}_i}^{*,*}, \sigma_{\texttt{SNR}}] \\ \cdot \Pr[\mathbf{x}_i|\mathbf{z}_i, \mathcal{F}_{\mathbf{y}_i,*}^{*,*}] \end{aligned} \tag{19}$$

where the two first terms represent the side-channel leakage on $\mathbf{y}_i$ and $\mathbf{x}_i$, respectively. The last term stands for the information on $\mathbf{x}_i$ obtained from the fault injection on $\mathbf{y}_i$. This equation puts forward that both information from side-channel and fault attacks can be summarized in the same probability, which makes the extension to combined attacks straightforward.

4.3 Experimental Results

In the following, we describe the results of both side-channel and fault attacks assuming the model described in the previous section. In both cases, both the polynomials $\mathbf{z}$ and $\mathbf{c}$ are assumed to be known (e.g., through a valid signature). The efficiency of the attack is estimated through the median number of correctly recovered coefficients in the secret key polynomial $\mathbf{s}$ among the 256 ones. This is estimated from 100 independent experiments after 20 iterations of the BP algorithm described in Section 3.

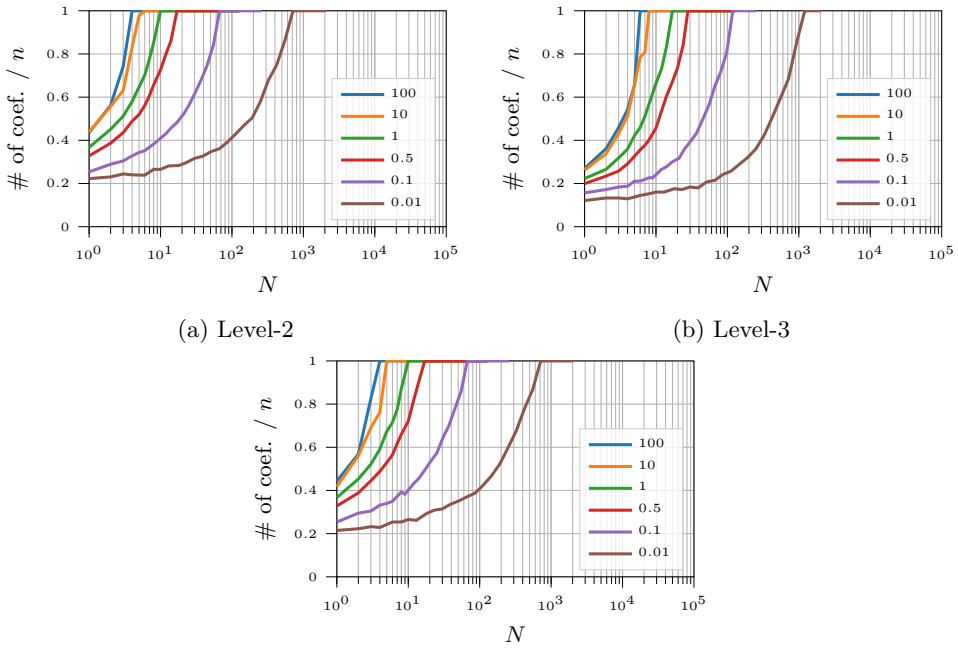
Recent works such as [DDGR20] have shown how to integrate information extracted from a side-channel into lattice reduction attacks against LWE-based schemes. However, up to our knowledge, it is not yet fully understood how to accurately quantify the extent of information required on the secret key coefficients to break `CRYSTALS-Dilithium`. This is more so challenging when dealing with soft/probabilistic information as is the case for common side-channel and fault attacks. This line of research is orthogonal to this work, and we leave it as an open question. In the following, we will consider that an attack is successful once the full secret key polynomial is recovered only from the physical attack.

Side-Channel Attacks. The results for side-channel attacks exploiting hamming weight leakage on all the coefficients of both polynomials $\mathbf{x}$ and $\mathbf{y}$ are reported in Figure 3. On these plots, each of the curves represents a different `SNR`. For level-2 parameters, only 4 traces are needed to obtain all the secret key polynomials when the noise is very low ($\texttt{SNR} = 100$). For $\texttt{SNR} = 0.1$, around 70 traces are needed and around 700 are needed with $\texttt{SNR} = 0.01$. As expected, as the noise increases, the number of traces required to mount the side-channel attack increases with an inversely proportional relationship. From these plots, we observe that both level-2 and level-5 have similar results, while level-3 requires slightly more traces. We expect that this difference is due to the larger secret key coefficient size η (see Table 2).

Fault Attacks. Similarly, the results of fault attacks with a single faulted coefficient $\mathbf{y}_0$ in a valid signature are reported in Figure 4. As a single coefficient leads to key dependent data, a single function node Σ_0 is added to the factor graph in Figure 1 per challenge $\mathbf{c}$. Similarly, as for side-channel attacks, we observe that the less precise the fault is, the less efficient the attack. As an example, when $\alpha = 1$ meaning that $\mathbf{y}_0$ is always set to zero, around 23 faulted and released signatures are required for level-2 parameter set. When $\alpha = 0.6$, meaning that the bias on $\mathbf{y}_0$ towards zero is weaker, around 2000 faults are needed.

Note that we report the number of faulted and released signatures. However, as `CRYSTALS-Dilithium` is a Fiat-Shamir with abort scheme, multiple signature trials are performed. This means that the faulted signature might not be among the released ones. Concretely, if a single fault is inserted during all signature attempts, the number of faults to insert is derived by multiplying the numbers from Figure 4 by the average number of repetitions from Table 2. We refer to [EAB+23] for an analysis of and strategies to deal with `CRYSTALS-Dilithium` signature aborts in combination with faults.

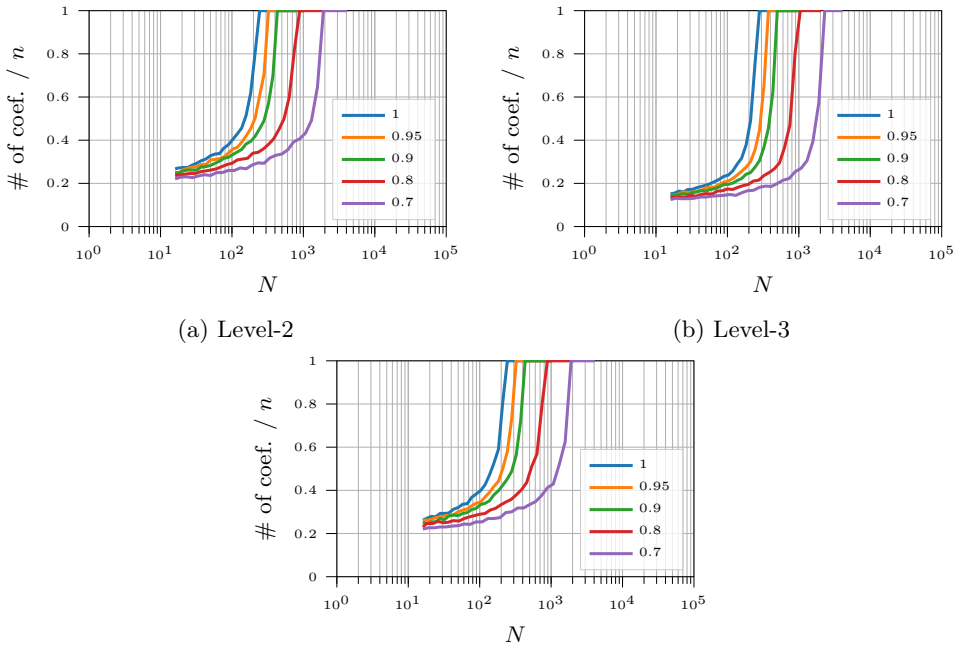
Eventually, we stress that if multiple faults are injected in a single signature attempt, then the same number of additional Σ_i nodes can be inserted in the factor graph. This has the effect of decreasing proportionally the required number of faulted and released signatures N. Indeed, the information extraction methodology is sensitive to the number of Σ nodes, independently of whether they are added through additional challenges $\mathbf{c}$ or through additional faults. Finally, we note that exactly the same methodology can be used to mount combined attacks as illustrated with Equation 19.



(a) Level-2 (b) Level-3

(c) Level-5

Figure 3: Side-channel attack with hamming weight leakages on polynomials $\mathcal{L}_{\mathbf{x}}^{0:31,\pm}$ and $\mathcal{L}_{\mathbf{y}}^{0:31,\pm}$. N is the number of released signatures. Curves correspond to the median proportion of correctly recovered coefficients for various `SNR` values.



(a) Level-2 (b) Level-3

(c) Level-5

Figure 4: Fault attack on $\mathbf{y}_0$ with model $\mathcal{F}_{\mathbf{y}_i,\alpha}^{0:23,+}$. N is the number of faulted and released signatures. Curves correspond to the median proportion of correctly recovered coefficients for various induced bias α values.

5 Physical Attacks with Rejected Signatures

In the previous section, we demonstrated that leakage on $\mathbf{y}$ and/or $\mathbf{x}$ can be used in combination with a valid signature $(\mathbf{z}, \mathbf{c})$ in order to retrieve the secret key polynomials. In this section, we will demonstrate that the combination of leakage on $\mathbf{y}$, the knowledge of the challenge $\mathbf{c}$ (e.g., through side-channel analysis) and the fact that a polynomial is rejected or not leads to exploitable information on the secret key. This applies to implementations with and without an early-abort strategy.

5.1 Initialization of the Factor Graph with Early-Abort

We denote the event of a coefficient in the signature polynomial $\mathbf{z}_i$ as being rejected as $R_i = r_i$. Namely, $r_i = 1$ (resp. $r_i = 0$) if the coefficient is rejected (resp. accepted). In case the early-abort is implemented, the coefficients of $\mathbf{z}$ are checked individually and sequentially. The process is aborted as soon as one coefficient is rejected, leaking r_i through timing. For consistency, we next denote the information induced by the physical attack such as side-channel leakage or a fault as $\mathcal{P}$. From this, we will estimate $\Pr[\mathbf{x}_i|R_i, \mathcal{P}]$ which is the probability distribution of the coefficient $\mathbf{x}_i$ knowing if $\mathbf{z}_i$ has been rejected and the information extracted from the physical attack.

Concretely, the probability that the coefficient $\mathbf{z}_i$ has been rejected given a value of $\mathbf{x}_i$ is expressed as

$$\Pr[R_i = 1|\mathbf{x}_i, \mathcal{P}] = \sum_{\mathbf{y}_i} \Pr[R_i = 1|\mathbf{x}_i, \mathbf{y}_i, \mathcal{P}] \cdot \Pr[\mathbf{y}_i|\mathcal{P}], \tag{20}$$

which sums over all the combinations $(\mathbf{y}_i, \mathbf{x}_i)$. In the previous expression, we note that the term $\Pr[R_i = 1|\mathbf{x}_i, \mathbf{y}_i, \mathcal{P}]$ is a compatibility function that is equal to one when $|\mathbf{x}_i + \mathbf{y}_i| \geq \gamma - \beta$ and equal to zero otherwise. Similarly, the probability that the same coefficient is accepted is given as

$$\Pr[R_i = 0|\mathbf{x}_i, \mathcal{P}] = 1 - \Pr[R_i = 1|\mathbf{x}_i, \mathcal{P}]. \tag{21}$$

Finally, the probability on $\mathbf{x}_i$ can be obtained thanks to Bayes's theorem according to

$$\Pr[\mathbf{x}_i|R_i = r_i, \mathcal{P}] = \frac{\Pr[R_i = r_i|\mathbf{x}_i, \mathcal{P}]}{\sum_{x^*} \Pr[R_i = r_i|x^*, \mathcal{P}]}, \tag{22}$$

similarly to Equation 14. In this equation, values for $\Pr[R_i = r_i|\mathbf{x}_i, \mathcal{P}]$ are derived with Equation 20 if $r_i = 1$ and Equation 21 otherwise. This probability can then directly be used to initialize the factor graph described in Section 3.

5.2 Initialization of the Factor Graph without Early Abort

We also study the case where the early-abort is not implemented. Hence, the adversary only knows if the full $\mathbf{z}$ has been rejected (resp. accepted), denoted with $R = 1$ (resp. $R = 0$), but does not have knowledge of which particular coefficient was rejected. Concretely, in the following, we will derive the expression of $\Pr[\mathbf{x}_i|R, \mathcal{P}]$. To do so, we first describe the expression for the probability that a coefficient is individually accepted which is

$$\Pr[R_i = 0|\mathcal{P}] = \sum_{\mathbf{x}_i \in [\![-\beta, \beta]\!]} \Pr[\mathbf{x}_i] \cdot \Pr[R_i = 0|\mathbf{x}_i, \mathcal{P}] \tag{23}$$

where $\Pr[R_i = 0|\mathbf{x}_i, \mathcal{P}]$ is obtained from Equation 21. The distribution $\Pr[\mathbf{x}_i]$ can be analytically computed because $\mathbf{x}_i$ is defined as the sum of τ uniform variables uniform on

$[\![-\eta, \eta]\!]$. Then, the probability that the polynomial is accepted given a single coefficient value $\mathbf{x}_i$ is given by

$$\Pr[R = 0|\mathbf{x}_i, \mathcal{P}] = \Pr[R_i = 0|\mathbf{x}_i, \mathcal{P}] \prod_{j \neq i} \Pr[R_j = 0|\mathcal{P}], \tag{24}$$

because the polynomial is accepted only if all the coefficients are individually accepted. Similarly to Equation 21, the probability that the polynomial is rejected is given by

$$\Pr[R = 1|\mathbf{x}_i, \mathcal{P}] = 1 - \Pr[R = 0|\mathbf{x}_i, \mathcal{P}]. \tag{25}$$

Finally, the posterior distribution of the coefficient $\mathbf{x}_i$ given $R = r$ can be expressed as

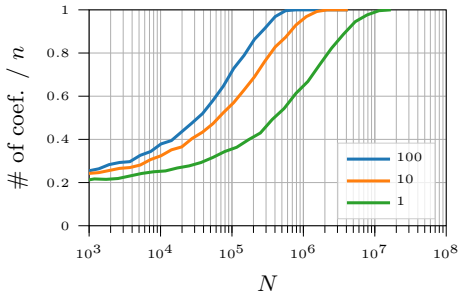
$$\Pr[\mathbf{x}_i|R = r, \mathcal{P}] = \frac{\Pr[R = r|\mathbf{x}_i, \mathcal{P}]}{\sum_{x^*} \Pr[R = r|x^*, \mathcal{P}]}. \tag{26}$$

Again, this expression can be directly used to initialize the factor graph.

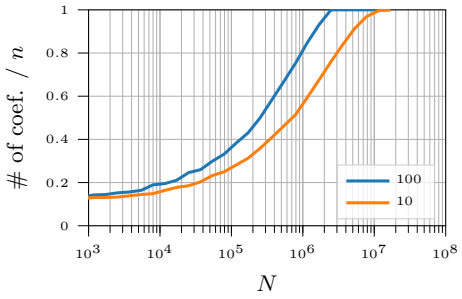
5.3 Experimental Results

Side-Channel Attack with Early-Abort. In Figure 5, the results of side-channel attacks against rejected signatures when early-abort is implemented are reported for all parameter sets. Concretely, the target is assumed to leak $\mathcal{L}_{\mathbf{y}}^{0:31,\pm}$ as well as the index of the rejected coefficient $\mathbf{z}_j$ such that $j = \min_i(R_i = 1)$. As a result, the adversary also gets knowledge that $R_i = 0$, for all $i < l$. From this information, we leverage Equation 22 in order to initialize the factor graph described in Section 3. Despite the fact that both accepted and rejected coefficients could be exploited to mount the attack, we noticed that the coefficient for which $R_j = 1$ provides the most information on $\mathbf{x}_j$. As a result, we only exploit the rejected coefficient in order to reduce the size of the factor graph (and its memory requirements). From these figures, we observe that with $\texttt{SNR} = 100$, around $7 \cdot 10^5$ (resp. $2.1 \cdot 10^6$) rejected polynomials are needed to recover the secret polynomial for Level-2 and Level-5 (resp. Level-3). As expected, decreasing the SNR, hence increasing noise, leads to an increased number of required signatures. Concretely, around 10^7 are required to recover the secret key polynomial when $\texttt{SNR} = 1$.

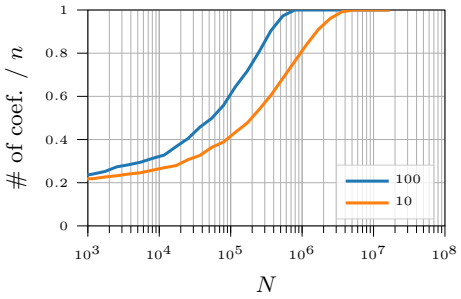
Side-Channel Attack without Early-Abort. Next, we discuss the previously described attack exploiting rejected polynomials when the early-abort is not implemented. That is, the index(es) of the rejected coefficient(s) is unknown to the adversary. Accordingly, the factor graph is initialized with Equation 26. As a result, the graph includes a node for every $\mathbf{x}_i$ as the adversary has no prior knowledge of which node is the most informative, as opposed to when an early-abort strategy is used. This results in a large factor graph which is challenging to process. We were not able to mount such an attack against standard CRYSTALS-Dilithium parameter sets because of insufficient memory in our setup. Therefore, we introduce a weakened parameter set called Level-0 which is equivalent to Level-2 except that $\tau = 14$. This weakened parameter set is used to put forward that secret information can also be recovered even if only the knowledge that $\mathbf{z}$ has been rejected is provided to the adversary in addition to leakage on $\mathbf{y}$. This applies to all the parameter sets despite the fact that we were not able to mount a full attack with our evaluation setup. Results of this attack on the so-called Level-0 are given in Figure 6. There, we show that for (almost) noise free Hamming Weight on $\mathbf{y}$, around $5 \cdot 10^5$ rejected polynomials are needed.



(a) Level-2



(b) Level-3



(c) Level-5

Figure 5: Side-channel attack with hamming weight leakages on polynomials $\mathcal{L}_{\mathbf{y}}^{0:31,\pm}$ with rejected signatures. Early-abort is implemented and the index of the rejected coefficient is known. N is the number of rejected signatures. Curves correspond to the median proportions of correctly recovered secret key coefficients for various `SNR` values.

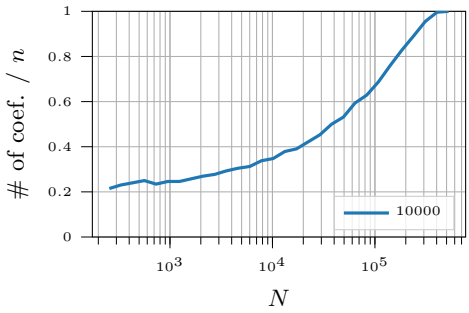


Figure 6: `CRYSTALS-Dilithium` Level-0. Side-channel attack with hamming weight leakages on polynomials $\mathcal{L}_{\mathbf{y}}^{0:31,\pm}$ with rejected signature. Early-abort is not implemented. N is the number of rejected polynomials. Curves correspond to the median proportion of correctly recovered coefficients for various `SNR` values.

6 Physical Attacks against Shuffled $\mathbf{y}$

In this section, we apply the methodology described in Section 3 in order to target an implementation where the polynomial $\mathbf{y}$ is protected with shuffling, as proposed in [ABC+22]. Note that the later version of this work [ABC+23] does not rely on shuffling anymore to protect the considered `CRYSTALS-Dilithium` implementation. In the following, we demonstrate that shuffling $\mathbf{y}$ is indeed not sufficient to protect against side-channel attacks. Next we follow the same approach as in previous section. Namely, we first describe how to initialize the factor graph in that setting, and then provide the results.

6.1 Initialization of the factor graph

In a shuffled implementation, the adversary does not know the order in which the coefficients in $\mathbf{y}$ are manipulated. That is, only the leakage of $\mathbf{y}^j$ is available, which is the leakage of the j-th manipulated coefficient. From this, the adversary can not relate this leakage to a single specific coefficient $\mathbf{y}_i$. As a result, we leverage the approach from [VMKS12], later refined in [ABG+22, Section 3], in order to attack shuffled implementations. The first step is to compute the probability of $\mathbf{y}_i$ given the leakage on all the coefficients of $\mathbf{y}$. Assuming that no leakage is available on the shuffle permutation itself, this expression is given by

$$\Pr[\mathbf{y}_i | l_{\mathbf{y}}, \mathcal{L}^{*,*}_{\mathbf{y}}, \sigma_{\texttt{SNR}}] \propto \sum_{i=0}^{n} \Pr[\mathbf{y}^j | l_{\mathbf{y}^j}, \mathcal{L}^{*,*}_{\mathbf{y}^j}, \sigma_{\texttt{SNR}}], \tag{27}$$

where each of the $\Pr[\mathbf{y}^j | l_{\mathbf{y}^j}, \mathcal{L}^{*,*}_{\mathbf{y}^j}, \sigma_{\texttt{SNR}}]$ is the probability for coefficient manipulated at index j. Its expression can be obtained similarly as for standard template attacks (see Equation 14). From this estimated probability on $\mathbf{y}_i$, the probability on each of the coefficient $\mathbf{x}_i$ can be derived with

$$\Pr[\mathbf{x}_i | \mathbf{z}_i, l_{\mathbf{y}_i}, \mathcal{L}^{*,*}_{\mathbf{y}_i}, \sigma_{\texttt{SNR}}] \propto \Pr[\mathbf{y}_i = \mathbf{z}_i - \mathbf{x}_i | l_{\mathbf{y}}, \mathcal{L}^{*,*}_{\mathbf{y}}, \sigma_{\texttt{SNR}}], \tag{28}$$

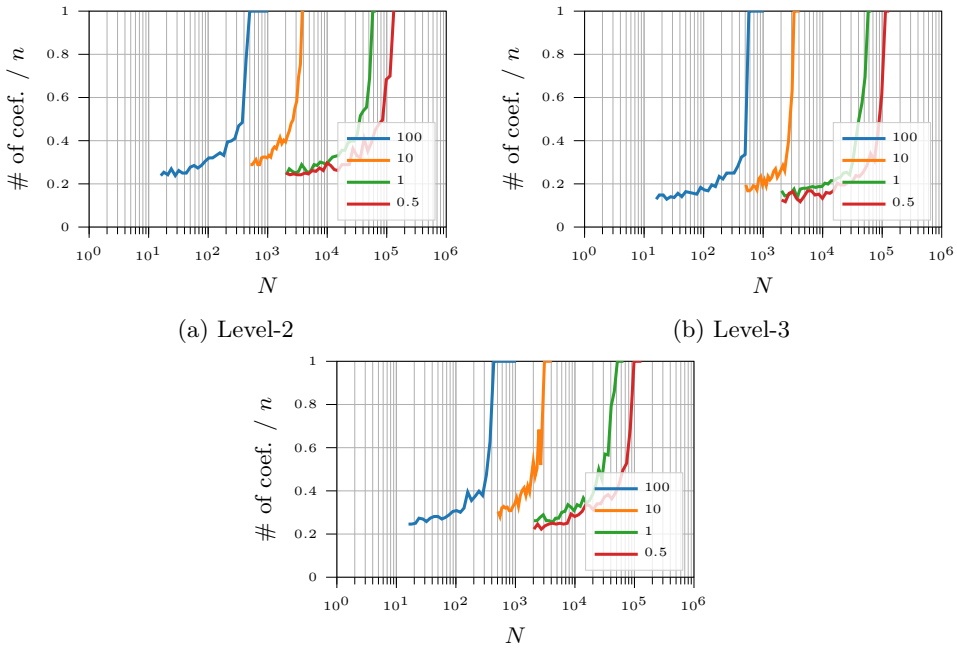
which is exactly similar to Equation 17. These probabilities are then directly used to initialize the factor graph. We note that Equation 27 is equivalent for every $\mathbf{y}_i$. As a result, it can be computed only once. In case permutation leakage is available (which is not assumed here), it can be incorporated into that equation, leading to different expression for every $\mathbf{y}_i$ [ABG+22].

6.2 Experimental Results

The result of the attacks against shuffled $\mathbf{y}$ are reported in Figure 7. There, we observe that for Dilithium Level-2, around $\approx 5 \cdot 10^3$ traces are needed when $\texttt{SNR} = 100$. When $\texttt{SNR} = 0.5$, around $1.2 \cdot 10^5$ are required. Interestingly we observe that when doubling the `SNR`, the attack complexity is increased by two. Meaning that the noise only linearly increase the complexity of the attack, as expected with shuffling [VMKS12]. These results can also be compared with Figure 3 where the $\mathbf{y}$ was not shuffled. It shows that the attack complexity is increased by a significant factor, but does not avoid the attack.

7 Conclusion

We put forward several new fault and side-channel attack vectors against `CRYSTALS-Dilithium` signature generation leading to key recovery. We have shown that even a slight bias on the distribution of $\mathbf{y}$ posterior to a physical attack leaks sensitive information. This is an improvement over previous work that requires high accuracy from the leakage traces



(a) Level-2

(b) Level-3

(c) Level-5

Figure 7: Side-channel attack with hamming weight leakages $\mathcal{L}_{\mathbf{y}}^{0:31,\pm}$ on shuffled polynomial. N is the number of released signatures. Curves correspond to the median proportion of correctly recovered coefficients for various `SNR` values.

or fault precision. Up to our knowledge, this work is the first one to demonstrate physical attacks exploiting rejected signatures, putting forward that all the iterations of the Fiat-Shamir with Abort are sensitive. Eventually, similar attacks should be applicable to various schemes such as Raccoon [dPPRS23] or HAETAE [CCD+23].

References

[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. *Fourth PQC Standardization Conference*, 2022.

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

[ABD+19] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 3:4, 2019.

[ABG+22] Melissa Azouaoui, Olivier Bronchain, Vincent Grosso, Kostas Papagiannopoulos, and François-Xavier Standaert. Bitslice masking and improved shuffling: How and when to mix them in software? *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):140–165, 2022.

[ABH+22] Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standaert. Systematic study of decryption and re-encryption leakage: The case of kyber. In Josep Balasch and Colin O'Flynn, editors, *Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings*, volume 13211 of *Lecture Notes in Computer Science*, pages 236–256. Springer, 2022.

[AHKS22] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Amber Sprenkels. Faster kyber and dilithium on the cortex-M4. In Giuseppe Ateniese and Daniele Venturi, editors, *ACNS 22: 20th International Conference on Applied Cryptography and Network Security*, volume 13269 of *Lecture Notes in Computer Science*, pages 853–871. Springer, Heidelberg, June 2022.

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

[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. Springer, Heidelberg, December 2018.

[BGR+21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):173–214, 2021.

[BS21] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):202–234, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8973.

[BVC+23] Alexandre Berzati, Andersson Calle Viera, Maya Chartouni, Steven Madec, Damien Vergnaud, and David Vigilant. A practical template attack on CRYSTALS-dilithium. Cryptology ePrint Archive, Report 2023/050, 2023. https://eprint.iacr.org/2023/050.

[CB23] Gaëtan Cassiers and Olivier Bronchain. Scalib: A side-channel analysis library. *Journal of Open Source Software*, 8(86):5196, 2023.

[CCD+23] Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. HAETAE: shorter lattice-based fiat-shamir signatures. *IACR Cryptol. ePrint Arch.*, page 624, 2023.

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

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[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. Springer, Heidelberg, August 2020.

[DLL+17] Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - dilithium: Digital signatures from module lattices. *IACR Cryptol. ePrint Arch.*, page 633, 2017.

[dPPRS23] Rafaël del Pino, Thomas Prest, Mélissa Rossi, and Markku-Juhani O. Saarinen. High-order masking of lattice signatures in quasilinear time. In *SP*, pages 1168–1185. IEEE, 2023.

[EAB+23] Mohamed ElGhamrawy, Melissa Azouaoui, Olivier Bronchain, Joost Renes, Tobias Schneider, Markus Schönauer, Okan Seker, and Christine van Vredendaal. From MLWE to RLWE: A differential fault attack on randomized & deterministic dilithium. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):262–286, 2023.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen ciphertext k-trace attacks on masked CCA2 secure kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):88–113, 2021. https://tches.iacr.org/index.php/TCHES/article/view/9061.

[HKL+22] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. First-order masked kyber on ARM cortex-m4. *IACR Cryptol. ePrint Arch.*, page 58, 2022.

[IMS+22] Saad Islam, Koksal Mus, Richa Singh, Patrick Schaumont, and Berk Sunar. Signature correction attack on dilithium signature scheme. In *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*, pages 647–663. IEEE, 2022.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[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. Springer, Heidelberg, June 2019.

[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.

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

[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. *IACR Cryptol. ePrint Arch.*, page 821, 2018.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[SCZ+23] Muyan Shen, Chi Cheng, Xiaohan Zhang, Qian Guo, and Tao Jiang. Find the bad apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of kyber. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):89–112, 2023.

[UMB+23] Vincent Quentin Ulitzsch, Soundes Marzougui, Alexis Bagia, Mehdi Tibouchi, and Jean-Pierre Seifert. Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ILP. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):367–392, 2023.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, Heidelberg, December 2012.

[XPR+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David F. Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Trans. Computers*, 71(9):2163–2176, 2022.