Uncompressing Dilithium's public key

Paco Azevedo-Oliveira[1,2], Andersson Calle Viera[1,3], Benoît Cogliati[1], and Louis Goubin[2]

[1] Thales DIS, France
paco.azevedo-oliveira@thalesgroup.com
andersson.calle-viera@thalesgroup.com
benoit-michel.cogliati@thalesgroup.com
[2] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
louis.goubin@uvsq.fr
[3] Sorbonne Université, CNRS, Inria, LIP6, F-75005 Paris, France

Abstract. The Dilithium signature scheme – recently standardized by NIST under the name ML-DSA – owes part of its success to a specific mechanism that allows an optimizaion of its public key size. Namely, among the data of the MLWE instance $(\mathbf{A}, \mathbf{t})$, which is at the heart of the construction of Dilithium, the least significant part of $\mathbf{t}$ – denoted by $\mathbf{t}_0$ – is not included in the public key. The verification algorithm had been adapted accordingly, so that it should not require the knowledge of $\mathbf{t}_0$. However, since it is still required to compute valid signatures, it has been made part of the secret key. The knowledge of $\mathbf{t}_0$ has no impact on the black-box cryptographic security of Dilithium, as can be seen in the security proof. Nevertheless, it does allow the construction of much more efficient side-channel attacks. Whether it is possible to recover $\mathbf{t}_0$ thus appears to be a sensitive question. In this work, we show that each Dilithium signature leaks information on $\mathbf{t}_0$, then we construct an attack that retrieves it from Dilithium signatures. Experimentally, depending on the Dilithium security level, between 200 000 and 500 000 signatures are sufficient to recover $\mathbf{t}_0$ on a desktop computer.

1 Introduction

Dilithium. Following NIST's Post-Quantum Cryptography competition, the Dilithium signature scheme [BDK+21] has been selected as one of the winners under the name ML-DSA. It belongs to the family of lattice-based signature schemes, and is an application of the Fiat-Shamir with abort [Lyu09] to the Module Learning-With-Errors (MLWE) problem. In general, the public key of such a scheme is an (M)LWE instance, which is to say a (matrix, vector) pair $(\mathbf{A}, \mathbf{t})$ such that two "small" *secret* vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ exist such that $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. One of the main selling points of Dilithium is its compressed public key. Indeed, the vector $\mathbf{t}$ is split coefficient-wise into a high and a low part, respectively denoted by $\mathbf{t}_1$ and $\mathbf{t}_0$. The Dilithium public key finally corresponds to the seed that was used to generate the matrix $\mathbf{A}$, along with $\mathbf{t}_1$, while $\mathbf{t}_0$ is considered to

be part of the secret key. The generation of the signature and its verification must be adapted so that the verifier is able to verify the signature without knowledge of $\mathbf{t}_0$.

On the status of $\mathbf{t_0}$. As stated previously, $\mathbf{t}_0$ is considered as a part of the secret key. However, the security proof of Dilitium [BDK+21] assumes that the whole vector $\mathbf{t}$ is given in the public key, which means that this compression is not intended to increase the security of Dilithium. Moreover, Lyubashevsky mentions $\mathbf{t}_0$ in a conference given in 2022 [Lyu22]: "$\mathbf{t}_0$ are not given but they are not secret, some informations is leaked with every signature. The security proof assumes that $\mathbf{t}_0$ is public.". This is further emphasized by the NIST draft standard for ML-DSA [NIS23], which states that $\mathbf{t}_0$ "can be reconstructed from a small number of signatures and, therefore, need not be regarded as secret". Unfortunately, it seems that this claim has never been formally studied. While this may seem benign due to the fact that the formal security of Dilithium does not rely on the secrecy of $\mathbf{t}_0$, it seems that its knowledge can be useful in the context of side-channel attacks.

Some papers assume that $\mathbf{t}_0$ is known to the attacker: in [RRB+19] we can read: "note that the security analysis of DILITHIUM is done with the assumption that the whole of $\mathbf{t}$ is declared as the public key. In addition to this, some information about $\mathbf{t}_0$ is leaked with every published signature and thus the whole of $\mathbf{t}$ can be reconstructed by just observing several signatures generated using the same secret key", but unfortunately again there is no argument or proof.

Others articles remain conservative and study their attacks in both cases: with or without the knowledge of $\mathbf{t}_0$. For example, in [EAB+23a] the authors state that: "the knowledge of $\mathbf{t}_0$ is not required for the MLWE to RLWE reduction part of our attack [...]. However, it has an impact on the resulting security of the RLWE problem making it harder to solve".

There are even papers that explicitly ask for a clarification of the role of $\mathbf{t}_0$ in the side-channel literature on Dilithium. In particular in [WNGD23]: "the main contribution of this paper is highlighting the possibility of recovering the complete secret vector $\mathbf{s}_1$ from a single trace with a non-negligible probability (9% in our experiments) in the case when $\mathbf{t}_0$ is known. None of the previous attacks on Dilithium can recover the full $\mathbf{s}_1$ from fewer than 100 traces. Our results demonstrate the necessity of protecting the secret key of Dilithium from single-trace attacks. They also prompt a reassessment of the role of $\mathbf{t}_0$ in the security of Dilithium implementations."

Finally, at least one paper considers that it is unrealistic to assume that a "real" attacker can find $\mathbf{t}_0$ in a side-channel setting. In [RJH+18] we read: "Thus, it might indeed be possible that the whole of $\mathbf{t}$ leaks as part of the signature and observations of sufficiently many signatures might lead to the recovery of the complete LWE instance, $\mathbf{t}$. But again, we expect the number of signatures and the computational effort to be very high, which cannot be expected in a practical SCA setting".

In conclusion, there is no consensus on the role of $\mathbf{t}_0$ in the case of side-channel attacks. As side-channel protections are known to be costly, it can be tempting to assume that attackers cannot recover $\mathbf{t}_0$, since it allows lighter countermeasures. In the present paper, we show that this assumption is false.

Our contribution. In this article, we study the possibility of recovering $\mathbf{t}_0$ from signatures corresponding to arbitrary messages. In more details, from each signature, we extract inequalities on the coefficients of $\mathbf{t}_0$, until we get a system of linear non-equalities that admit $\mathbf{t}_0$ as its only solution. In order to solve the system, we rely on Linear Programming. The key takeaways of our work are the following:

1. Between 200 000 and 500 000 signatures are needed to reliably recover the value of $\mathbf{t}_0$, depending on the security level.
2. As a consequence, this results in a very large system of inequalities, which is computationally heavy to solve. We built a more efficient approach that builds a sequence of filtered systems of inequalities that have an increasingly smaller number of solutions until $\mathbf{t}_0$ becomes the unique solution.

Overall, our method is relatively simple yet non-trivial, and allows to find $\mathbf{t}_0$ in all our experiments, each time in less than 4 hours on a desktop computer. Our result shows two points: firstly it confirms the fact that not knowing $\mathbf{t}_0$ does not strengthen Dilithium's security (which is not really surprising) and, more importantly, it shows that $\mathbf{t}_0$ can be quickly recovered in practice. Therefore assuming that it can be recovered by a physical attacker is a sound assumption.

Outline. This paper is organized as follows. In Section 2, we redefine the basic notions about Dilithium and recall some results from linear programming which will be useful in the rest of this article. In Section 3, we define and motivate the problem we will solve in the rest of the article. In Section 4, we propose an approach based on linear programming tools. Finally, Section 5 presents the experimental results obtained for this new attack and a brief discussion of our results.

2 Preliminary requirements

In this section we begin by briefly introducing the notations and main functions used in Dilithium. For a detailed description of Dilitihum, the reader is referred to [BDK+21]. We then review the main definitions and results of linear programming, which will be used in the next section.

2.1 Notations, hints and inequalities

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:*

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

Notation 1 *For an integer* $l \in \mathbb{N}^*$ *and for an element* $\mathbf{t}_0 \in \mathcal{R}^l$*, we will note* $\mathbf{t}_0 = (\mathbf{t}_0^{[0]}, ..., \mathbf{t}_0^{[l-1]}) \in \mathcal{R}^l$ *and* $\mathbf{t}_{0,i}^{[j]}$ *will be the i-th coefficient of the polynomial* $\mathbf{t}_0^{[j]}$*.*

Notation 2 *We will note* $[\![$ *statement* $]\!]$ *the boolean operator wich evaluates to* 1 *if statement is true, and to* 0 *otherwise.*

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

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

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

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

and for $\mathbf{w} = (\mathbf{w}^{[0]}, ..., \mathbf{w}^{[l-1]}) \in \mathcal{R}^l$*,*

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

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

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

Dilithium is a signature scheme based on structured lattices, we will therefore manipulate 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 Dilithium uses algorithms that splits elements in $\mathbb{Z}_q$. The first and most natural way is to use bit decomposition: for $r \in \mathbb{Z}_q$ and $d \in \mathbb{N}^*$, $r = r_1 2^d + r_0$ where $r_0 = r \bmod^{\pm} 2^d$ and $r_1 = (r - r_0)/2^d$. This is done with the algorithm $\texttt{Power2Round}_q$ defined in Algorithm 1 and is used to reduce the size of the public key $\mathbf{t}$.

Since the public key is not "completely" known to the verifier, the signer must add "hints" to the signature to allow its verification. Given $r \in \mathbb{Z}_q$ and a small element $z \in \mathbb{Z}_q$, the verifier must calculate the high bits of $z + r$ without knowing z. To do this, the authors have decided to use a slightly different split: for an even α divisor of $q - 1$ and $r \in \mathbb{Z}_q$ they define $r = r_1\alpha + r_0$ with $r_0 = r \bmod^{\pm}(\alpha)$ and $r_1 = (r - r_0)/\alpha$. We will call r_1 the high bits of r and r_0 the low bits of r. As shown in Figure 1, for $z \in \mathbb{Z}_q$ such that $|z| \leq \alpha/2$, adding z to r can only increase or decrease the high bits of r by ± 1. With this simple tweak one can calculate the high bits of $r + z$, only with the knowledge of z and a hint bit $h \in \{0, 1\}$. Algorithm 1 give the description of the algorithms and Lemma 1 recall the main property used.

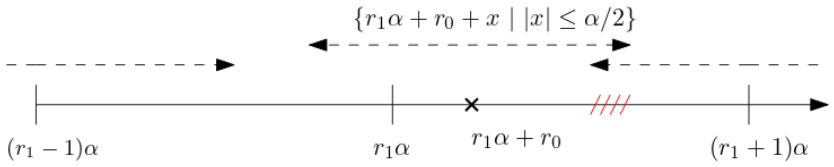

Fig. 1. carry caused by x

Algorithm 1 `Supporting algorithms for Dilithium`

$\texttt{Power2Round}_q(r, d)$:
1: $r = r \bmod^+ q$
2: $r_0 = r \bmod^\pm 2^d$
3: **return** $(r - r_0)/2^d, r_0)$

$\texttt{Decompose}_q(r, \alpha)$:
1: $r = r \bmod^+ q$
2: $r_0 = r \bmod^\pm \alpha$
3: **if** $r - r_0 = q - 1$ **then**
4: $\quad r_1 = 0$
5: $\quad r_0 = r_0 - 1$
6: **else** $r_1 = (r - r_0)/\alpha$
7: **return** (r_1, r_0)

$\texttt{HighBits}_q(r, \alpha)$:
1: $(r_1, r_0) = \texttt{Decompose}_q(r, \alpha)$
2: **return** r_1

$\texttt{LowBits}_q(r, \alpha)$:
1: $(r_1, r_0) = \texttt{Decompose}_q(r, \alpha)$
2: **return** r_0

$\texttt{MakeHint}_q(z, r, \alpha)$:
1: $r_1 = \texttt{HighBits}_q(r, \alpha)$
2: $v_1 = \texttt{HighBits}_q(r + z, \alpha)$
3: **return** $[\![r_1 \neq v_1]\!]$

$\texttt{UseHint}_q(h, r, \alpha)$:
1: $m = (q - 1)/\alpha$
2: $(r_1, r_0) = \texttt{Decompose}_q(r, \alpha)$
3: **if** $h = 1$ and $r_0 > 0$ **then**
4: $\quad$ **return** $(r_1 + 1) \bmod^+ m$
5: **if** $h = 1$ and $r_0 \leq 0$ **then**
6: $\quad$ **return** $(r_1 - 1) \bmod^+ m$
7: **return** r_1

Lemma 1 *[LDK+22] Let q and α be two positive integers such that $q > 2\alpha$, $q \equiv 1 \mod (\alpha)$ and α even. Let $\mathbf{r}$ and $\mathbf{z}$ be two vectors of $\mathcal{R}_q$ such that $||\mathbf{z}||_\infty \leq \alpha/2$ and let $\mathbf{h}, \mathbf{h}'$ be bit vectors. So the algorithms $\texttt{HighBits}_q$, $\texttt{MakeHint}_q$, $\texttt{UseHint}_q$ satisfy the properties:*

$$\texttt{UseHint}_q(\texttt{MakeHint}_q(\mathbf{z}, \mathbf{r}, \alpha), \mathbf{r}, \alpha) = \texttt{HighBits}_q(\mathbf{r} + \mathbf{z}, \alpha).$$

2.2 Algorithm description

Key Generation: The key generation algorithm is described in Algorithm 2. Dilithium 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, $\mathbf{A} \in \mathcal{R}_q^{k \times l}$ and $\mathbf{s}_1 \in S_\eta^l$ 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. This reduces the size of the public key by half at the cost of adding a few bits at the time of signing, so that the verifier does not need the knowledge of $\mathbf{t}_0$.

Algorithm 2 `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 3. The signer derives a masking vector $\mathbf{y} \in \mathcal{R}_q^l$, from which it calculates $\mathbf{w}_1$, the high 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:

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

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 $\texttt{HighBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t},\, 2\,\gamma_2)$. Using the method described in subsection 2.1, 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 the signature, 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 seed $\tilde{c}$, used to sample the challenge polynomial, the response vector $\mathbf{z}$, and the hint vector $\mathbf{h}$.

Algorithm 3 `Sig`

Require: sk, M
Ensure: $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$
$\mathbf{A} \in \mathcal{R}_q^{k \times l} := \texttt{ExpandA}(\rho)$
$\mu \in \{0,1\}^{512} := \mathrm{H}(tr \,||\, M)$
$\kappa := 0$, $(\mathbf{z}, \mathbf{h}) := \perp$
$\rho' \in \{0,1\}^{512} := \mathrm{H}(K \,||\, \mu)$
while $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
    $\mathbf{y} \in \tilde{S}_{\gamma_1}^l := \texttt{ExpandMask}(\rho', \kappa)$
    $\mathbf{w} := \mathbf{A}\,\mathbf{y}$
    $\mathbf{w}_1 = \texttt{HighBits}_q(\mathbf{w}, 2\,\gamma_2)$
    $\tilde{c} \in \{0,1\}^{256} := \mathrm{H}(\mu \,||\, \mathbf{w}_1)$
    $c \in B_\tau := \texttt{SampleInBall}(\tilde{c})$
    $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
    $\mathbf{r}_0 := \texttt{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\,\gamma_2)$
    if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
        $(\mathbf{z}, \mathbf{h}) := \perp$
    else
        $\mathbf{h} := \texttt{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
        if $||c\,\mathbf{t}_0||_\infty \geq \gamma_2$ or $|\mathbf{h}|_{\mathbf{h}_j=1} > \omega$ **then**
            $(\mathbf{z}, \mathbf{h}) := \perp$
    $\kappa := \kappa + l$
return $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Verification: The verification algorithm is described in Algorithm 4. To verify the signature, the verifier begins by reconstructing the matrix $\mathbf{A}$ and the polynomial c on which the signer has commited. Using the vector $\mathbf{h}$ of the signature and $\texttt{UseHint}_q$, $\mathbf{w}_1 = \texttt{HighBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\,\gamma_2)$ is recalculated. 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 4 `Ver`

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

Remark 1 *As shown above, in the formal definition of Dilithium,* $\mathbf{t}_0$ *is considered to be secret data even though it reveals nothing about the other polynomials of the secret key. Therefore, in a side channel attack, an attacker cannot use knowledge of* $\mathbf{t}_0$ *to attack a Dilithium implementation. Despite this, a large proportion of papers on side-channel or fault-based attacks against Dilithium [BVC+23] [RRB+18] [RRB+18] [EAB+23b] make the assumption that* $\mathbf{t}_0$ *is known. In the rest of the paper we will show that* $\mathbf{t}_0$ *can indeed be considered as part of the public key, since it can be reconstructed from Dilithium signatures.*

2.3 An overview of Polyhedral Theory

A polyhedron is a set of points verifying a finite number of inequalities, in other words: an instersection of a finite number of half-spaces. We are interested in this geometrical object because in Section 3 we will show that by querying Dilithium signatures generated under the same secret key, we will collect inequalities on the coefficients of the polynomial vector $\mathbf{t}_0$. $\mathbf{t}_0$ will therefore be in a bounded polyhedron, traditionally called a polytope. Obtaining information about this polytope will allow us to find $\mathbf{t}_0$ in Section 4. We refer to [NW88] for general definitions and unproven propositions.

Definition 4 *A polyhedron* $P \subset \mathbb{R}^n$ *is the set of points that satisfy a finite number of linear inequalities,* $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$ *where* (A, b) *is a* $m \times (n+1)$ *matrix.*

Definition 5 *A polyhedron* $P \subset \mathbb{R}^n$ *is bounded if there exists an* $w \in \mathbb{R}_+$ *such that* $P \subset \{x \in \mathbb{R}^n : \ -w \leq x_j \leq w \text{ for } j = 1, ..., n\}$. *A bounded polyhedron is called a polytope.*

Definition 6 *Let* $P \subset \mathbb{R}^n$ *be a polytope, we call the diameter of* P *and we note* $diam(P)$ *the quantity:*

$$diam(P) = \max_{p_1, p_2 \in P} ||p_1 - p_2||_\infty$$

Definition 7 *A polyhedron* P *is of dimension* k*, denoted by* $dim(P) = k$*, if the maximum number of affinely independent points in* P *is* $k + 1$.

Remark 2 *The definition of diameter and dimension provide an estimation on the number of elements in a polytope. In our case, we are going to collect inequalities verified by* $\mathbf{t}_0$*, so we will obtain a polytope containing* $\mathbf{t}_0$*. Estimating the dimension and diameter of this polytope allows us to obtain an estimation on the coefficients of* $\mathbf{t}_0$.

2.4 The basics of Linear Programming

The general linear programming problem is to find:

$$z_{LP} = \max\{cx : \ Ax \leq b, \ x \in \mathbb{R} \ \}$$

where A is a $m \times n$ matrix and c, b are $1 \times n$ and $m \times 1$ matrices. This problem is well defined is the sense that if it is feasible and does not have unbounded optimal values, then it has an optimal solution. In the rest of this paper, we will note (LP) and write it in the following form:

$$\begin{aligned} &\text{maximize } cx \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \in \mathbb{R} \end{aligned}$$

Remark 3 *Let P be a polytope described by a set of inequalities. Trivialy, finding an $x \in P$ (i.e. a point that satisfies all the inequalities that form the description of P) is an (LP) problem, as it can be solved by maximizing any function on P. Minimising a function on P is also an (LP) problem as it is sufficient to maximise its opposite.*

Proposition 1 *Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be a polytope, upper-bounding the dimension of P or calculating the diameter of P are two (LP) problems.*

Proof. For $i \in \{1, \ldots, n\}$, by solving the following two (LP) problems:

$$\begin{aligned} &\text{minimize } x_i \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \in \mathbb{R}^n \end{aligned} \qquad\qquad \begin{aligned} &\text{maximize } x_i \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \in \mathbb{R}^n \end{aligned}$$

Fig. 2. The 2×256 (LP) problems related to P.

Calculating $\text{card}(\{i \in \{1, \ldots, n\} : \exists w_i \in \mathbb{R}, \forall x \in P,\ x_i = w_i\})$ allows to upper-bound the dimension of P. Solving the same (LP) problems allow also to calculate the diameter of P.

□

Notation 3 *Let $P \subset \mathbb{R}^n$ be a polytope. The procedure for calculating a point by minimizing the null function on P is denoted* `lp_guess`*, and we denote* `calculate_diam` *the procedure which consists in computing the diameter of P.*

3 Problem definition and existing solutions

In the rest of the paper, we study the case of an attacker who tries to recover $\mathbf{t}_0$ based on knowledge of $pk = (\rho, \mathbf{t}_1)$ and a certain number of Dilithium signatures $\{\sigma_i\}_{i \in I}$ signed under the corresponding secret key $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$. In this section we show that each Dilithium signature provides information on the coefficients of $\mathbf{t}_0$, in the form of inequalities on its coefficients. Naturally, we will try to exploit this leakage of information by using linear programming theory to propose a solution.

Proposition 2 *Let $j \in \{0, \ldots, k-1\}$ and $i \in \{0, \ldots, 255\}$ and $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ be a signature of* `Sig`.

- *If* $\mathbf{h}_i^{[j]} = 0$*:*

$$|(-c\mathbf{t}_0)_i^{[j]} + \mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]}| \leq \gamma_2 - \beta - 1.$$

- *If* $\mathbf{h}_i^{[j]} = 1$ *and* $\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} > 0$*:*

$$(-c\mathbf{t}_0)_i^{[j]} \geq \gamma_2 + \beta + 1 - \mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} \geq 0.$$

- *If* $\mathbf{h}_i^{[j]} = 1$ *and* $\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} < 0$*:*

$$(-c\mathbf{t}_0)_i^{[j]} \leq -(\gamma_2 + \beta + 1) - \mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} \leq 0.$$

Remark 4 $\mathbf{A}$ *and* $\mathbf{t}_1$ *are publicly known and* $c, \mathbf{z}, \mathbf{h}$ *belongs to the signature, so an attacker can calculate* $\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$*. Since* γ_2 *and* β *are known parameters, an attacker can calculate the bound of the inequation obtained on* $\mathbf{t}_0$.

Proof. Let $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ be a signature of `Sig`, $j \in \{0, \ldots, k-1\}$ and $i \in \{0, \ldots, 255\}$.

- If $\mathbf{h}_i^{[j]} = 0$, by definition of $\mathbf{h}_i^{[j]}$:

$$\mathtt{HighBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)_i^{[j]} = \mathtt{HighBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]}.$$

  Using the relation $\mathbf{Az} - c\mathbf{t} = \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d - c\mathbf{t}_0$ and decomposing $\mathbf{Az} - c\mathbf{t}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d$ with their high and low bits, we obtain the following relation:

$$\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)_i^{[j]} = \mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} - (c\mathbf{t}_0)_i^{[j]},$$

  and by definition of the signature:

$$|\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)_i^{[j]}| < \gamma_2 - \beta.$$

  So we finally obtain:

$$|-(c\mathbf{t}_0)_i^{[j]} + \mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]}| \leq \gamma_2 - \beta - 1.$$

- If $\mathbf{h}_i^{[j]} = 1$ and $\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} > 0$, by definition of $\mathbf{h}_i^{[j]}$:

$$\mathtt{HighBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)_i^{[j]} = \mathtt{HighBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} + 1.$$

  Using the relation $\mathbf{Az} - c\mathbf{t} = \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d - c\mathbf{t}_0$ and decomposing $\mathbf{Az} - c\mathbf{t}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d$ with their high and low bits, we obtain the following relation:

$$\mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)_i^{[j]} = \mathtt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)_i^{[j]} - (c\mathbf{t}_0)_i^{[j]} - 2\gamma_2$$

By definition of the signature:

$$|\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\,\gamma_2)_i^{[j]}| < \gamma_2 - \beta.$$

By replacing $\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\,\gamma_2)_i^{[j]}$ with the equation above and using the fact that $||\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\,\gamma_2)||_\infty \leq \gamma_2$ and $||c\mathbf{t}_0||_\infty < \gamma_2$, we get:

$$(-c\mathbf{t}_0)_i^{[j]} \geq \gamma_2 + \beta + 1 - \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} \geq 0.$$

– If $\mathbf{h}_i^{[j]} = 1$ and $\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} < 0$:
By definition of $\mathbf{h}_i^{[j]}$:

$$\texttt{HighBits}_q(\mathbf{Az} - c\mathbf{t}, 2\,\gamma_2)_i^{[j]} = \texttt{HighBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} - 1.$$

Using the same reasoning, we obtain:

$$\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\,\gamma_2)_i^{[j]} = \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} - (c\mathbf{t}_0)_i^{[j]} + 2\,\gamma_2$$

and therefore:

$$|\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} - (c\mathbf{t}_0)_i^{[j]} + 2\,\gamma_2| < \gamma_2 - \beta.$$

Finally, because $||\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}, 2\,\gamma_2)||_\infty \leq \gamma_2$ and $||c\mathbf{t}_0||_\infty < \gamma_2$:

$$(-c\mathbf{t}_0)_i^{[j]} \leq -(\gamma_2 + \beta + 1) - \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} \leq 0.$$

□

The reasoning above, for the second case is summarised in Figure 3. In red the impossible values of $(-c\mathbf{t}_0)_i^{[j]}$ according to the value of $\mathbf{h}_i^{[j]}$. In purple, the impossible values of $(-c\mathbf{t}_0)_i^{[j]}$ according to the generation of the signature.

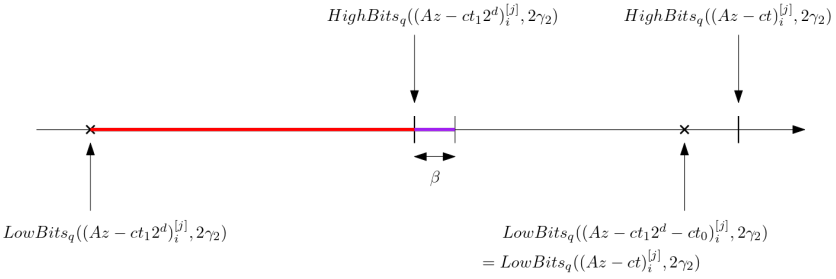

Fig. 3. Idea to obtain inequalities on $\mathbf{t_0}$.

To measure the frequency with which we obtain an inequation on the coefficient of $\mathbf{t}_0$, we collected 10 000 signatures for an equal number of random

messages for 10 random keys, for the three security level of Dilithium. The practical results are summarised in Table 1 below. For the sake of clarity, we have split the inequations obtained for coefficients of $\mathbf{h}$ equal to 0 (on the left) from the inequations obtained for coefficients of $\mathbf{h}$ equal to 1 (on the right).

NIST Level	II	III	V
Average inequation obtained	$1\,922 + 63$	$2\,996 + 38$	$3\,984 + 56$

Table 1. Average number of inequalities per signature, over 10 000 signatures, for different security levels.

Remark 5 *To visualise the information obtained, we can study the following problem: an attacker knows all the coefficients of* $\mathbf{t}_0^{[0]}$ *except the first two,* $\mathbf{t}_{0,0}^{[0]}$ *and* $\mathbf{t}_{0,1}^{[0]}$*. He queries signatures and obtains inequalities on the two missing coefficients, which can be represented as a point in the set of solutions to the inequalities it has collected.*

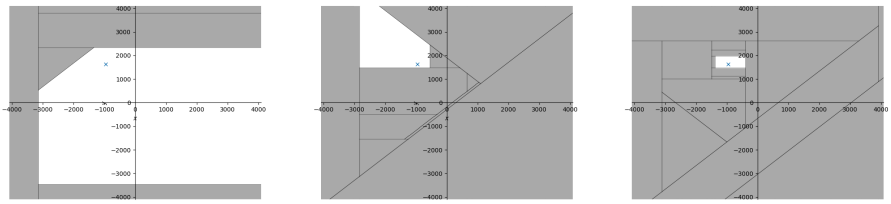

Fig. 4. Polytope containing $(\mathbf{t}_{0,0}^{[0]}, \mathbf{t}_{0,1}^{[0]})$ for $10, 50$ and 100 inequalities.

In Figure 3, the two coefficients the attacker is seeking are $(-961, 1631)$ *and the white part represents the polytope of solutions for* $10, 50$ *and* 100 *collected inequalities. As can be seen in the third image, we have an increasingly complex algebraic description (several dozen inequations, most of which are not useful) of a simple geometric object (a polytope with* 5 *faces).*

3.1 A state-of-the-art solution?

The paper [BDE+18] studies the following problem: A vector $\mathbf{s} \in \mathbb{Z}^n$ is fixed and secret, given m samples $(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + e_1), \ldots, (\mathbf{a}_m, \langle \mathbf{a}_m, \mathbf{s} \rangle + e_m)$, where for all i in $\{1, \ldots, m\}$, $\mathbf{a}_i \in \mathbb{Z}^n$ and $e_i \in \mathbb{Z}$ follows a probability distribution $(\chi_a)^n$ and χ_e respectively, is it possible to find $\mathbf{s}$? The paper answers yes to the question (under assumptions detailed later) and gives the answer in two steps: from the point of view of information theory, it gives a minimum bound for m (the number of

samples required). Then they give a practical way of finding $\mathbf{s}$ where the number of samples required is only slightly higher than the theoretical bound. For the practical solution, they use the method of least squares. To avoid overloading the paper, only the main results of the article are given.

Remark 6 *It is natural to relate this problem to the problem of finding* $\mathbf{t}_0$*. Each signature gives inequalities on a linear combination of the coefficients of* $\mathbf{t}_0$*. Consequently, each signature provides an approximate value on a linear combination of the coefficients of the polynomials of* $\mathbf{t}_0$*.*

Definition 8 *(ILWE Distribution [BDE+18]) For any vector* $\mathbf{s} \in \mathbb{Z}^n$ *and any two probability distributions* χ_a, χ_e *over* $\mathbb{Z}$*, the* $ILWE$ *distribution* $\mathcal{D}_{\mathbf{s},\chi_a,\chi_e}$ *associated with those parameters is the probability distribution over* $\mathbb{Z}^n \times \mathbb{Z}$ *defined as follows: samples from* $\mathcal{D}_{\mathbf{s},\chi_a,\chi_e}$ *are of the form*

$$(\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \quad \textit{with} \quad \mathbf{a} \leftarrow \chi_a^n \textit{ and } e \leftarrow \chi_e.$$

Definition 9 *(ILWE Problem [BDE+18]) The ILWE problem is the computational problem parametrized by* n, m, χ_a, χ_e *in which, given* m *samples* $\{(\mathbf{a}_i, b_i)\}_{1 \leq i \leq m}$ *from a distribution of the form* $\mathcal{D}_{\mathbf{s},\chi_a,\chi_e}$ *for some* $\mathbf{s} \in \mathbb{Z}^n$*, one is asked to recover the vector* $\mathbf{s}$*.*

Definition 10 *(Subgaussian random variable [BDE+18])*

A random variable X *over* $\mathbb{R}$ *is said to be* $\zeta-$*subgaussian for some* $\zeta > 0$ *if the following bound holds for all* $s \in \mathbb{R}$ *:*

$$\mathbb{E}[\exp(sX)] \leq \exp(\frac{\zeta^2 s^2}{2}).$$

A random vector $\mathbf{x} \in \mathbb{R}^n$ *is called a* $\zeta-$*subgaussian random vector if for all vectors* $\mathbf{u} \in \mathbb{R}^n$ *with* $||\mathbf{u}||_2 = 1$*, the inner product* $\langle \mathbf{u}, \mathbf{x} \rangle$ *is a* $\zeta-$*subgaussian random variable.*

Theorem 1 *([BDE+18]) Suppose that* χ_a *is a* ζ_a-*subgaussian and* χ_e *is* ζ_e-*subgaussian, and let* $(\mathbf{M}, \mathbf{b} = \mathbf{M}\mathbf{s}+\mathbf{e})$ *the data constructed from* m *samples of the* $ILWE$ *distribution* $\mathcal{D}_{\mathbf{s},\chi_a,\chi_e}$ *for some* $\mathbf{s} \in \mathbb{Z}^n$*. There exist constants* $C, C' > 0$ *such that if:*

$$m \geq Cn \quad \textit{and} \quad m \geq C' \times \frac{\sigma_e^2}{\sigma_a^2} \log(n)$$

then the least square estimators $\tilde{\mathbf{s}} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{b}$ *satisfies* $||\mathbf{s} - \tilde{\mathbf{s}}||_\infty < 1/2$*, and hence* $\lceil \tilde{\mathbf{s}} \rfloor = \mathbf{s}$*, with probability at least* $1 - 1/n$*.*

Building the ILWE system After collecting enough signatures, we will have multiple inequalities on the k polynomials of $\mathbf{t}_0$ independently, so we can split the problem into k smaller ones, one for each polynomial of the vector $\mathbf{t}_0$. For the sake of clarity, let us explain the methodology for a single polynomial of the

vector $\mathbf{t}_0 = (\mathbf{t}_0^{[0]}, \ldots, \mathbf{t}_0^{[k-1]})$. We select a signature that gives an inequation on $\mathbf{t}_0^{[0]}$. Let $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ be a signature and $i \in \{0, \ldots, 255\}$ such that $\mathbf{h}_i^{[0]} = 1$ and $\texttt{LowBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]} > 0$:

$$(-c\mathbf{t}_0)_i^{[0]} \geq \gamma_2 + \beta + 1 - \texttt{LowBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}, \tag{1}$$

$$\sum_{j=0}^{n-1} \mathbf{t}_{0,j}^{[0]}(-cx^j)_i \geq \gamma_2 + \beta + 1 - \texttt{LowBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}. \tag{2}$$

Since the polynomial c is known, σ gives us an inequality on the coefficients of $\mathbf{t}_0^{[0]}$. The case of $\mathbf{h}_i^{[0]} = 0$ and $\texttt{LowBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2) < 0$ are treated in the same way. Thus, with these signatures, we can construct two matrices $\mathbf{M}_+$ and $\mathbf{M}_-$ and four vectors $\mathbf{e}_+$, $\mathbf{b}_+$ and $\mathbf{e}_-$, $\mathbf{b}_-$ such that $\mathbf{b}_+ = \mathbf{M}_+\mathbf{t}_0^{[0]} + \mathbf{e}_+$ and $\mathbf{b}_- = \mathbf{M}_-\mathbf{t}_0^{[0]} + \mathbf{e}_-$. Each row of one of these matrices representing an inequality collected on $\mathbf{t}_0^{[0]}$. For the rest of this part, we assume that we have concatenated $\mathbf{M}_+, \mathbf{b}_+, \mathbf{e}_+$ and $\mathbf{M}_-, \mathbf{b}_-, \mathbf{e}_-$ to obtain $\mathbf{M}, \mathbf{b}, \mathbf{e}$ such that:

$$\mathbf{b} = \mathbf{M}\mathbf{t}_0^{[0]} + \mathbf{e}.$$

Remark 7 *To strictly apply Theorem 1, it must be proved that the coefficients of the matrix* $\mathbf{M}$*, as well as the error vector* $\mathbf{e}$*, follow subgaussian distributions, and then estimate the corresponding standard deviations* σ_a *and* σ_e*. As always when moving from theory to practice, certain simplifications are necessary.*

Estimating probability distributions We want now to calculate, if not estimate, the variance of the distribution followed by the coefficients of the matrix $\mathbf{M}$ and the error vector $\mathbf{e}$. Each row of the matrix $\mathbf{M}$ is built using the coefficients of a vector c generated with a Dilithium signature. Therefore, each line of $\mathbf{M}$ has its coefficients in $\{-1, 0, 1\}$, of which τ coefficients will be ± 1 (with equiprobability), the remainder being at zero. It is not true that the coefficients of the matrix $\mathbf{M}$ are independent. For example, the number of zeros in each row of M will always be $256 - \tau$. Even worse, by retrieving an inequation from a zero coefficient of $\mathbf{h}$, consecutive rows of the matrix $\mathbf{M}$ will be identical. Nevertheless, we make the heuristic assumption that it is the case, so that we can explicitly calculate the law followed by the coefficients of the matrix $\mathbf{M}$.

Lemma 2 *Let* $m \in \mathbb{N}$ *be the number of collected inequalities on* $\mathbf{t}_0^{[0]}$*, and* τ *the numbers of non-zero coefficient in* c*. Under the heuristic assumptions described above, the coefficients of the matrix* $\mathbf{M} = (a_{i,j})_{0 \leq i \leq m-1, 0 \leq j \leq n-1}$ *follow the law below:*

$$\mathbb{P}(a_{i,j} = 0) = \frac{256 - \tau}{256} \quad \textit{and} \quad \mathbb{P}(a_{i,j} = 1) = \mathbb{P}(a_{i,j} = -1) = \frac{\tau}{2 \times 256}.$$

Furthermore, this distribution is subgaussian and its variance can be calculated explicitly:

$$\sigma_a^2 = \frac{\tau}{256}.$$

Proof. According to [BDE+18], any real variable whose probability distribution has a mean of zero and is carried by a segment is subgaussian, which is the case for $a_{i,j}$. In addition, the variance formula gives:

$$\sigma_a^2 = \mathbb{E}(a_{i,j}^2) = \frac{\tau}{256}.$$

□

To demonstrate that the vector $\mathbf{e}$ is subgaussian, according to [BDE+18] it is sufficient to show that it has null expectation and that it is carried by a segment. Since $\mathbf{e}$ is defined by parts (depending on the inequalities we recover), it is sufficient to show that each part is subgaussian. In order not to overload this section, we will only give details of the inequalities obtained for coefficients of $\mathbf{h}$ at 0. Let $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ be a signature of $\texttt{Sig}$ and $i \in \{0, \ldots, 255\}$ such that $\mathbf{h}_i^{[0]} = 0$:

$$|(-c\mathbf{t}_0)_i^{[0]} + \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}| \leq \gamma_2 - \beta - 1.$$

The coefficients of $\mathbf{t}_0^{[0]}$ are fixed in $\{-2^{12}+1, \ldots, 2^{12}\}$ and from the generation of the signature $||c\mathbf{t}_0||_\infty < \gamma_2$, so we can assume that $(-c\mathbf{t}_0)_i^{[0]}$ follows a gaussian distribution centered and carried by $\{-\gamma_2+1, \ldots, \gamma_2-1\}$. If we also assume that all the coefficients of $\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)$ follow a uniform distribution in $\{-\gamma_2+1, \ldots, \gamma_2\}$, then $\gamma_2 - \beta - 1 - \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}$ follows a uniform distribution in $\{-\beta - 1, \ldots, (2\gamma_2 - \beta - 1) - 1\}$ By the same reasoning, $-(\gamma_2 - \beta - 1) - \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}$ follows a uniform distribution in $\{-(2\gamma_2 - \beta - 1), \ldots, \beta\}$. Finally, even if it means adding $1/2$ to the error obtained, the error vector obtained from the inequations where the coefficient of $\mathbf{h}$ is zero has a null expectation and is carried by the following segment: $\{-(3\gamma_2 - \beta - 2), \ldots, 3\gamma_2 - \beta - 2\}$. With the same assumptions and the same reasoning, we can show that the error vector obtained for the inequalities given by the coefficients of $\mathbf{h}$ at 1 is also of zero expectation and is carried by the segment $\{-(3\gamma_2 + \beta), \ldots, 3\gamma_2 + \beta\}$. Finally, the error vector obtained by concatenating all the inequalities has zero expectation and is carried by the segment $\{-(3\gamma_2 + \beta), \ldots, 3\gamma_2 + \beta\}$. According to [BDE+18], $\mathbf{e}$ is $(3\gamma_2 + \beta)$-subgaussian and:

$$\sigma_e^2 \leq (3\gamma_2 + \beta)^2.$$

Table 2 summarises the different values of τ, σ_a and σ_e (theoretical and experimental) according to the different security levels.[4]

[4] For each experimental measurement, the results are averages over 5 keys and the unbiased sample variance was calculated using 500 000 samples.

NIST Level	II	III	V
τ	39	49	60
σ_a^2 (theory)	0.1523	0.19141	0.234375
σ_a^2 (exp.)	0.1523	0.19139	0.234372
upper bound on σ_e^2	81 666 779 076	617 575 939 600	617 456 494 656
σ_e^2 (exp.)	11 411 519 137	30 832 793 508	30 752 629 506

Table 2. Value of the standard deviation according to the level of security.

It will be possible to find $\mathbf{t}_0$ using Theorem 1 with $\Omega((\sigma_e/\sigma_a)^2 \log(n))$ inequalities. However, as can be seen in Table 3 the variance of the error is significantly higher than the variance of the coefficients of the matrix $\mathbf{M}$. As it would be necessary to collect at least $m = (\sigma_e/\sigma_a)^2$ inequalities over $\mathbf{t}_0$, the size of the matrix $\mathbf{M} = (a_{i,j})_{0 \leq i \leq m-1, 0 \leq j \leq n-1}$ makes it impossible to calculate the least square estimator in practice (because it would be necessary to manipulate matrices of a size close to m). Furthermore according to [BDE+18], from an information-theoretic point of view, with fewer than $m = (\sigma_e/\sigma_a)^2$ inequalities it is not possible to distinguish the distribution of $\mathcal{D}_{\mathbf{t}_0,\chi_a,\chi_e}$ from $\mathcal{D}_{\mathbf{s},\chi_a,\chi_e}$ with $\mathbf{s} \neq \mathbf{t}_0$.

NIST Level	I	III	V
$\sigma_e^2/\sigma_a^2 \times \log(n)$	2^{39}	2^{40}	2^{39}

Table 3. Theoretical bound to distinguish $\mathbf{t}_0$.

Remark 8 *The ILWE system obtained is far too "noisy", but in our case we get more than an approximation of a linear combination of certain coefficients of* $\mathbf{t}_0$*, we also know whether this approximation is greater (in the case of a "upper inequality") or lower (in the case of a "lower inequality") than the value we are seeking. This detail changes the situation: taking this information into account by reformulating the problem of finding* $\mathbf{t}_0$ *into a linear programming problem will allow us to create an attack that will require far fewer inequations than the ILWE method.*

Nevertheless, this section shows that the distribution of the inequalities obtained for $\mathbf{t}_0$ follows a subgaussian distribution centered around the expected value. This result will be important in Section 4 when it comes to demonstrating the termination of our attack.

4 An attack methodology

The natural approach is to recover enough inequalities on $\mathbf{t}_0$ to form a system (LP) for which it is the unique solution. As we shall see later, this "naive" solution is not possible in our case.

Building the (LP) system After collecting enough signatures, we will have multiple inequalities on the k polynomials of $\mathbf{t}_0$ independently, so we can split the problem into k smaller ones, one for each polynomial of the vector $\mathbf{t}_0$. We explain the methodology for a single polynomial of the vector $\mathbf{t}_0 = (\mathbf{t}_0^{[0]}, ..., \mathbf{t}_0^{[k-1]})$. We select a signature that gives an inequation on $\mathbf{t}_0^{[0]}$. Let $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ be such a signature, with i such that $\mathbf{h}_i^{[0]} = 1$. Assuming $\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]} > 0$, one has

$$(-c\mathbf{t}_0)_i^{[0]} \geq \gamma_2 + \beta + 1 - \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}, \tag{3}$$

$$\sum_{j=0}^{n-1} \mathbf{t}_{0,j}^{[0]}(-cx^j)_i \geq \gamma_2 + \beta + 1 - \texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[0]}. \tag{4}$$

Since the polynomial c is known, σ gives an inequality on the coefficients of $\mathbf{t}_0^{[0]}$. The other two inequalities are treated in the same way. Thus, with these signatures, we can construct two matrices $\mathbf{M}_+$ and $\mathbf{M}_-$ and two vectors $\mathbf{b}_+$ and $\mathbf{b}_-$ such that $\mathbf{t}_0^{[0]} \in \{\mathbf{x} \in \{-2^{12}+1, \ldots, 2^{12}\}^n \mid \mathbf{M}_+\mathbf{x} \geq \mathbf{b}_+ \text{ and } \mathbf{M}_-\mathbf{x} \leq \mathbf{b}_-\}$. Each row of one of these matrices representing an inequality collected on $\mathbf{t}_0^{[0]}$. In particular, if we collect enough inequalities for $\mathbf{t}_0^{[0]}$ to be the only solution, we can find $\mathbf{t}_0^{[0]}$ by solving the following (LP) problem of dimension $n = 256$:

$$\begin{aligned} &\text{maximize } 0 \\ &\text{subject to } \mathbf{M}_+\mathbf{x} \geq \mathbf{b}_+ \\ &\qquad\qquad \mathbf{M}_-\mathbf{x} \leq \mathbf{b}_- \\ &\qquad\qquad \mathbf{x} \in [-2^{12}+1, 2^{12}]^n \end{aligned}$$

Fig. 5. The (LP) problem related to $\mathbf{t}_0^{[0]}$.

Results using the naive approach. Unless a huge number of inequalities is collected, as can be seen in the Table 4, $\mathbf{t}_0$ will never be the only solution. Nevertheless, we can assume that we know $\mathbf{t}_0$, in order to estimate the size of the polytope containing it. In Table 4, the attack time takes into account the time required to generate the signatures and the time required to build and solve the k (LP) systems associated with $\mathbf{t}_0$, however the number of inequalities only includes inequations associated with $\mathbf{t}_0^{[0]}$ and $\tilde{\mathbf{t}}_0^{[0]}$ is the polynomial obtained

by minimizing the null function on the polytope defined by the inequalities collected on $\mathbf{t}_0^{[0]}$. For the sake of clarity, we have split the inequations obtained for coefficients of $\mathbf{h}$ equal to 0 (on the left) from the inequations obtained for coefficients of $\mathbf{h}$ equal to 1 (on the right).

Number of signatures	Number of inequalities	$\|\|\mathbf{t}_0^{[0]} - \tilde{\mathbf{t}}_0^{[0]}\|\|_\infty$	Attack time
24	$9\,953 + 389$	$5\,649$	0h0m23s
117	$48\,456 + 1\,915$	$1\,031$	0h3m52s
583	$241\,541 + 9\,378$	247	1h55m47s

Table 4. Attack times and size of the (LP) system on $\mathbf{t}_0$.

The method provides us with a point close to $\mathbf{t}_0^{[0]}$ (because $\mathbf{t}_0^{[0]}$ is in the polytope constructed by the (LP) system by definition). Unfortunately, it is not possible to increase the number of inequations endlessly, as the calculation time depends polynomially on the number of inequations.

Remark 9 *If we denote P the polytope obtained on $\mathbf{t}_0^{[0]}$ with a large number of inequations, most of the inequations we collect are not "useful" in the sense that P remains unchanged whether the inequation is taken into account or not. This is illustrated in Figure 4: with* 100 *inequations collected, only* 5 *of them are actually useful in describing the polytope of solutions. By collecting lots of inequations, we get an increasingly complex algebraic description (a growing set of inequations) of a simple geometric object: a polytope with a few faces that approximates $\mathbf{t}_0^{[0]}$. We need a way of selecting "useful" and "useless" inequations to reduce the complexity of solving the (LP) problem associated with $\mathbf{t}_0^{[0]}$.*

4.1 Useful inequalities

In this subsection we assume that we know $\tilde{\mathbf{t}}_0^{[0]} \in \mathcal{R}_q$ and C such that $||\tilde{\mathbf{t}}_0^{[0]} - \mathbf{t}_0^{[0]}||_\infty \leq C$. In other words, $\mathbf{t}_0^{[0]} \in B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C)$. Given an inequation on $\mathbf{t}_0$, we want to determine efficiently if the intersection between $B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C)$ and the set of solutions of the inequation is non-trivial.

Definition 11 *Let $\tilde{\mathbf{t}}_0^{[0]} \in \mathcal{R}_q$ and $C \in \mathbb{R}_+$. We say that an inequation on $\mathbf{t}_0^{[0]}$ of the form $\{\mathbf{a}^\intercal\mathbf{x} - b \geq 0\}$ (resp. $\{\mathbf{a}^\intercal\mathbf{x} - b \leq 0\}$) is useful according to $\tilde{\mathbf{t}}_0^{[0]}$ and C if and only if:*

$$B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C) \not\subset \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^\intercal\mathbf{x} - b \geq 0\ \}\ \textit{(resp. } \mathbf{a}^\intercal\mathbf{x} - b \leq 0\textit{)}$$

Remark 10 *This definition is very natural and is illustrated with Figure 6.*

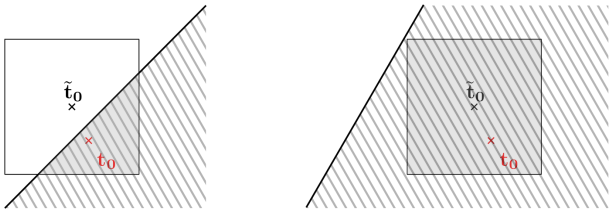

Fig. 6. On the left, a useful inequation. On the right a useless inequation.

Proposition 3 *An inequation on* $\mathbf{t}_0^{[0]}$ *of the form* $\{\mathbf{a}^\intercal\mathbf{x} - b \geq 0\}$ *is useful according to* $\tilde{\mathbf{t}}_0^{[0]}$ *and* C *if and only if:*

$$\mathbf{a}^\intercal\tilde{\mathbf{t}}_0^{[0]} - C||\mathbf{a}^\intercal||_\infty^* < b.$$

An inequation on $\mathbf{t}_0^{[0]}$ *of the form* $\{\mathbf{a}^\intercal\mathbf{x} - b \leq 0\}$ *is useful according to* $\tilde{\mathbf{t}}_0^{[0]}$ *and* C *if and only if:*

$$\mathbf{a}^\intercal\tilde{\mathbf{t}}_0^{[0]} + C||\mathbf{a}^\intercal||_\infty^* > b,$$

where $||.||_\infty^*$ *denote the operator norm.*

Proof. Let $f : \mathbf{x} \mapsto \mathbf{a}^\intercal\mathbf{x} - b$, then:

$$B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C) \subset \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \geq 0\} \iff \inf_{||\mathbf{u}||_\infty \leq 1}(f(\tilde{\mathbf{t}}_0^{[0]} + C\mathbf{u})) \geq 0.$$

In addition,

$$\begin{aligned}\inf_{||\mathbf{u}||_\infty \leq 1}(\mathbf{a}^\intercal(\tilde{\mathbf{t}}_0^{[0]} + C\mathbf{u}) - b) &= \mathbf{a}^\intercal\tilde{\mathbf{t}}_0^{[0]} - C\sup_{||\mathbf{u}||_\infty \leq 1}(\mathbf{a}^\intercal(-\mathbf{u})) - b \\ &= \mathbf{a}^\intercal\tilde{\mathbf{t}}_0^{[0]} - C||\mathbf{a}^\intercal||_\infty^* - b.\end{aligned}$$

This concludes the first relation, and the same reasoning can be used to deduce the second result.

□

Remark 11 *Proposition 3 allows us to calculate efficiently whether an inequation on* $\mathbf{t}_0$ *is useful or not according to* $\tilde{\mathbf{t}}_0$ *and* C*. Finally, it is important to note that the definitions and propositions stated here remain when the infinite norm is replaced by another norm, even if these formulations are not useful for us.*

Notation 4 *We will note* `generate_useful_ineq`$(\delta, \tilde{\mathbf{t}}_0^{[0]}, C)$ *the procedure for generating* δ *useful inequalities according to* $\tilde{\mathbf{t}}_0^{[0]}$ *and* C*.*

4.2 Formal description of the attack

With the different tools we need now defined, the main idea behind the attack strategy can be summed up in one sentence: 'Collect, guess, filter, repeat.' More precisely we will:

- Collect inequalities to obtain a polytope P_0. By definition, $\mathbf{t}_0^{[0]} \in P_0$.
- Calculate (or estimate heuristically) the diameter of P_0 to obtain C and $\tilde{\mathbf{t}}_0^{[0]}$ such that $\mathbf{t}_0^{[0]} \in B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C)$.
- Collect useful inequalities according to $\tilde{\mathbf{t}}_0^{[0]}$ and C to obtain P_1 and by construction $\mathbf{t}_0^{[0]} \in P_1$.
- Repeat until the polytope P verifies $diam(P) \leq 1/2$, in which case $\mathbf{t}_0^{[0]}$ as been recovered.

Algorithm 5 `Recovering` $\mathbf{t}_0^{[0]}$

Ensure: $\mathbf{t}_0^{[0]}$
Require: An inequation step δ

$\tilde{\mathbf{t}}_0^{[0]} = 0$
$C, diam = 2^{12}$
$P = \{-2^{12} + 1 \leq x_i \leq 2^{12}\}_{i=0,\ldots,255}$
while $C \geq 1$ **do**
    $\Delta = \delta$
    while $diam > C/2$ **do**
        $P =$ `generate_useful_ineq`$(\Delta, \tilde{\mathbf{t}}_0^{[0]}, C)$
        $diam =$ `calculate_diam`(P)
        $\Delta = 2 \times \Delta$
    $C = C/2$
    $\tilde{\mathbf{t}}_0^{[0]} =$ `round(lp_guess(P))`
return $\tilde{\mathbf{t}}_0^{[0]}$

Proposition 4 *For any $\delta \in \mathbb{N}^*$, Algorithm 5 terminates in a finite number of steps, giving $\mathbf{t}_0^{[0]}$.*

Proof. According to Section 3 , the error between the sum of some of the coefficients of $\mathbf{t}_0^{[0]}$ and the bound of the inequation obtained follows a subgaussian distribution. In particular, there are inequalities which are equalities verified by certain coefficients of $\mathbf{t}_0^{[0]}$. Thus the procedure `generate_useful_ineq` finishes in a finite time for any value of δ and $C > 0$. This ensures that Algorithm 5 finishes after a finite number of steps. In addition, at step i of Algorithm 5:

$\mathbf{t}_0^{[0]} \in B_\infty(\tilde{\mathbf{t}}_0^{[0]}, 2^{12-i})$ so that at the last step $\mathbf{t}_0^{[0]} \in B_\infty(\tilde{\mathbf{t}}_0^{[0]}, 1/2)$ and therefore $\mathbf{t}_0^{[0]} = \tilde{\mathbf{t}}_0^{[0]}$.

□

Remark 12 *Algorithm 5 is useful because it can be proved that it systematically finds* $\mathbf{t}_0^{[0]}$*. Unfortunately, in practice it is too complex to be used, as each call to the function* `calculate_diam` *requires the solution of* 2×256 (LP) *problems, each potentially containing several hundred thousand inequalities. Rather than calculating the size of the polytope containing* $\mathbf{t}_0^{[0]}$ *at each step, we estimate the number of inequations needed to make the size of the corresponding polytope small enough, without having to calculate it explicitly. This is formally described in Algorithm 6.*

Algorithm 6 `Recovering` $\mathbf{t}_0^{[0]}$ `heuristically`

Ensure: A candidate for $\mathbf{t}_0^{[0]}$
Require: An inequation step sequence $(\delta_i)_{i\in\{1,\dots,m\}}$, a radius sequence $C_m < C_{m-1} < \cdots < C_1 = 2^{12}$.

$\tilde{\mathbf{t}}_0^{[0]} = 0$
$i = 1$
$P = \{-2^{12} + 1 \leq x_i \leq 2^{12}\}_{i=1,\dots,256}$
while $i \leq m$ **do**
    $P =$ `generate_useful_ineq`$(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$
    $i = i + 1$
    $\tilde{\mathbf{t}}_0^{[0]} =$ `round(lp_guess(P))`
return $\tilde{\mathbf{t}}_0^{[0]}$

Remark 13 *If the sequence is not chosen carefully it may be that* $\mathbf{t}_0 \notin B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C)$ *at some step of Algorithm 6. As Figure 13 shows, if at any stage of the algorithm we have an* $\tilde{\mathbf{t}}_0^{[0]}$ *and* C *such that* $\mathbf{t}_0^{[0]} \notin B_\infty(\tilde{\mathbf{t}}_0^{[0]}, C)$*, collecting useful inequations according to* $\tilde{\mathbf{t}}_0^{[0]}$ *and* C *can lead to a point which deviates from* $\mathbf{t}_0^{[0]}$*, or even worse: an* (LP) *system without solution.*

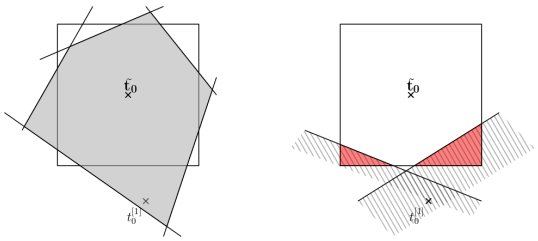

Fig. 7. (LP) system for a poorly choosen (δ_i).

5 Experimental results

We have used the reference implementation of Dilithium [DKL+22] to generate signatures and to solve the (LP) problems linked to $\mathbf{t}_0$ we have decided to use lp_solve [MB04], a free linear programming solver in C. Our attack method applies independently of lp_solve and any other solver could have been chosen. All the tests and results presented in this section were carried out on a laptop computer equipped with an Intel(R) Core(TM) i7-10850H 2.70GHz CPU. The code used for the attack will be made public in the final version of the article.

5.1 Attack on Dilithium-2

In this section we present the results obtained by applying Algorithm 6 with different choices of filtration (C_i) and inequation (δ_i) sequence. This gives rise to time/signature/memory trade-offs. To reduce the number of signatures required we decided to store the signatures generated at each stage of the algorithm since the same inequation can be useful for different filtration, again an attacker could do without it at the cost of having to collect more signatures.

Remark 14 *The choices of the different sequences (C_i) and (δ_i) are heuristic and are given as an example. Nothing prevents trying out other sequences and testing the results obtained.*

Attack methodology 1: The most conservative method is to follow as directly as possible Algorithm 5.We therefore choose to divide the radius by two at each stage *i.e.* $(C_i) = (4096, 2048, \ldots, 2, 1)$. We also choose a constant inequation sequence, which we have set at 50 000. In other words, at each step we collect 50 000 inequalities on each of the polynome of $\mathbf{t}_0$ before solving the k associated (LP) problems. Table 5 shows the results obtained for the first 10 KAT keys.

Signatures	Inequalities selected	Recovery probability	Average time	Median time
474 026	418 219 + 239 719	1	1h13m10s	1h12m55s

Table 5. Average results of the attack on $\mathbf{t}_0$

For greater clarity, we detail the results of Algorithm 6 for the first key of the KAT in Table 6. For this table only, the knowledge of $\mathbf{t}_0$ was used to illustrate the correctness of Algorithm 6. In Table 6, Time includes time to generate signatures as well as time to solve the $k = 4$ (LP) problems at each step of Algorithm6 and in "Inequalities selected" we detail the number of inequalities from zero and non-zero hints, for the polynomial $\mathbf{t}_0^{[0]}$. With our choice of parameters, this attack time is largely dominated by signature collection time.

Round	C_i	Signatures	Inequalities selected	$\|\|\mathbf{t}_0 - \tilde{\mathbf{t}}_0\|\|_\infty$	Time
1	4096	117	48 456 + 1 915	837	2m17s
2	2048	234	46 612 + 3 731	431	1m56s
3	1024	468	43 112 + 7 433	234	1m57s
4	512	937	37 172 + 13 540	119	1m58s
5	256	1 879	32 057 + 18 844	53	2m11s
6	128	3 743	28 787 + 21 863	27	2m37s
7	64	7 485	27 125 + 23 434	13	3m23s
8	32	14 989	26 250 + 23 434	7	3m32s
9	16	30 023	26 055 + 24 700	3	4m15s
10	8	60 200	25 735 + 25 098	1	5m7s
11	4	119 843	25 660 + 25 229	1	6m44s
12	2	238 674	25 437 + 25 009	0	13m34s
13	1	475 856	25 634 + 25 247	0	12m23s
Total	-	475 856	457 373 + 253 581	-	0h51m17s

Table 6. Detailed results of Algorithm 6 on the first KAT key.

Attack methodology 2: Another method of attack consists of choosing a more "relaxed" sequence , at the cost of queerying more signature at each step of Algorithm 6. This method is useful in a context where the attacker have a large computing capacity and need to find $\mathbf{t}_0$ with as few signatures as possible. For our tests, to maintain a manageable attack time we have chosen the sequence $(C_i) = (2^{12} = 4096, 2048, \ldots, 16, 8)$ with the associated sequence inequations: $(\delta_i) = (50\,000, \ldots, 50\,000, 150\,000)$. Table 7 shows the results obtained for the first 10 KAT keys. Again, we detail the results of the attack calculation for the first key of the KAT in Table 8, to compare it with the previous attack method.

Signatures	inequalities selected	Recovery probability	Average time	Median time
179 354	39 2696 + 21 3943	1	1h26m53s	1h24m8s

Table 7. Average results of the attack on $\mathbf{t}_0$

Round	C_i	Signatures	Inequalities selected	$\|\|\mathbf{t}_0 - \tilde{\mathbf{t}}_0\|\|_\infty$	Time
1	4096	117	48 456 + 1 915	1031	4m16s
2	2048	234	46 612 + 3 731	495	4m2s
3	1024	468	43 112 + 7 433	262	3m48s
4	512	937	37 172 + 13 540	135	3m44s
5	256	1 879	32 057 + 18 844	62	3m53s
6	128	3 743	28 787 + 21 863	37	3m53s
7	64	7 485	27 125 + 23 434	19	4m7s
8	32	14 989	26 250 + 23 434	10	4m48s
9	16	30 023	26 055 + 24 700	4	5m27s
10	8	179 515	76 487 + 74 192	0	47m5s
Total	-	179 515	392 113 + 213 853	-	1h25m3s

Table 8. Detailed results of the attack on the first KAT key.

For both methods, if we had sought to find $\mathbf{t}_0$ without filtration, each of the 4 (LP) problems associated would have contained at least 170 000 signatures. According to Section 3 we obtain on average 496 inequalities per polynomial, resulting in four huge (LP) systems of at least $170\,000 \times 496 = 84\,362\,500$ inequalities each, which is much more costly to solve than our sequence of small systems. Indeed, thanks to our natural definitions of "usefull" inequations, we were able to find an equivalent representation of each of the polytopes containing a polynomial of $\mathbf{t}_0$ with only 50 000 to 150 000 inequations.

5.2 Attack on Dilithium-3 and Dilthium-5

The theory remains true regardless of the security level of Dilithium, therefore there should not be differences by changing the security level. To provide complete results we tested the attack method presented in subsection 5.1 with the other two security levels of Dilithium. For the two security levels we tested the same sequence $(C_i) = (2^{12} = 4096, 2048, \ldots, 16, 2)$ associated with the following sequence of inequations $(\delta_i) = (60\,000, \ldots, 60\,000)$. Table 9 and Table 10 show the results obtained.

Signatures	inequalities selected	Recovery probability	Average time	Median time
523 534	398 704 + 210 140	1	1h 30min 05sec	1h 30min 33sec

Table 9. Average results of the attack on $\mathbf{t}_0$ for Dilithium-3.

Signatures	inequalities selected	Recovery probability	Average time	Median time
513 106	481 181 + 249 566	1	3h 21min 23sec	3h 22min 52sec

Table 10. Average results of the attack on $\mathbf{t}_0$ for Dilithium-5.

Remark 15 *The attack is slightly less effective on Dilithum security levels* 3 *and* 5*, which is not surprising:* $\mathbf{t}_0$ *contains* 6 *and* 8 *polynomials respectively, instead of* 4 *for level* 2*, so there are just as many* (LP) *problems to solve at each step. It should also be noted that the other security levels are slower to sign messages.*

6 Conclusion

Even if the state of art of attacks against Dilithium continues to evolve, the corresponding ML-DSA standard has recently been published. Dilithium has thus become a leading standard, and as such, is implemented in a quickly growing number of products, some of which are vulnerable to side-channel attacks. In order to design and implement effective protections against such attacks, it is necessary to have a complete knowledge of the data that are available to potential attackers. In particular, as discussed in the introduction, the question of the exact vulnerability level of one of the secret key vectors $\mathbf{t}_0$ was up to now one of the grey areas that was greatly in need of clarification.

Our paper shows that $\mathbf{t}_0$ can be reconstructed from sufficiently many Dilithium signatures (between 200 000 and 500 000, depending on the security level) and in just a few hours of computation. In other words, in theory as in practice, it is reasonable to consider $\mathbf{t}_0$ as part of the public key. The results are not surprising but, in our opinion, it is striking that the attack is so effective. For instance, for Dilithium 2, $\mathbf{t}_0$ is a vector of 4 polynomials of degree 256, whose coefficients belong to the interval $\{-2^{12}+1, \ldots, 2^{12}\}$, so that the set of potential solutions is enormous. Despite this, in all our experiments, we were able to find $\mathbf{t}_0$ in a few hours, with less than 200 000 signatures.

Finally, even if the various filtration techniques we used are efficient enough to recover $\mathbf{t}_0$ with a reastic complexity, there is still room for optimization in terms of both the number of signatures and the computation time.

References

BDE+18. Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 494–524. Springer, Heidelberg, December 2018.

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`.

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.

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

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

EAB+23b. 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`.

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

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

Lyu22. Vadim Lyubashevsky. Nist conference. NIST website, 2022. `https://www.nist.gov/video/fourth-pqc-standardization-conference-virtual-day-1-part-1`.

MB04. Peter Notebaert Michel Berkelaar, Kjell Eikland. lp solve. `https://lpsolve.sourceforge.net/5.5`, 2004. Open source (Mixed-Integer) Linear Programming system.

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`.

NW88. George L. Nemhauser and Laurence A. Wolsey. Integer and combinatorial optimization. In *Wiley interscience series in discrete mathematics and optimization*, 1988.

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, Paper 2018/821, 2018. `https://eprint.iacr.org/2018/821`.

RRB+18. Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Number "not used" once - practical fault attack on pqm4 implementations of nist candidates. Cryptology ePrint Archive, Paper 2018/211, 2018. `https://eprint.iacr.org/2018/211`.

RRB+19. Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Number "not used" once - practical fault attack on pqm4 implementations of NIST candidates. In Ilia Polian and Marc Stöttinger, editors, *COSADE 2019*, volume 11421 of *LNCS*, pages 232–250. Springer, Heidelberg, April 2019.

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.