An attack on ML-DSA using an implicit hint

Paco Azevedo-Oliveira[1,2], Jordan Beraud[2], and Louis Goubin[2]

[1] Thales DIS, France
paco.azevedo-oliveira@thalesgroup.com
[2] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
jordan.beraud@uvsq.fr, louis.goubin@uvsq.fr

Abstract. The security of ML-DSA, like most signature schemes, is partially based on the fact that the nonce used to generate the signature is unknown to any attacker. In this work, we exhibit a lattice-based attack that is possible if the nonces share implicit or explicit information. From a collection of signatures whose nonces share certain coefficients, it is indeed possible to build a collection of non full-rank lattices. Intersecting them, we show how to create a low-rank lattice that contains one of the polynomials of the secret key, which in turn can be recovered using lattice reduction techniques.

There are several interpretations of this result: firstly, it can be seen as a generalization of a fault-based attack on BLISS presented at SAC'16 by Thomas Espitau *et al.*. Alternatively, it can be understood as a side-channel attack on ML-DSA, in the case where an attacker is able to recover only one of the coefficients of the nonce used during the generation of the signature. For ML-DSA-II, we show that 4×160 signatures and few hours of computation are sufficient to recover the secret key on a desktop computer. Lastly, our result shows that simple countermeasures, such as permuting the generation of the nonce coefficients, are not sufficient.

Keywords: ML-DSA, side-channel attacks, lattice-based cryptanalysis.

1 Introduction

After several years of competition, the NIST published in August 2023 the final version of the ML-DSA [NIS23] signature standard, previously known as Dilithium [BDK$^+$21]. ML-DSA has been selected as the main signature standard to be used worldwide and in most cases. A proof of security, detailed in [BDK$^+$21], provides a strong security guarantee for ML-DSA. In addition many papers [BVC$^+$23] [CKA$^+$21] [KLH$^+$20], [MUTS22] [RJH$^+$18], [EAB$^+$23] [RCDB22] have studied the security of ML-DSA in the event that the attacker has access to auxiliary information during signature generation. Other papers [BBK16] [BP18] [BAE$^+$24] [WNGD23] [KPLG24] have studied its security under the condition that an attacker is able to inject one or more faults during signature generation.

In most signature schemes, the generation of a strong nonce – unknown to the attacker – is one of the keystones of the security, which collapses as soon as information on the nonce is revealed. For ECDSA, one of the pre-quantum signature standards, [FGR13] shows that it is sufficient for the attacker to obtain a set of signatures whose random values share a few bits, to find the secret key using lattice reduction techniques. Following these results, two papers [EFGT17], [LZS$^+$21] study the security of ML-DSA under the hypothesis of an attacker who would be able to recover (explicit or implicit) information about the nonce. In [EFGT17], the authors show that an attacker capable of introducing a fault at the time of nonce generation, a polynomial vector named $\mathbf{y}$ in ML-DSA, to obtain a polynomial $\mathbf{y}$ of abnormally low degree, can recover the secret key. Their attack is customized to BLISS, another signature algorithm based on structured lattices, but can be easily modified to be applied to ML-DSA. In [LZS$^+$21] the authors prove that an attacker able to retrieve one precise bit of information on **each** coefficient of the nonce $\mathbf{y}$ will need 10 000 signatures to find the secret key.

In this paper, we extend the ML-DSA security analysis under the hypothesis of an attacker who is capable of learning information about the nonce $\mathbf{y}$. More specifically, we show that the ideas used in [EFGT17] give rise to a more general attack. We demonstrate that knowing one or more coefficients of the nonce used to generate a signature allows to reduce the rank of the lattice in which one of the polynomials of the secret key, namely $\mathbf{s}_1$, is contained. If one does not know sufficiently many coefficients, using a lattice reduction is not practical. On the other hand, with a collection of signatures, and by intersecting the different lattices obtained, we can create a lattice of low rank that contains $\mathbf{s}_1$ by construction, and thus retrieve $\mathbf{s}_1$. The main results of our article can be summarised as follows:

– Seen as a result of a fault attack, our attack generalizes that of [EFGT17], and becomes drastically more efficient if an attacker can recover at least two faulty signatures.
– Seen as a side-channel attack, if we assume that an attacker is able to find one of the coefficients of $\mathbf{y}$ thanks to the signature generation execution trace, he will need 4×160 signatures and few hours of computation to find the secret key, for ML-DSA-II.
– Seen as an exploitation of implicit information on the nonce, we obtain the same result as [FGR13] transposed to ML-DSA: assuming that the attacker knows a set of signatures whose nonces share an unknown coefficient in common, he can retrieve the position of the unknown coefficients and apply the method described in the previous point to find the secret key.

Outline. This paper is organized as follows. In Section 2, we review the necessary background about ML-DSA and lattice theory. In Section 3, we define an attack scenario, and show how it can be reformulated in terms of lattice reduction. In Section 4, we propose different variations of the result of Section 3. In addition to the theoretical results presented in Section 3, that apply regardless of the security level of ML-DSA, we present in Section 5 the practical results we obtained when attacking ML-DSA-II. Finally, in Section 6 we discuss the obtained results, their limitations and their implications.

2 Summary of ML-DSA and lattices

In this section we begin by briefly introducing the notations and main functions used in ML-DSA. For a detailed description of ML-DSA, the reader is referred to [BDK$^+$21]. We then review the main definitions and results on lattice theory, which will be used in the next section.

2.1 Notations

Definition 1 *Let α be an even (resp. odd) integer. We define $r' := r \bmod^{\pm}(\alpha)$ the unique -$\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$ (resp. $-\frac{\alpha-1}{2} \leq r' \leq \frac{\alpha-1}{2}$) such that $r' = r \mod (\alpha)$. We will speak of centered reduction modulo α. We define $r'' := r \bmod^{+}(\alpha)$ the unique $0 \leq r'' < \alpha$ such that $r'' = r \mod (\alpha)$.*

Definition 2 *We define $\phi_n = x^n + 1$ with n a power of 2 and q a prime, and introduce the following rings:*

$$\mathcal{R} := \mathbb{Z}[x]/(\phi_n) \text{ and } \mathcal{R}_q := \mathbb{Z}_q[x]/(\phi_n).$$

Definition 3 *For $w \in \mathbb{Z}_q$:*

$$||w||_\infty := |w \bmod^{\pm} (q)|.$$

For $\boldsymbol{w} = \sum_i w_i x^i \in \mathcal{R}$:

$$||\boldsymbol{w}||_\infty := \max_i ||w_i \bmod^{\pm}(q)||_\infty \text{ and } ||\boldsymbol{w}|| := \Big(\sum_i ||w_i||_\infty^2\Big)^{1/2}$$

and for $\mathbf{w} = (\mathbf{w}^{[1]}, ..., \mathbf{w}^{[l]}) \in \mathcal{R}^l$,

$$||\mathbf{w}||_\infty := \max_i ||\mathbf{w}^{[i]}||_\infty \text{ and } ||\mathbf{w}|| := \Big(\sum_i ||\mathbf{w}^{[i]}||^2\Big)^{1/2}.$$

Finally, we define two sets $S_\eta, \tilde{S_\eta} \subset \mathcal{R}$ as follows:

$$S_\eta := \{\boldsymbol{w} \in \mathcal{R} \mid ||\boldsymbol{w}||_\infty \leq \eta\} \text{ and } \tilde{S_\eta} := \{\boldsymbol{w} \bmod^{\pm} (2\eta) \mid \boldsymbol{w} \in \mathcal{R}\}.$$

Notation 1 *For an element $\mathbf{z}_1 \in \mathcal{R}^l$ we will note $\mathbf{z}_1 = (\mathbf{z}_1^{[1]}, ..., \mathbf{z}_1^{[l]}) \in \mathcal{R}^l$ and $\mathbf{z}_{1,i}^{[j]}$ will be the $i-th$ coefficient of the polynomial $\mathbf{z}_1^{[j]}$.*

2.2 Algorithm description

ML-DSA is a signature scheme based on structured lattice, it uses matrices and vectors of $\mathcal{R}$ or $\mathcal{R}_q$, with the values of n and q fixed at $n = 256$ and $q = 2^{23} - 2^{13} + 1 = 8\,380\,417$ regardless of the security level. In addition, to reduce the size of the public key and to generate the signature ML-DSA uses algorithms that splits elements in $\mathbb{Z}_q$, namely $\texttt{Decompose}_q$, $\texttt{Power2Round}_q$, $\texttt{MakeHint}_q$ and $\texttt{UseHint}_q$. For a complete description of these algorithms, we refer to the ML-DSA specification [BDK+21].

Key Generation: The key generation algorithm is described in Algorithm 1. ML-DSA is based on the Module-LWE problem, a variant of the LWE problem introduced by Regev in [Reg05], which we will not recall here. From some seeds, $\mathbf{A} \in \mathcal{R}_q^{k\times l}$, $\mathbf{s}_1 \in S_\eta^l$ and $\mathbf{s}_2 \in S_\eta^k$ are generated and then $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ is computed. Two optimisations are made to the public key, which is traditionally $(\mathbf{A}, \mathbf{t})$, to reduce its size. The first optimisation, the most natural, consists of transmitting only the seed used to generate the matrix $\mathbf{A}$. For the second optimisation, only $\mathbf{t}_1$ (the high part of $\mathbf{t}$ computed with $\texttt{Power2Round}_q$) is considered to be part of the public key.

Algorithm 1 KeyGen

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

Signature: The signature algorithm is described in Algorithm 2. The signer derives a masking vector $\mathbf{y} \in \mathcal{R}_q^l$, from which it calculates $\mathbf{w}_1$, the most significant bits of $\mathbf{w} := \mathbf{A}\,\mathbf{y}$ and then a challenge $c \in \mathcal{R}$ which is a sparse polynomial whose coefficients are in $\{-1, 0, 1\}$. It then calculates $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$, the main part of the signature, which verifies the following equation, used for verification:

$$\texttt{HighBits}_q(\mathbf{Az} - c\mathbf{t},\, 2\,\gamma_2) = \texttt{HighBits}_q(\mathbf{Ay} - c\mathbf{s}_2,\, 2\,\gamma_2).$$

The signer then checks that $\mathbf{z}$ does not give information about the secret key; if it does, it starts again by drawing another masking vector. Once $\mathbf{z}$ has passed the tests, we have the following equation:

$$\mathbf{w}_1 = \texttt{HighBits}_q(\mathbf{A}\mathbf{y} - c\mathbf{s}_2,\, 2\,\gamma_2) = \texttt{HighBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t},\, 2\,\gamma_2).$$

Since $\mathbf{t}_0$ is not known, anyone attempting to verify the signature cannot directly compute $\texttt{HighBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t},\, 2\,\gamma_2)$. The signer adds $\mathbf{h} = \texttt{MakeHint}_q(-c\mathbf{t}_0, \mathbf{A}\mathbf{y} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ to allow the verifier to calculate $\texttt{HighBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t},\, 2\,\gamma_2)$, without the knowledge of $\mathbf{t}_0$. Finally, the signature is composed of the challenge c, $\mathbf{z}$, and the hint vector $\mathbf{h}$.

Algorithm 2 Sig

```
Require: $sk, M$
Ensure: $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$
 1: $\mathbf{A} \in \mathcal{R}_q^{k\times l} := \texttt{ExpandA}(\rho)$
 2: $\mu \in \{0,1\}^{512} := \mathrm{H}(tr \,||\, M)$
 3: $\kappa := 0,\ (\mathbf{z}, \mathbf{h}) := \perp$
 4: $\rho' \in \{0,1\}^{512} := \mathrm{H}(K \,||\, \mu)$
 5: while $(\mathbf{z}, \mathbf{h}) = \perp$ do
 6:     $\mathbf{y} \in \tilde{S}_{\gamma_1}^{l} := \texttt{ExpandMask}(\rho',\, \kappa)$
 7:     $\mathbf{w} := \mathbf{A}\,\mathbf{y}$
 8:     $\mathbf{w}_1 = \texttt{HighBits}_q(\mathbf{w},\, 2\,\gamma_2)$
 9:     $\tilde{c} \in \{0,1\}^{256} := \mathrm{H}(\mu \,||\, \mathbf{w}_1)$
10:     $c \in B_\tau := \texttt{SampleInBall}(\tilde{c})$
11:     $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
12:     $\mathbf{r}_0 := \texttt{LowBits}_q(\mathbf{w} - c\mathbf{s}_2,\, 2\,\gamma_2)$
13:     if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ then
14:         $(\mathbf{z}, \mathbf{h}) := \perp$
15:     else
16:         $\mathbf{h} := \texttt{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
17:         if $||c\,\mathbf{t}_0||_\infty \geq \gamma_2$ or $|\mathbf{h}|_{\mathbf{h}_j=1} > \omega$ then
18:             $(\mathbf{z}, \mathbf{h}) := \perp$
19:     $\kappa := \kappa + l$
20: return $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$
```

Verification: The verification algorithm is described in Algorithm 3. To verify the signature, it is sufficient to reconstruct the matrix $\mathbf{A}$ and the polynomial c on which the signer has commited. Using the vector $\mathbf{h}$ of the signature, we can recalculate $\mathbf{w}_1 = \texttt{HighBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t},\, 2\,\gamma_2)$, by using $\texttt{UseHint}_q$. Finally, the signature will be accepted if it is possible to reconstruct the correct c from $\mathbf{w}_1$ and if $\mathbf{z}$ meets the security conditions imposed during signature generation.

Algorithm 3 `Ver`

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

2.3 An overview of Lattice Theory

The classic lattice definitions and unproven results from this Section are quoted from the book "The LLL Algorithm" [NV09].

Definition 4 *Let $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Q}^n$. Denote by $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ the set of all integral linear combinations of the $\mathbf{b}_i$'s:*

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \{\sum_{i=1}^{d} n_i \mathbf{b}_i \mid n_1, \ldots, n_d \in \mathbb{Z}\}.$$

When $L = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, we say that L is the lattice spanned by the $\mathbf{b}_i$'s, and that the $\mathbf{b}_i$'s are generators. When the $\mathbf{b}_i$'s are further linearly independent, we say that $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is a basis of the lattice L in which case:

$$\forall \mathbf{v} \in L,\ \exists!\, n_1, \ldots, n_d \in \mathbb{Z},\ \mathbf{v} = \sum_{i=1}^{d} n_i \mathbf{b}_i.$$

Definition 5 *We define the dimension or rank of a lattice L in $\mathbb{R}^n$, denoted by* $\dim(L)$, *as the dimension d of its linear span denoted by* $\mathrm{span}(L)$. *The lattice is said to be full-rank when $d = n$.*

Definition 6 *The dual of a lattice L is the set L^* define as:*

$$L^* = \{\mathbf{x} \in \mathrm{span}(L) \mid \forall \mathbf{y} \in L,\ \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}.$$

Proposition 1 *The dual of a lattice with basis $\mathbf{B}$ is a lattice with basis* $\mathbf{D} = \mathbf{B}(\mathbf{B}^\intercal \mathbf{B})^{-1}$.

Definition 7 *A non-singular matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}^{m\times n}$ is in Hermite normal form (HNF) iff*

- *There exists $1 \le i_1 < \cdots < i_h \le m$ such that $b_{i,j} \neq 0 \implies j < h$ and $i \ge i_j$.*
- *For all $k > j$, $0 \le b_{i_j,k} < b_{i_j,j}$ i.e all elements at rows i_j are reduced modulo $b_{i_j,j}$.*

Proposition 2

1. *There exists a polynomial algorithm that transforms a matrix into HNF in a polynomial number of operations.*
2. *If $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{H})$ and $\mathbf{B}, \mathbf{H}$ are HNF then $\mathbf{B} = \mathbf{H}$.*
3. *Let $\mathbf{B} \in \mathbb{Z}^{m \times n}$, there exists an HNF matrix $\mathbf{H}$ such that $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{H})$.*

Proposition 3 *There exists a polynomial time algorithm that, on input a lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ and an affine subspace S, outputs a new basis $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n)$ such that $\mathcal{L}(\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_d) = S \cap \mathcal{L}(\mathbf{B})$, where d is the dimension of $S \cap \operatorname{span}(\mathbf{B})$.*

Proposition 4 *There exists a polynomial time algorithm that, on input two lattice bases $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{Z}^{n \times n}$, outputs a new basis $\mathbf{D}$ such that $\mathcal{L}(\mathbf{D}) = \mathcal{L}(\mathbf{B}_1) \cap \mathcal{L}(\mathbf{B}_2)$*

Proof. See [Mic08] for proof.

We note that Proposition 4 can be generalized to the case of non-full-rank lattices. To the best of our knowledge, this generalization is not known in the literature, we state it below together with a proof.

Proposition 5 *There exists a polynomial time algorithm that, on input two lattices basis $\mathbf{B}_1 \in \mathbb{Z}^{m_1 \times n}, \mathbf{B}_2 \in \mathbb{Z}^{m_2 \times n}$, outputs a new basis $\mathbf{D}$ such that $\mathcal{L}(\mathbf{D}) = \mathcal{L}(\mathbf{B}_1) \cap \mathcal{L}(\mathbf{B}_2)$.*

Proof. Let $\mathbf{B}'$ a basis of $\operatorname{span}(\mathbf{B}_1) \cap \operatorname{span}(\mathbf{B}_2)$. We can easily compute a basis $\tilde{\mathbf{B}}_i$ of $\mathcal{L}(\mathbf{B}_i) \cap \operatorname{span}(\mathbf{B}')$ according to Proposition 3. We have

$$\mathcal{L}(\mathbf{B}_1) \cap \mathcal{L}(\mathbf{B}_2) = \mathcal{L}(\tilde{\mathbf{B}}_1) \cap \mathcal{L}(\tilde{\mathbf{B}}_2).$$

Moreover,

$$\operatorname{rank}(\operatorname{span}(\tilde{\mathbf{B}}_i)) = \operatorname{rank}(\operatorname{span}(\mathbf{B}_i) \cap \operatorname{span}(\mathbf{B}')) = \operatorname{rank}(\operatorname{span}(\mathbf{B}'))$$

and

$$\operatorname{span}(\tilde{\mathbf{B}}_1) = \operatorname{span}(\tilde{\mathbf{B}}_2).$$

So $\mathcal{L}(\tilde{\mathbf{B}}_1)$ and $\mathcal{L}(\tilde{\mathbf{B}}_2)$ are full rank lattices in $\mathcal{L}(\tilde{\mathbf{B}}_1, \tilde{\mathbf{B}}_2)$. We then use Propostion 2 to compute basis of $\mathcal{L}(\tilde{\mathbf{B}}_1, \tilde{\mathbf{B}}_2)$ and Propostion 4 to compute the intersection $\mathcal{L}(\tilde{\mathbf{B}}_1) \cap \mathcal{L}(\tilde{\mathbf{B}}_2)$ in $\mathcal{L}(\tilde{\mathbf{B}}_1, \tilde{\mathbf{B}}_2)$.

Remark 1 *The lattice intersection algorithm is based on the calculation of the HNF and the calculation of some dual basis. It is not difficult to implement an algorithm that calculates the HNF in a polynomial number of operations, however the resulting algorithm may run in super-polynomial time due to the explosion of the size of the coefficients during intermediate calculations.*

Worse still, for a lattice $L \subset \mathbb{Z}^n$, it is not generally true that $L^* \subset \mathbb{Z}^n$. The HNF must therefore be calculated for matrices with coefficients in $\mathbb{Q}$, whose numerators and denominators explode. In the case of the lattices considered in practice in Section 5, this can lead to rounding errors when calculating the HNF and an explosion in calculation time. For this reason, in Section 3, we will take a different approach to intersect lattices from the state-of-the-art.

Hard lattices problems:

Definition 8 *Let L be a lattice of $\mathbb{R}^n$. For all $1 \leq i \leq \dim(L)$, the $i-$th minimum $\lambda_i(L)$ is defined as the minimum of $\max_{1\leq j\leq i}(||\mathbf{v}_j||)$ over all i linearly independent lattice vectors $\mathbf{v}_1, \ldots, \mathbf{v}_i \in L$.*

Problem 1 *Shortest Vector Problem (SVP): Given a lattice basis $\mathbf{B}$ find a nonzero lattice vector of lenght at most $\lambda_1(\mathcal{L}(\mathbf{B}))$.*

Problem 2 *Closest Vector Problem (CVP): Given a lattice basis $\mathbf{B}$ and a target vector $\mathbf{t}$ find a nonzero lattice vector within distance $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$.*

Definition 9 *The Gram determinant of $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^n$, denoted by $\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is the determinant of the $m \times m$ Gram matrix $(\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1\leq i,j\leq m}$. Furthermore, when the $\mathbf{b}_i$'s are linearly independent, $\sqrt{\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_m)}$ is the $m-$dimensional volume of the parallelepiped spanned by the $\mathbf{b}_i$'s.*

Definition 10 *The volume of a lattice $L = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is defined as:*

$$\mathrm{vol}(L) = \sqrt{\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_d)}$$

which is independent of the choice of the basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of the lattice L.

Definition 11 *Let L be a sublattice in $\mathbb{R}^n$. A sublattice of L is a lattice M included in L: sublattices of L are the subgroups of L. If the rank of M is equal to the rank of L, we say that M is a full-rank sublattice of L. In which case the group index $[L : M]$ is finite and:*

$$\mathrm{vol}(M) = \mathrm{vol}(L) \times [L : M].$$

Theorem 1 *The `LLL` algorithm given as input a basis of a $d-$dimensional integer lattice $L \subset \mathbb{Z}^n$, outputs a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of L in time polynomial in the size of the basis such that:*

$$||\mathbf{b}_1|| \leq 2^{(d-1)/4} \mathrm{vol}(L)^{1/d}.$$

More precisely, the complexity is $\mathcal{O}(d^5(d + log(B))log(B))$ where $B = \max_i(||\mathbf{b}_i||_2)$.

Theorem 2 *The `NearestPlane` algorithm given as input a basis of a integer lattice $L = \mathcal{L}(\mathbf{B}) \subset \mathbb{Z}^n$ and a target vector $\mathbf{t}$ output a lattice point $\mathbf{v}$ of L in time polynomial in the size of the basis such that:*

$$\forall i \in \{1, \ldots, n\},\ \langle \mathbf{t} - \mathbf{v}, \mathbf{b}_i^* \rangle / ||\mathbf{b}_i^*|| \in [-1/2, 1/2),$$

where $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ denote the Gram-Schmidt matrix of $\mathbf{B}$.

Assumption 1 *Let L be a $d-$dimensional lattice in $\mathbb{R}^n$. The Gaussian Heuristic "predicts" that $\lambda_1(L)$ would be close to:*

$$\left(\frac{\mathrm{vol}(L)}{\nu_d}\right)^{1/d} \approx \sqrt{\frac{d}{2\pi e}}\mathrm{vol}(L)^{1/d},$$

where ν_d denotes the volume of the closed unitary ball of $\mathbb{R}^d$.

Remark 2 *It is common to assume that if a vector $\mathbf{v} \in L$ is smaller than the Gaussian heuristic, then it is the smallest vector of L. It is then possible to solve the SVP, because a shortest vector of L can be found in its* `LLL`*-reduced basis, provided that the bound of Theorem 1 is sufficiently small. Similarly, the CVP can be solved by pre calculating the* `LLL`*-reduced basis and then applying the* `NearestPlane` *algorithm.*

3 Problem definition and attack methodology

Following the same motivations as in [FGR13], we examine the security of ML-DSA in a scenario where an attacker obtains m messages signed under the same secret key and such that the nonces used during the generation of the signature share a certain number of coefficients. To simplify the presentation, in this section we assume that the common coefficients between the nonces are those of highest degree. Furthermore as $\mathbf{y} \in \mathcal{R}_q^l$ is a vector of polynomials, we present the reasoning only on $\mathbf{y}^{[1]}$, the first polynomial of $\mathbf{y}$. For the moment we assume we know m signatures $\sigma_1, \ldots, \sigma_m$ of `Sig` signed under the same secret key, such that:

$$\forall i \in \{1, \ldots, m\},\ \mathbf{y}_i^{[1]} = \sum_{j=0}^{d} \mathbf{y}_{i,j}^{[1]} x^j + \sum_{j=d+1}^{n-1} \mathbf{y}_{1,j}^{[1]} x^j$$

where $0 \leq d \leq n-2$ is unknown. By definition, $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$ so even if it means subtracting $\mathbf{z}_1$ from all the other $\mathbf{z}_i$, without loss of generality we can assume the last coefficients shared by the nonce to be zero, formally:

$$\forall i \in \{1, \ldots, m\},\ \mathbf{z}_i^{[1]} = \mathbf{y}_i^{[1]} + c_i \mathbf{s}_1^{[1]} \ \text{ with } \ \deg(\mathbf{y}_i^{[1]}) \leq d < n.$$

Lemma 1 *Assuming that for $i \in \{1, \ldots, m\}$ $c_i \in \mathcal{R}_q$ behaves like a random element of the ring $\mathcal{R}_q$, c_i is invertible in $\mathcal{R}_q$ with probability at least 99%.*

Proof. By the Chinese remainder theorem, $\mathcal{R}_q \cong \mathbb{Z}_q^n$, assuming that for $i \in \{1, \ldots, m\}$ c_i behaves like a random element of $\mathcal{R}_q$, the probability that c_i is invertible is $(1 - (1/q))^n \approx 0.999969$.

Therefore, in the rest of the article, the vectors c_i will be considered invertible without loss of generality. For each $i \in \{1, \ldots, m\}$, it exists a $P_i \in \mathcal{R}$ such that $c_i \times c_i^{-1} = 1 + qP_i$, where c_i^{-1} denotes the inverse of c_i seen as a element of $\mathcal{R}_q$. Therefore, $c_i^{-1}\mathbf{z}_i^{[1]} = c_i^{-1}\mathbf{y}_i^{[1]} + \mathbf{s}_1^{[1]} + qP_i\mathbf{s}_1^{[1]}$ and for all $i \in \{1, \ldots, m\}$:

$$\mathbf{s}_1^{[1]} \in L_i := \mathcal{L}\left(c_i^{-1}\mathbf{z}_i^{[1]}, \{c_i^{-1}x^j\}_{j\in\{0,\ldots,d\}}, \{qx^j\}_{j\in\{0,\ldots,n-1\}}\right).$$

This relation can be rewritten as:

$$\forall i \in \{1,\ldots,m\},\quad \mathbf{s}_1^{[1]} \in \overline{L_i} := \operatorname{span}_{\mathbb{F}_q}\left(c_i^{-1}\mathbf{z}_i^{[1]}, \{c_i^{-1}x^j\}_{j\in\{0,\ldots,d\}}\right).$$

We have reformulated the problem of finding $\mathbf{s}_1^{[1]}$, as the calculation of a lattice intersection (or as an intersection of $\mathbb{F}_q$ vectorial spaces), indeed:

$$\mathbf{s}_1^{[1]} \in \bigcap_{1\le i\le m} L_i \quad \text{and} \quad \mathbf{s}_1^{[1]} \in \bigcap_{1\le i\le m} \overline{L_i}.$$

Switching between $\mathbb{F}_q$ subspaces and lattices: One method would be to calculate the intersection of the lattices L_i using duality and the results of Section 2, although polynomial this method can be inefficient for large lattices, mainly due to rounding errors when calculating the HNF. The definition of $\overline{L_i}$'s simplifies this, allowing to calculate only intersections of vector spaces. Unfortunately, it is not possible to find a small element in a vector space directly. This is why we apply the following method:

1. Compute a basis $\mathbf{B} = (\mathbf{b}_1,\ldots,\mathbf{b}_\alpha)$ of the $\overline{L_i}$'s intersection, named $\overline{L}$.
2. View $\overline{L}$ as a integer lattice, by considering $L = \mathcal{L}(\mathbf{B}, \{qx^j\}_{j\in\{0,\ldots,n-1\}})$.
3. Retrieve $\mathbf{s}_1^{[1]}$ using `LLL`, according to the Gaussian heuristic, $\mathbf{s}_1^{[1]}$ will be the shortest vector of L.

This method allows to efficiently compute the intersection of L_i's because the lattice L calculated in steps 1 and 2 satisfies:

$$L = \bigcap_{1\le i\le m} L_i.$$

Remark 3 *Steps* 1 *and* 2 *provide a generic way of computing intersections of modular lattices. This method is more efficient than the classical one (which uses duality) described in Section 2.*

Since L is of full rank in $\mathbb{Z}^n$, `LLL` cannot be applied directly (as the approximation factor depends exponentially on the dimension). A classical way to overcome the problem, described in [EFGT17], consists in projecting L in lower dimension to recover $\mathbf{s}_1^{[1]}$ piece by piece.

Theorem 3 *Let $M_1,\ldots,M_m$ be m messages with the associated signatures $\sigma_1,\ldots,\sigma_m$ such that, for $i \in \{1,\ldots,m\}$, $\deg(\mathbf{y}_i^{[1]}) = d$. Under Assumption 1, $\mathbf{s}_1^{[1]}$ can be computed in polynomial time as soon as:*

$$d < n - 2 \quad \textit{and} \quad m \geq \frac{n-\alpha}{n-d-1}$$

where $\alpha \in \mathbb{N}$ denotes an integer where it is considered realistic to recover the shortest vector by applying `LLL` *or any other lattice reduction algorithm, on a lattice of dimension α.*

Proof.

$$\forall i \in \{1, \ldots, m\}, \quad \dim_{\mathbb{F}_q}\left(\overline{L_i}\right) = d + 1 < n - 1,$$

Let us consider, for $i \in \{1, \ldots, m\}$, $\phi_i : \mathbb{F}_q^n \to \mathbb{F}_q^{n-(d+1)}$ which verifies, $\overline{L_i} = \ker_{\mathbb{F}_q}(\phi_i)$. Therefore, by defining $\phi : x \to (\phi_1(x), \ldots, \phi_m(x))$:

$$\begin{aligned}\dim_{\mathbb{F}_q}\left(\bigcap_{1 \leq i \leq m} \overline{L_i}\right) &= \dim(\ker_{\mathbb{F}_q}(\phi)) \\ &= n - \mathrm{rank}_{\mathbb{F}_q}(\phi).\end{aligned}$$

Assuming ϕ to be a random application, it has maximum rank and thus:

$$\dim_{\mathbb{F}_q}\left(\overline{L}\right) = \dim_{\mathbb{F}_q}\left(\bigcap_{1 \leq i \leq m} \overline{L_i}\right) = n - m(n - (d+1)) \leq \alpha.$$

Let $L = \mathcal{L}(\mathbf{B}, \{qx^j\}_{j \in \{0,\ldots,n-1\}})$, where $\mathbf{B}$ is a basis of $\overline{L}$ and $I \subset \{0, \ldots, n-1\}$, with $|I| = l$ and $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^l$ the projection associated with I, then $L_I := \varphi_I(L)$ is an integer lattice in $\mathbb{Z}^l$ and $\varphi_I(\mathbf{s}_1^{[1]}) \in L_I$. According to the ML-DSA parameters, $||\mathbf{s}_1^{[1]}||_2 \leq \sqrt{n}\eta$ where $\eta \in \{2, 4\}$ depending on the security level, so $||\varphi_I(s_1^{[1]})||_2 \leq \sqrt{l}\eta$. Furthermore,

$$\mathrm{vol}(L_I) = \frac{\mathrm{vol}(q\mathbb{Z}^l)}{[L_I : q\mathbb{Z}^l]} = q^{l-\alpha}.$$

$\mathbf{s}_1^{[1]}$ will be the shortest vector of L_I, as soon as:

$$||\varphi(\mathbf{s}_1^{[1]})||_2 = \sqrt{l}\eta \ll \sqrt{\frac{n}{2\pi e}}\mathrm{vol}(L_I)^{1/l} = \sqrt{\frac{l}{2\pi e}}q^{1-m/l}.$$

In other words, with $C = \sqrt{2\pi e}\eta$:

$$l \geq \frac{\alpha}{1 - log(C)/log(q)} \approx \alpha(1 + \frac{log(C)}{log(q)}).$$

If it is considered realistic to solve the SVP in L_I of dimension l, then $\mathbf{s}_1^{[1]}$ can be found by applying `LLL` or any other lattice reduction algorithm.

3.1 Affine space optimisation

Even if Theorem 3 is sufficient to find $\mathbf{s}_1^{[1]}$ efficiently with a reasonable number of signatures, the $\overline{L_i}$'s are $\mathbb{F}_q$-subspaces of dimension $d+1$, and for their intersection to be non-trivial, it is necessary that $d < n-2$. In the following sub-section, we propose an affine optimisation which allow to intersect the $\overline{L_i}$'s in a non trivial way, even when the degree of the nonce is $n-2$ (*i.e.* when only the last coefficient of $\mathbf{y}$ is 0). More precisely, we had for all $i \in \{1, \dots, m\}$:

$$\mathbf{s}_1^{[1]} \in L_i := \mathcal{L}\left(c_i^{-1}\mathbf{z}_i^{[1]}, \{c_i^{-1}x^j\}_{j\in\{0,\dots,d\}}, \{qx^j\}_{j\in\{0,\dots,n-1\}}\right).$$

But since $c_i^{-1}\mathbf{z}_i^{[1]} = c_i^{-1}\mathbf{y}_i^{[1]} + \mathbf{s}_1^{[1]} + qP_i\mathbf{s}_1^{[1]}$, the coordinate of $\mathbf{s}_1^{[1]}$ in $c_i^{-1}\mathbf{z}_i^{[1]}$ is always 1, it is possible to consider affine lattices, of dimension d. Formally for $i \in \{1, \dots, m\}$:

$$\mathbf{s}_1^{[1]} \in L_{A_i} := c_i^{-1}\mathbf{z}_i^{[1]} + \mathcal{L}\left(\{c_i^{-1}x^j\}_{j\in\{0,\dots,d\}}, \{qx^j\}_{j\in\{0,\dots,n-1\}}\right).$$

As described previously, it is possible to write this intersection using $\mathbb{F}_q-$subspaces:

$$\forall i \in \{1, \dots, m\}, \quad \mathbf{s}_1^{[1]} \in \overline{L_{A_i}} := c_i^{-1}\mathbf{z}_i^{[1]} + \operatorname{span}_{\mathbb{F}_q}\left(\{c_i^{-1}x^j\}_{j\in\{0,\dots,d\}}\right).$$

So, finally:

$$\mathbf{s}_1^{[1]} \in L_A := \bigcap_{1\le i\le m} L_{A_i} \quad \text{and} \quad \mathbf{s}_1^{[1]} \in \overline{L_A} := \bigcap_{1\le i\le m} \overline{L_{A_i}}.$$

We cannot apply directly the results of Section 2, since the A_i's are affine spaces and not a vector spaces. The following strategy is proposed:

1. Compute a basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_\alpha)$ and a vector $\mathbf{v}$ such that:
$$\overline{L_A} = \mathbf{v} + \operatorname{span}(\mathbf{B}).$$
2. View $\overline{L_A}$ as an affine lattice, by considering:
$$L_A = \mathbf{v} + \mathcal{L}\left(\mathbf{B}, \{qx^j\}_{j\in\{0,\dots,n-1\}}\right).$$
3. Retrieve $\mathbf{s}_1^{[1]}$ using the `NearestPlane` algorithm, according to the Gaussian heuristic, $\mathbf{s}_1^{[1]} - \mathbf{v}$ will be the closest vector of $-\mathbf{v}$ in L_A.

Theorem 4 *Let m messages $M_1, \dots, M_m$ with the associated signatures $(\mathbf{z}_i, c_i)_{i\in\{1,\dots,m\}}$ such that for $i \in \{1, \dots, m\}$, $\deg(\mathbf{y}_i^{[1]}) = d$. Under Assumption 1, $\mathbf{s}_1^{[1]}$ can be computed in polynomial time as soon as:*

$$d < n-1 \quad \text{and} \quad m \ge \frac{n-\alpha}{n-d}$$

where $\alpha \in \mathbb{N}$ denotes an integer where it is considered realistic to recover the shortest vector by apply `NearestPlane` *or any other CVP solver algorithm on a lattice of size α.*

Proof. Since the proof of Theorem 3 is similar, fewer details are provided.

$$\forall i \in \{1, \ldots, m\}, \quad \dim_{\mathbb{F}_q}\left(\overline{L_{A_i}}\right) = d < n - 1,$$

and therefore:

$$\dim_{\mathbb{F}_q}\left(\overline{L_A}\right) = \dim_{\mathbb{F}_q}\left(\bigcap_{1 \leq i \leq m} \overline{L_{A_i}}\right) = n - m(n-d) \leq \alpha.$$

Let $L_A = \mathbf{v} + \mathcal{L}(\mathbf{B}, \{qx^j\}_{j \in \{0,\ldots,n-1\}})$, where $\mathbf{B}$ is a basis of the vectorial space $\overline{L_A}$ and $\mathbf{v} \in \overline{L_A}$. L_A is an affine lattice, and $\mathbf{s}_1^{[1]}$ will be the shortest vector on L_A, as in the proof of Theorem 3 we cannot use lattice reduction due to the dimension of L_A, but using an appropriate projection, one can retrieve $\mathbf{s}_1^{[1]}$ using the `NearestPlane` algorithm, according to the Gaussian heuristic, $\mathbf{s}_1^{[1]} - \mathbf{v}$ will be the closest vector of $-\mathbf{v}$ in $\mathcal{L}(\mathbf{B}, \{qx^j\}_{j \in \{0,\ldots,n-1\}})$.

4 Generalisation and further developments

In Section 3, it was assumed that the nonces had the same coefficients of higher degree. This section demonstrates that the two theorems of Section 3 continue to apply even when the positions of the common coefficients are unknown to the attacker.

In this section, we know m messages $M_1, \ldots, M_m$ with their associated signatures $\sigma_1, \ldots, \sigma_m$ such that the $\mathbf{y}_i$'s used during the signature share a block of ℓ coefficients in common[3], whose position is unknown. Again even if it means subtracting $\mathbf{z}_1$ from all the other $\mathbf{z}_i$'s, one can consider that the coefficients shared by $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are zero. To simplify the calculations, we make the heuristic assumption that the coefficients of the $\mathbf{z}_i$'s are uniformly distributed in $\{-\gamma_1 + 1, \gamma_1\}$. According to the definition of c and $\mathbf{s}_1$, $||c\mathbf{s}_1||_\infty \leq \beta$. So ℓ coefficients lie in $\{-\beta, \beta\}$. However, for any m signatures, the probability of this happening is:

$$\left(\frac{2\beta + 1}{2\gamma_1}\right)^{\ell \times m}. \tag{1}$$

As soon as this probability is low enough, we can find the block of ℓ coefficients in common by looking at the family of coefficients of the $\mathbf{z}_i$'s. The same kind of reasoning can be applied if an attacker is able to retrieve a coefficient of $\mathbf{y}$ without knowing its exact position. Formally, we know a message M associated with its signature $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ and α, for which there exists an unknown i in $\{1, \ldots, 256\}$ such that $\mathbf{y}_i^{[1]} = \alpha$. It is possible to find i as soon as

$$\left| \{\mathbf{z}_i^{[1]}\}_{i \in \{1,\ldots,n\}} \bigcap \{\alpha - \beta, \ldots, \alpha + \beta\} \right| = 1.$$

[3] The case considered in Section 3 corresponds to a block of size $\ell = n - d$.

If not, the attacker cannot exploit the signature and proceeds with another one. Therefore, it will be possible to find i with probability:

$$\left(1 - \frac{2\beta + 1}{2\gamma_1}\right)^{n-1}. \qquad (2)$$

We have chosen to present our result in the case where the nonce share a block of ℓ common coefficients, but in reality, due to the nature of $\mathcal{R}$, this result can be extended to many other contexts as we will discuss below.

Blocks of hidden shared bits: The attack described previously can be generalized to the case where the nonces share multiple blocks of coefficients. According to the beginning of the present section, it is possible to statistically find the position of the common coefficients: We therefore assume that we know m messages $M_1, \ldots, M_m$ with their associated signatures $\sigma_1, \ldots, \sigma_m$ such that the $\mathbf{y}_i$'s used during the signature share a total of ℓ coefficients in common, divided into δ different blocks as shown in Figure 1.

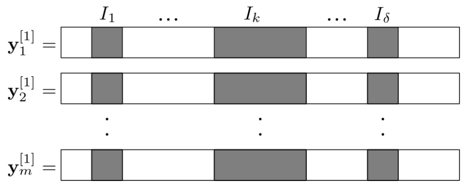

Fig. 1. Biased nonces

Formally for all i in $\{1, \ldots, m\}$:

$$\mathbf{y}_i^{[1]} = \sum_{k=1}^{\delta} \sum_{j \in I_k} \mathbf{y}_{1,j}^{[1]} x^j + \sum_{j \notin \cup I_k} \mathbf{y}_{i,j}^{[1]} x^j$$

where $I_1, \ldots, I_\delta$ represent the coefficients of each block. If $\ell_1, \ldots, \ell_\delta$ denote the length of each block, by definition $\sum \ell_i = \ell$. Even if it means subtracting $\mathbf{z}_1$ from all the $\mathbf{z}_i$'s, it can be assumed that the shared coefficients of $(\mathbf{y}_i)_{i \in \{1,\ldots,m\}}$ are zero. Moreover by posing $P_k(x) = x^{n+1-b_k}$, where b_k denote the last element on the interval I_k for k in $\{1, \ldots, \delta\}$:

$$\deg(P_k \mathbf{y}_i^{[1]}) \leq n - 1 - b_k.$$

It is then sufficient to apply the attack described in Section 3 to the family $\{P_k \mathbf{z}_i\}_{(i,k) \in \{1,\ldots,m\} \times \{1,\ldots,\delta\}}$.

Applications to fault attacks: Our attack, inspired by the result in [EFGT17], can also be seen as a fault attack on the generation of $\mathbf{y}$, since an attacker who injects a fault to stop the generation of the coefficients of $\mathbf{y}^{[1]}$ prematurely will obtain a signature with a corresponding $\mathbf{y}^{[1]}$ of unusually low degree. In [EFGT17] the authors focus on the injection of a single fault on the BLISS signature scheme, we show that this attack works on ML-DSA and is drastically more effective when an attacker is able to retrieve more than one faulty signature. In particular, with sufficiently many faults injected, any part of the generation of $\mathbf{y}$ is vulnerable.

Applications to side-channel attacks: This attack also applies to the case where, by analyzing the generation of the signature, an attacker is able to find one (one more) of the coefficients of the nonce used to generate the signature. To simplify the presentation, we assume that we know m messages $M_1, \dots, M_m$ with their associated signatures $(\mathbf{z}_i, c_i)_{i \in \{1,\dots,m\}}$ and $i_1, \dots, i_m$ such that the attacker knows $\mathbf{y}^{[1]}_{1,i_1}, \dots, \mathbf{y}^{[1]}_{m,i_m}$. To reduce the problem to the one described in Section 3, it is sufficient to pose the following.

$$\forall j \in \{1, \dots, m\},\ P_j(x) = (\mathbf{y}^{[1]}_{j,i_j})^{-1} x^{n-1-i_j}.$$

Therefore, according to the definition of the P_j's, the last coefficient of each $P_j \mathbf{y}^{[1]}_j$ is 1, so this returns to the case of nonce which have their highest degree coefficients in common, as described in Section 3.

5 Experimental results

To demonstrate our theoretical results in practice, we focused on finding the first polynomial of $\mathbf{s}_1$, $\mathbf{s}_1^{[1]}$. However, the reasoning for finding the other polynomials of $\mathbf{s}_1$ is strictly identical. Furthermore, the theory presented does not depend on the level of security, we have only tested the results on ML-DSA level 2 security. As the code is a proof of concept, the results are far from being optimized, all the tests and results presented in this section were carried out using `SageMath`, on a laptop computer equipped with an Intel(R) Core(TM) i7-10850H 2.70GHz CPU. The code used for the attack is available at `https://github.com/AzevedoPaco/AttackML-DSA`. The various constants for the three ML-DSA security levels are shown in Table 2. For Tables 3, 4 and 5, the success statistics are based on an average of 10 keys.

Security level	II	III	V
(k,l)	$(4,4)$	$(6,5)$	$(8,7)$
γ_1	2^{17}	2^{19}	2^{19}
γ_2	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
η	2	4	2
τ	39	49	60
β	78	196	120

Table 1. Settings for different security levels of ML-DSA

Statistics for ML-DSA's parameters: The probabilities formally described in Section 4 are calculated. One wants to find the position of ℓ coefficients in common for the nonces corresponding to m different messages, as the following paragraphs will show, the parameters useful in practical tests are typically: $(m,\ell) \approx (1,150)$ (which was the case studied in the paper [EFGT17]), $(m,\ell) \approx (2,75)$ and $(m,\ell) \approx (150,1)$. For these values the quantity $m \times \ell$ is constant. Table 2 calculates the probabilities described in Section 4, for the three security level of ML-DSA.[4]

Security level	II	III	V
Numeric application in (1)	2^{-1605}	2^{-1707}	2^{-1813}
Numeric application in (2)	0.736	0.826	0.889

Table 2. Numeric application for probabilities in (1) and (2)

Retrieving $\mathbf{s_1}$ with a single faulty signature: As mentioned in Section 4, the attack can be seen as a generalisation of the one described in [EFGT17], with a single signature, our results fall back to those against BLISS described in [EFGT17]. For a single signature and if we note d the degree of $\mathbf{y}^{[1]}$ at the time of its generation, l_{min} the minimum dimension in which the network can be projected, Table 3 summarizes the obtained results.

d	20	40	60	80	90
Theoretical l_{min}	27	53	79	105	118
l in practice	27	53	79	115	188
Probability of success	1	1	1	1	4/5
recover $\mathbf{s}_1^{[1]}$	0.272s	2.65s	13.69s	60.49s	866.9s
used algorithm	`LLL`	`BKZ25`	`BKZ25`	`BKZ30`	`BKZ30`

Table 3. Attack results with a single signature against ML-DSA-II

[4] for m and ℓ such that $m \times \ell \approx 150$.

Retrieving $\mathbf{s_1}$ with few faulty signatures: The main interest of our generalization is to show that the attack described in [EFGT17] becomes drastically more efficient when we assume the attacker is capable of injecting the same fault for at least two executions, Table 4 summarizes the results obtained for two faulty signatures (where the degree of each $\mathbf{y}^{[1]}$ is d).

d	90	110	130	150	170
$\dim_{\mathbb{F}_q}(L)$	1	1	6	46	86
Theoretical l_{min}	1	1	8	53	98
l in practice	50	50	50	60	110
Success for our l	1	1	1	1	1
recover $\mathbf{s}_1^{[1]}$	0.18s	0.198s	0.42s	2.44s	29.6s
used algorithm	`LLL`	`LLL`	`LLL`	`LLL`	`BKZ25`

Table 4. Attack results with two signatures against ML-DSA-II

With just one additional faulty signature, the attack is much more effective, when $m \leq 125$, using lattice reduction is useless since the intersection is trivial (it is a line directed by $\mathbf{s}_1^{[1]}$). By using 3 or more faulty signatures, it would be possible to find $\mathbf{s}_1^{[1]}$ with fewer $\mathbf{y}$ coefficients set to 0.

Retrieving $\mathbf{s_1}$ with a single known coefficient: This paragraph presents the results of the attack, as it was originally described in Section 3, Table 5 summarizes the attack results when the nonce of m signatures share the same last coefficient of highest degree.

m	220	200	180	160
$\dim_{\mathbb{F}_q}(L)$	36	56	76	96
Theoretical l_{min}	41	64	87	109
l in practice	50	65	90	120
Success for our l	1	1	1	1
recover $\mathbf{s}_1^{[1]}$	89.38s	84.27s	84.9s	1744.9s
used algorithm	`LLL`	`BKZ25`	`BKZ25`	`BKZ30`

Table 5. Attack results with m signatures against ML-DSA-II

6 Conclusion

This paper, built on [EFGT17]'s ideas, proposes an attack on ML-DSA under the assumption that an attacker has access to information (implicit or explicit) on the nonce. More precisely, our attacks assume redundancy of information on the nonces used during the generation of some signature, either in the form of unknown common coefficients or null coefficients or in the case where the attacker can know the value of one of the coefficients. In all these cases, we show that ML-DSA's secret key can be extracted from such a set of signatures, in a reasonable time and always with less than 4×160 signatures for ML-DSA-II.

Finding appropriate countermeasures against our new attacks is not obvious. For instance, a technique based on swapping the order of generation of the coefficients of $\mathbf{y}$ was suggested by the authors of [EFGT17] and seemed sufficient to thwart their attacks. However, this technique alone is not able to protect ML-DSA implementations against our improved fault attacks. This suggests the absence of a simple countermeasure and seems to indicate that it is necessary to mask the entire sampling algorithm.

Acknowledgments

This work received funding from the France 2030 program, managed by the French National Research Agency under grant agreements No. ANR-22-PETQ-0008 (PQ-TLS) and ANR-22-CMAS-0001 (QuanTEdu-France), and from ANRT (Association nationale de la recherche et de la technologie).

References

BAE$^+$24. Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes, and Tobias Schneider. Exploiting small-norm polynomial multiplication with physical attacks: Application to crystals-dilithium. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):359–383, Mar. 2024.

BBK16. Nina Bindel, Johannes Buchmann, and Juliane Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. In *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, August 16, 2016*, pages 63–77. IEEE Computer Society, 2016.

BDK$^+$21. Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Algorithm specifications and supporting documentation (version 3.1), 2021. `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`.

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

BVC$^+$23. Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting intermediate value leakage in dilithium: A template-based approach. *IACR TCHES*, 2023(4):188–210, 2023.

CKA$^+$21. 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 *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 583–590, 2021.

EAB$^+$23. Mohamed ElGhamrawy, Melissa Azouaoui, Olivier Bronchain, Joost Renes, Tobias Schneider, Markus Schönauer, Okan Seker, and Christine van Vredendaal. From mlwe to rlwe: A differential fault attack on randomized & deterministic dilithium. Cryptology ePrint Archive, Paper 2023/1074, 2023. `https://eprint.iacr.org/2023/1074`.

EFGT17. Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1857–1874. ACM Press, October / November 2017.

FGR13. Jean-Charles Faugère, Christopher Goyet, and Guénaël Renault. Attacking (EC)DSA given only an implicit hint. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012*, volume 7707 of *LNCS*, pages 252–274. Springer, Heidelberg, August 2013.

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

KPLG24. Elisabeth Krahmer, Peter Pessl, Georg Land, and Tim Güneysu. Correction fault attacks on randomized crystals-dilithium. Cryptology ePrint Archive, Paper 2024/138, 2024. `https://eprint.iacr.org/2024/138`.

LZS$^+$21. Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage. *IEEE Transactions on Information Forensics and Security*, 16:1868–1879, 2021.

Mic08. Daniele Micciancio. Efficient reductions among lattice problems. In *ACM-SIAM Symposium on Discrete Algorithms*, 2008.

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

NIS23. NIST. Fips 204 (draft): Module-lattice-based digital signature standard. Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2023. `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.ipd.pdf`.

NV09. Phong Q. Nguyen and Brigitte Vallée. The lll algorithm - survey and applications. In *Information Security and Cryptography*, 2009.

RCDB22. Prasanna Ravi, Anupam Chattopadhyay, Jan Pieter D'Anvers, and Anubhab Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. Cryptology ePrint Archive, Paper 2022/737, 2022. `https://eprint.iacr.org/2022/737`.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.

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

WNGD23. Ruize Wang, Kalle Ngo, Joel Gärtner, and Elena Dubrova. Single-trace side-channel attacks on CRYSTALS-dilithium: Myth or reality? Cryptology ePrint Archive, Paper 2023/1931, 2023.