An attack on ML-DSA using an implicit hint

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

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

THALES
Building a future we can all trust

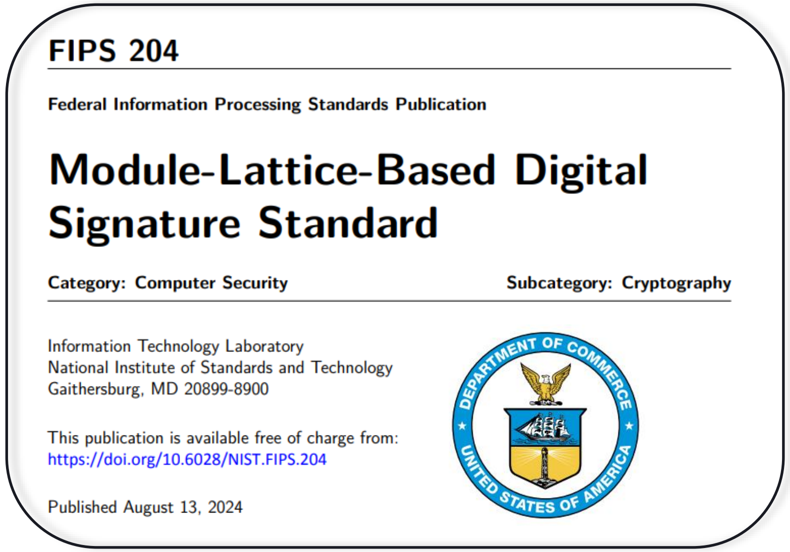
Table of contents

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales **© 2024 THALES. All rights reserved.**

Context

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

ML-DSA is recommended for computing quantum-secure signatures in most use cases.



FIPS 204

Federal Information Processing Standards Publication

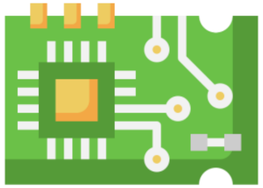
Module-Lattice-Based Digital Signature Standard

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 investigate the security of embedded implementations. The security of ML-DSA against Side-Channel Attacks (SCA) and Fault Attacks (FA) thus needs to be carefully assessed.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

An overview of ML-DSA

ML-DSA 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 **KeyGen**

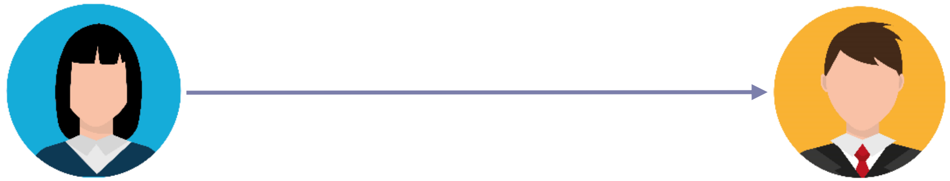
Ensure: (pk, sk)

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

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

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

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)

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

An overview of ML-DSA

ML-DSA uses two rings:

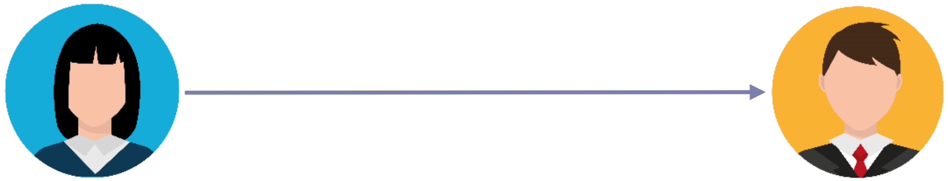
$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

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

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



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

α an even integer which 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: $\left\{-\frac{\alpha}{2} + 1, \dots, 0, \dots, \frac{\alpha}{2}\right\}$

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

One note:

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

THALES
Building a future we can all trust
REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

An overview of ML-DSA

ML-DSA uses two rings:

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

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

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

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

Algorithm KeyGen

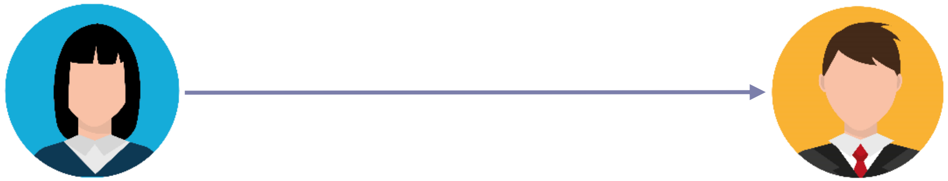
Ensure: (pk, sk)

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

$$P = \left(P^{[1]}, \dots, P^{[l]}\right)$$

$$P^{[i]} = \sum p_i x^i$$

$$HighBits_q(P^{[i]}, \alpha) = \sum HighBits_q(p_i, \alpha) x^i$$



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

$$HighBits_q(P, \alpha) = \left(HighBits_q(P^{[1]}, \alpha), \dots, HighBits_q(P^{[l]}, \alpha)\right)$$

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

An overview of ML-DSA

Algorithm 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 − c s2, 2γ2)||_∞ ≥ γ2 − β then
        z := ⊥
    end if
end while
return σ = (c, z)
```

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

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

Alice draws a polynomial vector at random:

$$y \in_R R^l, \quad ||y||_\infty \leq \gamma_1.$$

She computes a random challenge that depends on the message:

$$c = H\left(M \,||\, HighBits_q(Ay, 2\gamma_2)\right).$$

She provides a response to the challenge:

$$z = y + c s_1$$

By definition of z:

$$Az - ct = Ay - c s_2.$$

The signature will be:

$$\sigma = (c, z).$$

But..

THALES
Building a future we can all trust
REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

An overview of ML-DSA

Algorithm Sig

Require: sk, M

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

$\mathbf{z} = \perp$

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

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

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

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

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

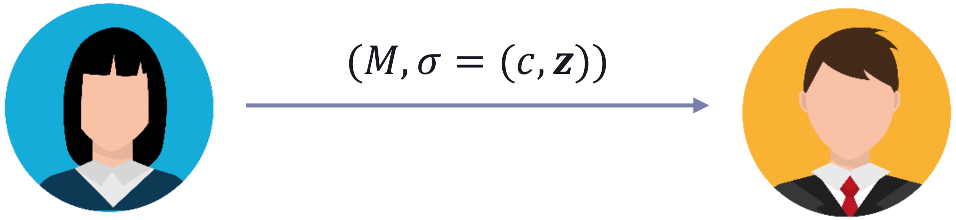
  if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $\mathtt{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)||_\infty \geq \gamma_2 - \beta$ **then**

    $\mathbf{z} := \perp$

  end if

end while

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



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

But..

By definition of z:

$$z = y + cs_1$$

Two conditions must be fulfilled:

$$\begin{cases} ||z||_\infty < max_y\left(||y||_\infty\right) - max_{\{c,s_1\}}\left(||cs_1||_\infty\right) \\ HighBits_q(Ay, 2\gamma_2) = HighBits_q(Ay - cs_2, 2\gamma_2) \end{cases}$$

The first condition is for security, the second for verification and security.

With these conditions:

$$HighBits(Az - ct) = HighBits(Ay - cs_2) = HighBits(Ay)$$

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

An overview of ML-DSA

Algorithm 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 − c s2, 2γ2)||∞ ≥ γ2 − β then
        z := ⊥
    end if
end while
return σ = (c, z)
```

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:

$$\begin{aligned} \mathbf{w_1} &= HighBits_q(Ay, 2\gamma_2) \\ &= HighBits_q(Ay - cs_2, 2\gamma_2) \\ &= HighBits_q(Az - ct, 2\gamma_2) \\ &= w'_1 \end{aligned}$$

(A, t, s_1, s_2)

$(M, \sigma = (c, \mathbf{z}))$ →

(A, t)

THALES
Building a future we can all trust
REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

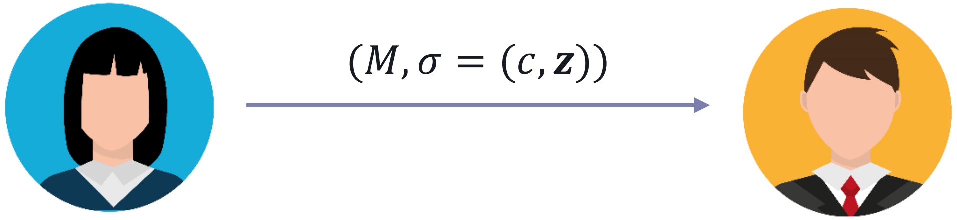
An overview of ML-DSA

Algorithm Sig

Require: sk, M

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

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



(A, t, s_1, s_2) $\quad (M, \sigma = (c, \mathbf{z})) \quad$ (A, t)

Algorithm 1 Ver

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

Bob can recompute w_1:

$$\begin{aligned} \mathbf{w}_1 &= HighBits_q(Ay, 2\gamma_2) \\ &= HighBits_q(Ay - cs_2, 2\gamma_2) \\ &= HighBits_q(Az - ct, 2\gamma_2) \\ &= w_1' \end{aligned}$$

All that aside, the most important relation is:

$$z = y + cs_1$$

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

Existing fault attack on ML-DSA

www.thalesgroup.com

THALES
Building a future we can all trust

A fault attack on ML-DSA

Loop-Abort Faults on Lattice-Based Fiat–Shamir and Hash-and-Sign Signatures

Thomas Espitau[4], Pierre-Alain Fouque[2], Benoît Gérard[1], and Mehdi Tibouchi[3]

[EFGT17]: Published at SAC2017 and describes a fault attack against BLISS.

Algorithm 1 Sig

Require: sk, M

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

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

Main Idea: Inject a fault to obtain one of the coefficients of $\mathbf{y}$ of abnormally small degree.

They consider a signature $\sigma = (c, z)$ with

$$z^{[1]} = y^{[1]} + c s_1^{[1]} \quad \text{and} \quad \deg\left(y^{[1]}\right) = m \ll n$$

This will make $s_1{}^{[1]}$ the smallest vector in a lattice of sufficiently small dimension to find it.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA

Single fault attack:

One has:

$$z^{[1]} = y^{[1]} + cs_1^{[1]}$$

Thus if c is invertible:

$$s_1^{[1]} = c^{-1}z^{[1]} - \sum_{i=0}^{m} y_i^{[1]}(cx)^i \; mod(q).$$

Therefore,

$$s_1^{[1]} \in L(c^{-1}z^{[1]}, \{ (cx)^i \}_{i\in\{0,\dots,m\}})$$

If m is sufficently small, $s_1^{[1]}$ can be recovered using lattice reduction technique (LLL or BKZ).

Practical results:

Fault after iteration number $m =$	20	40	60	80	100
Theoretical minimum dimension $\ell_{\min}$	22	44	66	88	110
Dimension ℓ in our experiment	24	50	80	110	140
Lattice reduction algorithm	LLL	BKZ–20	BKZ–25	BKZ–25	BKZ–25
Success probability (%)	100	100	100	100	—
Avg. CPU time to recover ℓ coeffs. (s)	0.23	7.3	119	941	10500
Avg. CPU time for full key recovery	5 s	80 s	14 min	80 min	12 h

Conclusion:

The fault attack is plausible.

The fault needs to be injected before the generation of the 100 first coefficients.

Proposed countermeasure: Shuffling the order of the coefficient's generation.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

THALES
Building a future we can all trust

Our results

www.thalesgroup.com

A fault attack on ML-DSA

Limitations:

- Less realistic for ML-DSA.
- Simple countermeasures.
- Single fault attack?

Our questions:

- Applicable to ML-DSA?
- Possible to improve with more faults?
- Possible to overcome the simple countermeasure?
- Turn it into a passive attack?

Algorithm 34 ExpandMask(ρ, μ)

Samples a vector $\mathbf{y} \in R^\ell$ such that each polynomial $\mathbf{y}[r]$ has coefficients between $-\gamma_1 + 1$ and γ_1.

Input: A seed $\rho \in \mathbb{B}^{64}$ and a nonnegative integer μ.
Output: Vector $\mathbf{y} \in R^\ell$.

```
1: c ← 1 + bitlen (γ1 − 1)                      ▷ γ1 is always a power of 2
2: for r from 0 to ℓ − 1 do
3:     ρ′ ← ρ||IntegerToBytes(μ + r, 2)
4:     v ← H(ρ′, 32c)                            ▷ seed depends on μ + r
5:     y[r] ← BitUnpack(v, γ1 − 1, γ1)
6: end for
7: return y
```

Algorithm 19 BitUnpack(v, a, b)

Reverses the procedure BitPack.

Input: $a, b \in \mathbb{N}$ and a byte string v of length $32 \cdot \text{bitlen}\ (a+b)$.
Output: A polynomial $w \in R$ with coefficients in $[b - 2^c + 1, b]$, where $c = \text{bitlen}\ (a+b)$.
When $a + b + 1$ is a power of 2, the coefficients are in $[-a, b]$.

```
1: c ← bitlen (a + b)
2: z ← BytesToBits(v)
3: for i from 0 to 255 do
4:     w_i ← b − BitsToInteger((z[ic], z[ic + 1], ... z[ic + c − 1]), c)
5: end for
6: return w
```


This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: Improvement

Let $\sigma_1, \dots, \sigma_m$ be m signatures such that:

$$\forall i \in \{1, \dots, m\}, z_i^{[1]} = y_i^{[1]} + c s_1^{[1]} \text{ with } \deg\left(y_i^{[1]}\right) \leq d < n - 1.$$

Then one can construct m lattices such that:

$$\forall i \in \{1, \dots, m\}, \quad \dim(L_i) \leq d + 2 \quad \text{and} \quad s_1^{[1]} \in L = \bigcap L_i$$

THALES
Building a future we can all trust

REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: Improvement

Let $\sigma_1, \dots, \sigma_m$ be m signatures such that:

$$\forall i \in \{1, \dots, m\}, z_i^{[1]} = y_i^{[1]} + c s_1^{[1]} \text{ with } \deg\left(y_i^{[1]}\right) \leq d < n - 1.$$

Then one can construct m lattices such that:

$$\forall i \in \{1, \dots, m\}, \qquad \dim(L_i) \leq d + 2 \quad \text{and} \quad s_1^{[1]} \in L = \bigcap L_i$$

Formally:

$$\forall i \in \{1, \dots, m\}, \qquad L_i = L\left(c_i^{-1} z_i^{[1]}, \{ (c_i x)^j \}_{j \in \{0, \dots, d\}}\right) \quad and \quad L = \bigcap L_i$$

THALES
Building a future we can all trust
REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: Improvement

Let $\sigma_1, \dots, \sigma_m$ be m signatures such that:

$$\forall i \in \{1, \dots, m\}, z_i^{[1]} = y_i^{[1]} + c s_1^{[1]} \text{ with } \deg\left(y_i^{[1]}\right) \leq d < n - 1.$$

Then one can construct m lattices such that:

$$\forall i \in \{1, \dots, m\}, \quad \dim(L_i) \leq d + 2 \text{ and } s_1^{[1]} \in L = \bigcap L_i$$

Formally:

$$\forall i \in \{1, \dots, m\}, \quad L_i = L\left(c_i^{-1} z_i^{[1]}, \{ (c_i x)^j \}_{j \in \{0, \dots, d\}}\right) \quad and \quad L = \bigcap L_i$$

We have reformulated the problem of finding $s_1^{[1]}$, as the calculation of a lattice intersection

To have $d + 2 < n$, one needs $d \leq n - 3$.

The attack requires knowledge of 2 coefficients.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: How to intersect lattices efficently?

Classic method: Using duality

Let $L_1 = L(B_1)$ and $L_2 = L(B_2)$ be two lattices.

Union: $L_1 \cup L_2 = L\big(HNF(B_1|B_2)\big)$ and Duality relation: $(L_1 \cup L_2)^* = L_1^* \cap L_2^*$

Lead to:

$$L_1 \cap L_2 = \Big(L\big(HNF(D_1|D_2)\big)\Big)^*,$$

with D_1, D_2 such that $L_1^* = L(D_1)$ and $L_2^* = L(D_2)$.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: How to intersect lattices efficently?

Classic method: Using duality

Let $L_1 = L(B_1)$ and $L_2 = L(B_2)$ be two lattices.

Union: $L_1 \cup L_2 = L\big(HNF(B_1|B_2)\big)$ and Duality relation: $(L_1 \cup L_2)^* = L_1^* \cap L_2^*$

Lead to:

$$L_1 \cap L_2 = \Big(L\big(HNF(D_1|D_2)\big)\Big)^*,$$

with D_1, D_2 such that $L_1^* = L(D_1)$ and $L_2^* = L(D_2)$.

Problems: For $L \subset \mathbb{Z}^n$ generally $L^* \not\subset \mathbb{Z}^n$. One have to compute HNF over $\mathbb{Q}$, and numerators and denominators explode. This leads to rounding errors when calculating the HNF and an explosion in calculation time.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: How to intersect lattices efficently?

Optimized method: Using $\mathbb{F}_q$-subspaces.

Let $L_1 = L(B_1)$ and $L_2 = L(B_2)$ be two lattices, such that $L_1, L_2 \subset q\mathbb{Z}^n$

1. View $\overline{L_1}, \overline{L_2}$ as $\mathbb{F}_q$-subspaces
2. Compute an intersection of subspaces: $\overline{L} = \overline{L_1} \cap \overline{L_2}$ and B a basis of $\overline{L}$.
3. View $\overline{L}$ as an integer lattice by considering: $L = L\left(B, \{qx^j\}_{j\in\{0,\ldots,n-1\}}\right)$

Solution: No need to work in rationnal field. Better complexity.

1. Attack can be improved with more faults
2. No restriction on fault injection at the time of y generation

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: How to intersect lattices efficently?

Optimized method: Using $\mathbb{F}_q$-subspaces.

Let $L_1 = L(B_1)$ and $L_2 = L(B_2)$ be two lattices, such that $L_1, L_2 \subset q\mathbb{Z}^n$

1. View $\overline{\mathrm{L}_1}, \overline{L_2}$ as $\mathbb{F}_q$-subspaces
2. Compute an intersection of subspaces: $\overline{L} = \overline{L_1} \cap \overline{L_2}$ and B a basis of $\overline{L}$.
3. View $\overline{L}$ as an integer lattice by considering: $L = L\left(B, \{qx^j\}_{j \in \{0,\ldots,n-1\}}\right)$

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: How to intersect lattices efficently?

Optimized method: Using $\mathbb{F}_q$-subspaces.

Let $L_1 = L(B_1)$ and $L_2 = L(B_2)$ be two lattices, such that $L_1, L_2 \subset q\mathbb{Z}^n$

1. View $\overline{\mathrm{L}_1}, \overline{L_2}$ as $\mathbb{F}_q$-subspaces
2. Compute an intersection of subspaces: $\bar{L} = \overline{L_1} \cap \overline{L_2}$ and B a basis of $\bar{L}$.
3. View $\bar{L}$ as an integer lattice by considering: $L = L\left(B, \{qx^j\}_{j\in\{0,\dots,n-1\}}\right)$

Solution: No need to work in rationnal field. Better complexity.

1. Attack can be improved with more faults
2. No restriction on fault injection at the time of y generation

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: How to intersect lattices efficently?

Optimized method: Using $\mathbb{F}_q$-subspaces.

Let $L_1 = L(B_1)$ and $L_2 = L(B_2)$ be two lattices, such that $L_1, L_2 \subset q\mathbb{Z}^n$

1. View $\overline{\mathrm{L}_1}, \overline{L_2}$ as $\mathbb{F}_q$-subspaces
2. Compute an intersection of subspaces: $\overline{L} = \overline{L_1} \cap \overline{L_2}$ and B a basis of $\overline{L}$.
3. View $\overline{L}$ as an integer lattice by considering: $L = L\left(B, \{qx^j\}_{j \in \{0,\ldots,n-1\}}\right)$

Solution: No need to work in rationnal field. Better complexity.

1. Attack can be improved with more faults
2. No restriction on fault injection at the time of y generation

But how do you turn it into a passive attack?

THALES
Building a future we can all trust
REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: Considering affine lattices

To switch from a fault-based attack to a side channel attack, the attack must operate with a single coefficient.

But:

$$\dim\left(L_i = L\left(c_i^{-1} z_i^{[1]}, \{\,(c_i x)^j\,\}_{j\in\{0,\dots,d\}}\right)\right) = d + 2.$$

To have $d + 2 < n$, one needs $d \leq n - 3$.

The attack requires knowledge of 2 coefficients.

THALES
Building a future we can all trust
REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007
This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: Considering affine lattices

Easy fix: By considering affine lattices,

$$\forall i \in \{1, \dots, m\}, \qquad A_i = c_i^{-1} z_i^{[1]} + L\left(\{\, (c_i x)^j \,\}_{j \in \{0, \dots, d\}}\right) \;\; and \;\; A = \bigcap A_i$$

This time, $\dim(A_i) = d + 1$. We simply need to adapt the attack to the affine case:

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

A fault attack on ML-DSA: Considering affine lattices

Easy fix: By considering affine lattices,

$$\forall i \in \{1, \dots, m\}, \qquad A_i = c_i^{-1} z_i^{[1]} + L\left(\left\{ (c_i x)^j \right\}_{j \in \{0, \dots, d\}}\right) \ \ and \ \ A = \bigcap A_i$$

This time, $\dim(A_i) = d + 1$. We simply need to adapt the attack to the affine case:

$L_i = L\left(c_i^{-1} z_i^{[1]}, \left\{ (c_i x)^j \right\}_{j \in \{0, \dots, d\}}\right) \ \ and \ \ L = \bigcap L_i$	$A_i = c_i^{-1} z_i^{[1]} + L\left(\left\{ (c_i x)^j \right\}_{j \in \{0, \dots, d\}}\right) \ \ and \ \ A = \bigcap A_i$
Computing $\bar{L} = \bigcap \bar{L_i}$	Computing $\bar{A} = \bigcap \bar{A_i}$
Using LLL	Using Babai's NPA

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

THALES
Building a future we can all trust

Practical results

www.thalesgroup.com

A fault attack on ML-DSA: Results

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

Attack results with a single signature against ML-DSA-II

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

Attack results with m signatures against ML-DSA-II

- **The attack is applicable to ML-DSA and more effective with a few faults.**
- **The suggested countermeasure is not sufficient.**

- **If the attacker knows a single coefficient, he needs 160 signatures to find the secret key.**

The code is publicly available: [GitHub - AzevedoPaco/AttackML-DSA]

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.

THALES

Building a future we can all trust

Thank you

www.thalesgroup.com

References:

[EFGT17]: Thomas Espitau, Pierre-Alain Fouque, Benoit Gérard, Mehdi Tibouchi. Loop abort Faults on LatticeBased Fiat-Shamir & Hash'n Sign signatures. 23rd Conference on Selected Area In Cryptography, Aug 2016, Saint John's, Canada.

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of Thales © 2024 THALES. All rights reserved.