Some applications of linear programming to Dilithium

Paco Azevedo-Oliveira [1,2]

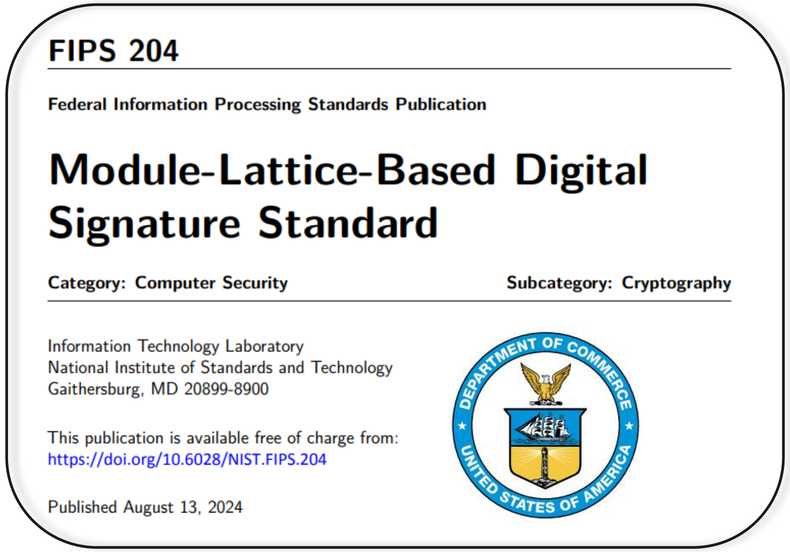
[1] Thales, France
[2] UVSQ, France

Summary

Context

Dilithium is a signature algorithm recently standardized by NIST under the name ML-DSA.

Dilithium is recommended for calculating signatures in most use cases.



FIPS 204

Federal Information Processing Standards Publication

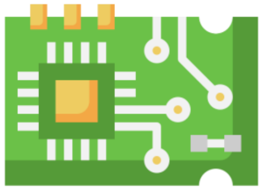
Module-Lattice-Based Digital Signature Standard

Category: Computer Security　　Subcategory: Cryptography

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

This publication is available free of charge from:
https://doi.org/10.6028/NIST.FIPS.204

Published August 13, 2024

$m \rightarrow$  $\rightarrow \sigma_m$

It is necessary to study the security of embedded implementations. The security of Dilithium against side-channel attacks (SCA) and fault attacks (FA) must be carefully evaluated.

Simplified Dilithium

Dilithium uses two rings:

$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

With: $n = 256$ and $q = 8380417$

Algorithm 1 KeyGen

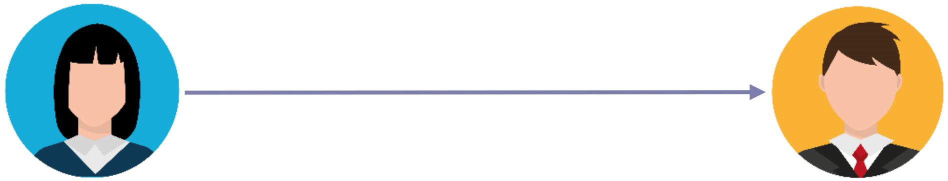
Ensure: (pk, sk)

$\mathbf{A} \leftarrow \mathcal{R}_q^{k \times l}$

$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k$

$\mathbf{t} := \mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2$

return $pk = (\mathbf{A}, \mathbf{t}),\ sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2)$



(A, t, s_1, s_2) $\qquad$ (A, t)

Simplified Dilithium

Dilithium uses two rings:

$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

With: $n = 256$ and $q = 8380417$

Algorithm 1 KeyGen

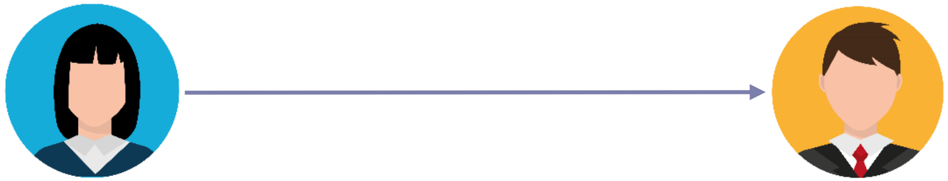
Ensure: (pk, sk)

$\mathbf{A} \leftarrow \mathcal{R}_q^{k \times l}$

$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k$

$\mathbf{t} := \mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2$

return $pk = (\mathbf{A}, \mathbf{t}),\ sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2)$



(A, t, s_1, s_2) (A, t)

α an even integer that divides $q - 1$ and:

$$r = r_1\alpha + r_0 \ with\ r_0 = r \bmod^{\pm}(\alpha)\ and\ r_1 = \frac{r - r_0}{\alpha}$$

Possible values of r_0 are in $\left\{-\frac{\alpha}{2} + 1, \dots, 0, \dots, \frac{\alpha}{2}\right\}$

Possible values of $r_1\alpha$ are in $\{0, \alpha, 2\alpha, \dots, q - 1\}$

One note:

$HighBits_q(r, \alpha) = r_1$ and $LowBits_q(r, \alpha) = r_0$

Simplified Dilithium

Dilithium uses two rings:

$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

With: $n = 256$ and $q = 8380417$

Algorithm 1 KeyGen

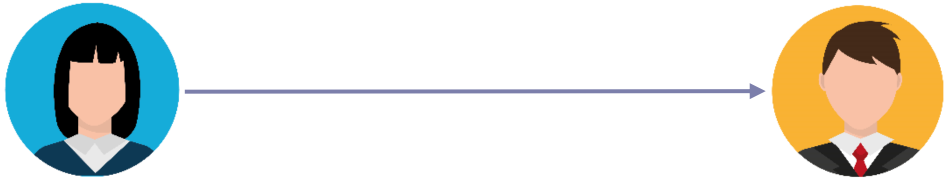
Ensure: (pk, sk)

$\mathbf{A} \leftarrow \mathcal{R}_q^{k \times l}$

$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k$

$\mathbf{t} := \mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2$

return $pk = (\mathbf{A}, \mathbf{t}),\ sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2)$



(A, t, s_1, s_2) (A, t)

$$r = HighBits_q(r, \alpha) \times \alpha + LowBits_q(r, \alpha)$$

$$P \in \mathcal{R}_q^l,\ P = (P_1, P_2, \ldots, P_l)$$

$$P_1 := \sum p_i x^i \in \mathcal{R}_q,$$

$$\mathtt{HighBits}_q(P_1) := \sum \mathtt{HighBits}_q(p_i) x^i$$

Simplified Dilithium

B_τ is the set of elements of R that have τ coefficients equal to -1 or 1 and the rest equal to 0.

Algorithm 1 Sig

Require: sk, M
Ensure: $\sigma = (c, \mathbf{z})$

$\mathbf{z} = \perp$
while $\mathbf{z} = \perp$ **do**
  $\mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$
  $\mathbf{w}_1 := \texttt{HighBits}(\mathbf{Ay}, 2\gamma_2)$
  $c \in B_\tau := H(M \| \mathbf{w}_1)$
  $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\texttt{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta$ **then**
    $\mathbf{z} := \perp$
  end if
end while
return $\sigma = (c, \mathbf{z})$

By definition of z:

$$Az - ct = Ay - cs_2$$

Rejection

y is chosen such that:

$$HighBits_q(Ay, 2\gamma_2) = HighBits_q(Ay - cs_2, 2\gamma_2)$$

$(M, \sigma = (c, \mathbf{z}))$

(A, t, s_1, s_2) (A, t)

Simplified Dilithium

B_τ is the set of elements of R that have τ coefficients equal to -1 or 1 and the rest equal to 0.

Algorithm 1 Sig

Require: sk, M
Ensure: $\sigma = (c, \mathbf{z})$

```
z = ⊥
while z = ⊥ do
    y ← S̃^l_{γ1}
    w1 := HighBits(Ay, 2γ2)
    c ∈ B_τ := H(M||w1)
    z := y + c s1
    if ||z||∞ ≥ γ1 − β or LowBits(Ay − cs2, 2γ2)||∞ ≥ γ2 − β then
        z := ⊥
    end if
end while
return σ = (c, z)
```

By definition of z:

$$Az - ct = Ay - cs_2$$

Rejection

y is chosen such that:

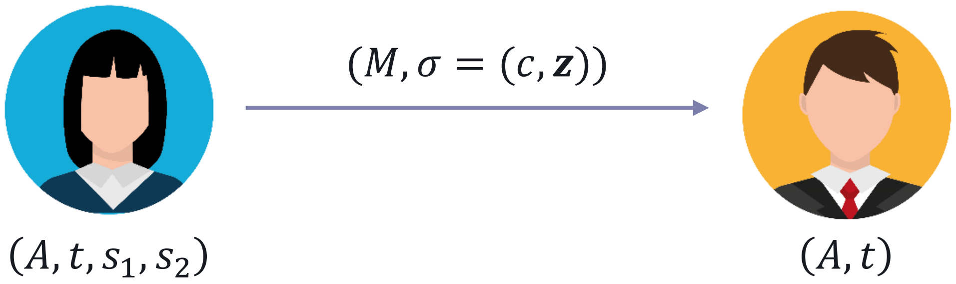
$$HighBits_q(Ay, 2\gamma_2) = HighBits_q(Ay - cs_2, 2\gamma_2)$$

Algorithm 1 Ver

```
w'1 := HighBits(Az − ct, 2γ2)
Accept if ||z||∞ ≤ γ1 − β and c = H(M||w'1)
```

Bob can recompute w_1:

$$\begin{aligned} \mathbf{w}_1 &= HighBits_q(Ay, 2\gamma_2) \\ &= HighBits_q(Ay - cs_2, 2\gamma_2) \\ &= HighBits_q(Az - ct, 2\gamma_2) \\ &= w'_1 \end{aligned}$$



Simplified Dilithium

How does rejection sampling impact Dilithium's security?

Algorithm 1 Sig

Require: sk, M

Ensure: $\sigma = (c, \mathbf{z})$

$\mathbf{z} = \perp$

while $\mathbf{z} = \perp$ **do**

  $\mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$

  $\mathbf{w}_1 := \mathtt{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$

  $c \in B_\tau := H(M \| \mathbf{w}_1)$

  $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$

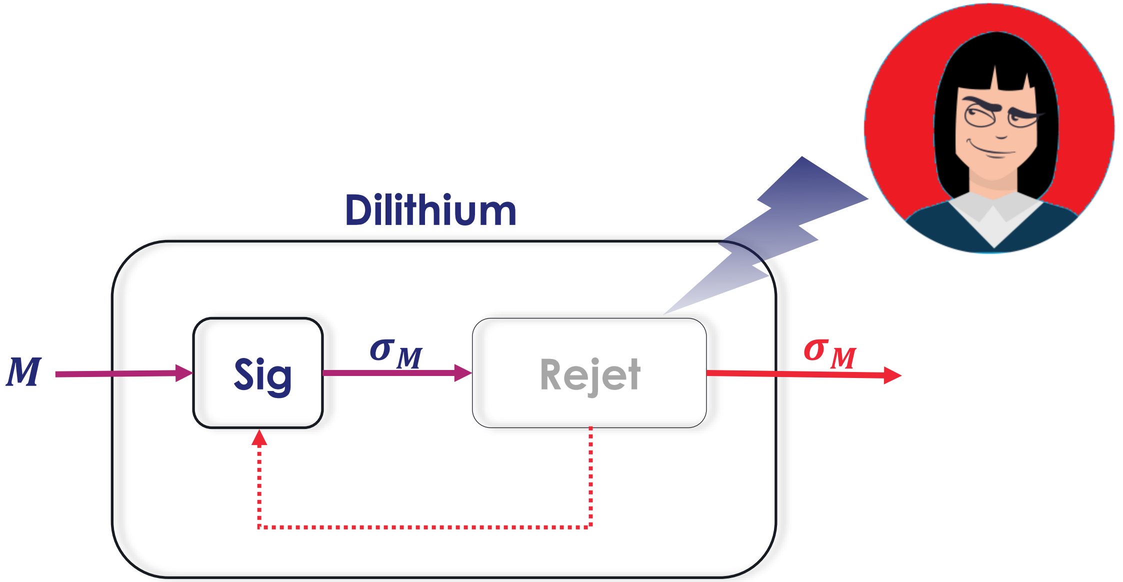
  if $\boxed{\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta}$ or $\boxed{\|\mathtt{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta}$ **then**

    $\mathbf{z} := \perp$

  end if

end while

return $\sigma = (c, \mathbf{z})$



Simplified Dilithium

How does rejection sampling impact Dilithium's security?

Algorithm 1 Sig

Require: sk, M

Ensure: $\sigma = (c, \mathbf{z})$

$\mathbf{z} = \perp$

while $\mathbf{z} = \perp$ **do**

  $\mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$

  $\mathbf{w}_1 := \mathtt{HighBits}(\mathbf{Ay}, 2\gamma_2)$

  $c \in B_\tau := H(M || \mathbf{w}_1)$

  $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$

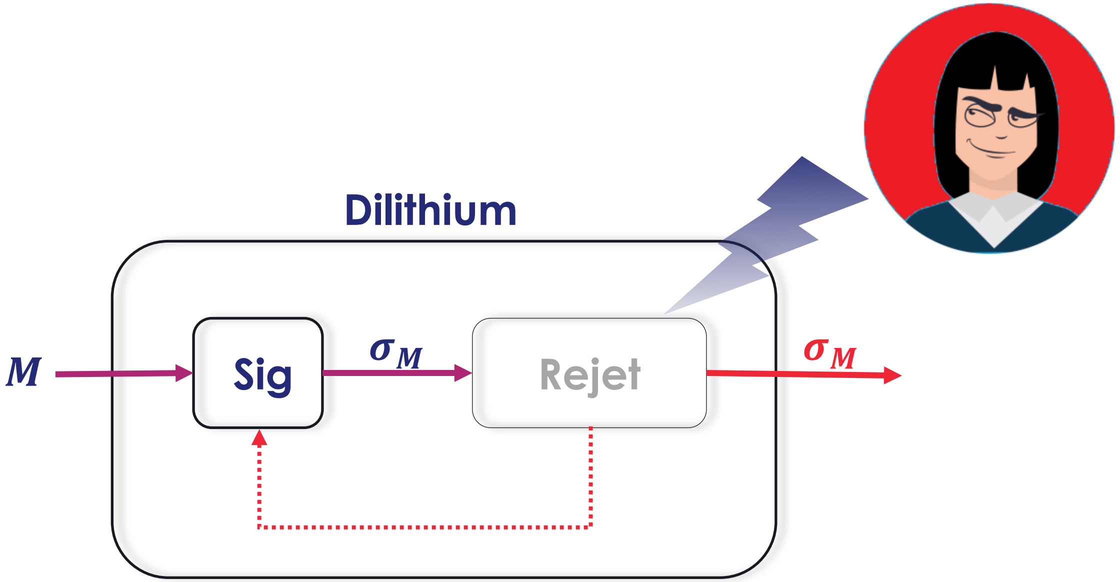
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\boxed{\|\mathtt{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta}$ **then**

    $\mathbf{z} := \perp$

  end if

end while

return $\sigma = (c, \mathbf{z})$



Finding a polytope: A practical fault attack against Dilithium

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

Natural question: What is the impact of rejection?

$\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
if $\boxed{\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta}$ or $\boxed{\|\texttt{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta}$ **then**
  $\mathbf{z} := \perp$
end if
end while
return $\sigma = (c, \mathbf{z})$

Algorithm 1 Ver

$\mathbf{w}_1' := \texttt{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$
Accept if $||\mathbf{z}||_\infty \leq \gamma_1 - \beta$ **and** $c = H(M||\mathbf{w}_1')$

The signature (c, z) is not valid and rejected by Ver. One has:

$$\left|\left|LowBits_q(Ay - cs_2)\right|\right|_\infty \geq \gamma_2 - \beta$$

$$w_1 = HighBits_q(Ay, 2\gamma_2) \neq w_1' = HighBits_q(Ay - cs_2, 2\gamma_2).$$

Natural question: What is the impact of rejection?

$\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
if $\boxed{\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta}$ or $\boxed{\|\mathtt{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta}$ **then**
  $\mathbf{z} := \perp$
end if
end while
return $\sigma = (c, \mathbf{z})$

Algorithm 1 Ver

$\mathbf{w}_1' := \mathtt{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$
Accept if $\|\mathbf{z}\|_\infty \leq \gamma_1 - \beta$ **and** $c = H(M \| \mathbf{w}_1')$

The signature (c, z) is not valid <u>and</u> rejected by Ver. One has:

$$\left\| LowBits_q(Ay - cs_2) \right\|_\infty \geq \gamma_2 - \beta$$

$$\mathbf{w}_1 = HighBits_q(Ay, 2\gamma_2) \neq w_1' = HighBits_q(Ay - cs_2, 2\gamma_2).$$

Two possible scenarios:

Scenario n°1 :

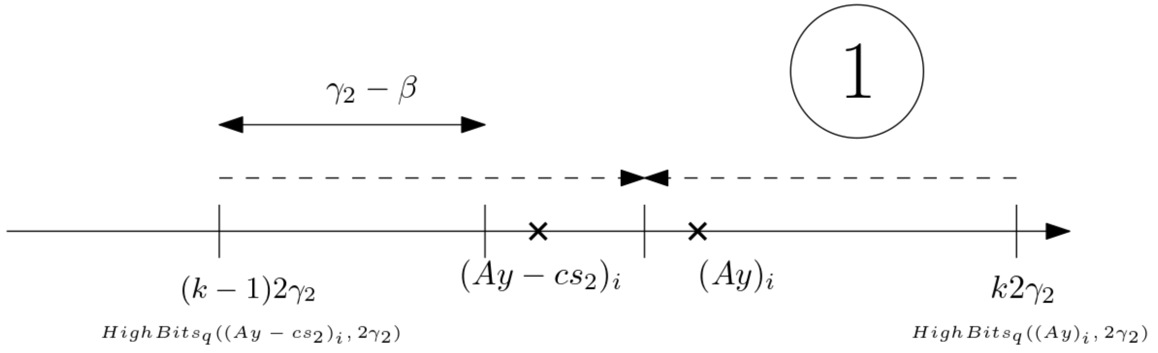
$$HighBits_q\big((Ay)_i, 2\gamma_2\big) = HighBits_q\big((Ay - cs_2)_i, 2\gamma_2\big) + 1$$

Scenario n°2 :

$$HighBits_q\big((Ay)_i, 2\gamma_2\big) = HighBits_q\big((Ay - cs_2)_i, 2\gamma_2\big) - 1$$

Natural question: What is the impact of rejection?

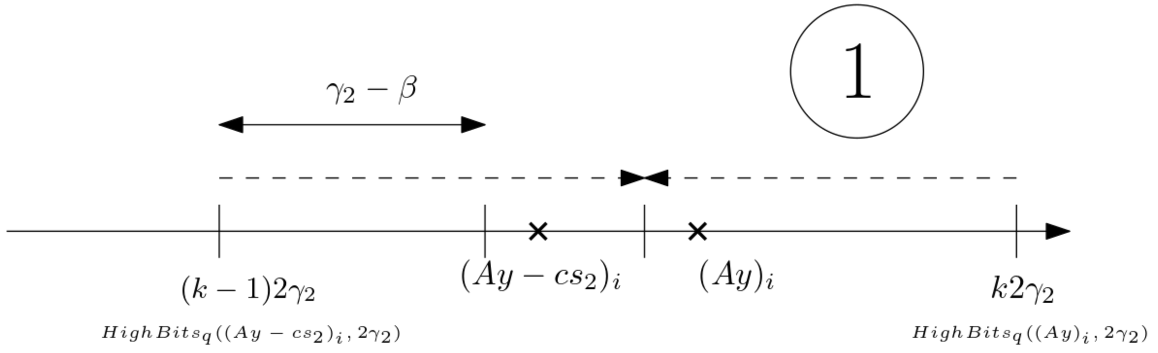
Scenario n°1 : $\boldsymbol{HighBits_q\big((Ay)_i, 2\gamma_2\big) = HighBits_q\big((Ay - cs_2)_i, 2\gamma_2\big) + 1.}$



The coefficient of $(cs_2)_i$ is positive, and we even have: $\boldsymbol{(cs_2)_i \geq \gamma_2 - LowBits_q((Az - ct)_i, 2\gamma_2)}$.

Natural question: What is the impact of rejection?

Scenario n°1 : $HighBits_q((Ay)_i, 2\gamma_2) = HighBits_q((Ay - cs_2)_i, 2\gamma_2) + 1.$



The coefficient of $(cs_2)_i$ is positive, and we even have: $(cs_2)_i \geq \gamma_2 - LowBits_q((Az - ct)_i, 2\gamma_2)$.

$c \in R$ is a public polynomial, with coefficients in {-1,0,1}, so one obtain inequations:

$$-1 \times (s_2)_{10} + (s_2)_{68} - \cdots + (s_2)_{56} \geq b_1$$

$$-1 \times (s_2)_{25} + (s_2)_{34} - \cdots + (s_2)_{243} \geq b_2$$

$$\cdots$$

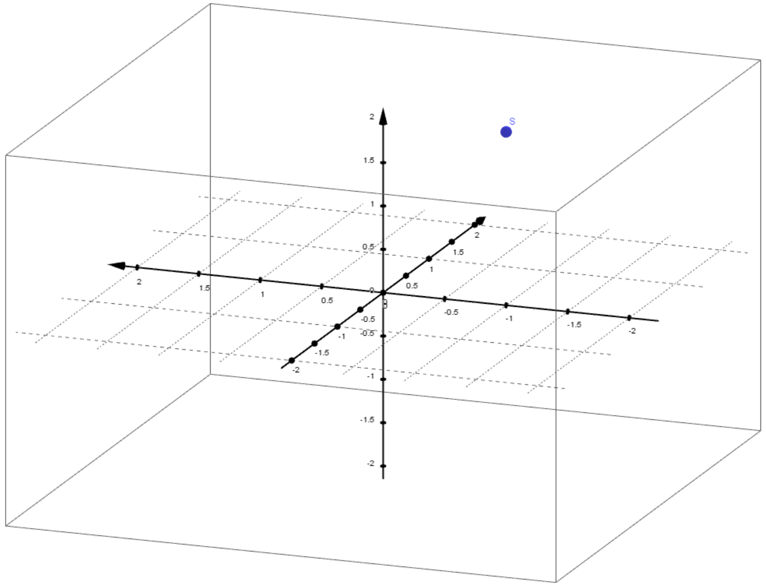
$$-1 \times (s_2)_{118} + (s_2)_{87} - \cdots + (s_2)_{174} \geq b_k$$

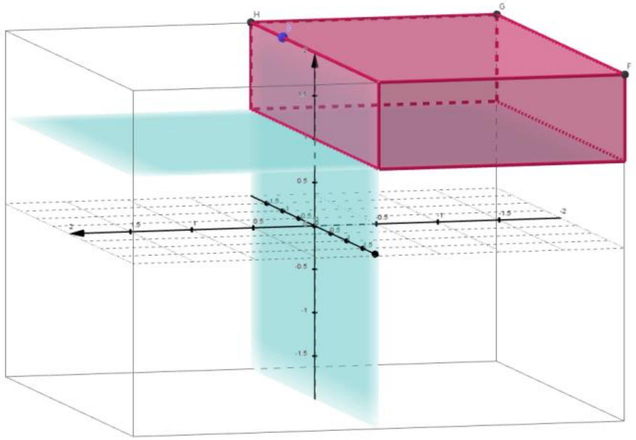
Natural question: What is the impact of rejection?

Example for n=3 :

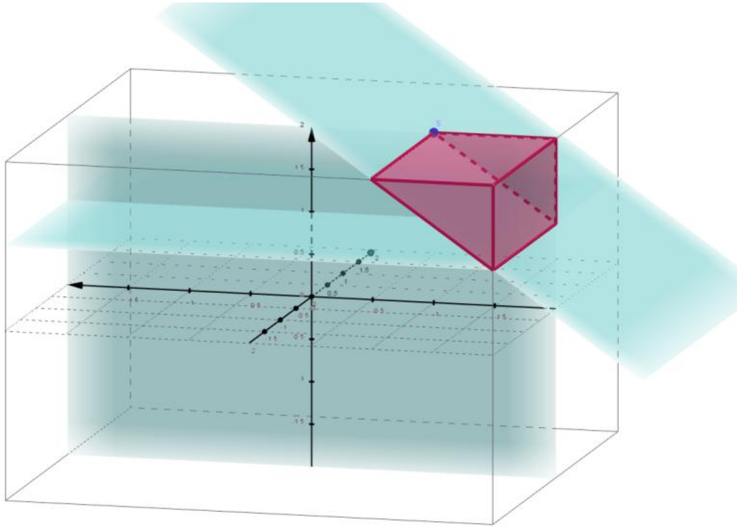
Suppose all the coefficients of the key are known, except for $(s_2)_0, (s_2)_1, (s_2)_2$.
The unknowns are the coordinates of a point in $[-2, 2]^3 \cap Z$

For $s_2 = (2,0,1,\dots)$: signing multiple times with the same key will produce inequalities.





Solution of inequations:
$x \leq 0$
$y \geq 1$



Solution of inequations :
$x \leq 0$
$y \geq 1$
$y - z \geq -3$

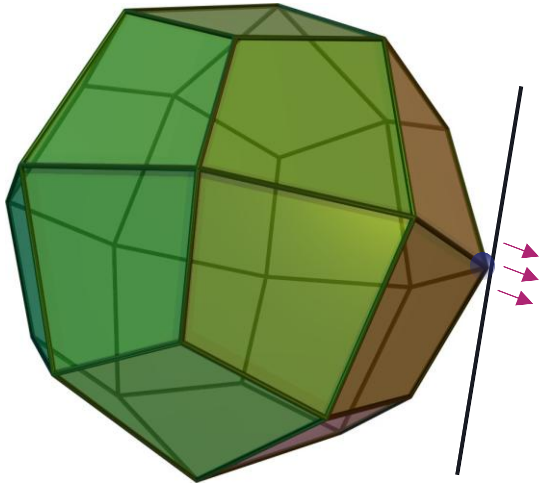
Natural question: What is the impact of rejection?

In mathematics, it is easy to solve linear systems $Ax = b$

What about inequality systems? $Ax \leq b$

We use linear programming methods. These are polynomial algorithms that provide a solution to the following problem:

$$\begin{array}{ll} \text{Find a vector} & \mathbf{x} \\ \text{that maximizes} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0}. \end{array}$$



Objective: Collect enough inequalities so that s_2 is the only associated solution, then solve the corresponding (LP) system.

Natural question: What is the impact of rejection?

If the solution is unique, it suffices to minimize the zero function.

Finding the min/max of functions $x \to x_i$, one can find the dimension of the polytope.

Exemple : All coordinates of s_2 are assumed to be known, except the first 40.

Even though $P \subset [-2, 2]^{40}$ the polytope is actually included in a square!

```
Starting attack
------

Pour s0:

Minimum: -2
Maximum: -2
------

Pour s1:

Minimum: -1
Maximum: -1
------

Pour s2:

Minimum: 2
Maximum: 2
------

Pour s3:

Minimum: 0
Maximum: 0
------

Pour s4:

Minimum: 1
Maximum: 1
------

Pour s25:

Minimum: 1
Maximum: 1
------

Pour s26:

Minimum: -2
Maximum: -2
------

Pour s27:

Minimum: 1
Maximum: 2
------

Pour s28:

Minimum: -1
Maximum: -1
------

Pour s29:

Minimum: 2
Maximum: 2
------

Pour s35:

Minimum: -1
Maximum: -1
------

Pour s36:

Minimum: -2
Maximum: -2
------

Pour s37:

Minimum: -1
Maximum: -1
------

Pour s38:

Minimum: 2
Maximum: 2
------

Pour s39:

Minimum: 0
Maximum: 1

Attack time: 0.988412618637085
```

Natural question: What is the impact of rejection?

We estimate the number of inequalities needed using statistics:

Unknown coefficients	32	64	128	256
Nb tests	100	100	100	-
Inequalities	323	1306	3917	10 445 (predicted)
Polytopes dimensions	0	0	0	-
Attack time	1.36 s	17.4s	227.3s	-

Table 1. Evolution of the dimension as a function of the unknowns

We collect enough signatures so that the polytope defined by the inequalities contains only s_2.

Signatures	Average inequalities	Success probability	Average time	Median Time
1 250 000	11 085	0.99	277.53s	180.00s

Table 3. Average results of the attack on `F-Sig`

Conclusion: The attack illustrates the power of LP methods; we are looking for a point in the interval $[-2,2]^{256}$ and we find it in a few minutes. This also that the attack is realistic:

It is necessary to protect tests against faults.

Hidden problems..

The Dilithium version is not complete. The public key is compressed:

$$t = t_1 \times 2^d + t_0$$

The least significant bits of the coefficients of t are not given, verification is no longer possible:

Hidden problems..

The Dilithium version is not complete. The public key is compressed. :

$$t = t_1 \times 2^d + t_0$$

The least significant bits of the coefficients of t are not given, verification is no longer possible:

$$\mathbf{t} = (\mathbf{t}^{[0]}, \ \ldots \ , \mathbf{t}^{[k-1]})$$

$$\mathbf{t}^{[0]} = \sum_{i=0}^{n-1} \mathbf{t}_i^{[0]} x^i$$

$$\mathbf{t}_0^{[0]} = \underbrace{0\ 0\ 1\ 1\ 0\ \cdots \overbrace{\square\ \square\ \square\ \square}^{13}}_{23}$$

Hidden problems..

The Dilithium version is not complete. The public key is compressed. :

$$t = t_1 \times 2^d + t_0$$

The least significant bits of the coefficients of t are not given, verification is no longer possible:

Algorithm 1 Ver
1: $\mathbf{w}'_1 := \mathtt{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$
2: **Accept if** $\|\|\mathbf{z}\|\|_\infty \leq \gamma_1 - \beta$ **and** $c = H(M\|\|\mathbf{w}'_1)$

Bob could recompute w_1 because:

$$\mathrm{w}_1 = HighBits_q(Ay, 2\gamma_2) = HighBits_q(Az - ct, 2\gamma_2)$$

Bob can now compute: $HighBits_q\left(Az - ct_1\, 2^d, 2\gamma_2\right) \neq HighBits_q\left(Az - ct_1\, 2^d - ct_0, 2\gamma_2\right).$

Hidden problems..

The Dilithium version is not complete. The public key is compressed. :

$$t = t_1 \times 2^d + t_0$$

The least significant bits of the coefficients of t are not given, verification is no longer possible:

Algorithm 1 Ver
1: $\mathbf{w}_1' := \texttt{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$
2: **Accept if** $\|\mathbf{z}\|_\infty \leq \gamma_1 - \beta$ **and** $c = H(M \| \mathbf{w}_1')$

Bob could recompute w_1 because:

$$\mathrm{w}_1 = HighBits_q(Ay, 2\gamma_2) = HighBits_q(Az - ct, 2\gamma_2)$$

Bob can now compute: $HighBits_q(Az - ct_1\,2^d, 2\gamma_2) \neq HighBits_q(Az - ct_1\,2^d - ct_0, 2\gamma_2)$.

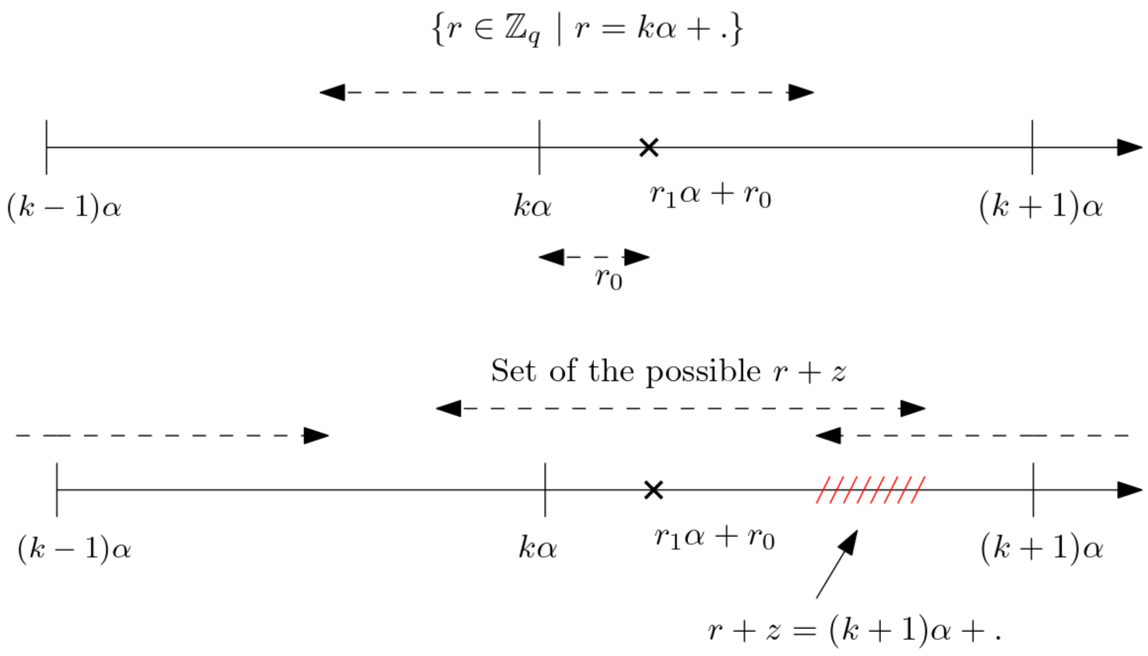
Algorithm: One knows $r = r_1 \times \alpha + r_0$ and $z = 0 \times \alpha + z_0$,

How to find $HighBits_q(r + z, \alpha)$ without knowing z ?

Hidden problems..

$$HighBits_q(Az - ct_1\, 2^d, 2\gamma_2) \neq HighBits_q(Az - ct_1\, 2^d - ct_0, 2\gamma_2)$$

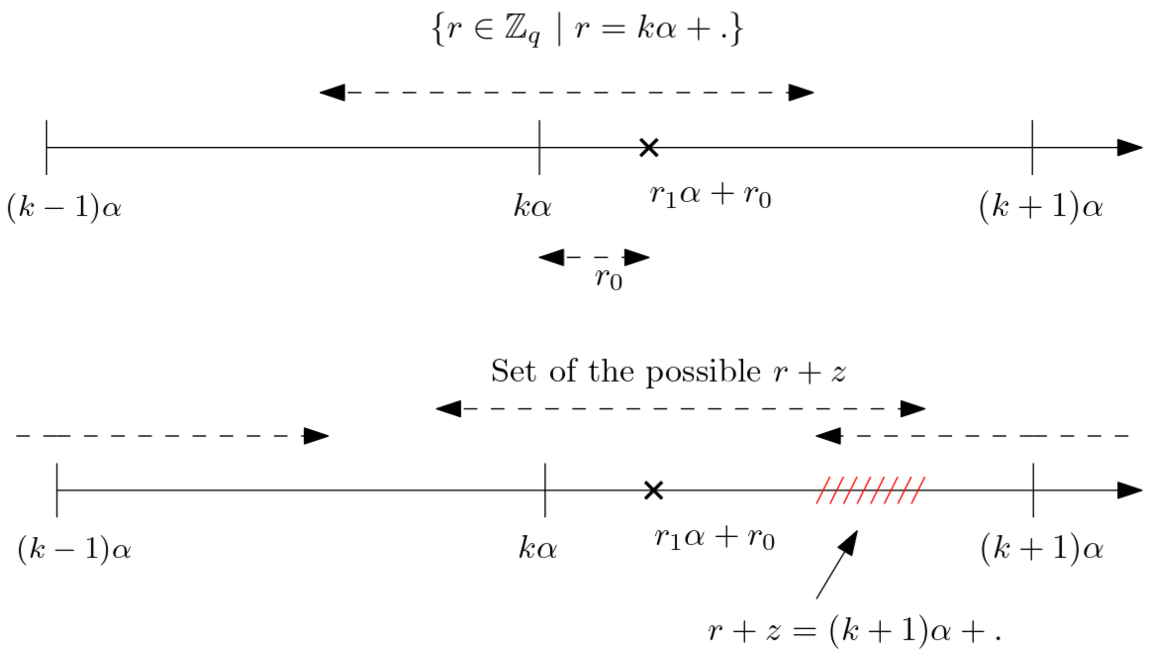
Algorithm: One knows $r = r_1\alpha + r_0$ and $z = 0 \times \alpha + z_0$ we want to find $HighBits_q(r + z, \alpha)$ without knowing z.



Hidden problems..

$$HighBits_q\left(Az - ct_1\, 2^d, 2\gamma_2\right) \neq HighBits_q\left(Az - ct_1\, 2^d - ct_0, 2\gamma_2\right)$$

Algorithm: One knows $r = r_1\alpha + r_0$ and $z = 0 \times \alpha + z_0$ we want to find $HighBits_q(r + z, \alpha)$ without knowing z.



Algorithm 1 Supporting algorithms for Dilithium

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

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

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

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

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

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

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

Hidden problems: The complete Dilithium

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:*

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

Algorithm 1 KeyGen

Ensure: (pk, sk)

1: $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times l}$
2: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k$
3: $\mathbf{t} := \mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2$
4: **return** $pk = (\mathbf{A}, \mathbf{t})$, $sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2)$

Algorithm 2 $\texttt{KeyGen}$

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: $\boxed{(\mathbf{t}_1,\ \mathbf{t}_0) := \texttt{Power2Round}_q(\mathbf{t},\ d)}$
7: $tr \in \{0,1\}^{256} := \mathrm{H}(\rho \,||\, \mathbf{t}_1)$
8: **return** $\boxed{pk = (\rho,\ \mathbf{t}_1),\ sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)}$

Hidden problems: The complete Dilithium

Algorithm 1 Sig

Require: sk, M

Ensure: $\sigma = (c, \mathbf{z})$

$\mathbf{z} = \perp$

while $\mathbf{z} = \perp$ **do**

  $\mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$

  $\mathbf{w}_1 := \mathtt{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$

  $c \in B_\tau := H(M || \mathbf{w}_1)$

  $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$

  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\mathtt{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)||_\infty \geq \gamma_2 - \beta$ **then**

    $\mathbf{z} := \perp$

  end if

end while

return $\sigma = (c, \mathbf{z})$

Algorithm 3 Sig

Require: sk, M

Ensure: $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

$\mathbf{A} \in \mathcal{R}_q^{k \times l} := \mathtt{ExpandA}(\rho)$

$\mu \in \{0,1\}^{512} := \mathrm{H}(tr \,||\, M)$

$\kappa := 0,\ (\mathbf{z}, \mathbf{h}) := \perp$

$\rho' \in \{0,1\}^{512} := \mathrm{H}(K \,||\, \mu)$

while $(\mathbf{z}, \mathbf{h}) = \perp$ **do**

  $\mathbf{y} \in \tilde{S}^{l}_{\gamma_1} := \mathtt{ExpandMask}(\rho', \kappa)$

  $\mathbf{w} := \mathbf{A}\,\mathbf{y}$

  $\mathbf{w}_1 = \mathtt{HighBits}_q(\mathbf{w}, 2\gamma_2)$

  $\tilde{c} \in \{0,1\}^{256} := \mathrm{H}(\mu \,||\, \mathbf{w}_1)$

  $c \in B_\tau := \mathtt{SampleInBall}(\tilde{c})$

  $\mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$

  $\mathbf{r}_0 := \mathtt{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$

  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**

    $(\mathbf{z}, \mathbf{h}) := \perp$

  else

    $\boxed{\mathbf{h} := \mathtt{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)}$

    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 $\boxed{\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})}$

Hidden problems: The complete Dilithium

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:*

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

Algorithm 1 Ver

$\mathbf{w}_1' := \texttt{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$

Accept if $||\mathbf{z}||_\infty \leq \gamma_1 - \beta$ **and** $c = H(M||\mathbf{w}_1')$

Algorithm 4 Ver

Require: pk, σ

$\mathbf{A} \in \mathcal{R}_q^{k \times l} := \texttt{ExpandA}(\rho)$

$\mu \in \{0,1\}^{512} := \mathrm{H}(\mathrm{H}(\rho \,||\, \mathbf{t}_1) \,||\, M)$

$c := \texttt{SampleInBall}(\tilde{c})$

$\boxed{\mathbf{w}_1' := \texttt{UseHint}_q(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)}$

return $[\![||\mathbf{z}||_\infty < \gamma_1 - \beta]\!]$ and $[\![\tilde{c} = \mathrm{H}(\mu \,||\, \mathbf{w}_1')]\!]$ and $[\![|\mathbf{h}|_{\mathbf{h}_j=1} \leq \omega]\!]$

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

NIST

thus producing the polynomial vector $\mathbf{t}_1$. This compression is an optimization for performance, not security. The low order bits of t can be reconstructed from a small number of signatures and, therefore, need not be regarded as secret.

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

NIST

thus producing the polynomial vector $\mathbf{t}_1$. This compression is an optimization for performance, not security. The low order bits of t can be reconstructed from a small number of signatures and, therefore, need not be regarded as secret.

RRB+19

There is a subtle but considerable difference with respect to publicly revealed LWE instances in the Dilithium scheme. The public key reveals only $\mathbf{t}_1$, the d higher order bits of $\mathbf{t}$, while $\mathbf{t}_0$ (the lower order component) is part of the secret key. Even on ensuring nonce-reuse, we would not be able to trivially solve for the secret $\mathbf{s}$ from the faulty public key. But, 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 [1]. Thus it

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

NIST

thus producing the polynomial vector $\mathbf{t}_1$. This compression is an optimization for performance, not security. The low order bits of t can be reconstructed from a small number of signatures and, therefore, need not be regarded as secret.

RRB+19

There is a subtle but considerable difference with respect to publicly revealed LWE instances in the Dilithium scheme. The public key reveals only $\mathbf{t}_1$, the d higher order bits of $\mathbf{t}$, while $\mathbf{t}_0$ (the lower order component) is part of the secret key. Even on ensuring nonce-reuse, we would not be able to trivially solve for the secret $\mathbf{s}$ from the faulty public key. But, 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 [1]. Thus it

References

1. Suppressed for blind review

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

EAB+23

Dilithium [BBK16, RJH+19]. In our attack, the knowledge of $\mathbf{t_0}$ is not required for the MLWE to RLWE reduction part of our attack as $\mathbf{t_0}$ can be embedded into the additive noise vector (moving $\mathbf{t_0}$ to the right part of Equation 6). However, it has an impact on the resulting security of the RLWE problem making it harder to solve. As it is unclear if $\mathbf{t_0}$ must be considered secret or public (it has been hinted that $\mathbf{t_0}$ can be recovered from enough signatures in [Lyu22, RJH+18, RRB+19]), we take a worst-case approach for the rest of this work. If not specified in the following sections, the full $\mathbf{t}$ is assumed to be public. In Subsection 5.3, we derive the impact of fully secret $\mathbf{t_0}$ on the complexity of the (reduced) RLWE instance. We hope that this approach gives a complete view to the reader about the applicability of the attack to Dilithium.

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

EAB+23

Dilithium [BBK16, RJH+19]. In our attack, the knowledge of $\mathbf{t_0}$ is not required for the MLWE to RLWE reduction part of our attack as $\mathbf{t_0}$ can be embedded into the additive noise vector (moving $\mathbf{t_0}$ to the right part of Equation 6). However, it has an impact on the resulting security of the RLWE problem making it harder to solve. As it is unclear if $\mathbf{t_0}$ must be considered secret or public (it has been hinted that $\mathbf{t_0}$ can be recovered from enough signatures in [Lyu22, RJH+18, RRB+19]), we take a worst-case approach for the rest of this work. If not specified in the following sections, the full $\mathbf{t}$ is assumed to be public. In Subsection 5.3, we derive the impact of fully secret $\mathbf{t_0}$ on the complexity of the (reduced) RLWE instance. We hope that this approach gives a complete view to the reader about the applicability of the attack to Dilithium.

RJH+18

Thus, it might indeed be possible that the whole of $\bar{t}$ leaks as part of the signature and observations of sufficiently many signatures might lead to the recovery of the complete LWE instance, $\bar{t}$. 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.

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

EAB+23

Dilithium [BBK16, RJH+19]. In our attack, the knowledge of $\mathbf{t_0}$ is not required for the MLWE to RLWE reduction part of our attack as $\mathbf{t_0}$ can be embedded into the additive noise vector (moving $\mathbf{t_0}$ to the right part of Equation 6). However, it has an impact on the resulting security of the RLWE problem making it harder to solve. As it is unclear if $\mathbf{t_0}$ must be considered secret or public (it has been hinted that $\mathbf{t_0}$ can be recovered from enough signatures in [Lyu22, RJH+18, RRB+19]), we take a worst-case approach for the rest of this work. If not specified in the following sections, the full $\mathbf{t}$ is assumed to be public. In Subsection 5.3, we derive the impact of fully secret $\mathbf{t_0}$ on the complexity of the (reduced) RLWE instance. We hope that this approach gives a complete view to the reader about the applicability of the attack to Dilithium.

WNGD23

ferent from profiling device is non-negligible (9%). The success rate approaches 100% if multiple traces are available for the attack. Our results demonstrate the necessity of protecting the secret key of CRYSTALS-Dilithium from single-trace attacks and call for a reassessment of the role of compression of the public key vector $\mathbf{t}$ in the security of CRYSTALS-Dilithium implementations.

RJH+18

Thus, it might indeed be possible that the whole of $\bar{t}$ leaks as part of the signature and observations of sufficiently many signatures might lead to the recovery of the complete LWE instance, $\bar{t}$. 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.

Hidden problems: A status for t_0?

Formally, t_0 is part of the private key, but it is a performance optimization. The security proof considers it public, but what about side-channel attacks?

EAB+23

Dilithium [BBK16, RJH+19]. In our attack, the knowledge of $\mathbf{t_0}$ is not required for the MLWE to RLWE reduction part of our attack as $\mathbf{t_0}$ can be embedded into the additive noise vector (moving $\mathbf{t_0}$ to the right part of Equation 6). However, it has an impact on the resulting security of the RLWE problem making it harder to solve. As it is unclear if $\mathbf{t_0}$ must be considered secret or public (it has been hinted that $\mathbf{t_0}$ can be recovered from enough signatures in [Lyu22, RJH+18, RRB+19]), we take a worst-case approach for the rest of this work. If not specified in the following sections, the full $\mathbf{t}$ is assumed to be public. In Subsection 5.3, we derive the impact of fully secret $\mathbf{t_0}$ on the complexity of the (reduced) RLWE instance. We hope that this approach gives a complete view to the reader about the applicability of the attack to Dilithium.

WNGD23

ferent from profiling device is non-negligible (9%). The success rate approaches 100% if multiple traces are available for the attack. Our results demonstrate the necessity of protecting the secret key of CRYSTALS-Dilithium from single-trace attacks and call for a reassessment of the role of compression of the public key vector $\mathbf{t}$ in the security of CRYSTALS-Dilithium implementations.

RJH+18

Thus, it might indeed be possible that the whole of $\bar{t}$ leaks as part of the signature and observations of sufficiently many signatures might lead to the recovery of the complete LWE instance, $\bar{t}$. 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.

Finding a polytope: A practical fault attack against Dilithium

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

$$(cs_2)_i \geq \gamma_2 - LowBits_q((Az - ct)_i, 2\gamma_2)\ldots$$

How to find t_0 ?

The inequation ideas presented above work! Each Dilithium signature provides inequations on the coefficients of t_0.

How to find t_0 ?

The inequation ideas presented above work! Each Dilithium signature provides inequations on the coefficients of t_0.

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* $\texttt{Sig}$.

– *If* $\mathbf{h}_i^{[j]} = 0$:

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

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

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

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

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

How to find t_0 ?

The inequation ideas presented above work! Each Dilithium signature provides inequations on the coefficients of t_0.

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

– *If* $\mathbf{h}_i^{[j]} = 0$:

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

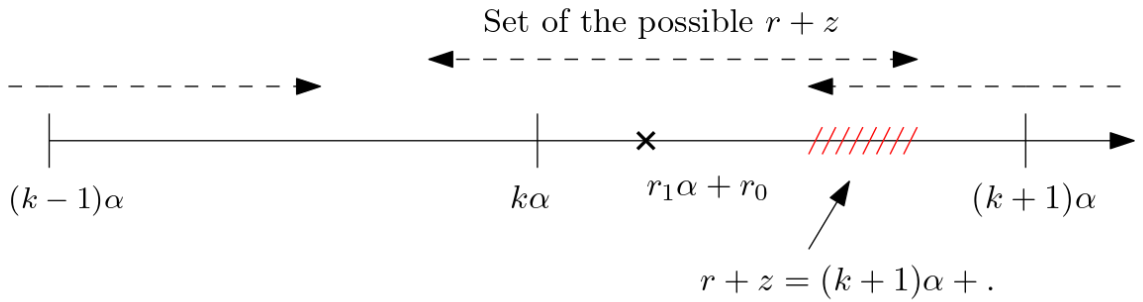
– *If* $\mathbf{h}_i^{[j]} = 1$ *and* $\texttt{LowBits}_q(\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\,\gamma_2)_i^{[j]} > 0$:

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

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

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

Proof (idea):



h gives information about the size of ct_0.

$$HighBits_q(Az - ct_1\, 2^d, 2\gamma_2) \overset{?}{\neq} HighBits_q(Az - ct_1\, 2^d - ct_0, 2\gamma_2)$$

How to find t_0 ?

Naive attack method: We try to repeat the attack as before.

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 $\mathbf{t}_0$.

How to find t_0 ?

Naive attack method: We try to repeat the attack as before.

Number of signatures	Number of inequalities	$\lVert \mathbf{t}_0^{[0]} - \tilde{\mathbf{t}}_0^{[0]} \rVert_\infty$	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 $\mathbf{t}_0$.

First problem:

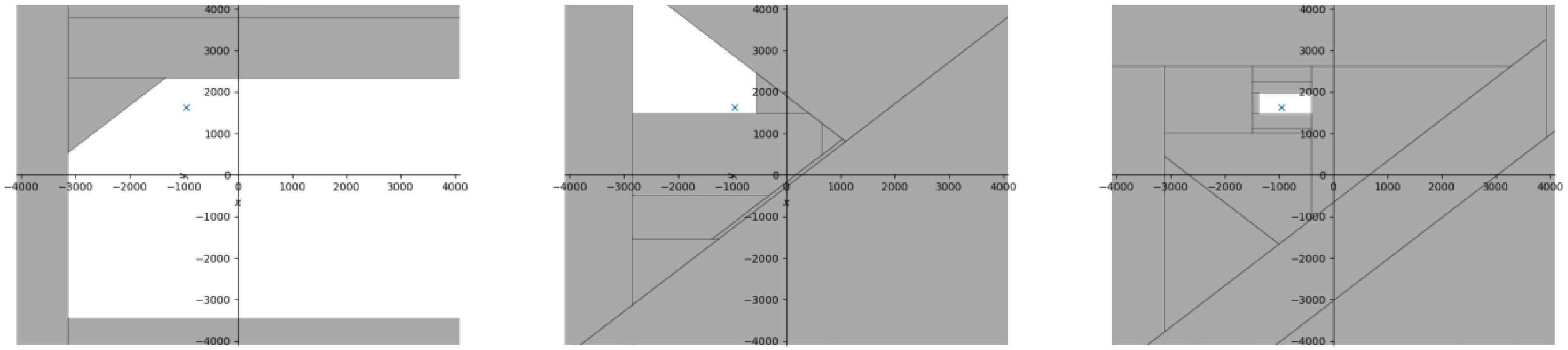


Fig. 4. Polytope containing $(\mathbf{t}_{0,0}^{[0]}, \mathbf{t}_{0,1}^{[0]})$ for 10, 50 and 100 inequalities.

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

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

How to find t_0 ?

Naive attack method: We try to repeat the attack as before.

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 $\mathbf{t}_0$.

Second problem:

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.

We obtain many more inequalities per signature, the (LP) problem to be solved is too complex.

How to find t_0 ?

Result of the naive method:

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 $\mathbf{t}_0$.

How to find t_0 ?

Result of the naive method:

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

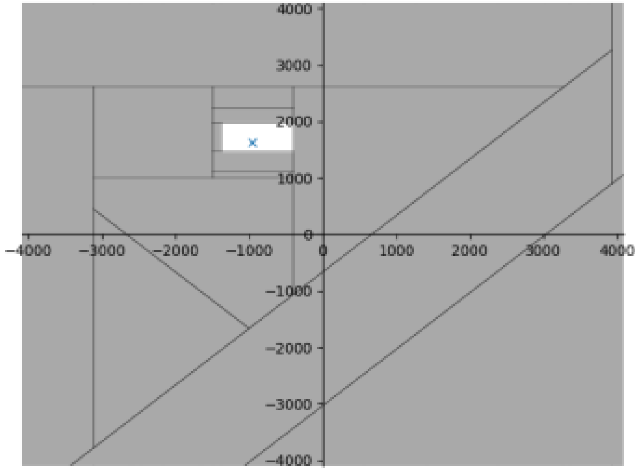
Table 4. Attack times and size of the (LP) system on $\mathbf{t}_0$.

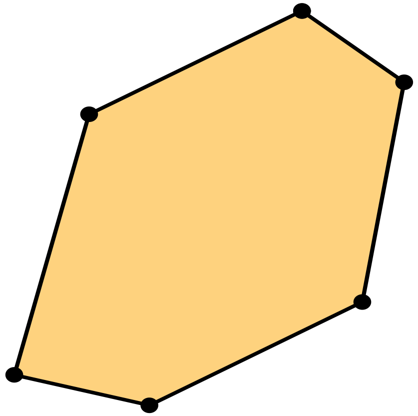
Idea:

We have a complex algebraic representation (lots of inequalities) of a simple geometric object (a polytope with a few faces).

The complexity of the LP solver depends on the number of inequalities:

We need to find a way to filter out useful inequalities.





How to find t_0 ? Filtrations

We suppose know a radius C and a polynomial $\tilde{t_0}$ such that $t_0 \in B_\infty(\tilde{t_0}, C)$.

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:*

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

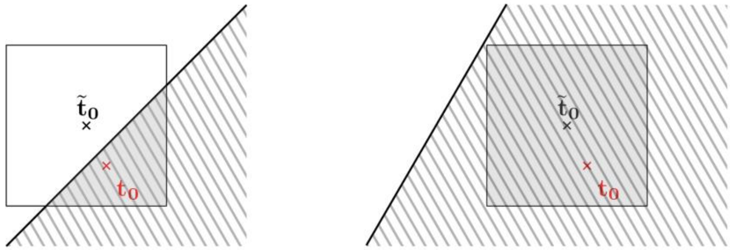


Fig. 6. On the left, a useful inequation. On the right a useless inequation.

How to find t_0 ? Filtrations

We suppose know a radius C and a polynomial $\tilde{t_0}$ such that $t_0 \in B_\infty(\tilde{t_0}, C)$.

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:*

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

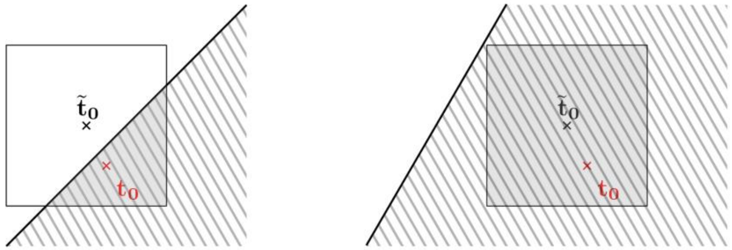


Fig. 6. On the left, a useful inequation. On the right a useless inequation.

We show that it is possible to efficiently calculate whether an inequality is useful or not:

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.

How to find t_0 ? Attack idea

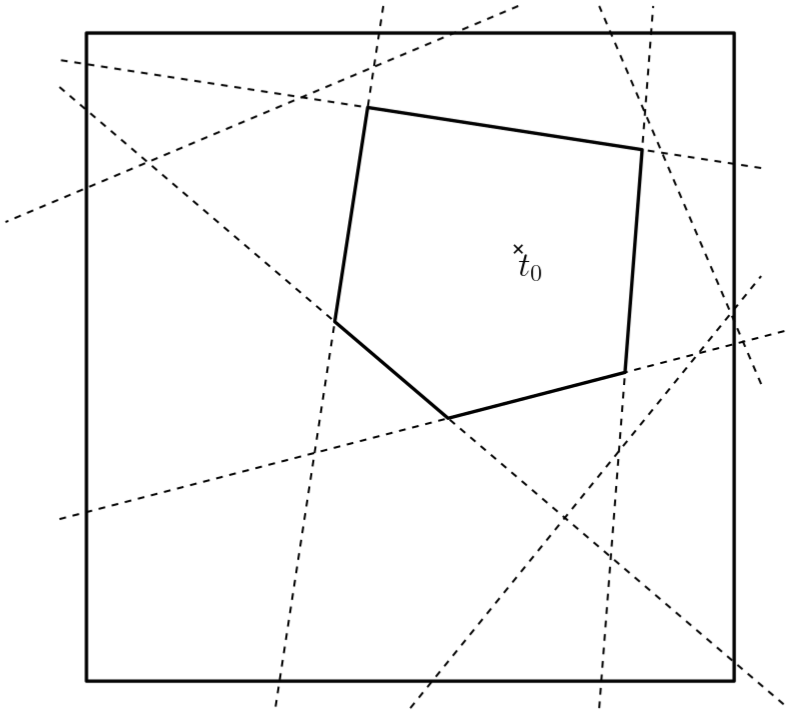
We use the strategy « Collect,guess, filter, repeat. »

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

Ensure: A candidate for $\mathbf{t}_0^{[0]}$

Require: An inequation step sequence $(\delta_i)_{i\in\{1,\ldots,m\}}$, a radius sequence $C_m < C_{m-1} < \cdots < C_1 = 2^{12}$.

1: $\tilde{\mathbf{t}}_0^{[0]} = 0$
2: $i = 1$
3: $P = \{-2^{12}+1 \leq x_i \leq 2^{12}\}_{i=1,\ldots,256}$
4: **while** $i \leq m$ **do**
5: $\quad P = \texttt{generate\_useful\_ineq}(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$
6: $\quad i = i + 1$
7: $\quad \tilde{\mathbf{t}}_0^{[0]} = \texttt{round(lp\_guess(P))}$
8: **return** $\tilde{\mathbf{t}}_0^{[0]}$



How to find t_0 ? Attack idea

We use the strategy « Collect,guess, filter, repeat. »

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

Ensure: A candidate for $\mathbf{t}_0^{[0]}$

Require: An inequation step sequence $(\delta_i)_{i\in\{1,\ldots,m\}}$, a radius sequence $C_m < C_{m-1} < \cdots < C_1 = 2^{12}$.

1: $\tilde{\mathbf{t}}_0^{[0]} = 0$

2: $i = 1$

3: $P = \{-2^{12} + 1 \leq x_i \leq 2^{12}\}_{i=1,\ldots,256}$

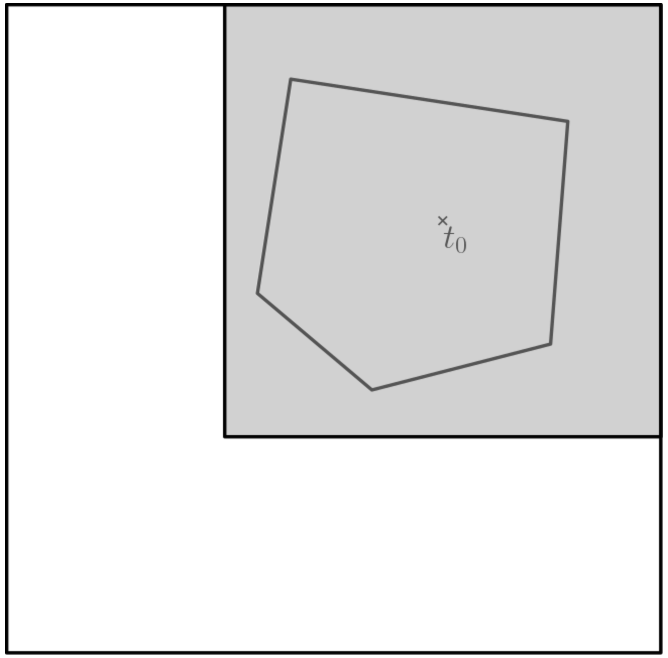
4: **while** $i \leq m$ **do**

5: $\quad P = \texttt{generate\_useful\_ineq}(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$

6: $\quad i = i + 1$

7: $\quad \tilde{\mathbf{t}}_0^{[0]} = \texttt{round(lp\_guess(P))}$

8: **return** $\tilde{\mathbf{t}}_0^{[0]}$



How to find t_0 ? Attack idea

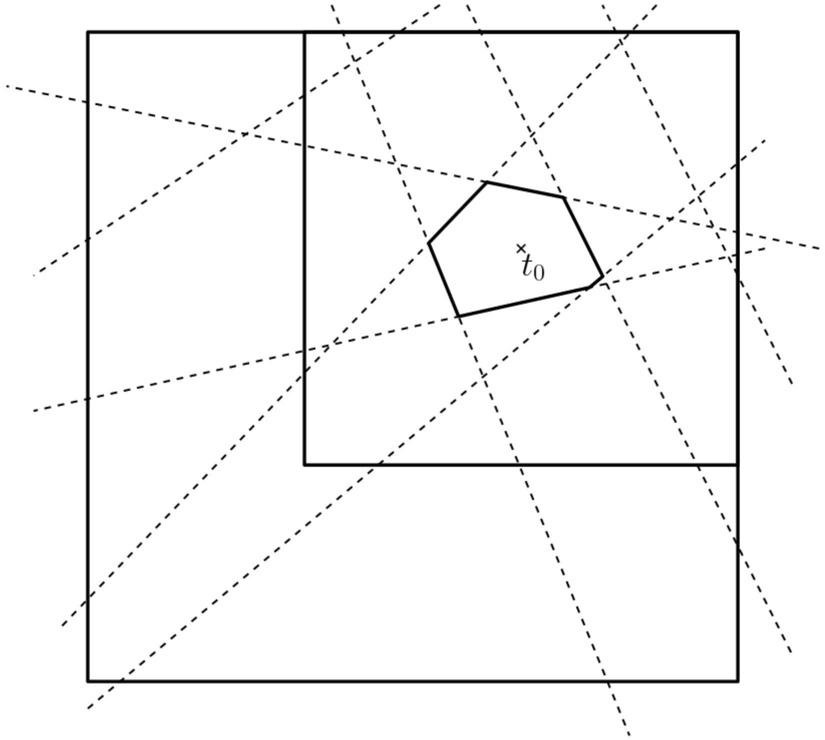
We use the strategy « Collect,guess, filter, repeat. »

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 $C_m < C_{m-1} < \dots < C_1 = 2^{12}$.

1: $\tilde{\mathbf{t}}_0^{[0]} = 0$
2: $i = 1$
3: $P = \{-2^{12} + 1 \le x_i \le 2^{12}\}_{i=1,\dots,256}$
4: **while** $i \le m$ **do**
5: $\quad P = \texttt{generate\_useful\_ineq}(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$
6: $\quad i = i + 1$
7: $\quad \tilde{\mathbf{t}}_0^{[0]} = \texttt{round(lp\_guess(P))}$
8: **return** $\tilde{\mathbf{t}}_0^{[0]}$



How to find t_0 ? Attack idea

We use the strategy « Collect,guess, filter, repeat. »

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

Ensure: A candidate for $\mathbf{t}_0^{[0]}$

Require: An inequation step sequence $(\delta_i)_{i\in\{1,\ldots,m\}}$, a radius sequence $C_m < C_{m-1} < \cdots < C_1 = 2^{12}$.

1: $\tilde{\mathbf{t}}_0^{[0]} = 0$

2: $i = 1$

3: $P = \{-2^{12} + 1 \le x_i \le 2^{12}\}_{i=1,\ldots,256}$

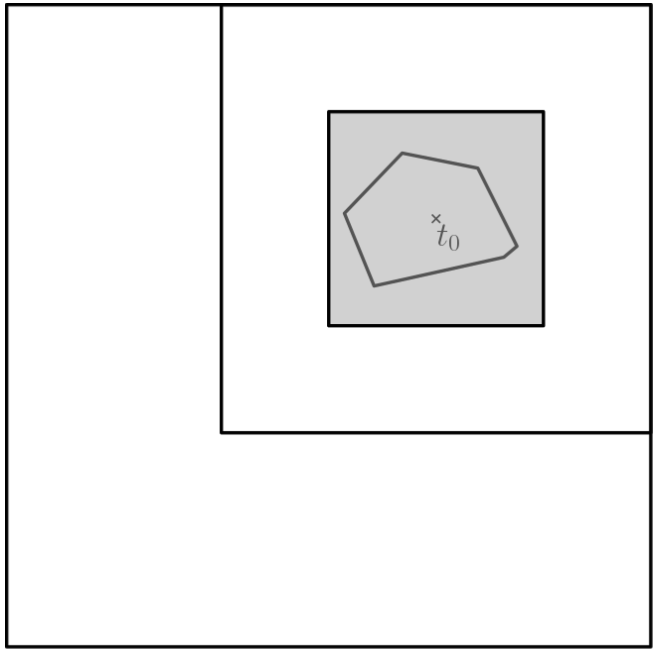
4: **while** $i \le m$ **do**

5: $\quad P = \texttt{generate\_useful\_ineq}(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$

6: $\quad i = i + 1$

7: $\quad \tilde{\mathbf{t}}_0^{[0]} = \texttt{round(lp\_guess(P))}$

8: **return** $\tilde{\mathbf{t}}_0^{[0]}$



How to find t_0 ? Attack idea

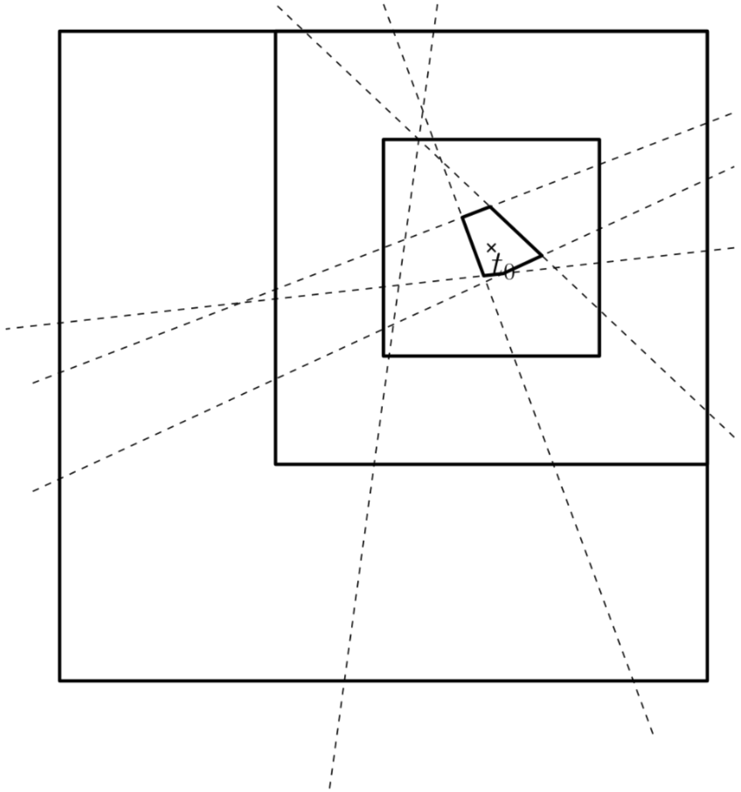
We use the strategy « Collect,guess, filter, repeat. »

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

Ensure: A candidate for $\mathbf{t}_0^{[0]}$

Require: An inequation step sequence $(\delta_i)_{i\in\{1,\ldots,m\}}$, a radius sequence $C_m < C_{m-1} < \cdots < C_1 = 2^{12}$.

1: $\tilde{\mathbf{t}}_0^{[0]} = 0$
2: $i = 1$
3: $P = \{-2^{12} + 1 \leq x_i \leq 2^{12}\}_{i=1,\ldots,256}$
4: **while** $i \leq m$ **do**
5: $\quad P = \texttt{generate\_useful\_ineq}(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$
6: $\quad i = i + 1$
7: $\quad \tilde{\mathbf{t}}_0^{[0]} = \texttt{round(lp\_guess(P))}$
8: **return** $\tilde{\mathbf{t}}_0^{[0]}$



How to find t_0 ? Attack idea

We use the strategy « Collect,guess, filter, repeat. »

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

Ensure: A candidate for $\mathbf{t}_0^{[0]}$

Require: An inequation step sequence $(\delta_i)_{i\in\{1,\ldots,m\}}$, a radius sequence $C_m < C_{m-1} < \cdots < C_1 = 2^{12}$.

$\tilde{\mathbf{t}}_0^{[0]} = 0$

$i = 1$

$P = \{-2^{12} + 1 \leq x_i \leq 2^{12}\}_{i=1,\ldots,256}$

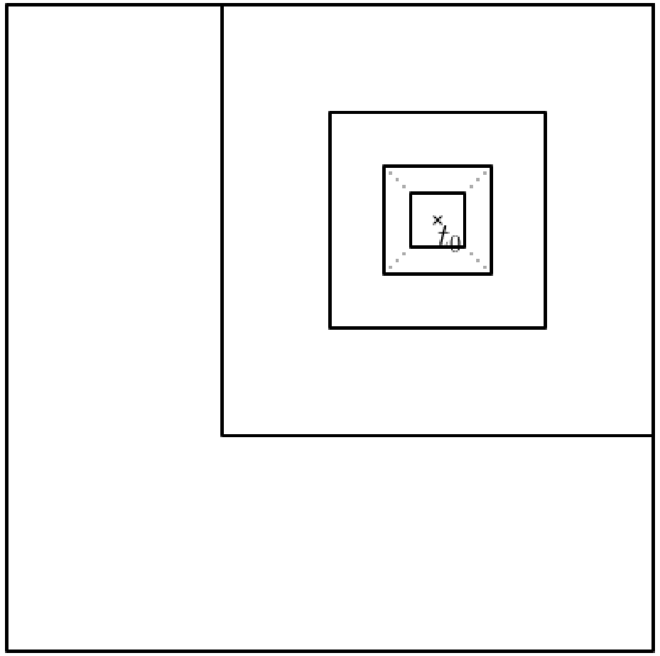
while $i \leq m$ **do**

  $P = \texttt{generate\_useful\_ineq}(\delta_i, \tilde{\mathbf{t}}_0^{[0]}, C_i)$

  $i = i + 1$

  $\tilde{\mathbf{t}}_0^{[0]} = \texttt{round(lp\_guess(P))}$

return $\tilde{\mathbf{t}}_0^{[0]}$



How to find t_0 ? Attack idea

We use the strategy « Collect,guess, filter, repeat. »

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 $C_m < C_{m-1} < \dots < C_1 = 2^{12}$.

$\tilde{\mathbf{t}}_0^{[0]} = 0$

$i = 1$

$P = \{-2^{12} + 1 \leq x_i \leq 2^{12}\}_{i=1,\dots,256}$

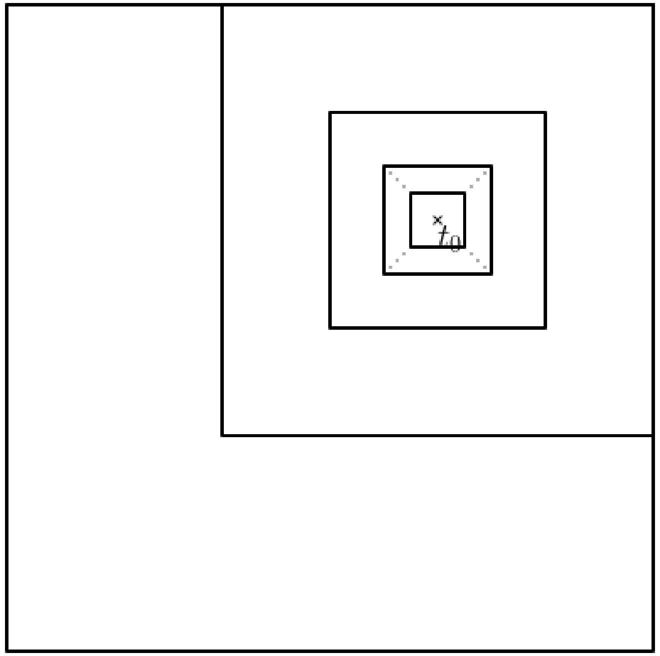
while $i \leq m$ **do**

  $P =$ `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 $\tilde{\mathbf{t}}_0^{[0]}$



How do you choose the sequence of radius? And the number of inequalities?

How to find t_0 ? Attack idea

We use statistics to verify that it is heuristically correct.

We have chosen $C_i = (4096, 2048, 1024, \dots, 16, 8)$ and a constant number of collected inequations: $50\,000$.

One can suppose t_0 know to better understand what is happening:

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.

How to find t_0 ? Results

Attack result:

Signatures	inequalities selected	Recovery probability	Average time	Median time
179 354	392 696 + 213 943	1	1h26m53s	1h24m8s

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

Without filtration: Each signatures gives ≈ 500 inequations on each polynomial of t_0.

It would be necessary to solve a problem (LP) of approximately $100\,000\,000$ inequations in 256 variables, which is difficult even for modern solvers.

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

Conclusion:

Linear programming tools are often used in symmetric cryptanalysis, but relatively rarely in public-key cryptanalysis.

By exploiting the integer decomposition algorithms in Dilithium, we obtain two results:

1. **A fault attack that exploits invalid Dilithium signatures.**

2. **We show that Dilithium's public key can be decompressed from enough signatures generated with the same secret key.**

THALES

Building a future we can all trust

Merci !

www.thalesgroup.com