THALES
Building a future we can all trust

Finding a polytope:

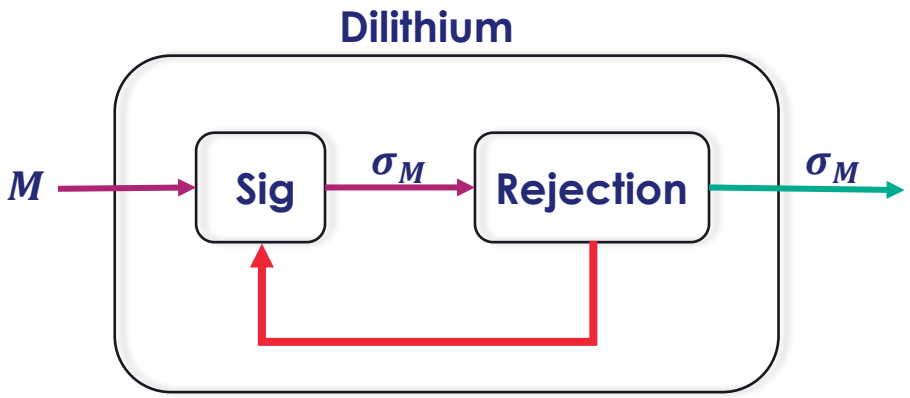
A practical fault attack against Dilithium

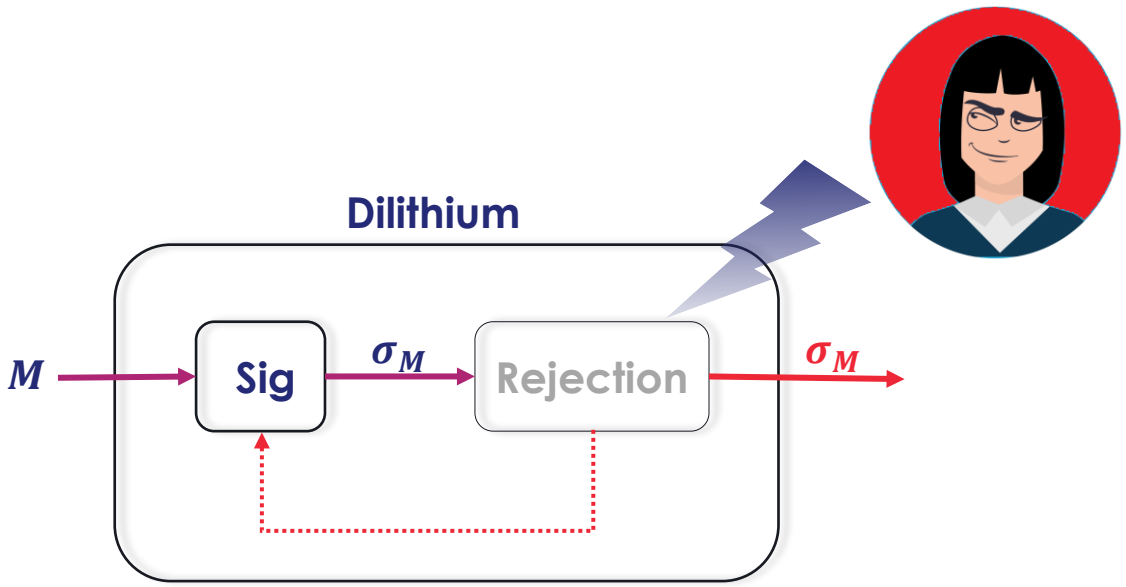
Paco Azevedo-Oliveira [1,2]
Andersson Calle Viera [1,3]
Benoît Cogliati [1]
Louis Goubin [2]

[1] Thales CDI, France
[2] UVSQ, France
[3] INRIA, LIP6, France

Table of contents

Conclusion

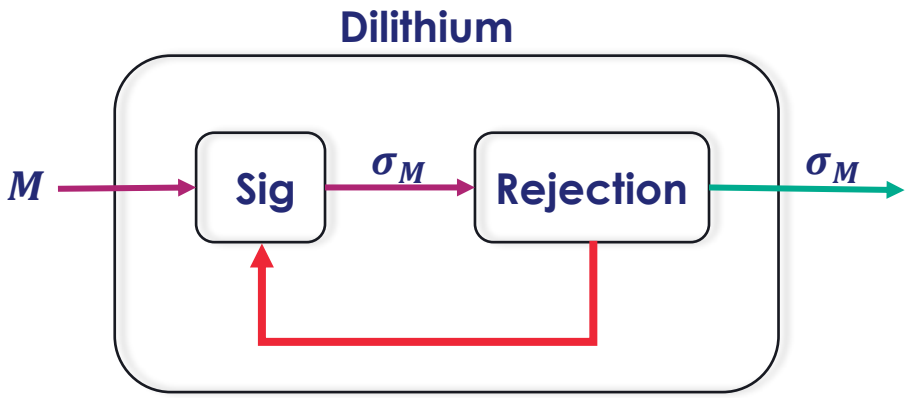


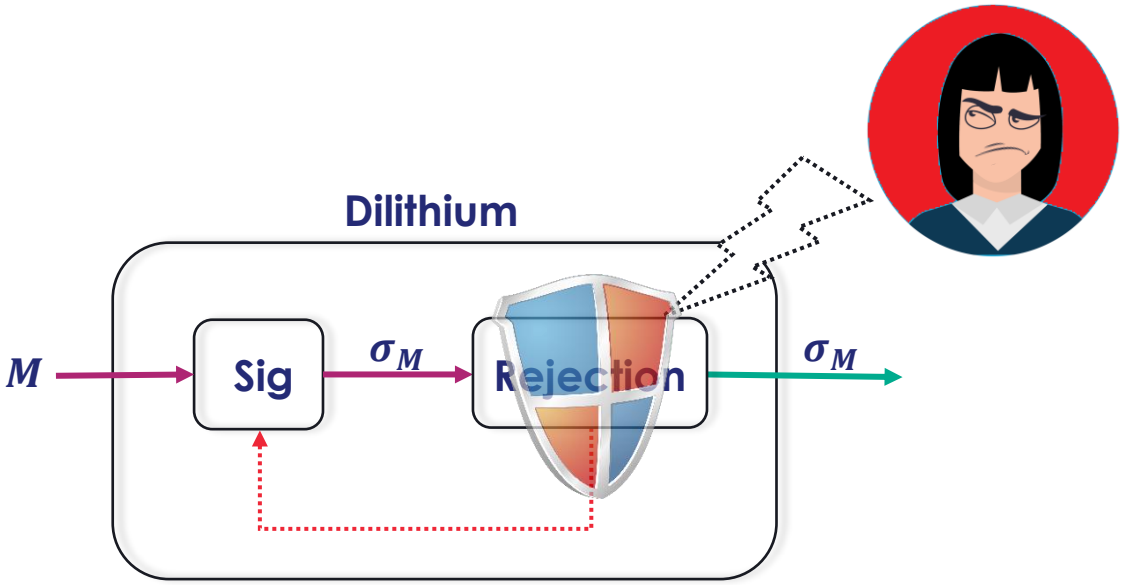


Dilithium is a signature algorithm recently standardized by NIST under the name ML-DSA.

In Dilithium, the rejection sampling step is crucial for the proof of security and correctness of the scheme. We show that an adversary with enough rejected signatures can recover Dilithium's secret key in less than half an hour on a computer.

Conclusion



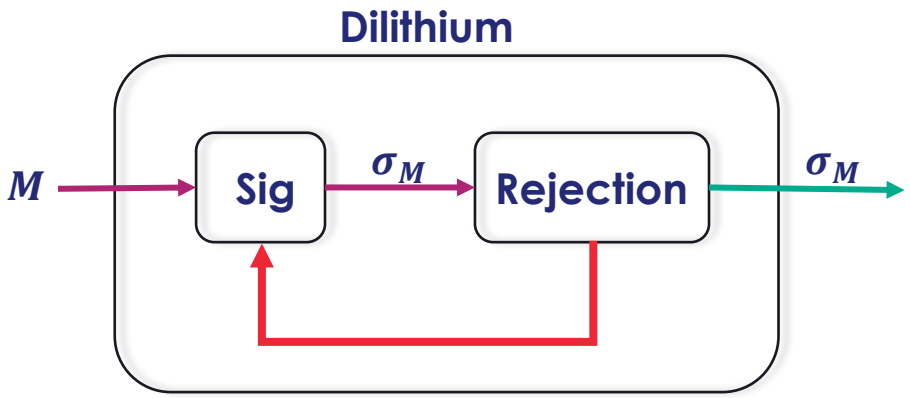


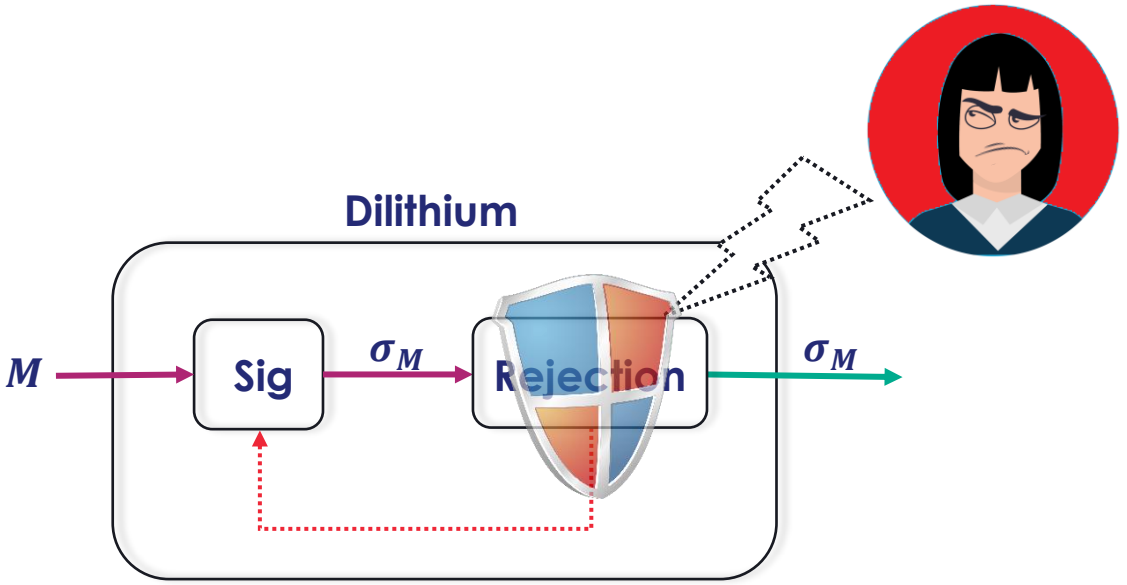
Dilithium is a signature algorithm recently standardized by NIST under the name ML-DSA.

In Dilithium, the rejection sampling step is crucial for the proof of security and correctness of the scheme. We show that an adversary with enough rejected signatures can recover Dilithium's secret key in less than half an hour on a computer.

Security	With rejection	Without rejection
Dilithium-II	2^{128}	2^{20}
Dilithium-III	2^{192}	2^{21}
Dilithium-V	2^{256}	2^{22}

Conclusion





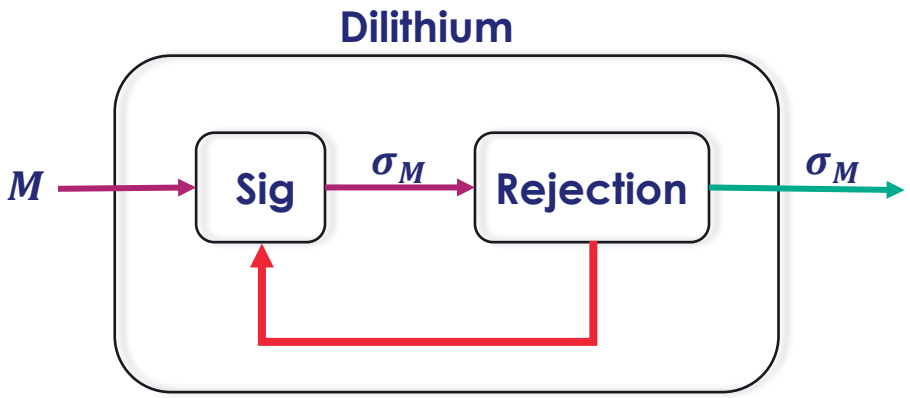
Dilithium is a signature algorithm recently standardized by NIST under the name ML-DSA.

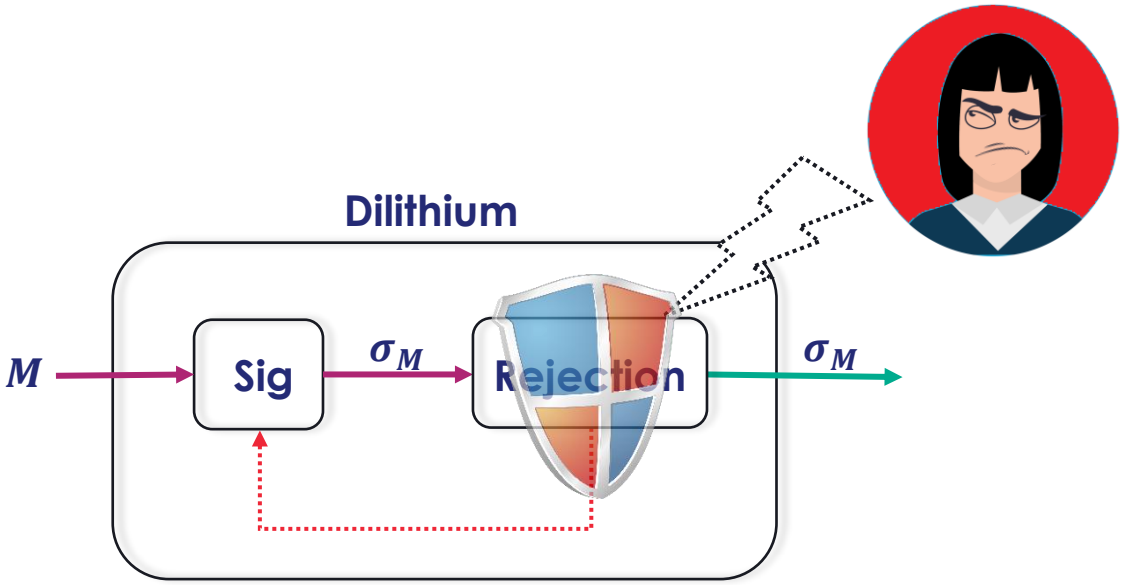
In Dilithium, the rejection sampling step is crucial for the proof of security and correctness of the scheme. We show that an adversary with enough rejected signatures can recover Dilithium's secret key in less than half an hour on a computer.

Security	With rejection	Without rejection
Dilithium-II	2^{128}	2^{20}
Dilithium-III	2^{192}	2^{21}
Dilithium-V	2^{256}	2^{22}

From our paper: the tests must be protected and not just the values manipulated during the test.

Conclusion





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

In Dilithium, the rejection sampling step is crucial for the proof of security and correctness of the scheme. We show that an adversary with enough rejected signatures can recover Dilithium's secret key in less than half an hour on a computer.

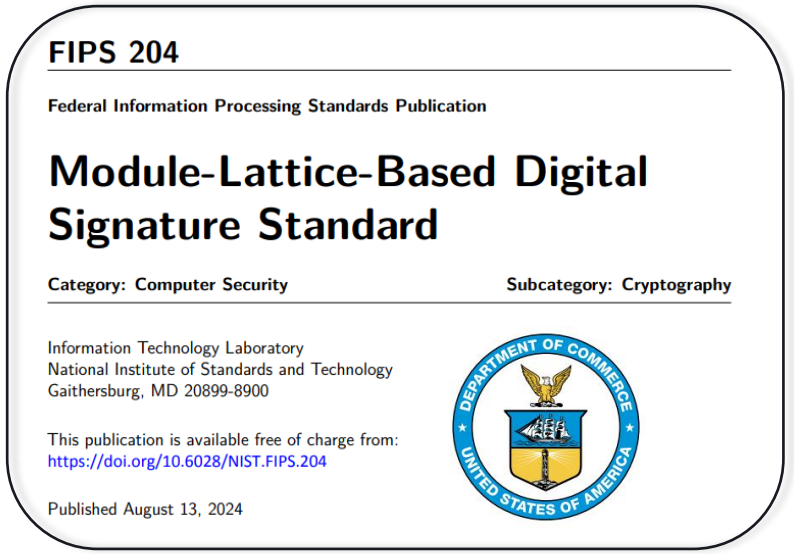
Security	With rejection	Without rejection
Dilithium-II	2^{128}	2^{20}
Dilithium-III	2^{192}	2^{21}
Dilithium-V	2^{256}	2^{22}

The code is publicly available: GitHub - anders1901/Polytope_attack

Context

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

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



FIPS 204

Federal Information Processing Standards Publication

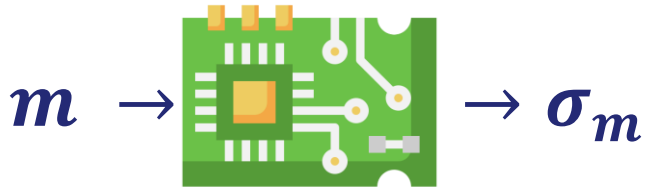
Module-Lattice-Based Digital Signature Standard

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



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

Dilithium in details

Dilithium uses two rings:

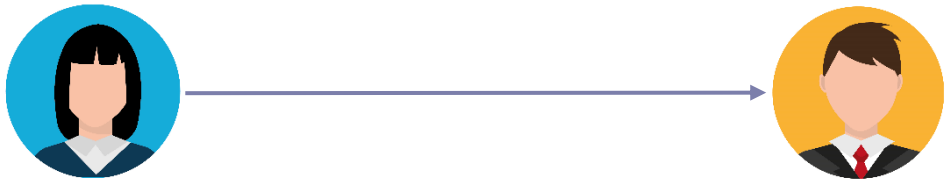
$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

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

Algorithm 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)

Dilithium in details

Dilithium uses two rings:

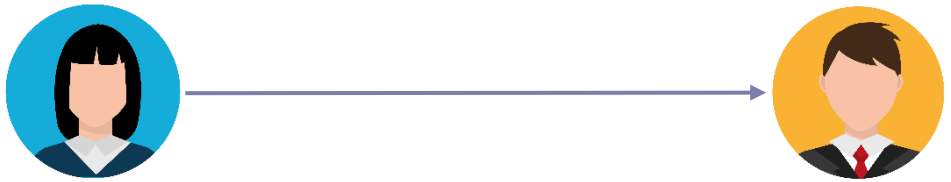
$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

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

Algorithm 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:

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

Possible values of r_0: $\left\{-\frac{\alpha}{2} + 1, \ldots, 0, \ldots, \frac{\alpha}{2}\right\}$

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

One note:

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

Dilithium in details

Dilithium uses two rings:

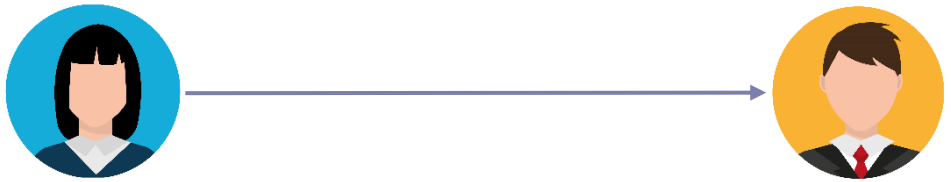
$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \qquad \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

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

Algorithm 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)

$$r = \mathbf{HighBits}_q(r, \alpha) \times \alpha + \mathbf{LowBits}_q(r, \alpha)$$

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

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

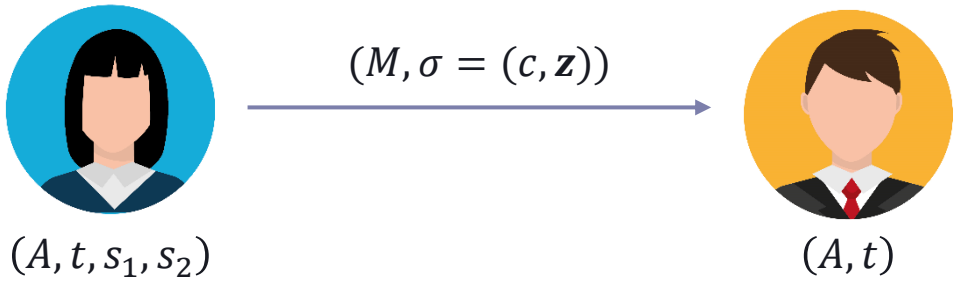
$$\texttt{HighBits}_q(P_1) := \sum \texttt{HighBits}_q(p_i) x^i$$

Dilithium in details

Algorithm Sig

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

1: $\mathbf{z} = \perp$
2: **while** $\mathbf{z} = \perp$ **do**
3: $\quad \mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$
4: $\quad \mathbf{w}_1 := \texttt{HighBits}(\mathbf{Ay}, 2\gamma_2)$
5: $\quad c \in B_\tau := H(M||\mathbf{w}_1)$
6: $\quad \mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
7: $\quad$ **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**
8: $\quad\quad \mathbf{z} := \perp$
9: $\quad$ **end if**
10: **end while**
11: **return** $\sigma = (c, \mathbf{z})$



Alice draws a polynomial vector at random:

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

She calculates a random challenge that depends on the message:

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

She provides an response to the challenge:

$$z = y + cs_1$$

By definition of z:

$$Az - ct = Ay - cs_2$$

The signature will be:

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

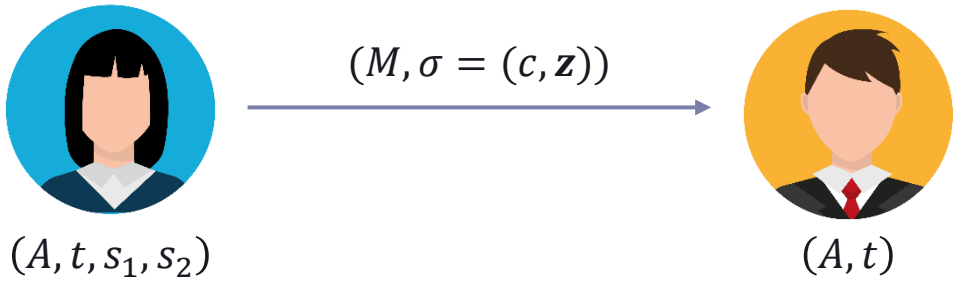
But..

Dilithium in details

Algorithm Sig

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

1: $\mathbf{z} = \perp$
2: **while** $\mathbf{z} = \perp$ **do**
3: $\quad \mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$
4: $\quad \mathbf{w}_1 := \texttt{HighBits}(\mathbf{Ay}, 2\gamma_2)$
5: $\quad c \in B_\tau := H(M||\mathbf{w}_1)$
6: $\quad \mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
7: $\quad$ **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**
8: $\quad\quad \mathbf{z} := \perp$
9: $\quad$ **end if**
10: **end while**
11: **return** $\sigma = (c, \mathbf{z})$



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:

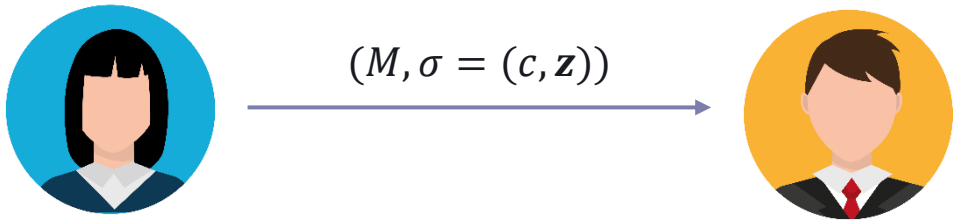
$$HighBits(Az - ct) = HighBits(Ay - cs_2) = HighBits(Ay)$$

Dilithium in details

Algorithm Sig

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

1: $\mathbf{z} = \perp$
2: **while** $\mathbf{z} = \perp$ **do**
3: $\quad \mathbf{y} \leftarrow \tilde{S}^{l}_{\gamma_1}$
4: $\quad \mathbf{w}_1 := \mathtt{HighBits}(\mathbf{Ay}, 2\gamma_2)$
5: $\quad c \in B_\tau := H(M||\mathbf{w}_1)$
6: $\quad \mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
7: $\quad$ **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**
8: $\quad\quad \mathbf{z} := \perp$
9: $\quad$ **end if**
10: **end while**
11: **return** $\sigma = (c, \mathbf{z})$



By definition of z:

$$Az - ct = Ay - cs_2$$

y is chosen such that: Rejection

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

Algorithm 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 can recalculate w_1:

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

Dilithium in details

Dilithium's version is simplified. The public key is compressed:

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

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

Algorithm 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 can only compute:

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

Dilithium in details

Dilithium's version is simplified. The public key is compressed:

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

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

Algorithm 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 can only compute:

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

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:*

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

The real Dilithium

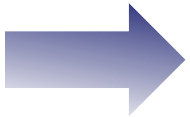
Lemma 1 *[LDK+22] Let q and α be two positive integers such that $q > 2\alpha$, $q \equiv 1 \mod (\alpha)$ and α even. Let $\mathbf{r}$ and $\mathbf{z}$ be two vectors of $\mathcal{R}_q$ such that $||\mathbf{z}||_\infty \leq \alpha/2$ and let $\mathbf{h}, \mathbf{h}'$ be bit vectors. So the algorithms $\texttt{HighBits}_q$, $\texttt{MakeHint}_q$, $\texttt{UseHint}_q$ 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 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 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** $pk = (\rho, \mathbf{t}_1)$, $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$

The real Dilithium

Algorithm Sig

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

1: $\mathbf{z} = \perp$
2: **while** $\mathbf{z} = \perp$ **do**
3: $\quad \mathbf{y} \leftarrow \tilde{S}^l_{\gamma_1}$
4: $\quad \mathbf{w}_1 := \texttt{HighBits}(\mathbf{Ay}, 2\gamma_2)$
5: $\quad c \in B_\tau := H(M||\mathbf{w}_1)$
6: $\quad \mathbf{z} := \mathbf{y} + c\,\mathbf{s}_1$
7: $\quad$ **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**
8: $\quad\quad \mathbf{z} := \perp$
9: $\quad$ **end if**
10: **end while**
11: **return** $\sigma = (c, \mathbf{z})$

Algorithm Sig

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

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

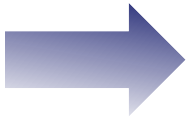
The real Dilithium

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

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** $c = H(M||\mathbf{w}'_1)$



Algorithm 4 Ver

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

THALES
Building a future we can all trust

Attack method

www.thalesgroup.com

Attack method: First step

Algorithm F-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 $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **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 $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Algorithm Ver

Require: pk, σ

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

$$\underbrace{\boldsymbol{HighBits_q(Ay, 2\gamma_2)}}_{\boldsymbol{w_1}} \stackrel{?}{=} \underbrace{\boldsymbol{HighBits_q(Ay - cs_2, 2\gamma_2)}}_{\boldsymbol{w'_1}}$$

Attack method: First step

Algorithm F-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 $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **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 $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Algorithm Ver

Require: pk, σ

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

$$\underbrace{\boldsymbol{HighBits_q(Ay, 2\gamma_2)}}_{\boldsymbol{w_1}} \stackrel{?}{=} \underbrace{\boldsymbol{HighBits_q(Ay - cs_2, 2\gamma_2)}}_{\boldsymbol{w'_1}}$$

Assumption 1: With overwhelming probability, for a signature of F-Sig the polynomial vector $w_1 - w_1'$ has at most one non-zero coefficient.

Proposition: Under **Assumption 1**, it is possible to recover w_1 from the knowledge of w_1'.

Attack method: First step

Algorithm F-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 $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **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 $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Algorithm 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 $[\![|\mathbf{h}|_{\mathbf{h}_j=1} \leq \omega]\!]$

Assumption 1: With overwhelming probability, for a signature of F-Sig the polynomial vector $w_1 - w_1'$ has at most one non-zero coefficient.

Proposition: Under **Assumption 1**, it is possible to recover w_1 from the knowledge of w_1'.

Proof: If the signature is rejected, one can carry an exhaustive research, knowing the relation:

$$c = H(\mu || w_1)$$

Attack method: First step

Algorithm F-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 $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **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 $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Algorithm Ver

Require: pk, σ

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

Assumption 1: With overwhelming probability, for a signature of F-Sig the polynomial vector $w_1 - w_1'$ has at most one non-zero coefficient.

Proposition: Under **Assumption 1**, it is possible to recover w_1 from the knowledge of w_1'.

Attack method: First step

Algorithm F-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 $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **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 $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Algorithm 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 $[\![|\mathbf{h}|_{\mathbf{h}_j=1} \leq \omega]\!]$

Assumption 1: With overwhelming probability, for a signature of F-Sig the polynomial vector $w_1 - w_1'$ has at most one non-zero coefficient.

Proposition: Under **Assumption 1**, it is possible to recover w_1 from the knowledge of w_1'.

Remark: If the hypothesis is not verified, simply ignore the signature

Attack method: Second step

Algorithm F-Sig

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

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

Each signature not accepted by the verification algorithm provide an inequality:

Proposition *For any $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ signature of* F-Sig *that is not accepted by the verification algorithm, there exists a unique $j \in \{1, ..., k\}$ and a unique $i \in \{0, ..., 255\}$ such that:*

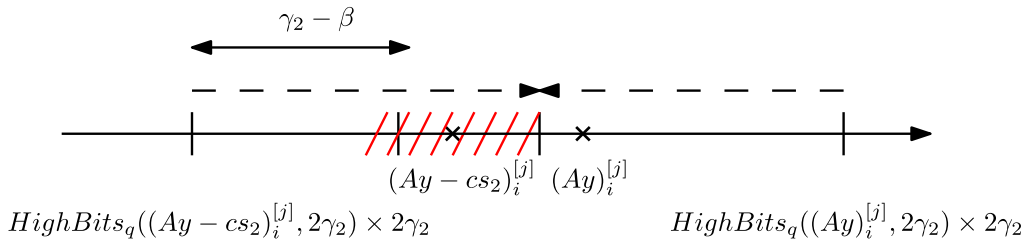
- *if* $(\mathbf{w}_1 - \mathbf{w}_1')_i^{[j]} = 1$*:*

$$(c\mathbf{s}_2)_i^{[j]} \geq \gamma_2 - \mathbf{r}_{0,i}^{[j]} \geq 0,$$

- *if* $(\mathbf{w}_1 - \mathbf{w}_1')_i^{[j]} = -1$*:*

$$(c\mathbf{s}_2)_i^{[j]} \leq -\gamma_2 - \mathbf{r}_{0,i}^{[j]} \leq 0.$$

Proof (sketch):

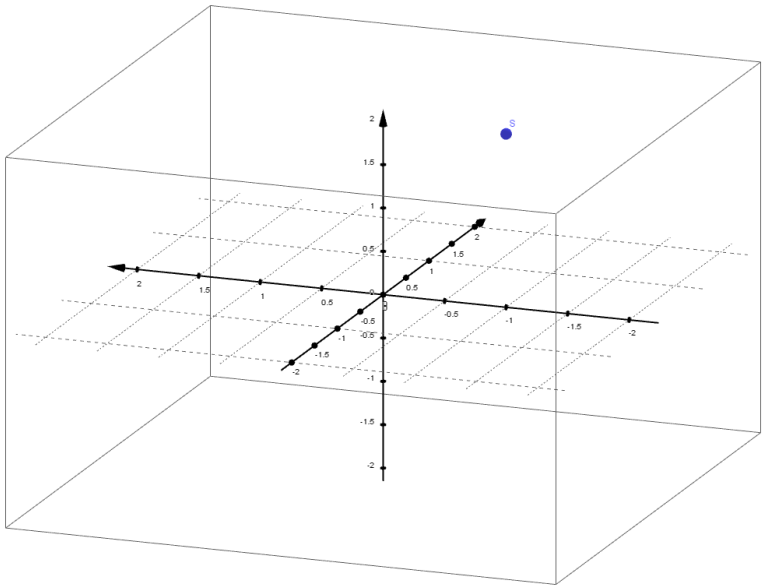


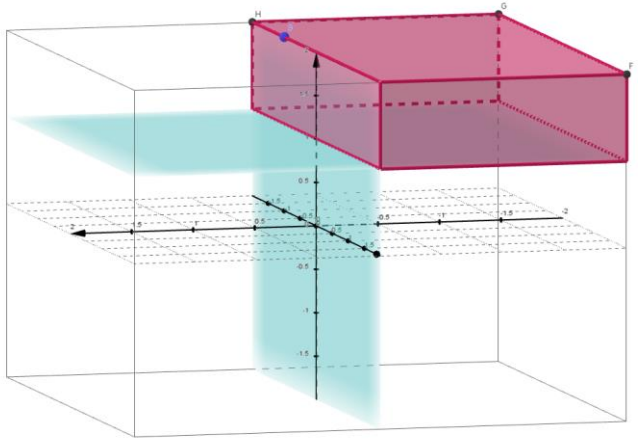
Attack method: Second step

Example for n=3 :

Let's assume that all the key coefficients are known, apart from $(s_2)_0, (s_2)_1, (s_2)_2$. The unknowns are the coordinates of a point in $[-2,2]^3 \cap Z$

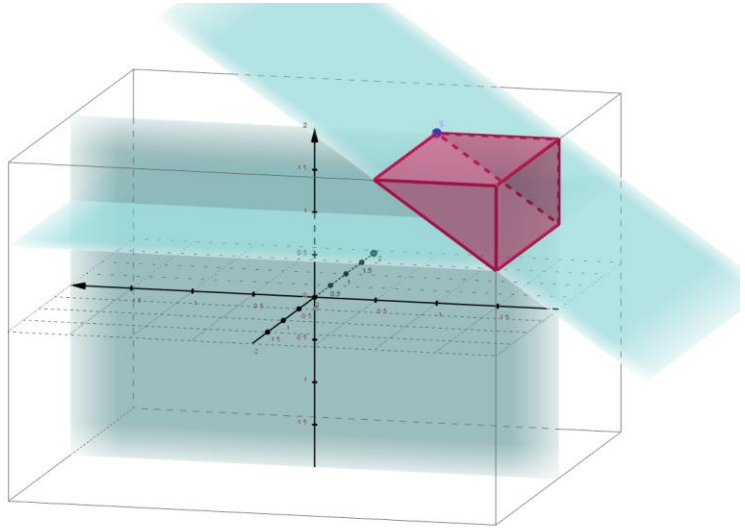
For example for $s_2 = (2,0,1,\ldots)$: signing several times with the same key will produce inequalities.





Solution of inequalities :

$$x \leq 0$$
$$y \geq 1$$

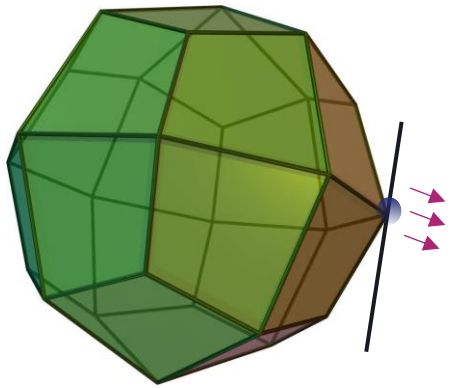


Solution of inequalities:

$$x \leq 0$$
$$y \geq 1$$
$$y - z \geq -3$$

Attack method: Second step

We use linear programming (LP) methods.



Upper bound the number of solutions of is a (LP) problem:

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

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

If inequalities are collected so that s_2 is the only solution, it suffices to maximize any function:

$$\begin{aligned} &\text{maximize} && 0 \\ &\text{subject to} && A_+ x \geq b_+ \\ & && A_- x \leq b_- \\ & && x \in [-\eta, \eta]^n \end{aligned}$$

THALES
Building a future we can all trust

Practical results

www.thalesgroup.com

Practical results

The number of inequalities required can be estimated 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	-

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

Conclusion : The attack illustrates the power of LP methods: we search for a point in $[-2, 2]^{256}$ is found under one hour:

Tests must be protected against faults.

THALES
Building a future we can all trust

Hidden problems

www.thalesgroup.com

Hidden problems: Implementation VS Specification

Algorithm Sig

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

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

$\texttt{HighBits}_q(r, \alpha)$:
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

The “specification” version calls the decompose function twice, which is costly.

This can be avoided by making an equivalent and less costly test, which uses a little more memory.

Hidden problems: Implementation VS Specification

Algorithm $\overline{\text{Sig}}_{\text{Ref}}$

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

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

$\texttt{HighBits}_q(r, \alpha)$:
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

The test is equivalent and saves a call to the decompose function, with the cost of storing w_0 in memory.

Hidden problems: Implementation VS Specification

Algorithm $\overline{\text{Sig}}_{\text{Ref}}$

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

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

$\texttt{HighBits}_q(r, \alpha)$:
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

The test is equivalent and saves a call to the decompose function, with the cost of storing w_0 in memory.

These versions are no longer equivalent without the second test!

Hidden problems: Implementation VS Specification

For Specification:

Assumption 1: With overwhelming probability, for a signature of $F - Sig$ the polynomial vector $w_1 - w_1{}'$ has at most one non-zero coefficient.

Proposition *For any $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ signature of* `F-Sig` *that is not accepted by the verification algorithm, there exists a unique $j \in \{1, ..., k\}$ and a unique $i \in \{0, ..., 255\}$ such that:*

- *if* $(\mathbf{w}_1 - \mathbf{w}'_1)_i^{[j]} = 1$*:*

$$(c\mathbf{s}_2)_i^{[j]} \geq \gamma_2 - \mathbf{r}_{0,i}^{[j]} \geq 0,$$

- *if* $(\mathbf{w}_1 - \mathbf{w}'_1)_i^{[j]} = -1$*:*

$$(c\mathbf{s}_2)_i^{[j]} \leq -\gamma_2 - \mathbf{r}_{0,i}^{[j]} \leq 0.$$

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

Hidden problems: Implementation VS Specification

For Specification:

Assumption 1: With overwhelming probability, for a signature of $F - Sig$ the polynomial vector $w_1 - w_1'$ has at most one non-zero coefficient.

Proposition *For any $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ signature of* `F-Sig` *that is not accepted by the verification algorithm, there exists a unique $j \in \{1, ..., k\}$ and a unique $i \in \{0, ..., 255\}$ such that:*

- *if* $(\mathbf{w}_1 - \mathbf{w}'_1)_i^{[j]} = 1$*:*

$$(c\mathbf{s}_2)_i^{[j]} \geq \gamma_2 - \mathbf{r}_{0,i}^{[j]} \geq 0,$$

- *if* $(\mathbf{w}_1 - \mathbf{w}'_1)_i^{[j]} = -1$*:*

$$(c\mathbf{s}_2)_i^{[j]} \leq -\gamma_2 - \mathbf{r}_{0,i}^{[j]} \leq 0.$$

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

For Implementation:

Assumption 2: The signature made by $F - Sig_{REF}$ will always be accepted by the verification algorithm.

Proposition *Under Assumption 2, let $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ be a signature of* $\texttt{F-Sig}_{\texttt{Ref}}$*, then either* $\mathbf{w}_1 = \texttt{HighBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$ *or there exists at least one $j \in \{1, ..., k\}$ and at least one $i \in \{0, ..., 255\}$ such that:*

- *if* $(\mathbf{w}'_1 - \texttt{HighBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2))_i^{[j]}$ *is positive:*

$$(c\mathbf{s}_2)_i^{[j]} \geq \gamma_2 - \boldsymbol{r}_{0,i}^{[j]} \geq 0,$$

- *if* $(\mathbf{w}'_1 - \texttt{HighBits}_q(\mathbf{Az} - c\mathbf{t}, 2\gamma_2))_i^{[j]}$ *is negative:*

$$(c\mathbf{s}_2)_i^{[j]} \leq -\gamma_2 - \boldsymbol{r}_{0,i}^{[j]} \leq 0.$$

Signatures	Average inequalities	Sucess probability	Average time	Median time
1 250 000	11 083	0.98	261.79s	148.79s

Hidden problems: Second conclusion

Algorithm $\mathtt{Ver}$

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** $c = H(M||\mathbf{w}_1')$

Algorithm $\mathtt{Ver}$

Require: pk, σ

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

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

3: $c := \mathtt{SampleInBall}(\tilde{c})$

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

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

Hidden problems: Second conclusion

Algorithm 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** $c = H(M||\mathbf{w}_1')$

Algorithm Ver

Require: pk, σ

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

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

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

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

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

In a fault framework: Verification in the implementation is weaker than when t_0 is known.

Incorrect Dilithium signatures, which provide information about the secret key, are considered valid by the Dilithium reference verification.

THALES
Building a future we can all trust

Thank you

www.thalesgroup.com