# Implementations of Post-Quantum Cryptography Algorithms Secured Against Physical Attacks

CALLE VIERA Andersson
Director : VERGNAUD Damien
Supervisor: BERZATI Alexandre
Almasty Workshop 2024, 12 Jul. 2024

[1] Thales DIS, France
[2] Sorbonne Université, France

# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

**Importance:** These algorithms will be implemented **securely** in a variety of use cases

# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

**Importance:** These algorithms will be implemented **securely** in a variety of use cases

Study PQC

# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

**Importance:** These algorithms will be implemented **securely** in a variety of use cases

# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

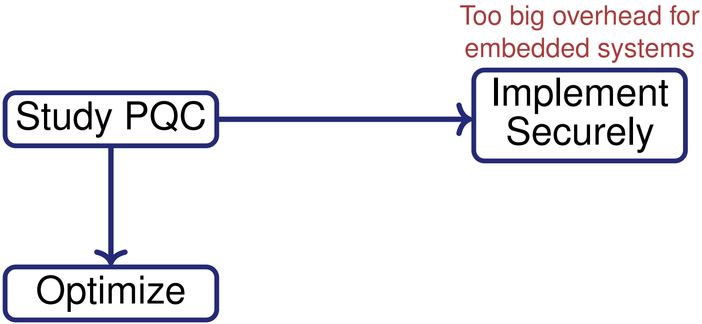
**Importance:** These algorithms will be implemented **securely** in a variety of use cases

# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

**Importance:** These algorithms will be implemented **securely** in a variety of use cases

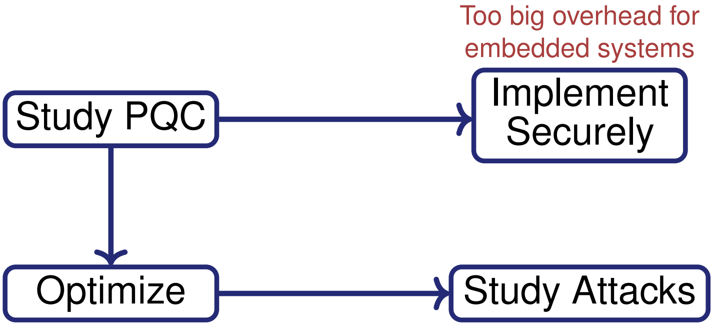


# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

**Importance:** These algorithms will be implemented **securely** in a variety of use cases

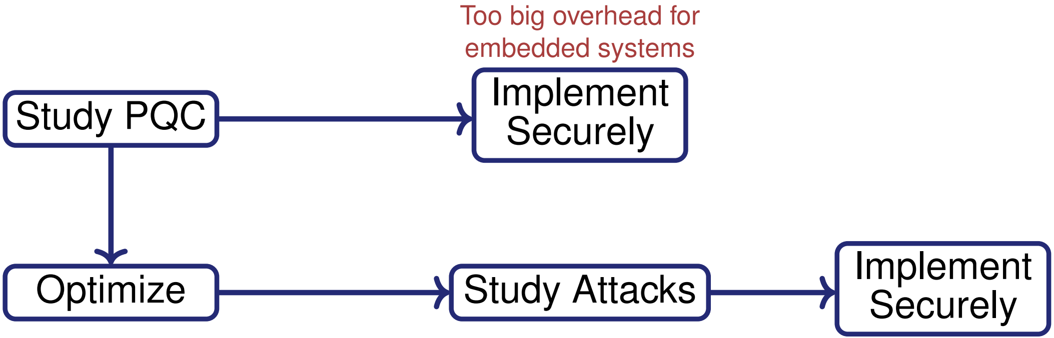


# Context

NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

**Importance:** These algorithms will be implemented **securely** in a variety of use cases

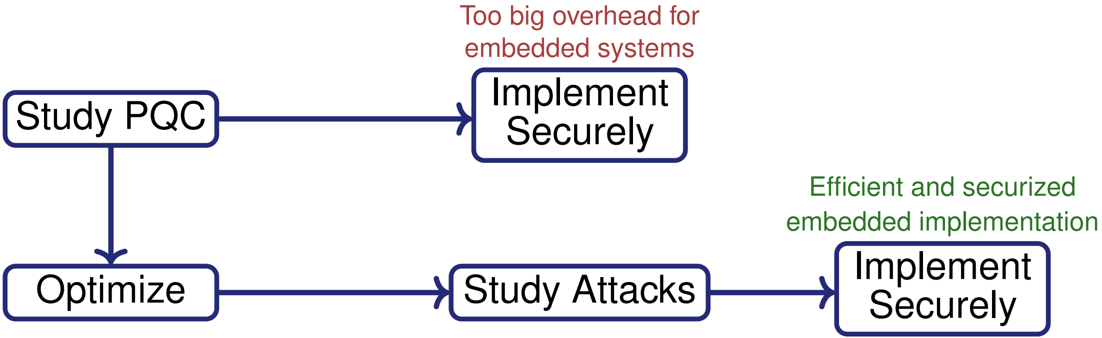


# Context

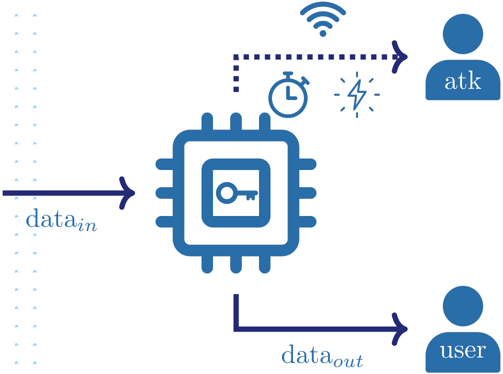
NIST: National Institute of Standards and Technology

- 2024: First KEM (**Kyber**) and DSA (**Dilithium**/Falcon/SPHINCS+) standards finalized

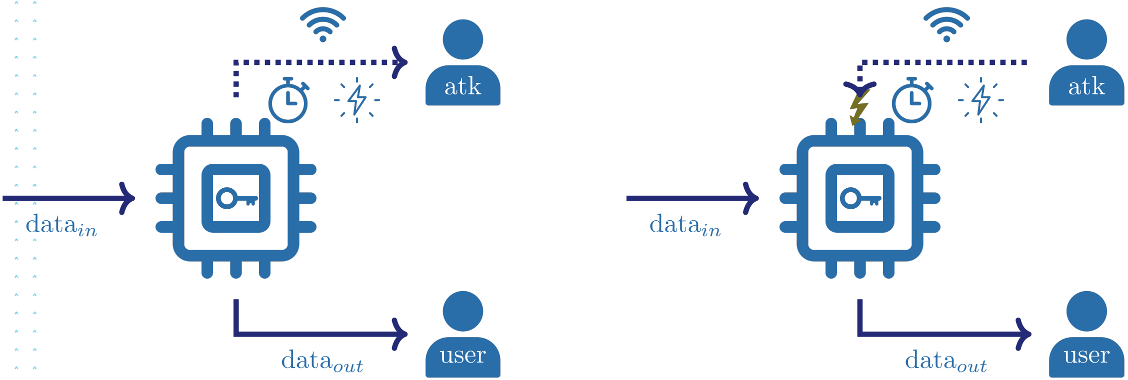
**Importance:** These algorithms will be implemented **securely** in a variety of use cases



# Side Channel and Fault Attacks



# Side Channel and Fault Attacks



# Dilithium

- Public key signature algorithm, based on hard problems on Lattices

M-LWE

M-SIS

- Easy to implement and secret-independent execution time

# Dilithium

- Public key signature algorithm, based on hard problems on Lattices
  - M-LWE
  - M-SIS
- Easy to implement and secret-independent execution time
- Three security levels: Dilithium-2, Dilithium-3, Dilithium-5
- Two versions: deterministic and hedged (randomized)

# Dilithium

- Public key signature algorithm, based on hard problems on Lattices — M-LWE, M-SIS
- Easy to implement and secret-independent execution time
- Three security levels: Dilithium-2, Dilithium-3, Dilithium-5
- Two versions: deterministic and hedged (randomized)
- Quotient Ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ where $n = 2^8$ and $q = 2^{23} - 2^{13} + 1$
  - Most of the time we work with vectors of $k$ or $l$ elements in $\mathcal{R}_q$
  - Polynomial multiplication using the Number Theoretic Transform (NTT)

# KeyGen:

1 $A \in \mathcal{R}_q^{k \times l}$

2 $(s_1, s_2) \in S_\eta^l \times S_\eta^k$

3 $t = A s_1 + s_2 \in \mathcal{R}_q^k$

4 $(t_1, t_0) = \texttt{Power2Round}(t, d)$

5 return $pk = (A, t_1), sk = (A, s_1, s_2, t_0, pk)$

# KeyGen:

1 $A \in \mathcal{R}_q^{k\times l}$

2 $(s_1, s_2) \in S_\eta^l \times S_\eta^k$

3 $t = A s_1 + s_2 \in \mathcal{R}_q^k$

4 $(t_1, t_0) = \texttt{Power2Round}(t, d)$

5 return $pk = (A, t_1), sk = (A, s_1, s_2, t_0, pk)$

# KeyGen:

1 $A \in \mathcal{R}_q^{k \times l}$

2 $(s_1, s_2) \in S_\eta^l \times S_\eta^k$

3 $t = A\, s_1 + s_2 \in \mathcal{R}_q^k$

4 $(t_1, t_0) = \texttt{Power2Round}(t, d)$

5 return $pk = (A, t_1)$, $sk = (A, s_1, s_2, t_0, pk)$

# KeyGen:

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

$(s_1, s_2) \in S_\eta^l \times S_\eta^k$

$t = A\, s_1 + s_2 \in \mathcal{R}_q^k$

$(t_1, t_0) = \texttt{Power2Round}(t, d)$

return $pk = (A, t_1)$, $sk = (A, s_1, s_2, t_0, pk)$

| | | | | |
|---|---|---|---|---|
| $t_{0,0}$ | $t_{0,1}$ | $\cdots$ | $t_{0,n-2}$ | $t_{0,n-1}$ |
| $t_{1,0}$ | $t_{1,1}$ | $\cdots$ | $t_{1,n-2}$ | $t_{1,n-1}$ |
| | | $\cdots$ | | |
| $t_{k-2,0}$ | $t_{k-2,1}$ | $\cdots$ | $t_{k-2,n-2}$ | $t_{k-2,n-1}$ |
| $t_{k-1,0}$ | $t_{k-1,1}$ | $\cdots$ | $t_{k-1,n-2}$ | $t_{k-1,n-1}$ |

# KeyGen:

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

$(s_1, s_2) \in S_\eta^l \times S_\eta^k$

$t = A\, s_1 + s_2 \in \mathcal{R}_q^k$

$(t_1, t_0) = \texttt{Power2Round}(t, d)$

return $pk = (A, t_1)$, $sk = (A, s_1, s_2, t_0, pk)$

| | | | | |
|---|---|---|---|---|
| $t_{0,0}$ | $t_{0,1}$ | $\cdots$ | $t_{0,n-2}$ | $t_{0,n-1}$ |
| $t_{1,0}$ | $t_{1,1}$ | $\cdots$ | $t_{1,n-2}$ | $t_{1,n-1}$ |
| | | $\cdots$ | | |
| $t_{k-2,0}$ | $t_{k-2,1}$ | $\cdots$ | $t_{k-2,n-2}$ | $t_{k-2,n-1}$ |
| $t_{k-1,0}$ | $t_{k-1,1}$ | $\cdots$ | $t_{k-1,n-2}$ | $t_{k-1,n-1}$ |

# KeyGen:

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

$(s_1, s_2) \in S_\eta^l \times S_\eta^k$

$t = A\, s_1 + s_2 \in \mathcal{R}_q^k$

$(t_1, t_0) = \texttt{Power2Round}(t, d)$

return $pk = (A, t_1), sk = (A, s_1, s_2, t_0, pk)$

| | | | | |
|---|---|---|---|---|
| $t_{0,0}$ | $t_{0,1}$ | $\cdots$ | $t_{0,n-2}$ | $t_{0,n-1}$ |
| $t_{1,0}$ | $t_{1,1}$ | $\cdots$ | $t_{1,n-2}$ | $t_{1,n-1}$ |
| | | $\cdots$ | | |
| $t_{k-2,0}$ | $t_{k-2,1}$ | $\cdots$ | $t_{k-2,n-2}$ | $t_{k-2,n-1}$ |
| $t_{k-1,0}$ | $t_{k-1,1}$ | $\cdots$ | $t_{k-1,n-2}$ | $t_{k-1,n-1}$ |

# KeyGen:

1. $A \in \mathcal{R}_q^{k \times l}$
2. $(s_1, s_2) \in S_\eta^l \times S_\eta^k$
3. $t = A\, s_1 + s_2 \in \mathcal{R}_q^k$
4. $(t_1, t_0) = \texttt{Power2Round}(t, d)$
5. return $pk = (A, t_1)$, $sk = (A, s_1, s_2, t_0, pk)$

| | | | | |
|---|---|---|---|---|
| $t_{0,0}$ | $\boxed{t_{0,1}}$ | $\cdots$ | $t_{0,n-2}$ | $t_{0,n-1}$ |
| $t_{1,0}$ | $t_{1,1}$ | $\cdots$ | $t_{1,n-2}$ | $t_{1,n-1}$ |
| | | $\cdots$ | | |
| $t_{k-2,0}$ | $t_{k-2,1}$ | $\cdots$ | $t_{k-2,n-2}$ | $t_{k-2,n-1}$ |
| $t_{k-1,0}$ | $t_{k-1,1}$ | $\cdots$ | $t_{k-1,n-2}$ | $t_{k-1,n-1}$ |

# Sign($M$, $sk = (A,\ s_1,\ s_2,\ t_0,\ pk)$):

$(z, h) = \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}^l_{\gamma_1}$
    $w = A\, y$
    $w_1 = \texttt{HighBits}(w)$
    $c \in B_\tau = \mathrm{H}(pk\,||\,M\,||\,w_1)$
    $z = y + c\, s_1$
    $r_0 = \texttt{LowBits}(w - c\, s_2)$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) = \perp$
    else
        $h = \texttt{MakeHint}(-c\, t_0,\ w - c\, s_2 + c\, t_0)$
        if $\|c\, t_0\|_\infty \geq \gamma_2$, then $(z, h) = \perp$
return $\sigma = (c,\ z,\ h)$

# Sign($M$, $sk = (A,\ s_1,\ s_2,\ t_0,\ pk)$):

$(z, h) = \perp$

while $(z, h) = \perp$ do

  $y \in \tilde{S}_{\gamma_1}^{l}$

  $w = A\, y$

  $w_1 = \texttt{HighBits}(w)$

  $c \in B_\tau = \mathtt{H}(pk \,||\, M \,||\, w_1)$

  $z = y + c\, s_1$

  $r_0 = \texttt{LowBits}(w - c\, s_2)$

  if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) = \perp$

  else

    $h = \texttt{MakeHint}(-c\, t_0,\ w - c\, s_2 + c\, t_0)$

    if $||c\, t_0||_\infty \geq \gamma_2$, then $(z, h) = \perp$

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

# Sign($M$, $sk = (A, s_1, s_2, t_0, pk)$):

$(z, h) = \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}^{l}_{\gamma_1}$
    $w = A\,y$
    $w_1 = \texttt{HighBits}(w)$
    $c \in B_\tau = \texttt{H}(pk \,||\, M \,||\, w_1)$
    $z = y + c\, s_1$
    $r_0 = \texttt{LowBits}(w - c\, s_2)$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) = \perp$
    else
        $h = \texttt{MakeHint}(-c\, t_0,\, w - c\, s_2 + c\, t_0)$
        if $||c\, t_0||_\infty \geq \gamma_2$, then $(z, h) = \perp$
return $\sigma = (c, z, h)$

# Sign($M$, $sk = (A,\ s_1,\ s_2,\ t_0,\ pk)$):

```
(z, h) = ⊥
while (z, h) = ⊥ do
    y ∈ S̃^l_{γ1}
    w = A y
    w1 = HighBits(w)
    c ∈ B_τ = H(pk || M || w1)
    z = y + c s1
    r0 = LowBits(w − c s2)
    if ||z||_∞ ≥ γ1 − β or ||r0||_∞ ≥ γ2 − β, then (z, h) = ⊥
    else
        h = MakeHint(−c t0, w − c s2 + c t0)
        if ||c t0||_∞ ≥ γ2, then (z, h) = ⊥
return σ = (c, z, h)
```

# Sign($M$, $sk = (A,\ s_1,\ s_2,\ t_0,\ pk)$):

$(z, h) = \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}^l_{\gamma_1}$
    $w = A\,y$
    $w_1 = \texttt{HighBits}(w)$
    $c \in B_\tau = \texttt{H}(pk \,||\, M \,||\, w_1)$
    $z = y + c\,s_1$
    $r_0 = \texttt{LowBits}(w - c\,s_2)$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) = \perp$
    else
        $h = \texttt{MakeHint}(-c\,t_0,\ w - c\,s_2 + c\,t_0)$
        if $\|c\,t_0\|_\infty \geq \gamma_2$, then $(z, h) = \perp$
return $\sigma = (c,\ z,\ h)$

# Verify($pk = (A, t_1)$, $M$, $\sigma = (c, z, h)$):

1 $w_1' = \texttt{UseHint}(h, A\,z - c\,t_1 2^d)$

# Verify($pk = (A, t_1)$, $M$, $\sigma = (c, z, h)$):

$A\,z - c\,t_1 2^d$

1 $w_1' = \texttt{UseHint}(h, \boxed{A\,z - c\,t_1 2^d})$

# Verify($pk = (A, t_1)$, $M$, $\sigma = (c, z, h)$):

$$A\,z - c\,t_1 2^d = A\,\overbrace{(y + c\,s_1)}^{z} - c\,\overbrace{(A\,s_1 + s_2 - t_0)}^{t_1 2^d}$$

1 $w_1' = \texttt{UseHint}(h, \boxed{A\,z - c\,t_1 2^d})$

# Verify($pk = (A, t_1), M, \sigma = (c, z, h)$):

$$
\begin{aligned}
A\,z - c\,t_1 2^d &= A\,\overbrace{(y + c\,s_1)}^{z} - c\,\overbrace{(A\,s_1 + s_2 - t_0)}^{t_1 2^d} \\
&= \underbrace{A\,y}_{w} - cs_2 + ct_0 \\
&= w - cs_2 + ct_0
\end{aligned}
$$

1 $w_1' = \texttt{UseHint}(h, \boxed{A\,z - c\,t_1 2^d})$

# Verify($pk = (A, t_1)$, $M$, $\sigma = (c, z, h)$):

$$\begin{aligned} A\,z - c\,t_1 2^d &= A\,\overbrace{(y + c\,s_1)}^{z} - c\,\overbrace{(A\,s_1 + s_2 - t_0)}^{t_1 2^d} \\ &= \underbrace{A\,y}_{} - c s_2 + c t_0 \\ &= w - c s_2 + c t_0 \end{aligned}$$

Lemma 1.1 [1] $\Longrightarrow \texttt{UseHint}(h, \; w - c s_2 + c\,t_0) = \texttt{HighBits}(w - c s_2)$

1 $w_1' = \texttt{UseHint}(h, \boxed{A\,z - c\,t_1 2^d})$

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé,
CRYSTALS - Dilithium: Digital Signatures from Module Lattices

# Verify($pk = (A, t_1), M, \sigma = (c, z, h)$):

$$A z - c\, t_1 2^d = A \overbrace{(y + c\, s_1)}^{z} - c \overbrace{(A\, s_1 + s_2 - t_0)}^{t_1 2^d}$$
$$= \underbrace{A\, y}_{} - c s_2 + c t_0$$
$$= w - c s_2 + c t_0$$

Lemma 1.1 [1] $\implies$ $\texttt{UseHint}(h, w - c s_2 + c\, t_0) = \texttt{HighBits}(w - c s_2)$

Lemma 2 [1] $\implies$ $\texttt{HighBits}(w - c s_2) = \underbrace{\texttt{HighBits}_q}_{}(w)$
$$= w_1$$

1 $w_1' = \texttt{UseHint}(h, \boxed{A\, z - c\, t_1 2^d})$

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

# Verify$(pk = (A, t_1), M, \sigma = (c, z, h))$:

$$
\begin{aligned}
A\,z - c\,t_1 2^d &= A\,\overbrace{(y + c\,s_1)}^{z} - c\,\overbrace{(A\,s_1 + s_2 - t_0)}^{t_1 2^d} \\
&= \underbrace{A\,y}_{} - c s_2 + c t_0 \\
&= w - c s_2 + c t_0
\end{aligned}
$$

Lemma 1.1 [1] $\implies \texttt{UseHint}(h, w - c s_2 + c\,t_0) = \texttt{HighBits}(w - c s_2)$

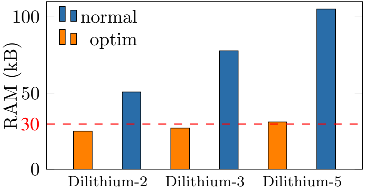
Lemma 2 [1] $\implies \texttt{HighBits}(w - c s_2) = \underbrace{\texttt{HighBits}_q}_{}(w) = w_1$

1 $w'_1 = \texttt{UseHint}(h, \boxed{A\,z - c\,t_1 2^d})$
2 if $||z||_\infty < \gamma_1 - \beta$ and $c = \mathtt{H}(pk\,||\,M\,||\,w'_1)$ and $\#\,1$'s in $h \leq \omega$
3 return $\mathit{True}$
4 else
5 return $\mathit{False}$

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

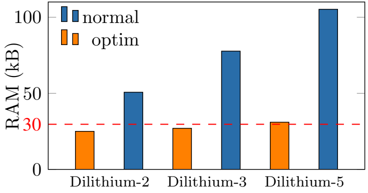
# Optimizing Dilithium Signature Scheme

- Key size larger than secure element RAM size ($\approx 30$kB)
- A lot of RAM consumption for the $3$ security levels of Dilithium
  - Each polynomial is $256 \times 4$ bytes, so $1$kB/polynomial
  - Matrix $A$ of $k \times l$ polynomials with $k$ and $l$ up to $8$ and $7$!



# Optimizing Dilithium Signature Scheme

- Key size larger than secure element RAM size ($\approx$ 30kB)
- A lot of RAM consumption for the $3$ security levels of Dilithium
  - Each polynomial is $256 \times 4$ bytes, so $1$kB/polynomial
  - Matrix $A$ of $k \times l$ polynomials with $k$ and $l$ up to $8$ and $7$!




- Main idea: perform operations polynomial-wise instead of vector-wise
- Proprietary implementation conform to standard Dilithium
- Up to $30\%$ reduction for Dilithium-5
- Only $\approx 3\%$ slower in average

OPEN

# A Practical Template Attack on Dilithium (CHES2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, CHARTOUNI Maya, MADEC Steven, VERGNAUD Damien, VIGILANT David

Suppose an attacker has access to several signatures $\sigma = (c, z, h)$

$$\begin{aligned} A\,z - c\,t_1\,2^d &= A\,(y + c\,s_1) - c\,(A\,s_1 + s_2 - t_0) \\ &= \underbrace{A\,y}_{w} - c s_2 + c t_0 \\ &= w_1\,2\,\gamma_2 + w_0 + c(t_0 - s_2) \end{aligned}$$

# A Practical Template Attack on Dilithium (CHES2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, CHARTOUNI Maya, MADEC Steven, VERGNAUD Damien, VIGILANT David

Suppose an attacker has access to several signatures $\sigma = (c, z, h)$

$$\begin{aligned} A\,z - c\,t_1\,2^d &= A\,(y + c\,s_1) - c\,(A\,s_1 + s_2 - t_0) \\ &= \underbrace{A\,y}_{w} - cs_2 + ct_0 \\ &= w_1\,2\,\gamma_2 + w_0 + c(t_0 - s_2) \end{aligned}$$

- Assuming an attacker is able to distinguish when $(w_0)_i = cst$ then

$$(A\,z - c\,t_1\,2^d)_i = (w_1)_i\,2\,\gamma_2 + cst + (c\,(t_0 - s_2))_i$$

# A Practical Template Attack on Dilithium (CHES2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, CHARTOUNI Maya, MADEC Steven, VERGNAUD Damien, VIGILANT David

Suppose an attacker has access to several signatures $\sigma = (c, z, h)$

$$
\begin{aligned}
A\,z - c\,t_1\,2^d &= A\,(y + c\,s_1) - c\,(A\,s_1 + s_2 - t_0) \\
&= \underbrace{A\,y}_{w} - c s_2 + c t_0 \\
&= w_1\,2\,\gamma_2 + w_0 + c(t_0 - s_2)
\end{aligned}
$$

- Assuming an attacker is able to distinguish when $(w_0)_i = 0$ then

$$(A\,z - c\,t_1\,2^d)_i = (w_1)_i\,2\,\gamma_2 + 0 + (c\,(t_0 - s_2))_i$$

OPEN

# A Practical Template Attack on Dilithium (CHES2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, CHARTOUNI Maya, MADEC Steven, VERGNAUD Damien, VIGILANT David

Suppose an attacker has access to several signatures $\sigma = (c, z, h)$

$$\begin{aligned} A\,z - c\,t_1\,2^d &= A\,(y + c\,s_1) - c\,(A\,s_1 + s_2 - t_0) \\ &= \underbrace{A\,y}_{w} - c s_2 + c t_0 \\ &= w_1\,2\,\gamma_2 + w_0 + c(t_0 - s_2) \end{aligned}$$

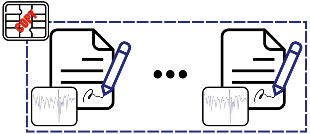
- Assuming an attacker is able to distinguish when $(w_0)_i = 0$ then

$$(A\,z - c\,t_1\,2^d)_i = (w_1)_i\,2\,\gamma_2 + \;0\; + (c\,(t_0 - s_2))_i$$

$$s_1 = (A^t\,A)^{-1}\,A^t\,(t_1\,2^d + (t_0 - s_2))$$

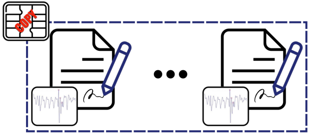
- Knowing $s_1$ suffices to sign arbitrary messages

# Dilithium Secret Key Retrieval

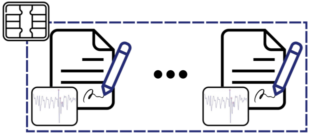


Learning phase
700 K traces

# Dilithium Secret Key Retrieval

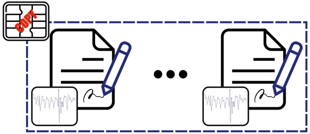



Learning phase
700 K traces

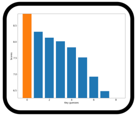


Matching phase
min. 1 trace per msg

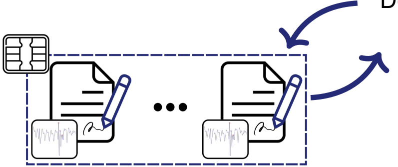
# Dilithium Secret Key Retrieval



Learning phase
700 K traces

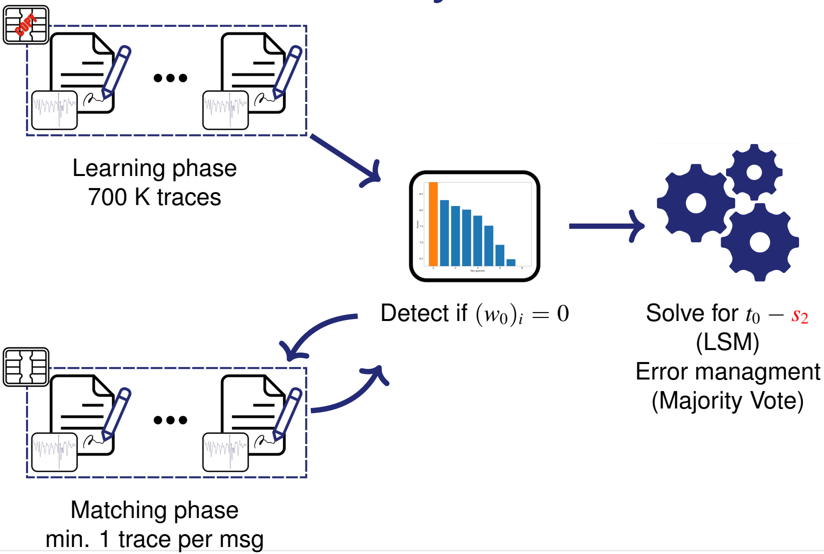


Detect if $(w_0)_i = 0$

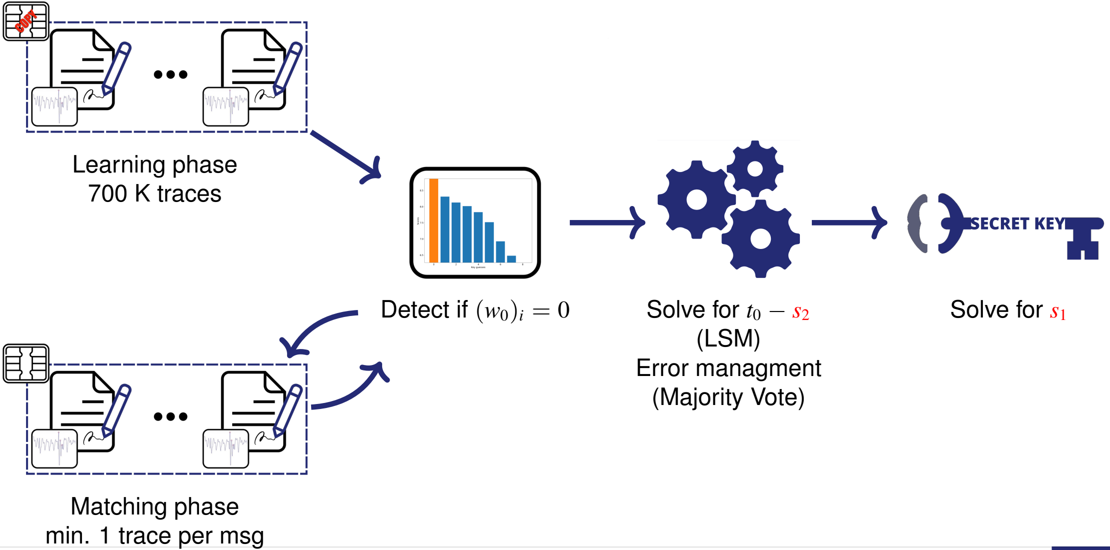


Matching phase
min. 1 trace per msg

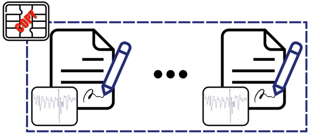
# Dilithium Secret Key Retrieval



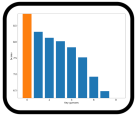
# Dilithium Secret Key Retrieval



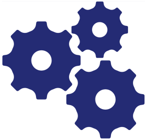
# Dilithium Secret Key Retrieval




Learning phase
700 K traces

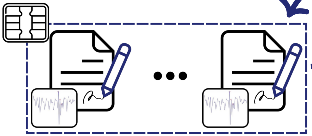


Detect if $(w_0)_i = 0$



Solve for $t_0 - s_2$
(LSM)
Error managment
(Majority Vote)


Solve for $s_1$



Matching phase
min. 1 trace per msg

- $\approx 2\,300\,000$ signatures to fully recover $t_0 - s_2$ (Dilithium-2)

# Fault Attacks sensitivity of Dilithium Verify (CARDIS2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, HEYDEMANN Karine

- Sensitivity Analysis of standard implementation of Verify
- Analyze usually unprotected operations
- Main idea: make $ct_1 2^d$ smaller than it is

$$1 \quad w_1' = \mathtt{UseHint}(h, Az - c\,t_1 2^d)$$

# Fault Attacks sensitivity of Dilithium Verify (CARDIS2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, HEYDEMANN Karine

- Sensitivity Analysis of standard implementation of Verify
- Analyze usually unprotected operations
- Main idea: make $ct_1 2^d$ smaller than it is

$$1 \quad w_1' = \texttt{UseHint}(h, Az - c t_1 2^d)$$

Zeroize polynomial $c$

# Fault Attacks sensitivity of Dilithium Verify (CARDIS2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, HEYDEMANN Karine

- Sensitivity Analysis of standard implementation of Verify
- Analyze usually unprotected operations
- Main idea: make $ct_1 2^d$ smaller than it is

$$1 \quad w_1' = \texttt{UseHint}(h, Az - c t_1 2^d)$$

Zeroize polynomial $c$

Change the exponent $d$

# Fault Attacks sensitivity of Dilithium Verify (CARDIS2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, HEYDEMANN Karine

- Sensitivity Analysis of standard implementation of Verify
- Analyze usually unprotected operations
- Main idea: make $ct_1 2^d$ smaller than it is

Skip the subtraction

$$1 \quad w_1' = \texttt{UseHint}(h, A\,z - c\,t_1 2^d)$$

Zeroize polynomial $c$

Change the exponent $d$

# Fault Attacks sensitivity of Dilithium Verify (CARDIS2023)

Authors: BERZATI Alexandre, CALLE VIERA Andersson, HEYDEMANN Karine

- Sensitivity Analysis of standard implementation of Verify
- Analyze usually unprotected operations
- Main idea: make $ct_1 2^d$ smaller than it is

Skip the subtraction

$$1 \quad w_1' = \texttt{UseHint}(h, Az - ct_1 2^d)$$

Zeroize polynomial $c$

Change the exponent $d$

- Allow to accept false signatures with few simple faults
- Simple and efficient countermeasures presented

# Finding a Polytope: Practical fault attack Against Dilithium

Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$

# Finding a Polytope: Practical fault attack Against Dilithium

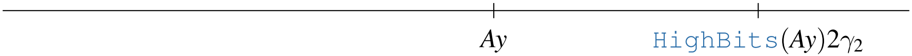
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

$Ay \in [\![0, q[\![$

From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

$\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$



# Finding a Polytope: Practical fault attack Against Dilithium

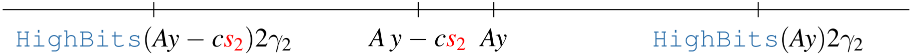
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$



# Finding a Polytope: Practical fault attack Against Dilithium

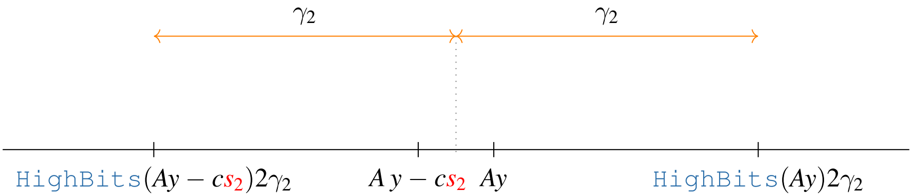
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$



# Finding a Polytope: Practical fault attack Against Dilithium

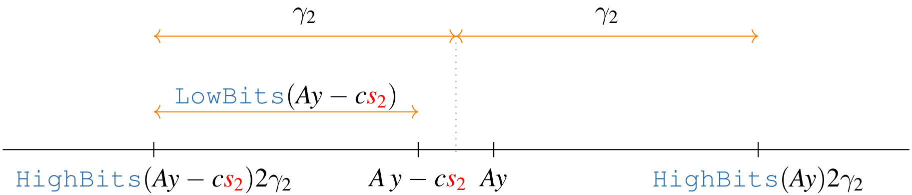
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$



# Finding a Polytope: Practical fault attack Against Dilithium

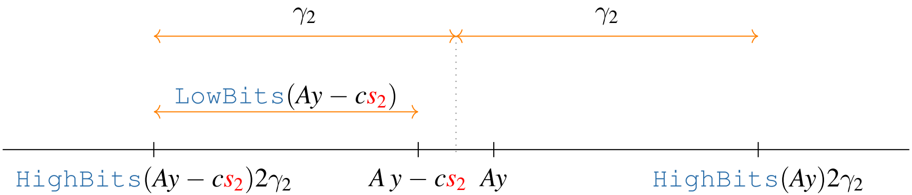
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$



# Finding a Polytope: Practical fault attack Against Dilithium

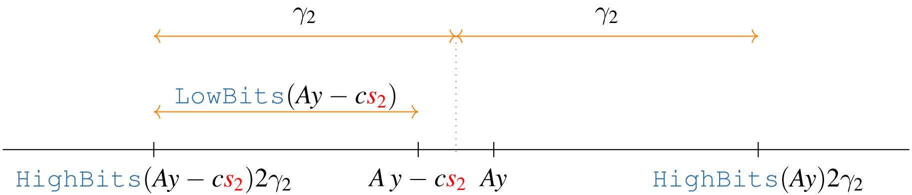
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$




- if $\texttt{HighBits}(Ay) - \texttt{HighBits}(Ay - cs_2) = 1$ then, $cs_2 \geq \gamma_2 - \texttt{LowBits}(Ay - cs_2) \geq 0$

# Finding a Polytope: Practical fault attack Against Dilithium

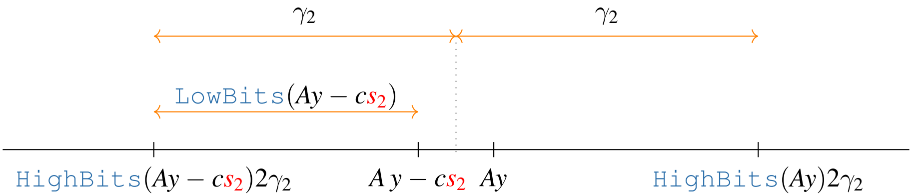
Authors: AZEVEDO OLIVEIRA Paco, CALLE VIERA Andersson, COGLIATI Benoit
GOUBIN Louis

- Dilithium Sign without condition $||r_0||_\infty \geq \gamma_2 - \beta$, use case: fault attacks

  $Ay \in [\![0, q[\![$

  From the specification $-\beta \leq cs_2 \leq \beta$, wlog. suppose $0 < cs_2$

  $\texttt{LowBits}(Ay - cs_2) + cs_2 \geq \gamma_2 \geq \texttt{LowBits}(Ay - cs_2)$




- if $\texttt{HighBits}(Ay) - \texttt{HighBits}(Ay - cs_2) = 1$ then, $cs_2 \geq \gamma_2 - \texttt{LowBits}(Ay - cs_2) \geq 0$
- if $\texttt{HighBits}(Ay) - \texttt{HighBits}(Ay - cs_2) = -1$ then, $cs_2 \leq -\gamma_2 - \texttt{LowBits}(Ay - cs_2) \leq 0$

# Solving the inequalities

Remember that, $s_2 = ((s_2)_0 x^0 + (s_2)_1 x^1 + \cdots + (s_2)_{255} x^{255}) = \sum_{a=0}^{255} (s_2)_a x^a$.

Therefore, $cs_2 = c\left(\sum_{a=0}^{255} (s_2)_a x^a\right) = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)$

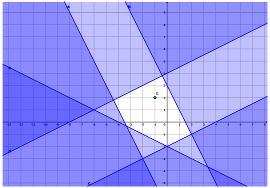
BUT, inequality on only one $j \in [\![0, 256[\![$, so: $(cs_2)_j = \left(c\left(\sum_{a=0}^{255} (s_2)_a x^a\right)\right)_j = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)_j$

# Solving the inequalities

Remember that, $s_2 = ((s_2)_0 x^0 + (s_2)_1 x^1 + \cdots + (s_2)_{255} x^{255}) = \sum_{a=0}^{255} (s_2)_a x^a$.

Therefore, $cs_2 = c\left(\sum_{a=0}^{255} (s_2)_a x^a\right) = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)$

BUT, inequality on only one $j \in [\![0, 256[\![$, so: $(cs_2)_j = \left(c\left(\sum_{a=0}^{255} (s_2)_a x^a\right)\right)_j = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)_j$

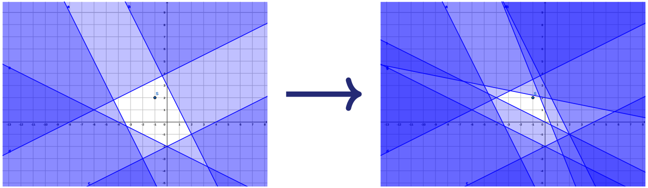


# Solving the inequalities

Remember that, $s_2 = ((s_2)_0 x^0 + (s_2)_1 x^1 + \cdots + (s_2)_{255} x^{255}) = \sum_{a=0}^{255} (s_2)_a x^a$.

Therefore, $cs_2 = c\left(\sum_{a=0}^{255} (s_2)_a x^a\right) = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)$

BUT, inequality on only one $j \in [\![0, 256[\![$, so: $(cs_2)_j = \left(c\left(\sum_{a=0}^{255} (s_2)_a x^a\right)\right)_j = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)_j$

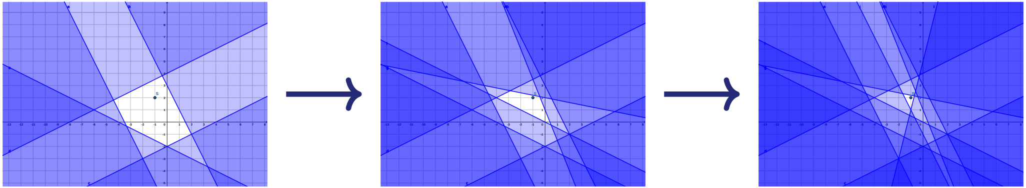


# Solving the inequalities

Remember that, $s_2 = ((s_2)_0 x^0 + (s_2)_1 x^1 + \cdots + (s_2)_{255} x^{255}) = \sum_{a=0}^{255} (s_2)_a x^a$.

Therefore, $cs_2 = c\left(\sum_{a=0}^{255} (s_2)_a x^a\right) = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)$

BUT, inequality on only one $j \in [\![0, 256[\![$, so: $(cs_2)_j = \left(c\left(\sum_{a=0}^{255} (s_2)_a x^a\right)\right)_j = \sum_{a=0}^{255} (s_2)_a \left(cx^a\right)_j$



- Collect enough inequalities ($\approx 11\,000$ over $1\,250\,000$ signatures)
- Solve the corresponding LP problem
- Round the result to get the correct solution

# Future Work

- Identify vulnerable operations within KEM schemes
  - SCA/FA on Kyber
- Study novel approaches for implementing Dilithium and Kyber
  - Balance security and efficiency (changes in arithmetic used for example)
- Evaluate efficient countermeasures for Dilithium/Kyber
  - Focus on polynomial multiplication

# Future Work

- Identify vulnerable operations within KEM schemes
  - SCA/FA on Kyber
- Study novel approaches for implementing Dilithium and Kyber
  - Balance security and efficiency (changes in arithmetic used for example)
- Evaluate efficient countermeasures for Dilithium/Kyber
  - Focus on polynomial multiplication

**Thank you**

Questions?