Breaking and Protecting the Crystal: Side-Channel Analysis of Dilithium in Hardware

Hauke Steffen[1], Georg Land[2], Lucie Kogelheide*[3], and Tim Güneysu[2,4]

[1] TÜV Informationstechnik GmbH, Essen, Germany
[2] Ruhr University Bochum, Horst Görtz Institute for IT Security, Germany
[3] BWI GmbH, Bonn, Germany
[4] DFKI GmbH, Cyber-Physical Systems, Bremen, Germany
`h.steffen@tuvit.de`, `mail@georg.land`, `lucie.kogelheide@bwi.de`,
`tim.gueneysu@rub.de`

Keywords: Dilithium · Side-Channel Analysis · FPGA · SPA · CPA · PQC

Abstract. The lattice-based CRYSTALS-Dilithium signature scheme has been selected for standardization by the NIST. As part of the selection process, a large number of implementations for platforms like x86, ARM Cortex-M4, or – on the hardware side – Xilinx Artix-7 have been presented and discussed by experts. While software implementations have been subject to side-channel analysis with several attacks being published, an analysis of Dilithium hardware implementations and their peculiarities has not taken place. With this work, we aim to fill this gap, presenting an analysis of vulnerable operations and practically showing a successful profiled Simple Power Analysis (SPA) and a Correlation Power Analysis (CPA) on a recent hardware implementation by Beckwith et *al.* Our SPA attack requires 700 000 profiling traces and targets the first Number-Theoretic Transform (NTT) stage. After finishing profiling, we can identify pairs of coefficients with 1 101 traces. The full CPA attack finds secret coefficients with as low as 66 000 traces. In response, we present specific countermeasures and show that they effectively prevent both attacks.

1 Introduction

Quantum computers pose a real threat to communication security. Currently deployed symmetric schemes can be adapted easily to withstand attacks even from large-scale quantum computers. In contrast, asymmetric schemes like RSA and ECC-based schemes can be broken without significant effort through Shor's algorithms [25]. Although it is not yet clear whether this threat will become a reality in the near future, it is undisputed that action needs to be taken early to prevent prospective damage. Therefore, the United States National Institute

---

* The respective work has been conducted as an employee of TÜV Informationstechnik GmbH.

for Standards and Technology (NIST) launched standardization efforts for post-quantum secure schemes for Key Encapsulation Mechanisms (KEMs) and digital signatures in 2017.

After three rounds, with several schemes being dropped due to cryptanalytic attacks, lacking efficiency, or missing confidence in their security assumptions, NIST announced the schemes to be standardized in July 2022. As KEM, Kyber has been selected, while four other schemes proceed to a fourth round and are considered for standardization in the future. For signature schemes, Dilithium, Falcon, and SPHINCS$^+$ are being standardized, with Dilithium being the primary choice.

Dilithium has undergone a thorough cryptanalytic process and guarantees security against Strong Existential Unforgeability under Chosen Message Attacks (SUF-CMA). Besides, concrete implementations can be attacked by employing side-channel analysis, exploiting dependencies of physical characteristics on secret values during computation. Several side-channel analyses have been published on Dilithium software implementations in this context. In [22], Ravi et *al.* show a signature forgery attack enabled by finding a partial secret key using a power analysis. This work is extended to fault attacks on pqm4 implementations of Dilithium and qTesla [23], also presenting a mitigation approach. Migliore et *al.* carry out a side-channel evaluation targeting the ARM Cortex-M4 platform [21]. They are also the first to introduce concrete masking countermeasures. Following this, Chen et *al.* present an efficient CPA attack on the Dilithium pqm4 software implementation [9], succeeding with only 157 power measurements. Karabulut et *al.* show that sampling of fixed-weight polynomials in Dilithium, NTRU, and NTRU Prime is vulnerable to side-channel analysis [16]. Finally, Marzougui et *al.* present a novel side-channel attack that exploits a vulnerability in a sampling procedure [20]. However, their attack requires many measurements and complex post-processing. Finally, a recent work by Azouaoui et *al.* [1] presents a thorough analysis of side-channel requirements for Dilithium, including state-of-the-art countermeasures.

All these works have in common that they target *software* platforms. At the same time, there is *no* dedicated side-channel analysis targeting hardware implementations, which is a glaring lack in light of Dilithium already being chosen for standardization. Our work aims to close this gap by analyzing a recent Field-Programmable Gate Array (FPGA) implementation, presenting a profiled SPA and a CPA attack. Additionally, we investigate and implement countermeasures, evaluating their efficacy against the before-proposed attacks.

Contribution. Hence, our contribution can be summarized as follows:

- We present the first power side-channel results of a Dilithium implementation in reconfigurable hardware.
- We show several profiled SPA attacks on Dilithium-2 and -5, including:
    - an evaluation of single-trace attacks on the decoding and the first NTT stage, with up to 94.2 % success probability to recover the correct coefficient.

  - multi-trace attacks on decoding with 50 000 profiling traces, capable of recovering the target coefficient with 130 traces during attack phase.
  - multi-trace attacks on first NTT stage with 350 000 profiling traces that enable full key recovery with a pair of target coefficients using 1 101 traces.
- We also show a CPA on the polynomial multiplication, recovering secret coefficients with 66 000 traces, which are agnostic to the parameter set and enable full key recovery.
- We present an analysis of how to apply masking as a countermeasure by proposing arithmetic masking, effectively prohibiting the presented attacks.

2 Preliminaries

2.1 Notation

Throughout this work, we will use and assume the following notation. Let n and q be two integers, such that $n = 256$ and $q = 2^{23} - 2^{13} + 1$. Further, let $\mathcal{R}_q$ be a polynomial ring with $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$. The infinity norm $||x||_\infty$ of a polynomial x is defined as the maximum absolute value among all its coefficients. For polynomial vectors, this norm is defined as the maximum infinity norm of all polynomials in the vector. Then, S_b denotes the set of polynomials in $\mathcal{R}_q$ with infinity norm b and $\tilde{S}_b$ denotes the same set but excluding coefficients with value $-b$. Furthermore, the set of polynomials in $\mathcal{R}_q$ with exactly τ non-zero coefficients and infinity norm 1 is denoted as B_τ. In addition, let us denote vectors in bold lower-case letters, e.g., $\mathbf{v}$, while matrices are denoted in bold upper-case letters, e.g., $\mathbf{A}$. Polynomials in NTT domain are indicated by a hat, e.g., $\hat{c}$. This is also used transitively; thus, $\hat{\mathbf{A}}$ denotes that each polynomial in $\mathbf{A}$ is transformed to NTT domain individually. Finally, we denote the pointwise multiplication with $\circ$.

2.2 CRYSTALS-Dilithium

As common for digital signature schemes, Dilithium provides the three core procedures for *key generation*, *signature generation*, and *signature verification*. In the following, we briefly explain the key generation and signing and leave a fully detailed description of the scheme to the official documentation [11].

Key Generation. Algorithm 1 shows the key generation of Dilithium. As can be seen there, finding the secret key from knowing the public key is basically the M-LWE problem. Moreover, once an attacker obtains either $\mathbf{s}_1$ or $\mathbf{s}_2$, she can directly obtain the other value since $\mathbf{A}$ and $\mathbf{t}$ are public values. However, Dilithium makes an interesting modification in moving the lower d bits of each coefficient in $\mathbf{t}$ to the secret key in order to reduce the public key size, which is what the function Power2Round does. Still, the polynomial vector $\mathbf{t}_0$, which contains these lower bits, is considered public information.

Algorithm 1 Dilithium key generation

```
$\zeta \leftarrow \{0,1\}^{256}$
$(\rho, \rho', K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} :=$ SHAKE-256($\zeta$)
sample $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$ deterministically in NTT domain from the output stream of SHAKE-128($\rho$)
sample $(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^\ell \times S_\eta^k$ from the output stream of SHAKE-256($\rho'$)
$\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
$(\mathbf{t_1}, \mathbf{t_0}) := \mathrm{Power2Round}_q(\mathbf{t}, d)$
$tr \in \{0,1\}^{256} :=$ SHAKE-256($\rho || \mathbf{t}_1$)
return $(pk = (\rho, \mathbf{t}_1), sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$
```

Algorithm 2 Dilithium signature generation

```
Require: secret key $sk$, message $M$
 1: $\kappa := 0$, sample $\mathbf{A}$ as in key generation
 2: $\mu \in \{0,1\}^{512} :=$ SHAKE-256($tr || M$)
 3: $\rho' \in \{0,1\}^{512} :=$ SHAKE-256($K || \mu$) for deterministic signing
    $\rho' \leftarrow \{0,1\}^{512}$ for randomized signing
 4: while true do
 5:     sample $\mathbf{y} \in \tilde{S}_{\gamma_1}^\ell$ deterministically based on $\rho', \kappa$
 6:     $\mathbf{w} := \mathbf{A}\mathbf{y}$
 7:     $\mathbf{w_1} := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
 8:     $\tilde{c} \in \{0,1\}^{256} :=$ SHAKE-256($\mu || \mathbf{w_1}$)
 9:     $c \in B_\tau :=$ SampleInBall($\tilde{c}$)
10:     $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
11:     $\mathbf{r_0} := \mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$
12:     if $||\mathbf{z}||_\infty < \gamma_1 - \beta$ and $||\mathbf{r_0}||_\infty < \gamma_2 - \beta$ then
13:         $\mathbf{h} := \mathsf{MakeHint}_q(-c\mathbf{t_0}, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t_0}, 2\gamma_2)$
14:         if $||c\mathbf{t_0}||_\infty < \gamma_2$ and the # of 1's in $\mathbf{h}$ is less than or equal to $\omega$ then
15:             return $(\mathbf{z}, \mathbf{h}, \tilde{c})$
16:     $\kappa := \kappa + \ell$
```

Signature Generation. Algorithm 2 describes the signature generation for a given message and secret key. Most notably, there is a big rejection loop that only terminates if the signature is approved not to leak any information on the secret key, which is ensured by the checks starting in Line 13. Inside the loop, the signing algorithm chooses a masking polynomial vector $\mathbf{y}$ with coefficients from $[-\gamma_1, \gamma_1)$, computes $\mathbf{w} = \mathbf{A}\mathbf{y}$, and rounds each coefficient of the resulting polynomial vector according to the $\mathsf{HighBits}_q$ function. From this and the message, the challenge polynomial c is sampled, which has precisely τ non-zero coefficients, which are either 1 or -1. Then, a signature candidate $\mathbf{z}$ is computed as $\mathbf{y} + c\mathbf{s}_1$. Following this, it is checked whether the broad "noise" generated by $\mathbf{y}$ actually hides $c\mathbf{s}_1$. Finally, using the $\mathsf{MakeHint}_q$ function, the signing algorithm generates "hints" for the verifier to compensate for the missing lower bits of $\mathbf{t_0}$. Note that all polynomial multiplications are performed using the NTT for efficiency.

Parameters. For Dilithium, three parameter sets are proposed, which aim at the NIST security categories 2, 3, and 5. The security is scaled primarily via increasing the matrix and vector dimensions (k, ℓ), which are $(4, 4)$ for level 2, $(6, 5)$ for level 3, and $(8, 7)$ for level 5. Another relevant parameter that changes over the parameter sets is the secret key range η, which is 2 for levels 2 and 5, and 4 for level 3.

2.3 Side-Channel Analysis

The field of side-channel analysis has been established with Kocher's seminal work [17] on timing side-channels. In the following, we briefly explain the two relevant approaches for our work.

Simple Power Analysis. This technique aims to analyze power traces directly to learn operations that have been executed and processed secrets. In the best case, a single measurement is sufficient to completely recover the key. The most important extension of SPA is *profiled* or template SPA. Here, the attack is performed in two phases. In the *profiling phase*, the attacker measures the target device performing several operations with known or chosen secret input, obtaining information about the device's behavior depending on the input. In the *attack phase*, she uses the knowledge from the first phase to recover the secret by measuring the target device performing the operation with secret input.

This requires an extension of the attacker model. When introducing profiling, the attacker must now have extended access to the target device, knowing or even being able to choose several usually secret inputs. She may use one or multiple traces in the attack phase, resulting in *single-trace* or *multi-trace* attacks.

Finding Points of Interest. To determine Points of Interest (POIs) that correspond to differences between the observed classes, we use the sum of squared pairwise t-differences (SOST) as the metric, which has been introduced in [13]. The idea is to measure many traces for each class, then compute the t-test traces between any possible pair of classes, square them point-wise, and accumulate the results. We then consider points if their SOST exceeds an adaptively chosen threshold based on the overall noise level.

Matching Power Traces to a Template. To match new traces to the prepared templates, we follow the approach first introduced in [8]. A template for a single class consists of a mean trace and the pooled noise covariance matrix (for a comprehensive definition, we refer to [10]). In the attack phase, when measuring a power trace, we compute the probability of matching each template using the probability density function of the multivariate normal distribution.

Updating the Ranking for Multi-Trace Attacks. Starting with one trace, we obtain probabilities for matching each class, as explained before. Subsequently, we analogously compute the probabilities for the following trace and update the classification probabilities according to Bayes' theorem.

Correlation Power Analysis. CPA [5] has a very different concept, as the attacker obtains many power measurements here. The idea then is to test all possible hypotheses for a part of a key (e.g., a single coefficient) by correlating a power model of an intermediate value that depends on the targeted key part with the power traces. For this, the attacker either must be able to choose or at least know the public input, which is in contrast to the profiled SPA, where she also is required to know or choose *secret* inputs in the first phase. In our case, for a digital signature scheme, the model is either the known or chosen message attacker for the CPA.

Finally, the hypothesis with the highest absolute correlation coefficient is identified as the correct key part. For this, Pearson's correlation is used, i.e., the covariance of power model output and sample value normalized over the product of the standard deviations of each of the two. As significance bound, we use $\sqrt{28/N}$, where N is the number of processed traces [19].

Countermeasures. Many countermeasures have been proposed to mitigate side-channel attacks. The straightforward idea is to purposefully decrease the target devices' signal-to-noise ratio (SNR) (where the signal is the leaking information). For instance, this can be achieved by noise generators that run in parallel to sensitive operations [15]. However, this usually aims to increase the number of measurements required for an attack.

If the algorithm whose implementation is to be secured allows re-ordering of operations, *shuffling* [26] can be an option to counter single-trace SPA. By this, the attacker may be able to recover the secret value but not its position within the complete secret. For a CPA, shuffling only decreases the SNR because a certain fraction of the measurements will have the operation that leaks the secret aligned, with all other measurements being noise concerning the attack.

Thus, *masking* has been introduced [7, 14] to counter this attack as well, which has its foundations in Shamir's secret sharing. Here, a secret value x is split into multiple uniform random shares. Regarding PQC, the two most common masking schemes are *Boolean* and *additive* masking, splitting secrets either in Boolean or additive shares. In order to process secret data, any linear function in the masking domain can be performed share-wise. Non-linear functions have a higher complexity growth and usually require refreshing the mask(s) during intermediate steps.

Consequently, the CPA attacker does not obtain any information about the secret, as only uniform random values are processed. This, of course, is only true if the attacker is restricted to only one probe. Once she can probe both shares, she can perform the same attack again. It follows that the masking degree is always chosen according to a given attacker model.

3 Conceptual Considerations

The first reported implementation of the current specification was presented by Land et *al.* [18]. This implementation heavily depends on Digital Signal Processors (DSPs) that speed up the NTT significantly. However, it is relatively slow and big compared to newer implementations. Instead, we target the state-of-the-art implementation by Beckwith et *al.* [4]. We are aware of the more recent work by Zhao et *al.* [27], which was unavailable at the start of our work. However, since the operations we exploit are rather algorithmic-specific, we expect broad applicability of our techniques. In the following, we explain and analyze several operations within the target implementation.

3.1 Bit-Packing and Decoding of Secret Polynomials $\mathbf{s_1}, \mathbf{s_2}$

In general, the specification describes encoding as follows: An integer $x \in [-\eta, \eta]$ is packed as $\eta - x$ such that the encoded value is non-negative. Particularly, $\eta = 2$ for Dilithium security levels 2 and 5, and $\eta = 4$ for security level 3. Five consecutive resulting three-bit values are packed to three bytes for all parameter sets. In our target implementation, chunks of 64 bits are processed rather than single coefficients, which is implemented with a FIFO, and then four coefficients are decoded in parallel.

Since the implementation uses an unsigned representation, the decoding operation (a subtraction) is modulo q. Thus, the decoded values are either close to zero or close to q. This results in vastly different HWs for the cases depicted in Table 1. As can be seen there, the particular value $q = 2^{23} - 2^{13} + 1$ additionally enables a clear distinction between the low-HW outputs, $q - 2$, and the high-HW outputs. We expect that the significant differences in the HW to lead to a distinguishable amount of power consumption, enabling SPA attacks.

3.2 Number-Theoretic Transform

After unpacking the secret polynomials in $\mathbf{s}_1$ and $\mathbf{s}_2$, they are transformed into NTT representation. The NTT, as used in Dilithium, can be seen as a discrete Fourier transform over polynomials in $\mathcal{R}_q$, where modular arithmetic of the

Table 1: Hamming weight (HW) differences of decoded coefficients in $\mathbf{s}_1$ and $\mathbf{s}_2$

(a) $\eta = 2$

in	out $= \eta -$ in mod q	HW(out)
0	`0x000002`	1
1	`0x000001`	1
2	`0x000000`	0
3	`0x7fe000`	10
4	`0x7fdfff`	22

(b) $\eta = 4$

in	out $= \eta -$ in mod q	HW(out)
0	`0x000004`	1
1	`0x000003`	2
2	`0x000002`	1
3	`0x000001`	1
4	`0x000000`	0
5	`0x7fe000`	10
6	`0x7fdfff`	22
7	`0x7fdffe`	21
8	`0x7fdffd`	21

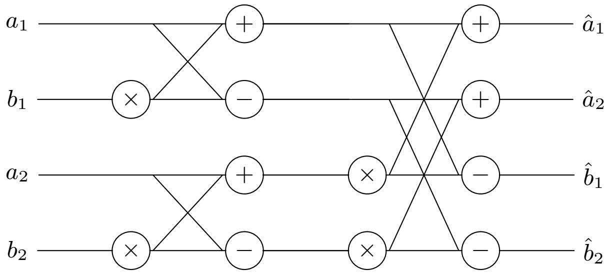


Fig. 1: 2x2 BFU construction

polynomial coefficients replaces the complex arithmetic. Since the ring structure enables negative wrapped convolution, we can use an n-point NTT for fast polynomial multiplication. For this, we transform both factor polynomials to the NTT domain, multiply coefficient-wise in the NTT domain, and then apply the inverse transform to the result to obtain the final product polynomial.

The core operation of the NTT is the so-called butterfly. Generally, the NTT is easily parallelizable and thus, it is possible to make a design choice of how many butterflies to instantiate. For the given $n = 256$, eight NTT layers must be processed. However, in the targeted implementation, a 2×2-Butterfly Unit (BFU) is deployed, which means that four butterflies are instantiated in a way that four input coefficients are processed first through two butterflies and then through the two others in order to perform two layers of NTT consecutively. This is depicted in Fig. 1. In the following, we refer to this as one stage of the NTT.

Note that for the butterfly, each output depends on all input values. Moreover, a_1 is spread without multiplication, b_1 is processed through one multiplication, a_2 through two multiplications, and b_2 through three. As the multiplications are with primitive roots of unity, which range over the whole $\mathbb{Z}_q$, intermediate values seem to be distributed uniformly in $\mathbb{Z}_q$ regardless of the input distribution. However, for $\mathbf{s}_1$ and $\mathbf{s}_2$ the input space to the first layer is bounded by η, which implicitly bounds the set of possible intermediate results and outputs of the BFU. We expect that this results in more distinguishable power signatures, facilitating more powerful SPA attacks.

3.3 Polynomial Multiplication

In Algorithm 2, we see that the secrets $\mathbf{s}_1$ and $\mathbf{s}_2$ are multiplied with the challenge polynomial c. If the signature candidate is accepted, the hash $\tilde{c}$ that is used to generate the challenge deterministically is part of the signature and thus publicly known. Besides, $\tilde{c}$ is the hash of μ, which directly depends on the message M, and $\mathbf{w_1}$. Therefore, for the deterministic signing procedure, c deterministically depends on the message. On the other hand, if randomized signing is deployed – introduced initially to counter fault attacks – c is also randomized even for a fixed message M through the randomization of $\mathbf{y}$, which is used to compute $\mathbf{w_1}$.

Moreover, the polynomial multiplications are performed in the NTT domain, which is essentially a coefficient-wise modular multiplication between $\hat{c}$ and the vectors $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$. This renders the aforementioned polynomial multiplications a natural target for a CPA attack since we can target the polynomial vector $\hat{\mathbf{s}}_1$ coefficient by coefficient.

The advantage of such an attack would be its weak attacker model. For the deterministic case, messages must be distinct, while for the randomized case there is no restriction on the messages. In both cases, though, the attacker must be able to trigger enough signings under the same secret key.

3.4 Measurement Setup

We perform all our attacks on a Xilinx Artix-7 100T FPGA – the hardware platform recommended by NIST for comparison of hardware implementations – running at 100 MHz. We measured the power consumption via peripheral components. Using an electromagnetic (EM) near-field probe, we measure the electromagnetic field of a capacitor on the board with a particularly low capacity of 47 nF. Since this capacitor is placed very close to the FPGA and in its power path, the capacitor's electromagnetic emanation directly depends on the power consumption of the FPGA. The advantage of this procedure is that no physical modifications are required on the target board. All measurements have been performed with 20 GS/s and a quantization of 12 bit.

4 Simple Power Analysis

In the following, we focus on the case $\eta = 2$ (Dilithium-2 and -5), which is more promising. Still, we evaluate and discuss the case $\eta = 4$ (Dilithium-3) at the end of this section.

4.1 Targeting Single Coefficients

As a first step towards a practical attack, we target single coefficients. We start by applying an attacker model, in which three out of the four secret coefficients decoded simultaneously are known, and the other one is attempted to recover. In practice this means that during the profiling phase, the attacker builds the templates knowing the three other secret coefficients. This results in less noise compared to the more realistic scenario in which the attacker does not know the other coefficients and thus would choose them randomly.

Interestingly, our countermeasures work also against this attacker. This results in an extended efficacy guarantee by deducting that the countermeasures effectively hinder *any weaker* SPA attacker, i.e., also the attacker that does not know the other three coefficients.

Attacking the Decoding Step. For this, we measure 55 000 traces, using a secret key as input fixed for all coefficients but one chosen randomly. We divide this trace set into the profiling set of 50 000 traces and the attack set of 5 000 traces. Subsequently, we prepare templates for three different attacks:

1. Five classes, aiming for the classification of the exact coefficient value
2. Four classes, aiming to distinguish between input classes
   - 0, 1 (yielding output HW 1)
   - 2 (yielding output HW 0)
   - 3 (yielding output HW 10)
   - 4 (yielding output HW 22)
3. Three classes, aiming to distinguish between input classes
   - 0, 1, 2 (yielding output HW 1 or 0)
   - 3 (yielding output HW 10)
   - 4 (yielding output HW 22)

Finally, we perform the three attacks on each subset of the attack set with the same key, obtaining the single-trace success probabilities.

As can be seen in Table 2, the results match the expectations, and classification works best for the case where three classes each internally have a very similar HW, recovering with high probability whether the targeted output is 4 or 3 or a member of the set $\{0, 1, 2\}$. Nevertheless, the classification model with the worst results, which is finding the exact coefficient value, also classifies each class correctly with a significantly higher probability than guessing, which would be 20 %.

When extending this attack to the multi-trace setting, the picture changes drastically. After at most 130 traces only, we can recover the correct coefficient for all classes.

Attacking the First NTT Layer. As explained before in Section 3.2, the four input coefficients to the BFU propagate differently as a_1 is only added or subtracted, while the others are also multiplied. We expect to classify coefficients for attacking this first NTT stage better than for targeting the decoding. That is because a small set of potential inputs is multiplied and reduced with the same constants, which results in a more diversified power signature and thus can be classified easier.

Table 2: Success rates of single-trace SPA on the decoder

Class					Avg.
0	1	2	3	4	
48.8 %	34.7 %	49.5 %	80.4 %	99.4 %	64.1 %
64.6 %		57.7 %	86.0 %	99.3 %	74.4 %
92.9 %			88.1 %	99.4 %	93.2 %

Table 3: Success rates for attacking the first NTT stage in the single- and multi-trace setting for $\eta = 2$ and $\eta = 4$

Target	$\eta = 2$							$\eta = 4$	
	Class					**Avg.**	**Multi-t.:**	**Avg.**	**Multi-t.:**
	0	1	2	3	4		**# Traces**		**# Traces**
a_1	60.1 %	59.1 %	92.2 %	89.6 %	97.8 %	79.8 %	34	57.3 %	87
b_1	89.1 %	88.4 %	100.0 %	89.3 %	92.4 %	91.8 %	4	74.5 %	10
a_2	83.5 %	88.1 %	93.8 %	96.6 %	100.0 %	92.5 %	4	84.0 %	45
b_2	88.0 %	90.2 %	99.8 %	94.6 %	97.7 %	94.2 %	3	76.2 %	23
Avg.	80.2 %	81.5 %	96.5 %	92.5 %	97.0 %	89.6 %		73.0 %	

The results in the left part of Table 3 show that the expectations again are met. Overall, this attack yields better results for all classes, as now, we can recover single coefficients that are processed as b_1, a_2, b_2 with a probability of over 90 %. In contrast, as expected, a_1 can be recovered with a lower probability.

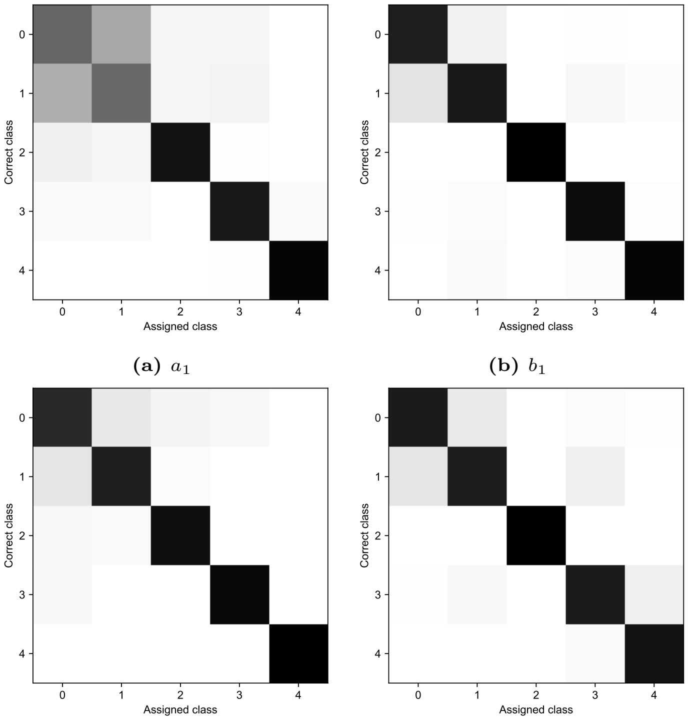
Furthermore, Fig. 2 visualizes the results of the single-trace attacks. The confusion matrices depict the probabilities of assigning each class during the attack phase given each (known) correct class. There, the darkness of a square quantifies the probability that, given the correct class for a trace (y-axis), a specific class (x-axis) has been assigned by the attack. As shown in Fig. 2a, the attack on a_1 mainly confuses class 1 for class 0 with low probability while correctly classifying all other classes with high probability. Note that the diagonals in Fig. 2 represent the single rows in Table 3.

For the multi-trace setting, Table 3 also shows how many traces are required to recover the correct coefficient with 100 % probability. This demonstrates the power of this attack, which requires at most 34 traces to recover any secret coefficient.

4.2 Extension to Multiple Coefficients

We extend our approach of targeting a single secret coefficient on the first NTT stage to attacking two coefficients simultaneously. A straightforward approach here would be to target all possible 5^4 combinations of (a_1, b_1, a_2, b_2). However, this would be a computationally very complex approach. Instead, we only target the first half of the BFU. The same operation is applied to the input tuples (a_1, b_1) and (a_2, b_2) independently. Thus, we can classify each possible input tuple by targeting $5 \times 5 = 25$ classes instead of 5^4. This comes at the cost of more profiling traces. Here, we require a profiling trace set with chosen secret coefficients, where (a_2, b_2) are kept steady for attacking (a_1, b_1) and vice versa. We increase the number of traces to 375 000 and divide them into 350 000 profiling traces and 25 000 attack traces to ensure the same number per class as for targeting single coefficients.

Fig. 3b shows the confusion matrix of this attack. As can be seen there, this attack succeeds with a high probability of assigning the correct class (the diag-



(a) a_1 **(b)** b_1

(c) a_2 **(d)** b_2

Fig. 2: Single-trace SPA confusion matrices for attacks on the first NTT stage with $\eta = 2$

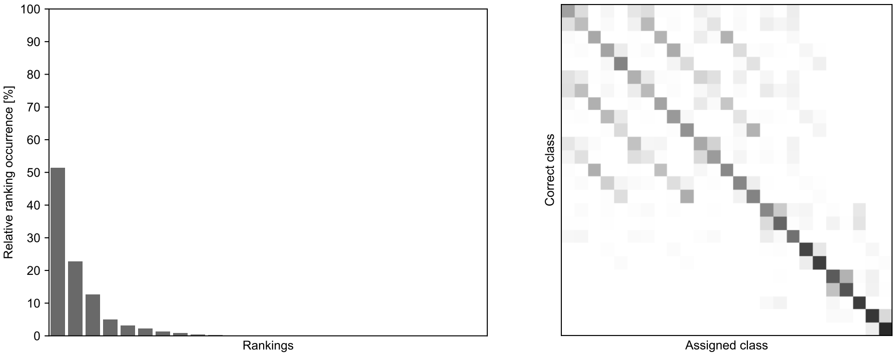
onal) but also shows some symmetry for assigning wrong classes, primarily due to confusing (a_1, b_1) with (b_1, a_1). On average, the attack succeeds in classifying the correct tuple with a probability of 51.5 %, vastly better than guessing, which would have a probability of 1/25. Moreover, in Fig. 3a, we see that the correct guess is within the top 5 with an overwhelming probability of 94.8 %.

Ultimately, we have also performed this attack in the multi-trace setting. Here, we are able to recover the correct combination of both secret coefficients after 1 101 traces. Using this approach, an attacker in the profiled SPA setting is able to recover the full secret polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$ with 700 000 profiling traces. In particular, the attacker would profile the device under test with 350 000 traces for all possible combinations of (a_1, b_1) and repeat the same for all possible combinations of (a_2, b_2). Then, according to our experiments, the device would be queried to perform 1 101 signing procedures (processing the secret key) and measure the first NTT stage of all secret polynomials either for $\mathbf{s}_1$ or $\mathbf{s}_2$ to recover it.

4.3 Attack on $\eta = 4$

For security level 3, where $\eta = 4$, the amount of classes increases from 5 to 9. The possible output HWs are shown in Table 1b. Similar to the results in Table 2, we can clearly distinguish between all groups with similar output HW when targeting the decoding. A multi-trace attack on the decoding finds the correct coefficient after 2 267 traces, compared to 130 for $\eta = 2$. This demonstrates that the increased number of possible coefficient values with similar HW downgrades the attack.

Targeting the BFU, we have performed experiments using 90 000 traces for profiling (i.e., 10 000 per class as for $\eta = 2$). The results are shown in the right part of Table 3. As expected and as it is the case for $\eta = 2$, the attack works better than those on the decoding, being capable of recovering the correct coefficient after one trace with a significantly higher probability than guessing, which would be 1/9. In the multi-trace setting, classifying the correct coefficient is possible after at most 87 traces. Overall, the SPA on Dilithium-3 is less feasible compared to the other parameter sets.



(a) Ranking distribution **(b)** Confusion matrix

Fig. 3: Single-trace SPA results for NTT inputs a_1 and b_1.

Table 4: Success probabilities for single-trace SPA on the combined a_1, b_1.

		b_1				
		0	1	2	3	4
a_1	0	37.1 %	25.8 %	34.1 %	35.6 %	48.8 %
	1	30.9 %	27.2 %	36.1 %	40.2 %	42.8 %
	2	34.4 %	39.4 %	46.1 %	46.9 %	48.2 %
	3	46.6 %	60.2 %	55.7 %	73.3 %	75.5 %
	4	64.1 %	66.9 %	76.3 %	78.5 %	83.2 %

5 Correlation Power Analysis on the Polynomial Multiplication

In addition to our SPA, we perform a CPA on the polynomial multiplication module, employing a weaker attacker model, as explained in Section 3.3.

For this attack, we observe many signature generations under the same secret key, and then, given the public challenge polynomial c, we target the pointwise multiplication $\hat{c} \circ \hat{\mathbf{s}}_1$. In this attack, we cannot exploit that each coefficient of $\mathbf{s}_1$ has a bounded norm since, during multiplication, the polynomial is processed in the NTT domain. Therefore, we have q hypotheses per coefficient in general.

5.1 Power Model

As a first approach, we chose to employ a HW model. As can be seen in Fig. 4, we show that the correct hypothesis reaches the first rank after 80 000 traces. However, there are also multiple wrong hypotheses exceeding the significance bound significantly.

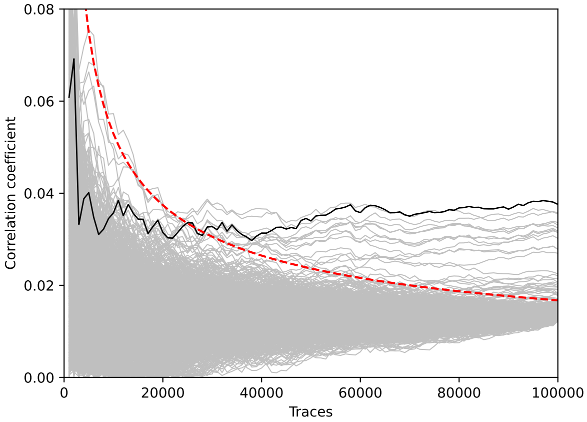
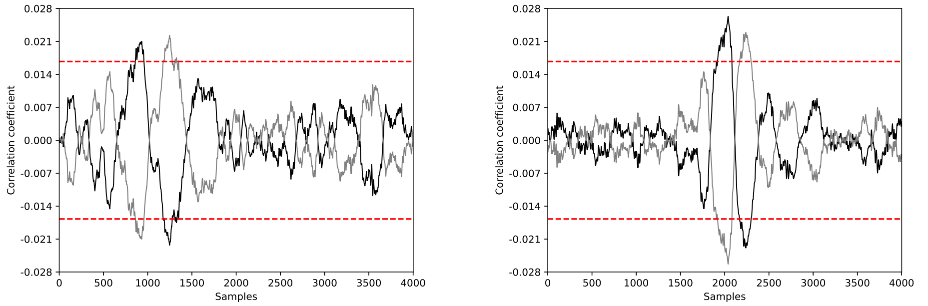


Fig. 4: CPA results – HW model, 1 000 most promising hypotheses shown, correct hypothesis in black, targeting $\hat{c} \circ \hat{\mathbf{t}}_0$

Instead, we adapt an idea from [9, Sec. III.B], where a software implementation is attacked and the hypothesis space is reduced by using the correlation peak polarity as additional information. We identify that targeting the least signification bit (LSB) of the product between the challenge polynomial coefficient and the hypotheses yields better results (i.e., no wrong hypotheses exceeding the significance bound significantly). Moreover, this approach allows cutting the number of hypotheses in half, resulting in a computationally less complex attack. We observe that for each hypothesis $h \in \mathbb{Z}_q \setminus \{0\}$ and each challenge polynomial coefficient $\hat{c}_i \in \mathbb{Z}_q \setminus \{0\}$ of the challenge $\hat{c}$, the following equation holds:

$$\mathsf{lsb}(\hat{c}_i \cdot h \bmod q) = 1 \oplus \mathsf{lsb}(\hat{c}_i \cdot (-h) \bmod q) \tag{1}$$

It follows that for this power model, the hypotheses h and $-h \bmod q$ yield inverted correlations. This can be used to halve the number of possible hypotheses to the range $[0, \lfloor q/2 \rfloor]$ by the following procedure. Fig. 5a shows the correlation of the LSB of the public coefficient $\hat{c}_i$ and the correlation with the LSB of the negative value. Note how there is first a positive peak and then a negative peak for the known $\hat{c}_i$. Correlations with other coefficients of the public polynomial c might also show an inverse peak polarity: first negative, then positive. In any case, the information of the correlation peak polarity is purely based on *public* information, and thus can be computed by the attacker in any case.



(a) Correlation of LSB of $\hat{c}_i$ (black) and $q - \hat{c}_i$ (gray)

(b) Correlation of LSB of $\hat{c}_i \cdot h \bmod q$ (black) and $\hat{c}_i \cdot (-h) \bmod q$ (gray)

Fig. 5: Correlation for 100 000 traces of the LSB of $\hat{c}_i$ and $\hat{c}_i \cdot h \bmod q$. For the highlighted (black) case, h is the correct hypothesis since both have a positive peak first, then a negative one.

Fig. 5b then shows very similar behavior for the correlation of the LSB of $\hat{c}_i \cdot h \bmod q$ and $\hat{c}_i \cdot (-h) \bmod q$. Our observation now is that if the correlation peak polarity is the same for the power correlation of $\hat{c}_i$ and $\hat{c}_i \cdot h \bmod q$ (where h is the hypothesis that yields the highest absolute correlation), h is the correct hypothesis. Otherwise, if the peak polarity does not match, $q - h$ is the correct hypothesis.

Thus, the attacker only needs to compute the correlations for half the hypotheses and then, after finding a hypothesis h with maximum absolute correlation coefficient, decides between h and $q - h$ based on whether the respective $\hat{c}_i$ yields

1. a positive, then a negative correlation peak. Then, if h yields
   (a) a positive, then a negative correlation peak, h is the sought coefficient.
   (b) a negative, then a positive correlation peak, $q-h$ is the sought coefficient.
2. a negative, then a positive correlation peak. Then, if h yields

(a) a positive, then a negative correlation peak, $q-h$ is the sought coefficient.
(b) a negative, then a positive correlation peak, h is the sought coefficient.

5.2 Noise

In the targeted implementation, the Keccak core works during all multiplications that include $\mathbf{s}_1$ or $\mathbf{s}_2$. This core generates most of the design's power consumption. This causes the problem that a lower quantization precision is left for the targeted value since the Keccak power consumption is noise to it. Both issues lead to requiring an increased number of traces for an attack. Thus, we investigate the attack in two different scenarios:

1. Evaluate $\hat{c} \circ \hat{\mathbf{t}}_0$, where no Keccak runs in parallel, and
2. Evaluate $\hat{c} \circ \hat{\mathbf{s}}_1$.

Compared to the first scenario, the concurrently operating Keccak module reduces the SNR by a factor of 25.

Therefore, the first scenario is a low-noise setting, and the second one is a high-noise setting, enabling a clear comparison between both. We expect that opening the FPGA packaging and probing the polynomial multiplication module locally using an EM near-field probe would result in a similar low-noise setting as for the first scenario.

5.3 Attacks

When targeting $\hat{c} \circ \hat{\mathbf{t}}_0$, we are able to recover the correct coefficients of $\hat{\mathbf{t}}_0$ after 66 000 traces, as can be seen in Fig. 6a. Moreover, after 22 000 traces, the correct hypothesis is within the top 2048 candidates, and after 57 000 traces, it is within the top 32 candidates.

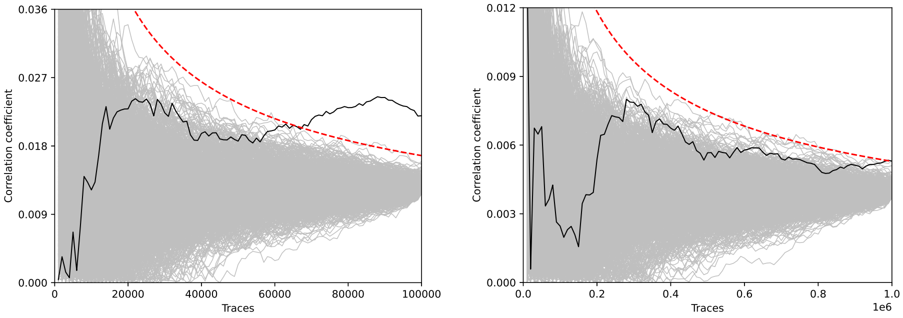
In Fig. 6b, it can be seen that the very same approach is becoming more difficult for attacking $\mathbf{s}_1$ for the reasons mentioned above due to a decreased SNR. Still, after 1 million traces, we can recover the correct coefficient. For this attack, the correct hypothesis is in the top 2048 after 240 000 traces and the top 32 after 850 000 traces.

In summary, it is possible to recover the secret in any case, even assuming a high-noise setting. Moreover, no invasive methods, such as opening the FPGA packaging, are required, which would be a much more specialized attack measuring the direct near-field EM emanation of the polynomial multiplication module. Finally, we want to stress that, contrary to the SPA, this attack works independently of η and thus is equally applicable to all security levels.

6 Countermeasures

6.1 Integration of Decoding into the First NTT Stage

Decoding the secrets $\mathbf{s}_1$ and $\mathbf{s}_2$ is an affine operation and thus can also be processed easily in a later phase of signature generation. Therefore, our first ap-



(a) Targeting $\hat{c} \circ \hat{\mathbf{t}}_0$ **(b)** Targeting $\hat{c} \circ \hat{\mathbf{s}}_1$

Fig. 6: CPA results – LSB model, 1 000 most promising hypotheses shown, correct hypothesis in black

proach aims at removing the parts of the decoder unit that process the targeted secret coefficients and integrate the decoding step into the first level of the NTT.

As explained before, we assume that the leakage of the decoding mainly depends on the differences of the HWs of the decoded values. Therefore, keeping all processed coefficients at a similar level of HW would be advantageous. We integrate the decoding into the BFU by feeding $q + \eta - x$ into each BFU input, where x is an encoded coefficient.

6.2 Masking

Masking must be deployed to counter both attacks through a comprehensive countermeasure. A comprehensive masking approach, where secret data is never processed nor transferred unmasked, requires that the secret key is already masked in the first place. In particular, we have the option to either apply arithmetic or Boolean masking. Applying arithmetic masking on $\mathbf{s}_1$ and $\mathbf{s}_2$, however, is not possible *efficiently* as it would induce an unnecessary high overhead factor for storing the masked key, since the coefficients are uniformly bounded by η rather than uniform in $\mathbb{Z}_q$. Because memory is an expensive resource on embedded hardware devices and the masking shares pose an overhead already, a masking countermeasure that requires no additional overhead would be desirable.

Specifically, in a real-world device, the secret key usually would be stored in a permanent memory outside of the FPGA, which would then have to be dimensioned bigger by a factor of 23/3 to account for the larger shares, and the key transfer would take equivalently longer compared to Boolean masking. The problem intensifies when the system includes multiple keys. In this case, external memory is virtually inevitable. Moreover, a smaller arithmetic masking domain could also be used, but this would also require a similarly expensive masking conversion compared to our proposal. Thus, only Boolean masking is feasible, which raises the necessity of converting efficiently from the encoded, Boolean-masked representation of $\mathbf{s}_1$ and $\mathbf{s}_2$, to a decoded and arithmetically masked representation.

Algorithm 3 First-order secure combined masking conversion and decoding, adapted from [12, Alg. 12]

```
Require: $b_0, b_1$ such that $b = b_0 \oplus b_1$
Ensure: $a_0, a_1$ such that $a = a_0 + a_1 = \eta - b \bmod q$
1: $X, R \leftarrow \mathbb{Z}_q \times \mathbb{Z}_{2^{23}}$
2: $Y_0 := ((X - \eta) + (2^{23} - q)) \oplus R$
3: $Y_1 := R$
4: $Z_0, Z_1 \leftarrow \mathsf{SecAdd}_q((b_0, b_1), (Y_0, Y_1))$    ▷ instantiate with $\mathsf{SecAdd}_q$ from [12, Alg. 8]
5: return $a_0 = X, a_1 = q - (Z_0 \oplus Z_1)$
```

As already introduced in [3] and further developed in [12], an efficient conversion from Boolean to arithmetic masking modulo q can be performed using a secure adder over Boolean shares, which have been studied extensively in [2,24]. It is possible to adapt this procedure to integrate the decoding step into the masking conversion.

The original idea from [12] is to sample a uniform random $A \in \mathbb{Z}_q$, then generate a fresh Boolean sharing of $(q - A) + (2^{23} - q)$ and add this with a secure adder as described in [12, Alg. 8] to the masked input. Note that in order to enable an easy reduction modulo q, this secure adder has the special property to subtract an additional constant of $2^{23} - q$, which explains the uncommon form of the input. The unmasked result of this operation then is one arithmetic share, and A is the other.

Instead, to include the decoding into the masking conversion, we adapt this procedure as shown in Algorithm 3:

1. We need two statistically random integers for the conversion, as shown in Line 1.
2. We generate a fresh Boolean sharing of $(X - \eta) + (2^{23} - q)$ in Lines 2 and 3 using R and X. Note that this operation can also be done offline or – for hardware – in parallel.
3. In Line 4, the Boolean masked input coefficient is added to the constructed Boolean sharing using the aforementioned special adder [12, Alg. 8], yielding a Boolean sharing of $X - \eta + 2^{23} - q + b - (2^{23} - q) = X - \eta + b$. Since X is uniformly random, it serves as an arithmetic mask and we can unmask the Boolean sharing without revealing the secret b.
4. In order to obtain a valid arithmetic sharing of $\eta - x$, we need to subtract the unmasked result from q, resulting in $\eta - b - X \bmod q$. Setting X as the other arithmetic share, we have completed the conversion with implicit decoding.

Following this, we can perform all linear operations in the masking domain simply by applying the function to each share. This includes both the NTT and multiplication with non-secret values like c. An implementation of this approach requires two different secure adders over Boolean shares:

1. For Step 1 in [12, Alg. 8], a 3 plus 23 bit adder is required.
2. For Step 4 in [12, Alg. 8], a 23 plus 23 bit adder with 12 of the input bits being hardcoded to zero, which enables substantial improvements compared to a generic secure adder

Note that this approach is not restricted to hardware implementations alone, but could very well also be done efficiently in a software implementation. For this, a secure bit-sliced adder, as proposed by [6], could be utilized, enabling parallelized processing of 32 or more coefficients.

It is possible to adapt this approach to an arbitrary masking order. For this, [6, Alg. 11] can be modified analogously to our method above. This requires an additional arithmetic-to-Boolean conversion to generate the Boolean sharing from Lines 2 and 3 in Algorithm 3. The additional conversion can be also performed offline and does not induce a further delay, even for higher orders.

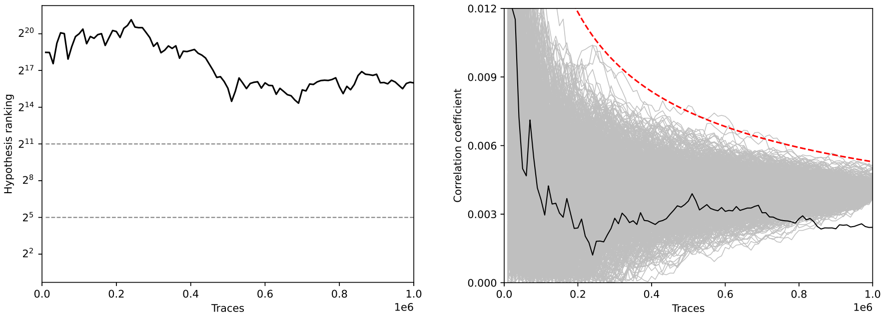
6.3 Evaluation

Decoding in the First NTT Stage. Integrating the decoding into the first NTT stage obviously eliminates the possibility of attacking the decoding as a standalone step. Nonetheless, we evaluate the effect of this countermeasure on the leakage of the BFU by performing the same single-coefficient attacks as explained before. Table 5 shows the attack's results compared to Table 3. Notably, even though the countermeasure is not intended to prevent this attack, it mitigates the SPA on the BFU. Additionally, the number of traces required to recover the coefficients is doubled. We suppose that Table 5 quantifies the impact of the diverse HWs of the first NTT stage while not altering the diversification of the power signature after the arithmetic operations.

Arithmetic Masking. We also evaluate the efficacy of arithmetic masking against the SPA and the CPA. First, we test whether the exact same CPA works as before. Fig. 7 shows the results for the low-noise setting that targets $\hat{c} \circ \hat{\mathbf{t}}_0$. As seen there, the correct hypothesis stays at about the same rank even after 1 million traces. Also, the absolute correlation does not come close to the

Table 5: SPA results on BFU with integrated decoding given as percent points (resp. difference of traces required in the multi-trace setting) with the $\eta = 2$ part of Table 3 as reference

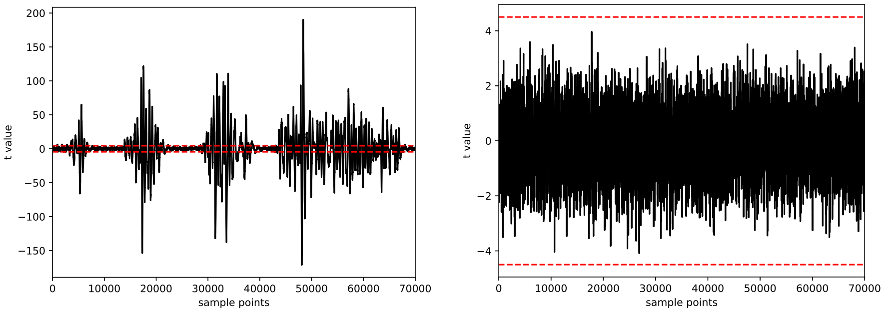
Target	Class 0	1	2	3	4	Average	Δ#Traces
a_1	-3.4 %	-3.8 %	+2.9 %	-18.0 %	-8.4 %	-5.7 %	+31
b_1	-23.0 %	-5.6 %	-17.7 %	-14.7 %	-14.1 %	-15.1 %	+3



(a) Correct hypothesis ranking progression

(b) Absolute correlation progression for the 1 000 most promising hypotheses (gray) and the correct hypothesis (black)

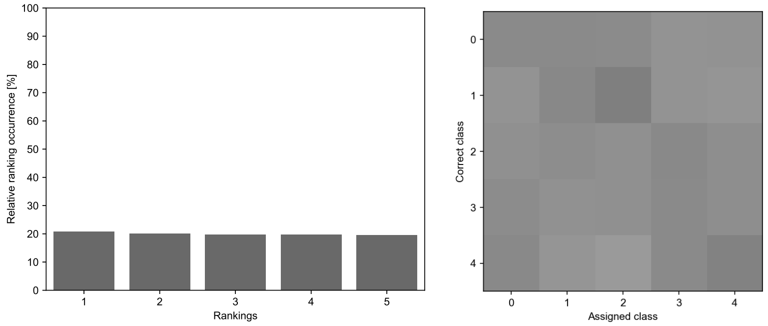
Fig. 7: CPA results for multiplication of $\hat{c}$ with masked $\hat{\mathbf{t}}_0$ for 1 000 000 traces



(a) Masking deactivated, 100 000 traces

(b) Masking activated, 1 000 000 traces

Fig. 8: Fixed-vs-random t-test for NTT



(a) Ranking distribution

(b) Confusion matrix

Fig. 9: SPA on NTT with masking, cf. Fig. 2

higher-ranked hypotheses or even the significance threshold. Since the attack does not work in the low-noise setting, we deduct that it does also not work when the Keccak module produces noise in parallel.

Then, to evaluate the effect of masking on the SPA, we perform a standard test-vector-based leakage assessment using a fixed-vs-random t-test on the NTT. As can be seen in Fig. 8, the masking effectively hinders any distinction between fixed and random input even after 1 000 000 traces.

Finally, we also attempt to perform the SPA on the whole BFU for single coefficients. For the evaluation, we increase the number of traces to 450 000 instead of 50 000 during the profiling phase, employ the same overly powerful attacker as before and confirm that the attack is not successful anymore. The respective confusion matrix and ranking distribution can be found in Fig. 9. From this, we deduce that no weaker SPA attacker can learn anything about the secret, e.g., also the one we present in Section 4.2. Finally, we could not recover any coefficient using a multi-trace attack with up to 10 000 traces per class.

7 Discussion and Future Work

Our work presents a first side-channel analysis of Dilithium in hardware. We demonstrate attack surfaces and feasibility for single- and multi-trace profiled SPA attacks, targeting the decoding of the secret polynomials and the first NTT stage. Beyond this strong attacker model of profiled SPA, we show a practical CPA attack on polynomial multiplication using power measurements. Regarding the applicability of these attacks on other implementations, we can summarize our findings as follow:

1. The SPA on the decoding exploits the specified range of the secret coefficients and their HW, which does not depend on our targeted implementation. Thus, we expect that the same attack surface exists for any implementation.
2. The SPA on the NTT similarly exploits the secret key range, benefitting from the more unique power signatures generated by the BFU. Following this, we expect that the attack works similarly for the implementations [18, 27], which also contain BFUs (as necessary for computing an NTT). However, the co-processor [18] detaches a "pre-computation" step from the signing procedure, which performs the NTT of the secrets once and then stores the transformed polynomial vectors for all subsequent signings under the same secret key. This could mitigate the SPA attack by potentially preventing the collection of multiple traces of the NTT transformation on $\mathbf{s}_1$ and $\mathbf{s}_2$.
3. The CPA on the polynomial multiplication is rather generic, as all implementations will perform the polynomial multiplication using the NTT, even though the specification does not strictly require it.

Moreover, our work shows that random noise generated by a Keccak module running in parallel to the multiplication does not effectively hinder either attack. Finally, we also present countermeasures and evaluate that arithmetic masking effectively prohibits all presented attacks.

For future work, we leave both higher-order attacks and efficient higher-order masking conversions with integrated decoding open. On a higher level, a complete masked hardware implementation of Dilithium is desirable.

Acknowledgments

We thank the reviewers for their constructive comments. Furthermore, we thank Pascal Sasdrich for the fruitful discussions. This work was supported by the German Research Foundation under Germany's Excellence Strategy – EXC 2092 CASA – 390781972, through the H2020 project PROMETHEUS (grant agreement ID 780701), CONVOLVE (grant agreement ID 101070374), and by the Federal Ministry of Education and Research of Germany through the QuantumRISC (16KIS1038), PQC4Med (16KIS1044), and 6GEM (16KISK038) projects.

References

1. Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Markus Schönauer, Tobias Schneider, François-Xavier Standaert, and Christine van Vredendaal. Protecting dilithium against leakage: Revisited sensitivity analysis and improved implementations. Cryptology ePrint Archive, Paper 2022/1406, 2022. `https://eprint.iacr.org/2022/1406`.
2. Florian Bache and Tim Güneysu. Boolean masking for arithmetic additions at arbitrary order in hardware. *Applied Sciences*, 12(5):2274, 2022.
3. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 354–384, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.
4. Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. High-performance hardware implementation of crystals-dilithium. In *International Conference on Field-Programmable Technology, (IC)FPT 2021, Auckland, New Zealand, December 6-10, 2021*, pages 1–10. IEEE, 2021.
5. 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.
6. 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.
7. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

8. 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.
9. Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma, and Jiwu Jing. An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In *39th IEEE International Conference on Computer Design, ICCD 2021, Storrs, CT, USA, October 24-27, 2021*, pages 583–590. IEEE, 2021.
10. Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.
11. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium – Algorithm Specifications and Supporting Documentation (Version 3.1). Technical report, 2021. `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`.
12. Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. Masked accelerators and instruction set extensions for post-quantum cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):414–460, 2022.
13. Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 15–29, Yokohama, Japan, October 10–13, 2006. Springer, Heidelberg, Germany.
14. Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Heidelberg, Germany.
15. Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 33–48, Nara, Japan, September 28 – October 1, 2011. Springer, Heidelberg, Germany.
16. Emre Karabulut, Erdem Alkim, and Aydin Aysu. Single-trace side-channel attacks on ω-small polynomial sampling: With applications to ntru, NTRU prime, and CRYSTALS-DILITHIUM. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2021, Tysons Corner, VA, USA, December 12-15, 2021*, pages 35–45. IEEE, 2021.
17. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.
18. Georg Land, Pascal Sasdrich, and Tim Güneysu. A hard crystal - implementing dilithium on reconfigurable hardware. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Con-*

ference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers, volume 13173 of *Lecture Notes in Computer Science*, pages 210–230. Springer, 2021.

19. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
20. 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`.
21. 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.
22. 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`.
23. Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Exploiting determinism in lattice-based signatures: Practical fault attacks on pqm4 implementations of NIST candidates. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *ASIACCS 19: 14th ACM Symposium on Information, Computer and Communications Security*, pages 427–440, Auckland, New Zealand, July 9–12, 2019. ACM Press.
24. Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic addition over Boolean masking - towards first- and second-order resistance in hardware. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 559–578, New York, NY, USA, June 2–5, 2015. Springer, Heidelberg, Germany.
25. 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.
26. 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, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.
27. Cankun Zhao, Neng Zhang, Hanning Wang, Bohan Yang, Wenping Zhu, Zhengdong Li, Min Zhu, Shouyi Yin, Shaojun Wei, and Leibo Liu. A compact and high-performance hardware architecture for crystals-dilithium. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):270–295, 2022.