Profiling Dilithium Digital Signature Traces for Correlation Differential Side Channel Attacks

Apostolos P. Fournaris[1](✉), Charis Dimopoulos[2], and Odysseas Koufopavlou[2]

[1] Industrial Systems Institute/Research Center ATHENA, Patras Science Park, Patras, Greece
fournaris@isi.gr

[2] Electrical and Computer Engineering Department, University of Patras, Rion Campus, Patras, Greece

Abstract. A significant concern for the candidate schemes of the NIST postquantum cryptography standardization project is the protection they support against side-channel attacks. One of these candidate schemes currently in the NIST standardization race is the Dilithium signature scheme. This postquantum signature solution has been analyzed for side channel attack resistance especially against timing attacks. Expanding our attention on other types of side-channel analysis, this work is focused on correlation based differential side channel attacks on the polynomial multiplication operation of Dilithium digital signature generation. In this paper, we describe how a Correlation Power Attack should be adapted for the Dilithium signature generation and describe the attack process to be followed. We determine the conditions to be followed in order for such an attack to be feasible, (isolation of polynomial coefficient multiplication inpower traces) and we create a power trace profiling paradigm for the Dilithium signature scheme executed in embedded systems to showcase that the conditions can be met in practice. Expanding the methodology of recent works that mainly use simulations for power trace collection, in this paper, power trace capturing and profiling analysis of the signature generation process was succesfully done on a, noisy, Commercial off-the-shelf ARM Cortex-M4 embedded system.

Keywords: Postquantum cryptography · Side channel attack · Secure embedded systems

1 Introduction

The research efforts in recent years are increasingly targeted towards the realization of a quantum computer. By exploiting properties that stem from quan-

This paper's work has received funding from the European Union's Horizon 2020 research and innovation programme CPSoSaware under grant agreement No 873718 and also the European Union's Horizon 2020 research and innovation programme CONCORDIA under grant agreement No 830927.

A. Orailoglu et al. (Eds.): SAMOS 2020, LNCS 12471, pp. 281–294, 2020.
https://doi.org/10.1007/978-3-030-60939-9_19

tum mechanics, these machines are able to bypass performance barriers traditional computers face. One key remark is that quantum computers are not able or designed to replace the existing computing paradigm, but to provide vast amounts of performance boost in specific applications powered by novel algorithms.

In the field of cryptography, Shor's algorithm [21] is able to solve the integer factorization problem in polynomial time when executed in a quantum computer thus breaking the majority of the public-key cryptography schemes currently deployed in many infrastructures. As quantum computer development is accelerating, new and quantum resilient schemes have emerged to replace the no-longer safe to use public-key algorithms like DSA, RSA and Elliptic Curve Cryptography. These new schemes are based on hard problems that retain their resistance even when paired against a quantum computer. The importance of discovering optimal postquantum cryptographic algorithms is demonstrated by, e.g., NIST's current call for post-quantum secure proposals [2] and the official recommendations regarding post-quantum security from the NSA [6]. Still, a fundamental security consideration in the postquantum cryptography schemes that may make them unsafe for real-world use is their vulnerabilities linked with the underlying implementation due to possible leakage from Side Channels (eg. power consumption or electromagnetic emission (EM) during processing, or timing delay). Side Channel leakage can be exploited by various relevant attacks including timing, power and EM attacks both simple or advanced ones []. Differential analysis attacks [11,13] constitute the most potent such attacks under an unknown implementation scenario while template [9,17] and Machine Learning attacks [15] are equally potent when a device under attack is in the full control of an attacker.

Lattice based cryptography (LBC) constitutes a very promising category of post-quantum cryptosystems that has a strong presence in the NIST postquantum cryptography contest [2] and can offer digital signatures, key exchange and encryption. Dilithium digital signature scheme [10] is one of the promising LBC proposal of the NIST content and can be implemented very efficiently since it has relatively small keys and parameters. It has been implemented with relative success in x86 based devices, in ARM based devices and also in hardware. There are several works that evaluate the Dilithium side channel attack resistance in emulated environments and provide generic leakage assessment results (eg. using the Welch t-test leakage assessment technique) but few works focus on specific attacks (like Differential Power Analysis or template based attacks [19]). Other works rely on protecting the scheme against side channel attacks (SCAs) but produce a considerable performance overhead [16]. It should be mentioned that very few works explore the potentials of DPA in the Dilithium Digital signature scheme in a practical level that include measurements. In general, works that collect side channel leakage from actual implementations, use dedicated side channel trace collection platforms like the Chipwhisperer boards [1] and not off-the-shelf devices that are found in real embedded systems [7,16]. Regarding Differential SCA on Dilithium, while the attack is considered possible due to the existence of linear operations in the Dilithium structure, there are no analytic and practical approaches on how to mount such an attack.

In this paper, the Dilithium signature generation operation is analyzed and interesting computation points are identified for mounting differential power analysis SCAs. The analysis is focused on the polynomial multiplication operation needed during the sample rejection loop of the algorithm (last loop iteration) regardless of the technique used to perform it (schoolbook polynomial multiplication, sparse matrix polynomial multiplication or Number Theoretic Transform (NTT) representation). Correlation Power Analysis (CPA) is a very promising type of attack to be mounted in the focused operation, so, in this paper, we describe how such an attack should be adapted for the Dilithium signature generation. It is concluded that a Dilithium based CPA attack is feasible, regardless of the polynomial multiplication technique that is used in the Dilithium implementation, as long as individual polynomial coefficient multiplication can be identified in the collected power traces. To evaluate the correctness of this assumption, in this paper, we showcase how to profile collected traces of Dilithium signature generation process even when those traces are captured from a Commercial, noisy, off-the-shelf (COTS) embedded system boards.The results verify that the identified, exploitable, interesting points are visible in the power trace signals and that they can be isolated so that CPA can be effectively mounted on Dilithium signature generation.

The rest of the paper is organized as follows. In Sect. 2 the Dilithium digital signature algorithm is described and how CPA can be mounted on the algorithm is presented in detail. In Sect. 3 the experimental process that was followed is presented and in Sect. 4 measurements for profiling Dilithium signature generation are described and analyzed. Finally, Sect. 5 concludes the paper.

2 Dilithium Postquantum Digital Signature Scheme

The Dilithium digital signature scheme [10] is one of the algorithms that participate in the second round in the NIST postquantum cryptography competition. Dilithium is a signature scheme based on the Fiat-Shamir with Abort framework [14] and is implemented with Modules. Thus, it can perform key generation, message signature generation and digital signature verification. The algorithm is using a variation of the Ring-LWE problem, i.e. the Module-LWE problem. It relies in rings R_q: $Z_q[X]/(X^n+1)$ where q is an integer. Thus, all elements in R_q are polynomials of degree n that have coefficients belonging to Z_q. In Dilithium scheme $q = 2^{23} - 2^{13} + 1 = 8380417$ and $n = 256$. Typically, as in all lattice based cryptography schemes, in Dilithium, the public key is a $k \times \ell$ matrix A where all its elements belong to R_q (thus $A \in R_q^{k \times \ell}$). However, Dilithium private keys S_1, S_2 are vectors of $R_q^{\ell \times 1}$ and $R_q^{k \times 1}$ respectively. A potential side channel attacker is expected to be interested in the key generation and most importantly in the message digital signature generation operation since in it, the secret key of the Dilithium scheme is used (the digital signature is verified using the public key so no sensitive information is involved in digital signature verification). Thus, in this paper, our analysis is focused on the Dilithium digital signature generation process (Algorithm 2). For the sake of completeness, the Key generation process is also presented (Algorithm 1). The algorithm uses a series of specialized truncation procedures i.e. ExpandA/Decompose/HighBits/LowBits/Power2Round/MakeHint. The reader can refer to the original Dilithium paper for more information on those functions [10]. Also, Dilithium has several parameters apart from q and n that can be tuned in order to provide different levels of security (weak, medium, high etc.). The parameters are presented in Table 1. As suggested by its creators, all polynomial multiplications in Dilithium are performed using NTT. For this reason, the table A, Y, W, Z as well as S_1 and S_2 are transformed in their NTT equivalent. However, there are works indicating that working in the normal domain could also be efficient in some corner cases.

Algorithm 1: Dilithium Key Generation Algorithm

```
Output: $P_{key} : (T_1, \rho)$ $S_{key} : (\rho, \hat{\rho}, S_1, S_2, T)$
1 Choose a random $\rho$ and $\hat{\rho}$ from $\{0,1\}^{256}$;
2 ExpandA to sample $A$ using XOF with seed $\rho$ ;
3 Randomly generate $S_1 \in R_\eta^{\ell \times 1}$ and $S_2 \in R_\eta^{k \times 1}$ using XOF with seed $\hat{\rho}$ ;
4 $T = A \cdot S_1 + S_2 \quad \in R_q^{k \times 1}$;
5 $(T_1, T_0) = \text{Power2Round}_q(T, d) \quad \in R_q^{k \times 1}$;
6 $P_{key} = (\rho, T_1)$;
7 $S_{key} = (\rho, \hat{\rho}, S_1, S_2, T_0)$;
8 return $P_{key}, S_{key}$
```

Algorithm 2: Dilithium Signature Generation Algorithm

```
Input: $S_{key} : (\rho, \hat{\rho}, S_1, S_2), T_0, m$
Output: $\sigma = (Z, H, C)$
1 ExpandA for $A$ using a XOF with input $\rho$ ;
2 $T_1 = \text{Power2Round}(T_0,\text{d})$;
3 Rejection sampling loop ;
4 $\rho' = \{0,1\}^{256}$;
5 $Y = Sample(\rho')$ using XOF ;
6 $W = A \cdot Y$;
7 $W_1 = HighBits(W, )$;
8 $C = H\{\rho, T_1, W, m\}$;
9 $Z = Y + C \cdot S_1$;
10 $R_0 = Lowbits(W - C \cdot S_2, 2\gamma_2)$;
11 if $||Z||_{\text{inf}} >= \gamma_1 - \beta$ go to 3 ;
12 if $||R_0||_{\text{inf}} >= \gamma_2 - \beta$ go to 3 ;
13 if $||C \cdot T_0||_{\text{inf}} >= \gamma_2$ go to 3;
14 $H = MakeHint(-C \cdot T_0, W - C \cdot S_2 + C \cdot T_0, 2\gamma_2)$;
15 return $\sigma = (Z, H, C)$
```

From an SCA adversary perspective, there are various computational points attacks can be mounted. In [16], the authors provide several such points of interest to be exploited in order to reveal the secret keys S_1 and S_2. More specifically, by attacking the rejection operation, an attacker can possibly extract partial information on the secret key during the key generation. Also in Algorithm 2, the matrix Y that is used for the signature generation or even on a rejected matrix Z can also leak information about the secret key [10]. Similarly, computations that are directly linked with S_1 and S_2 can also be targeted ie. the computation of $Z = Y + C \cdot S_1$ (targeting $C \cdot S_1$), $R_0 = Lowbits(W - C \cdot S_2, 2\gamma_2)$ (targeting $C \cdot S_2$) or $MakeHint(-C \cdot T_0, W - C \cdot S_2 + C \cdot T_0, 2\gamma_2)$.

Table 1. Dilithium parameters for various security levels [10]

	Weak	Medium	Recommended (high)	Very high
q	8380417	8380417	8380417	8380417
d	14	14	14	14
weight of c	60	60	60	60
$\gamma_1 = (q-1)/16$	523776	523776	523776	523776
$\gamma_1 = \gamma_1/2$	261888	261888	261888	261888
(k, ℓ)	(3,2)	(4,3)	(5,4)	(6,5)
η	7	6	5	3
β	375	325	275	175
ω	64	80	96	120
Public key size (bytes)	896	1184	1472	1760
Signature size (bytes)	1387	2044	2701	3366

2.1 On Side Channel Leakage of the Dilithium Secret Key

Inspecting the Dilithium specifications, reveals that the secret key S_1 and S_2 are polynomial vectors of ℓ and k elements respectively. Each polynomial, however, is not defined in the polynomial ring R_q but rather in $R_\eta : Z_\eta[X]/(X^n+1)$ where $n = 256$ and η is a Dilithium parameter that, as seen in Table 1, is a small integer number (3 to 7). So, practically, each polynomial in the secret key vector has 256 coefficients that each of them is an integer with values $\{-\eta, \eta\}$. Furthermore, in the Dilithium scheme specifications C is a n degree polynomial, deriving from a hash function (SHAKE256) sampling, that has 60 non zero coefficients of $\{-1, 1\}$. Given the above facts, computation in line 9 and line 10 of Algorithm 2 can become somewhat predictable as long as the computation of $C \cdot S_1$ and $C \cdot S_2$ can be identified in the side channel leakage traces. More specifically, focusing on the last iteration of the rejection sampling loop (where the sampling is not rejected and C is obtained), an attacker, by knowing the C value, can make a hypothesis on the value of each coefficient of S_1 or S_2 polynomial and compute by using a power model (Hamming weight or Hamming distance) this hypothesis expected leakage when $C \cdot S_1$ or $C \cdot S_2$ results are stored in some ARM processor's register. Then, using a distinguisher function (eg. Pearson Correlation), the attacker can evaluate if this hypothesis is correct using the collected leakage trace from the actual computation of $C \cdot S_1$ or $C \cdot S_2$. The above methodology constitutes the core of various divide and conquer Differential Side channel attacks (e.g Differential Power Analysis, DPA or Correlation Power Analysis, CPA) [8,13].

Let's assume a function $F(X)$ that consists of T different intermediate operations $f_i(X_i)$ where $i : \{1, ..T\}$ and each X or X_i is a set of $\mathbf{K}$ or k inputs respectively on $F()$ or $f_i()$ (eg. $X^{(i)} : \{I_1^{(i)}, I_2^{(i)}, I_3^{(i)}..I_k^{(i)}\}$). Consider also that some operation f_t uses as one of its input an unknown secret S to provide a result and that all other f_t inputs are known or can be computed from the provided $F(X)$ inputs X (ie.$f_t(X^{(t)})$ where $X^{(t)} : \{I_1^{(t)}, I_2^{(t)}..I_{k-1}^{(t)}, I_k^{(t)}\}$ and $I_k^{(t)} = S$). A Correlation based vertical side channel attack can be mounted as long as operation f_t can be identified in the $F(X)$ side channel signal.

A CPA can be mounted in three phases. Initially, we generate λ different, random, inputs X_j for $F(X)$, then perform $F(X_j)$ for each one of those inputs, collect the results and capture a side channel trace P_j for each one of those inputs ($j \in \{1, 2, 3, ..\lambda\}$). The samples on each one of the P_j traces that match the operation $f_t(x^{(t)})$ are isolated thus creating λ different $PK_j(f_t)$ useful traces with p samples each (one sample in each leakage trace is denoted in $PK_{j,v}$ where $v \in \{1, 2, ..d\}$). Apart from that, in the first phase, the intermediate inputs of f_t, denoted as $X_j^{(t)} : \{I_{1,j}^{(t)}, I_{2,j}^{(t)}..I_{k-1,j}^{(t)}\}$ ($I_{k,j}^{(t)} = S$ for all $j \in 1, 2, ..\lambda$), are computed for all X_j inputs or outputs of $F(X)$ (depending on how $F(X)$ is realized).

In the CPA second phase, the attacker creates a hypothesis h on the value of S, denoted S_h, for all possible X_j inputs of $F(X)$, then for each j input calculates the $f_t(X_{j,h}^{(t)})$ result where $X_{j,h}^{(t)} : \{I_{1,j}^{(t)}, I_{2,j}^{(t)}..I_{k-1,j}^{(t)}, S_{h,j}\}$ and generates its hypothetical side channel information using some model. In most cases, the dynamic power consumption leakage side channel information is used to mount an attack, so the Hamming Weight (HW) or Hamming distance (HD) model is adopted. Without loss of generality, in our analysis we adopt the Hamming Weight model, thus we compute the outcome of $f_t()$ for $X_{j,h}^{(t)}$ ($X_{j,h}^{(t)}$ is fully known if we include the hypothetical $S_{h,j}$ in the computation, and the hyphothetical leakage for $S_{h,j}$ would be $HWT_{j,h} = HW(f_t(X_{j,h}^{(t)}))$. The process is repeated with all possible h hypothesis of S thus m different $HWT_{j,h}$ hypothetical leakage values are obtained for each input $X_j^{(t)}$, where $h \in \{1, 2, ...m\}$. At the end of the second phase, $\lambda \times m$ different hypothetical leakage values are generated.

The third phase of the CPA attack, Pearson correlation is used as a distinguisher of false hypothesis based on the collected λ traces of d samples each. At the end of the computation, there should be one correlation value $r_{v,j}$ for each $v - th$ sample of each $j - th$ collected trace (where $v \in \{1, 2, ..p\}$ and $j \in \{1, 2, ..\lambda\}$. Pearson Correlation is performed using the following equation:

$$r_{v,j} = \frac{\sum_{w=0}^{\lambda}[(HWT_{w,v} - mean(HWT_{:,v})) \cdot (PK_{w,j} - mean(PK_{:,j}))]}{\sqrt{\sum_{w=0}^{\lambda}(HWT_{w,v} - mean(HWT_{:,v}))^2 \cdot \sum_{w=0}^{\lambda}(PK_{w,j} - mean(PK_{:,j}))^2}} \quad (1)$$

Thus, the outcome of the distinguisher operation would be a $p \times \lambda$ correlation matrix R. Each $j-th$ column of the R matrix corresponds to an S_j hypothesis. If some correlation value in the $j-th$ column is significantly higher that the other values on the same column then S_j hypothesis is correct (i.e the attacker winds the adversarial game by recovering S). In practice, a divide and conquer technique is applied in the above CPA approach, thus the attacker assumes that S can be partitioned into small components (eg. bytes) and focus his attack on an $f_t()$ operation that processes one S component at a time. Thus, CPA is repeated as many times as the S partitions.

Assuming that Dilithium signature generation acts as $F()$ operation, the various attack points identified in this paper and [16] can act as $f_t()$. Given the constrain that the $f_t()$ inputs be all known to the attacker apart from the secret S, the best match for CPA attack is the polynomial matrix multiplication of $C \cdot S_1$ or $C \cdot S_2$ (in line 9 and 10 of Algorithm 1 respectively) that happen at the last iteration of the rejection sampling loop (i.e where C, R_0 and Z are not rejected or the $C \cdot S_2$ in the $MakeHint$ operation (at line 14 of Algorithm 1). In all the above cases, C has a known value (is part of the digital signature σ) with $HW(C) = 60$ while the unknown secret S_1 or S_2 is a polynomial vector of ℓ or k elements that have small coefficients $(-\eta,))$. Thus, the CPA secret can be partitioned in individual polynomials and furthermore in individual coefficients so that using the divide and conquer technique the attacker can repeatedly retrieve individual coefficient values of S_1 or S_2 by partitioning, collecting traces and analyzing individual operations within $C \cdot S_1$ or $C \cdot S_2$ computation. Note, however, that this process is highly related to the technique used for performing the $C \cdot S_1$ or $C \cdot S_2$ computation. The first technique to be used, officially proposed in the Dilithium scheme specifications [10], is the number theoretic transform (NTT) technique, where all polynomial are transformed in the NTT domain, then their coefficients are multiplied and the result is transformed from the NTT to the normal domain. The second technique is the schoolbook and the sparse polynomial multiplication, on the normal polynomial domain, which, given the small values of the polynomial coefficients, might yield better performance in certain corner cases over area-constrained devices, due to its simplified control logic [12,18,20]. Additionally, since the signature component has to be generated in the normal domain, use of the schoolbook or the sparse polynomial multiplier technique does not need the NTT and inverse NTT computations.

Regarding the second technique, using the schoolbook method on polynomials $C(y)$ and $S_1(y)$ (or $S_2(y)$ respectively) can be performed if the following formula is used:

$$c(y) \cdot s_1(y) = \left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} c_i \cdot s_{1,j} x^{i+j}\right] mod \ \langle x^n + 1\rangle = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(-1)^{\lfloor \frac{i+j}{n} \rfloor} c_i \cdot s_{1,j} x^{i+j mod n} \quad (2)$$

Observing the above equation from the attacker's perspective reveals that as long as identifying and focusing on the side channel trace samples related to each computation of $c_i \cdot s_{1,j}$ is feasible, a CPA attack can be performed efficiently. Given that each c_i coefficient is a value between $(-1, 1)$ and that each $s_{1,j}$ is a value $(-\eta, \eta)$ the λ number of hypothetical values is small (based on Table 1 $\lambda \leq 15$) and the computations to find a hypothetical f_t result are simple. Thus, it can be concluded that this technique should be avoided in an unprotected Dilithium signature generation software or hardware implementation.

When the NTT technique is adopted in a Dilithium scheme implementation, both C and S_1 (or S_2) are represented in the NTT domain during the $C \cdot S_1$ or $C \cdot S_2$ computations. Assuming that $\hat{C}$ and $\hat{S_1}$ are the NTT versions of C and S_1 then performing $C \cdot S_1$ is as simple as performing $\hat{c_i} \cdot \hat{s_{1,j}}$ between the NTT coefficients of the polynomials and then applying a q modulo reduction on the result. These two operations are done sequentially thus each one of them may have a distinct presence in the power trace signal that can be assigned as $PK_{j,v}$. In fact, based on the conceptualization of CPA, an attacker is interested only on the operation $\hat{c_i} \cdot \hat{s_{1,j}}$ to act as $PK(f_t)$. However, in the NTT representation of all polynomial coefficients belong to Z_q (and not Z_η or Z_2) thus there will be $\lambda = q$ different hypothesis for $\hat{s_{1,j}}$ which may make the computational overhead of CPA considerable (i.e the correlation matrix). There are, however, research works suggest that this is feasible [22].

3 Experimental Process

Dilithium is highly optimized, in terms of performance, compared to other postquantum lattice based cryptography digital signature schemes and can be efficiently executed, apart from x86 machines, in ARM based processors of the cortex M family. In the previous section's analysis, it was shown that CPA is possible on the polynomial multiplications that involve the polynomials of S_1 and S_2, however, it was highlighted that the attack is feasible as long as the leakage samples of such operation can be identified and collected from the overall Dilithium signature generation function. This feat might be easily accomplished in a controlled environment (for embedded systems) like an emulator or a dedicated side channel assessment platform (eg. Chipwhisperer) but can be considerably more difficult in Commercial off-the-shelf (COTS) boards. In this section, a road map on how to collect and profile the appropriate traces for CPAs on COTS embedded system devices is proposed and described in detail. The trace analysis is made for NTT based implementations of Dilithium since they seem harder to attack (based on the previous section's analysis).

Setup: For our profiling experiments, the Post-quantum cryptographic library PQClean [4] was utlized, offering clean standalone implementations of most post-quantum schemes that are included in the NIST post-quantum project [2]. Aiming mostly towards the embedded system domain, the STM32 ST-Nucleo-F401RE [5] board was used as a testbench for the deployment of the Dilithium signature scheme. It features an ARM Cortex-M4 microprocessor that is currently being used by a plethora of embedded devices globally, making it an ideal reference system benchmark.

While the Dilithium signature scheme is being executed through the PQClean library, a probe is connected to the IDD current measurement jumper provided by the Nucleo, capturing the power consumption of the device. This power measurement signal is transferred through a pre-amplifier to a PicoScope 5000D Series Oscilloscope [3], which collects the generated traces with a sampling rate of 1 GS/s. Moreover, two additional probes are active throughout this process and connected to the GPIO pins of the Nucleo board. These probes act as trigger signals (one main, one secondary), assisting the oscilloscope to capture in a timely manner the correct portion of the received data. The secondary trigger is utilized as a marker to quickly identify the different operations of the signature scheme that are being computed at any given moment.

4 Profiling Process and Results

4.1 Methodology

Before commencing the profiling process, the PicoScope 5000D oscillator is configured properly in order to generate an additional math channel plot. This generated trace includes a measurement of the average value collected through multiple single-time signature samples. After sufficient number of these samples has been gathered (usually 100 single execution traces), this averaging process creates a clearer final trace with significantly less amount of noise, where the operations of the underlying computed signature generation scheme can be identified in a much easier manner compared to a single trace.

The signature generation algorithm as described by Algorithm 2 is being executed once every second, using the same randomly generated public-private key pair and the same message to be signed. This small delay allows the oscillator to conveniently capture the signal and add it to the averaging trace. As mentioned earlier, using the PQClean library's default implementation, the executed code can be viewed in Fig. 1, where the primary trigger signal is active throughout the duration of the whole signature generation process.

```
/* Generate Dilithium2 signature */
    main_trigger = 1;

    sign_ret = PQCLEAN_DILITHIUM2_CLEAN_crypto_sign_signature(signature,
         (size_t*)sizeof(signed_message), message, sizeof(message),
         secret_key, secondary_trigger);

    main_trigger = 0;
```

Fig. 1. PQClean Dilithium2 signature generation code snippet.

4.2 Signal Trace Acquisition

The complete Dilithium signature generation trace is presented in Fig. 2, where the blue upper plot represents a single signature generation capture, while the green plot is the averaging trace as described above. The red plot is the primary trigger, marking the beginning and end of the signature generation process. It can be observed by both plots how the algorithm is comprised of sequential rejection sampling loops (Step 3 to 13 of Algorithm 2), as well as the points in time when the $C \cdot S_1$ critical operation using the S_1 vector of the secret key is being computed inside the specific rejection loops.

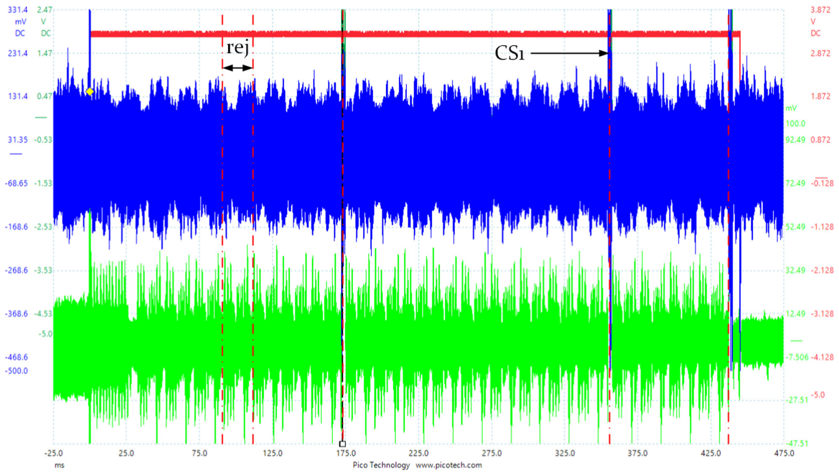

Fig. 2. Dilithium signature trace. (Color figure online)

By using the secondary trigger probe, the beginning and end of each individual process can be monitored. The first major one to be examined is the complete rejection loop as seen in Fig. 3. The various steps of the rejection loop are clearly visible in the averaging trace, starting with the *Y Vector Sampling* (Step 5), continuing with the matrix vector multiplication $W = A \cdot Y$ (Step 6) and the decomposing of the W matrix (Step 8). In the remainder of the rejection loop, one can observe the critical operations $C \cdot S_2$ and $C \cdot S_1$ involving the hidden coefficients S_2 and S_1 of the secret key respectively.

The $C \cdot S_2$ operation as is viewed in Fig. 3 is comprised of 4 distinct areas. This remark denotes the selection as default by the PQClean library of the value $k = 4$ of the Module-LWE which represents the number of rows of any given sample, as it is represented as a matrix of small polynomials A instead of a unique polynomial A. Focusing the secondary trigger towards one of those areas, the resulting trace portion $c_i \cdot s_{2,j}$ is presented in detail in Fig. 4.

Regarding the S_1 part of the secret key, the same logic is applied as in the $C \cdot S_2$ operation, with the main difference made apparent by looking at the trace

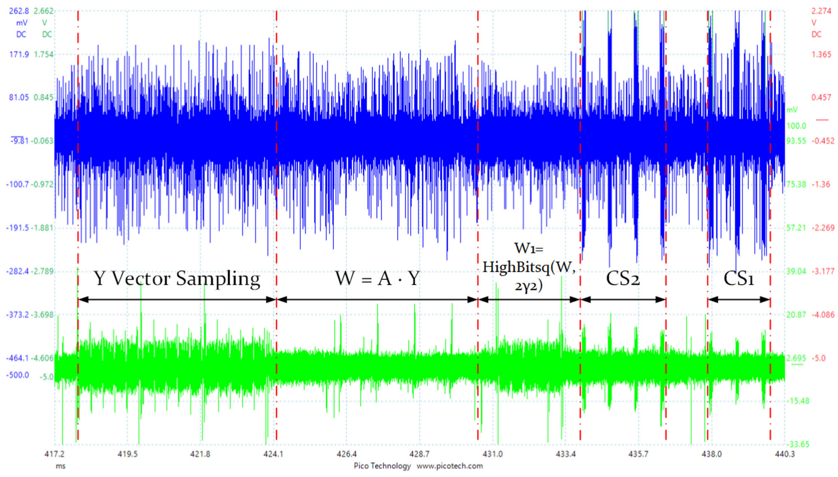

Fig. 3. Rejection loop trace.

associated with the selection of the value $l = 3$ as a parameter that represents the columns of the Module. The operation is, thus, repeated 3 times, each using a different part of the matrices. A sample code snipped mapped from the PQClean operation is presented in Fig. 5, denoting the activation of the secondary trigger during each of the 3 loops. The $c_i \cdot s_{1,j}$operation leakage is clearly seen in Fig. 6 while being differentiated from the adjacent computational leakages happening prior or after $C \cdot S_1(i)$. It should be noted at this point that the chosen rejection loop used in this analysis, is based on is the last rejection loop executed on the full signature generation algorithm, where all the conditions described in the Steps 11–13 of Algorithm 2 have been successfully met to generate the final signature value. Note also that our triggering involves only the polynomial multiplication in the NTT domain and the corresponding Montgomery reduction for one element of the S_1 vector. Matching the polynomial multiplication happening in the NTT domain coefficient per coefficient with the trace in Fig. 6 reveals that each trace peak corresponds to one coefficient multiplication $\hat{c_i} \cdot \hat{s_{1,j}}$ and can be directly used in the CPA approach described in the previous section

Given the generated traces for all the Steps that the Dilithium signature generation algorithm is consisted of and based on the security aspects of those operations as described in Sect. 2.1, it can be concluded that for medium security parameters shown in Table 1, a possible attacker can identify and extract the critical operations $c_i \cdot s_{1,j}$ and $c_i \cdot s_{2,j}$. This fact constitutes a valid requirement for an impacting realization of a possible CPA attack on the polynomial multiplication of the polynomials S_1 and S_2. It can be noted by observing at Figs. 6 and 4 the distinct waveform pattern produced by $c_i \cdot s_{1,j}$ and $c_i \cdot s_{2,j}$ polynomial operations.

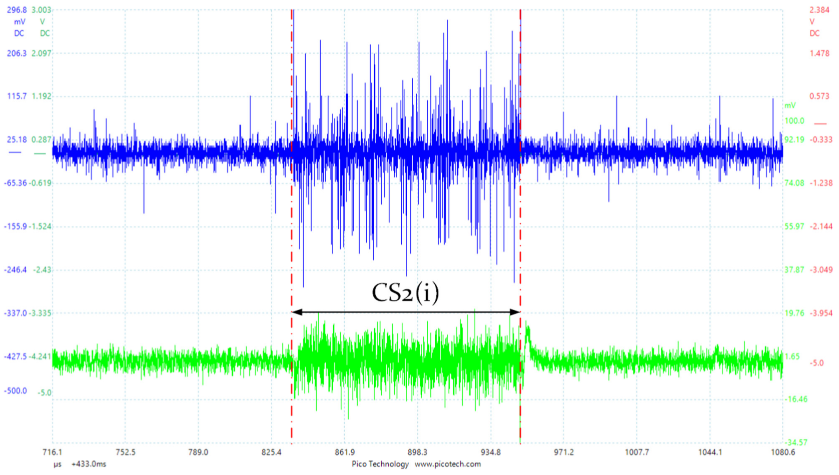

Fig. 4. $c_i \cdot s_{2,j}$ operation trace.

```
/* Compute z, reject if it reveals secret */
    for (size_t i = 0; i < L; ++i) {
        secondary_trigger = 1;
        PQCLEAN_DILITHIUM2_CLEAN_poly_pointwise_invmontgomery(&z.vec[i],
            &chat, &s1.vec[i]);
        secondary_trigger = 0;
        PQCLEAN_DILITHIUM2_CLEAN_poly_invntt_montgomery(&z.vec[i]);
    }
```

Fig. 5. PQClean $C \cdot S_1$ computation code snippet.

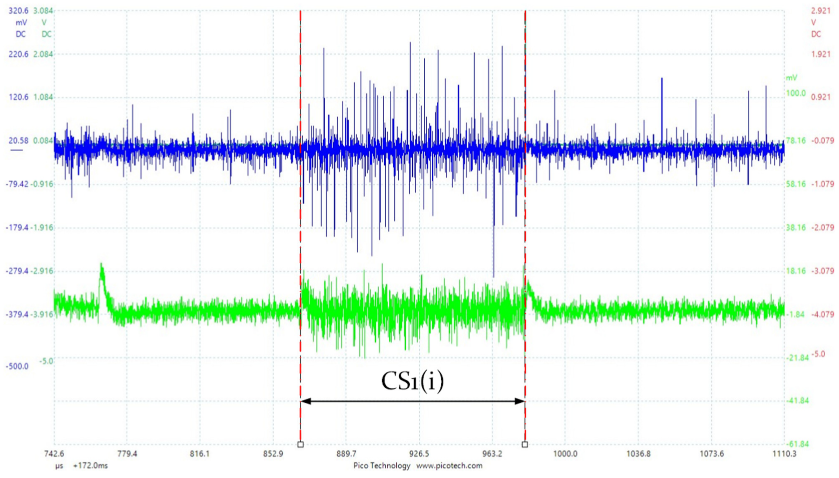

Fig. 6. $c_i \cdot s_{1,j}$ operation trace.

5 Conclusion

In this paper, an analysis on the Dilithium signature generation operation was done in order to identify interesting points for side channel analysis. We focused on the polynomial multiplication operation needed during the sample rejection loop of the algorithm (last loop iteration). In the paper, we also described how to perform CPA side channel attacks on this Dilithium operation regardless of how such operation is implemented. Our goal was to show that CPA is feasible as long as individual polynomial coefficient multiplication is traceable in collected power side channel information. By collecting traces from a COTS embedded system device that executes Dilithium Digital Signature generation on ARM Cortex M4 processor, we were able to demonstrate that indeed the polynomial operation is visible in the traces. Thus, it can be verified that the described attack is possible as long as the attacker has an analyzer/computer that can process the hypothesis values in correlation with the collected samples (as CPA dictates) for high bit length hypothesis data. As future work, we plan to perform the attack on extensive number of power trace datasets collected from the COTS device and evaluate the efficiency of the attack in relation to the number of such traces.

References

1. ChipWhisperer, NewAE Technology Inc. https://www.newae.com/chipwhisperer
2. NIST post-quantum cryptography project. https://csrc.nist.gov/projects/post-quantum-cryptography
3. PicoScope 5000D series oscillator. https://www.picotech.com/oscilloscope/5000/flexible-resolution-oscilloscope
4. PQClean library. https://github.com/PQClean/PQClean
5. STM32 ST-Nucleo-F401RE. https://os.mbed.com/platforms/ST-Nucleo-F401RE/
6. NSA/IAD. CNSA Suite and Quantum Computing FAQ. https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm (2016)
7. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Rossi, M., Tibouchi, M.: GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, pp. 2147–2164. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363223, https://doi.org/10.1145/3319535.3363223
8. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
9. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
10. Ducas, L., et al.: Crystals-Dilithium: a lattice-based digital signature scheme. IACR Trans. Cryptographic Hardware and Embed. Syst. 238–268 (2018)

11. Fournaris, A.P., Koufopavlou, O.: Protecting CRT RSA against fault and power side channel attacks. In: 2012 IEEE Computer Society Annual Symposium on VLSI, pp. 159–164 (2012)
12. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: a signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_31
13. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
14. Lyubashevsky, V.: Fiat-shamir with aborts: applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_35
15. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. IACR Cryptology ePrint Archive 2016, 921 (2016). http://eprint.iacr.org/2016/921
16. Migliore, V., Gérard, B., Tibouchi, M., Fouque, P.-A.: Masking Dilithium. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 344–362. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_17
17. Papachristodoulou, L., Fournaris, A.P., Papagiannopoulos, K., Batina, L.: Practical evaluation of protected residue number system scalar multiplication. IACR Trans. Cryptographic Hardware Embed. Syst. **2019**(1), 259–282 (2018). https://doi.org/10.13154/tches.v2019.i1.259-282. https://tches.iacr.org/index.php/TCHES/article/view/7341
18. Pöppelmann, T., Güneysu, T.: Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 139–158. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33481-8_8
19. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 513–533. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_25
20. Pöppelmann, T., Güneysu, T.: Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In: 2014 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2796–2799 (2014)
21. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172
22. Tunstall, M., Hanley, N., McEvoy, R.P., Whelan, C., Murphy, C.C., Marnane, W.P.: Correlation power analysis of large word sizes. In: IET Irish Signals and Systems Conference (ISSC), pp. 145–150 (2007)