Exploiting Intermediate Value Leakage in Dilithium:
A Template-Based Approach

Alexandre Berzati[1], **Andersson Calle Viera**[1,2],
Maya Chartouny[1,3], Steven Madec[1],
Damien Vergnaud[2], David Vigilant[1]
CHES 2023, 12 september 2023

[1] Thales DIS, France
[2] Sorbonne Université, France
[3] Université Paris-Saclay, France

Outline

Outline

Introduction

- Quantum threat: Shor’s quantum algorithm can break integer factorization and discrete logarithm in polynomial time
- PQC: Algorithms are currently under standardization with several international initiatives
- **Importance:** These new algorithms will be implemented securely in a variety of use cases

Banking

Personal Data

Communication

- ML-DSA draft specification is derived from Version 3.1 of CRYSTALS-Dilithium (Dilithium)
- CRYSTALS-Dilithium is the main PQC signature algorithm, selected in 2022 by the NIST

- **Our Contribution:** Template based exploitation of intermediate value on Dilithium

Dilithium

- Dilithium: public key signature algorithm
- Based on hard problems on Lattices
  - M-LWE
  - M-SIS
- Three security levels: Dilithium-2, Dilithium-3, Dilithium-5
- Two versions: deterministic and randomized
- Recommended as principal PQC signature scheme:
  - Adjusting security levels is simple
  - Minimal `pk` size + `sign` size
  - Already some constant time properties
- Advantage: No known efficient algorithm, classical or quantum, can solve these problems in less than exponential time

KeyGen:

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
where $n = 2^8$ and
$q = 2^{23} - 2^{13} + 1$

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

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

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

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

return $\texttt{pk} = (\rho, t_1), \texttt{sk} = (\rho, s_1, s_2, t_0, \texttt{H}(\texttt{pk}))$

KeyGen:

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
where $n = 2^8$ and
$q = 2^{23} - 2^{13} + 1$

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

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

5 return $\texttt{pk} = (\rho, t_1)$, $\texttt{sk} = (\rho, s_1, s_2, t_0, \texttt{H}(\texttt{pk}))$

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

Sign(M, sk):

$A \in \mathcal{R}_q^{k \times l} := \texttt{ExpandA}(\rho)$
$\mu := \texttt{H}(\texttt{H}(\texttt{pk}) \,||\, M),\ (z, h) := \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}_{\gamma_1}^{l}$
    $w := A\, y$
    $w_1,\ w_0 := \texttt{Decompose}_q(w,\ 2\,\gamma_2)$
    $c \in B_\tau := \texttt{H}(\mu \,||\, w_1)$
    $z := y + c\, s_1$
    $r_0 := w_0 - c\, s_2$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) := \perp$
    else
        $h := \texttt{MakeHint}_q(w_1,\ r_0 + c\, t_0,\ 2\,\gamma_2)$
        if $\|c\, t_0\|_\infty \geq \gamma_2$, then $(z, h) := \perp$
return $\sigma = (c,\ z,\ h)$

Verify(pk, M, σ):

1 $\mu \in \{0,1\}^{384} := \mathtt{H}(\mathtt{H}(\mathtt{pk}) \,||\, M)$

2 $w_1' := \mathtt{UseHint}_q(h, A\,z - c\,t_1 2^d, 2\,\gamma_2)$

3 if $||z||_\infty < \gamma_1 - \beta$ and $c == \mathtt{H}(\mu \,||\, w_1')$ and $\#\,1$'s in $h \leq \omega$ then return *True*

4 else

5 return *False*

Outline

Attack path

From the verification algorithm: $2\, w_1' := \texttt{UseHint}_q(h, A\,z - c\,t_1 2^d, 2\,\gamma_2)$

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

$$(A\,z - c\,t_1\,2^d)_i = (w_1)_i\,2\,\gamma_2 + cst + (c\,(t_0 - s_2))_i \tag{1}$$

Repeat for all the $k \times n$ coefficients

Attack path

From the verification algorithm: 2 $w_1' := \texttt{UseHint}_q(h, A\,z - c\,t_1 2^d, 2\,\gamma_2)$

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 $(w_0)_i = 0$ then

$$
(A\,z - c\,t_1\,2^d)_i = (w_1)_i\,2\,\gamma_2 + 0 + (c\,(t_0 - s_2))_i \tag{1}
$$

Repeat for all the $k \times n$ coefficients

Here, we consider exclusively the case $cst = 0$

Attack path

- $t_0 - s_2$ allows us to find s_1

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

A is not square, but $(A^t A)$ is square and invertible with high probability

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

- Knowing s_1 suffices to sign arbitrary messages

Remark: The attack's efficiency depends on how well we can differentiate for $(w_0)_i = 0$

Highlighting potential leakage spots

$A \in \mathcal{R}_q^{k \times l} := \texttt{ExpandA}(\rho)$
$\mu := \texttt{H}(\texttt{H}(\texttt{pk}) \,||\, M),\ (z, h) := \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}_{\gamma_1}^{l}$
    $w := A\, y$
    $w_1,\ w_0 := \texttt{Decompose}_q(w,\ 2\,\gamma_2)$
    $c \in B_\tau := \texttt{H}(\mu \,||\, w_1)$
    $z := y + c\, s_1$
    $r_0 := w_0 - c\, s_2$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) := \perp$
    else
        $h := \texttt{MakeHint}_q(w_1,\ r_0 + c\, t_0,\ 2\,\gamma_2)$
        if $\|c\, t_0\|_\infty \geq \gamma_2$, then $(z, h) := \perp$
return $\sigma = (c,\ z,\ h)$

1. Inside the decomposition
    - Direct use of w to produce w_0
2. Subtraction
    - Clear HW leakage

Highlighting potential leakage spots

$A \in \mathcal{R}_q^{k \times l} := \texttt{ExpandA}(\rho)$
$\mu := \texttt{H}(\texttt{H}(\texttt{pk}) \,||\, M),\ (z, h) := \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}_{\gamma_1}^{l}$
    $w := A\, y$
    $w_1,\ w_0 := \boxed{\texttt{Decompose}_q}(w,\ 2\,\gamma_2)$
    $c \in B_\tau := \texttt{H}(\mu \,||\, w_1)$
    $z := y + c\, s_1$
    $r_0 := w_0 - c\, s_2$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) := \perp$
    else
        $h := \texttt{MakeHint}_q(w_1,\ r_0 + c\, t_0,\ 2\,\gamma_2)$
        if $\|c\, t_0\|_\infty \geq \gamma_2$, then $(z, h) := \perp$
return $\sigma = (c,\ z,\ h)$

1 Inside the decomposition
- Direct use of w to produce w_0

2 Subtraction
- Clear HW leakage

Highlighting potential leakage spots

$A \in \mathcal{R}_q^{k \times l} := \texttt{ExpandA}(\rho)$
$\mu := \texttt{H}(\texttt{H}(\texttt{pk}) \,||\, M),\ (z, h) := \perp$
while $(z, h) = \perp$ do
    $y \in \tilde{S}_{\gamma_1}^{l}$
    $w := A\, y$
    $w_1,\ w_0 := \boxed{\texttt{Decompose}_q}(w,\ 2\,\gamma_2)$
    $c \in B_\tau := \texttt{H}(\mu \,||\, w_1)$
    $z := y + c\, s_1$
    $r_0 := \boxed{w_0 - c\, s_2}$
    if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta,$ then $(z, h) := \perp$
    else
        $h := \texttt{MakeHint}_q(w_1,\ r_0 + c\, t_0,\ 2\,\gamma_2)$
        if $\|c\, t_0\|_\infty \geq \gamma_2$, then $(z, h) := \perp$
return $\sigma = (c,\ z,\ h)$

1. Inside the decomposition
    - Direct use of w to produce w_0
2. Subtraction
    - Clear HW leakage

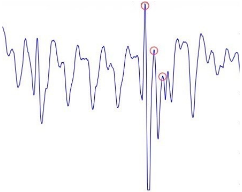
Template Attack (TPA) in theory

TPA are a powerful type of Side Channel Attacks

Step 1:

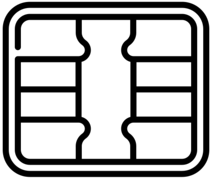
Record many power traces using different keys and inputs

Step 2:



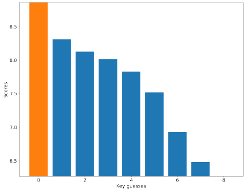
Create a template by selecting points of interest

Step 3:



Record few power traces using multiple plaintexts

Step 4:



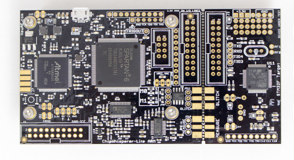
Apply the template to the attack traces

TPA in practice

PQClean implem of Dilithium

- Latest implem
- Deterministic
- Dilithium-2

ChipWhisperer



- Arm Cortex M4
- CPU: 32 bits
- RAM: 48kB

Side Channel:

- Leakage identification with power traces
- Without loss of generality the template is made on the first $(w_0)_0$
- Leakage model: HW of each of the 4 bytes of a $(w_0)_i$

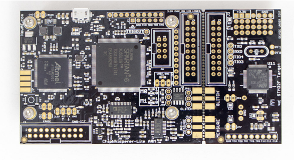
Goal: Differentiate efficiently for a $(w_0)_i = 0$

TPA in practice

PQClean implem of Dilithium

- Latest implem
- Deterministic
- Dilithium-2

ChipWhisperer



- Arm Cortex M4
- CPU: 32 bits
- RAM: 48kB

Side Channel:

- Leakage identification with power traces
- Without loss of generality the template is made on the first $(w_0)_0$
- Leakage model: HW of each of the 4 bytes of a $(w_0)_i$

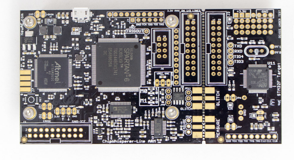
Goal: Differentiate efficiently for a $(w_0)_i = 0$

TPA in practice

PQClean implem of Dilithium

- Latest implem
- Deterministic
- Dilithium-2

ChipWhisperer



- Arm Cortex M4
- CPU: 32 bits
- RAM: 48kB

Side Channel:

- Leakage identification with power traces
- Without loss of generality the template is made on the first $(w_0)_0$
- Leakage model: HW of each of the 4 bytes of a $(w_0)_i$

Goal: Differentiate efficiently for a $(w_0)_i = 0$

Learning Phase (Step 1 and 2):

- Target the `Decompose` operation
- Collect suitable messages in C $\rightarrow$ 18 hours
- $700\,000$ power traces on the ChipWhisperer $\rightarrow$ 24 hours

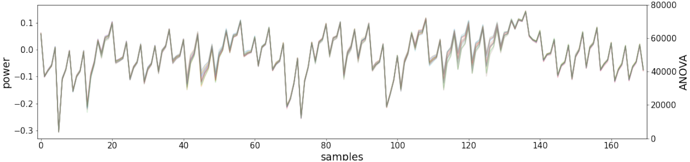


Figure: CW traces for $(w_0)_0$

Learning Phase (Step 1 and 2):

- Target the `Decompose` operation
- Collect suitable messages in C $\rightarrow$ 18 hours
- $700\,000$ power traces on the ChipWhisperer $\rightarrow$ 24 hours

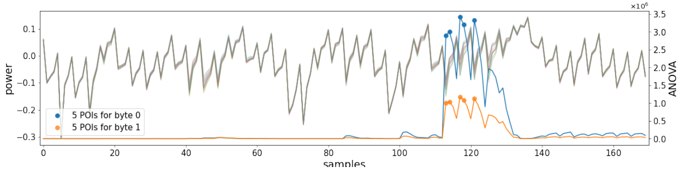


Figure: POIs selection for the two MSBs

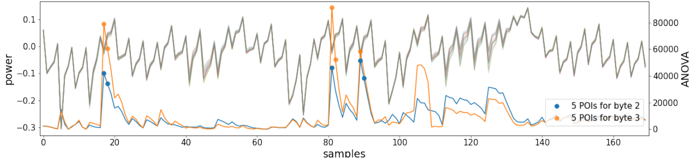


Figure: POIs selection the two LSBs

- ANOVA used to select the POIs and 5 peaks kept as POIs to build the template

Matching Phase (Step 3 and 4):

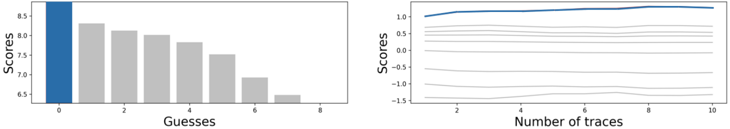


Figure: Matching value for LSB 0

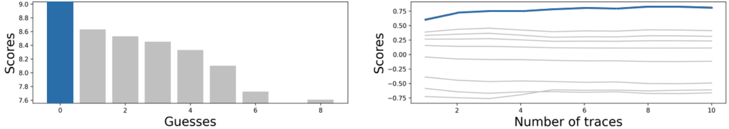


Figure: Matching value for LSB 1

- 0 value clearly distinguishable from the rest, even with 1 trace

Definition (False positives - False negatives)

False positives: predicting $w_0 = 0$ while it's not
False negatives: predicting $w_0 \neq 0$ while it's not

- fp: $0.067\% \Rightarrow \leq 1$ coeff from the $k \times n$
- fn: $0.174\% \Rightarrow$ more signatures to acquire

- Same results for ≈ 100 first coeffs

Filtering w_0 for efficiency

SCA measurements might be imperfect:

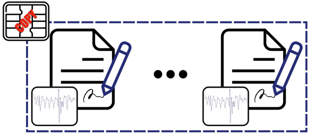
- False positives impact the success rate of the attack
- False negatives impact only the number of signatures needed

- We propose a filter on public values to avoid introducing equations with false positives

$$|(A\,z - c\,t_1\,2^d - w_1\,2\gamma_2)_{i,j}| \leq 2\sqrt{\frac{2^{2d}-1}{12}}\tau$$

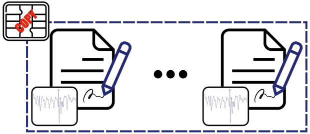
Discard ≈ **70%** of the $k \times n$ coeffs where we might not have $(w_0)_i = 0$ (impact on fp)
However ≈ **5%** of true $w_0 = 0$ are erroneously removed (impact on fn)

Dilithium Secret Key Retrieval

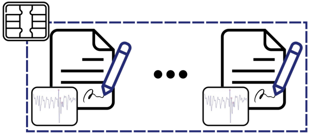


Learning phase
700 K traces

Dilithium Secret Key Retrieval

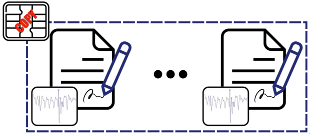


Learning phase
700 K traces

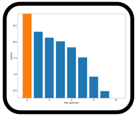


Matching phase
min. 1 trace per msg

Dilithium Secret Key Retrieval



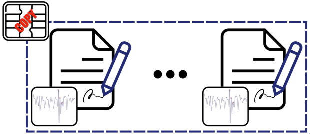
Learning phase
700 K traces



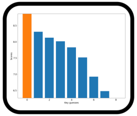
Detect if $(w_0)_i = 0$
min. 2.5 M signatures

Matching phase
min. 1 trace per msg

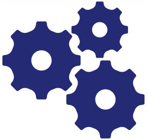
Dilithium Secret Key Retrieval



Learning phase
700 K traces



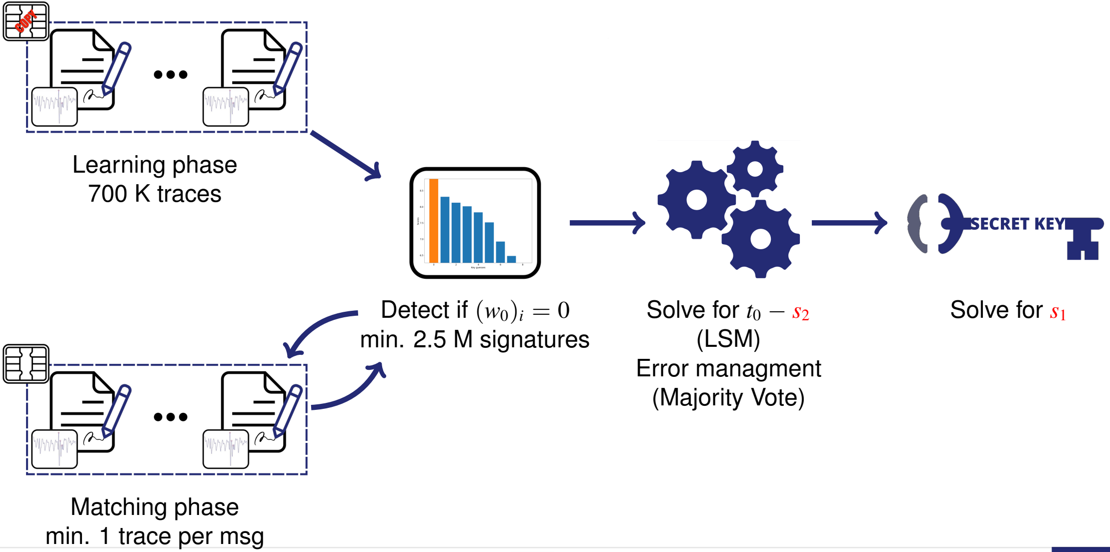
Detect if $(w_0)_i = 0$
min. 2.5 M signatures



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

Matching phase
min. 1 trace per msg

Dilithium Secret Key Retrieval



Outline

Countermeasures

Goal: Reduce the potential leakage spots

Simple countermeasures are known and efficient against this attack

- Shuffling of coefficient during sensitive steps (`Decompose` and `Subtraction`)
- Secret sharing/ Masking when manipulating w_0
  - Masking design of the `Decompose` function discussed in [ACNS2019, CHES2023, CHES2023]
  - For the `Subtraction` use masked $r_0 = \texttt{LowBits}_q(w - c\, s_2,\ 2\gamma_2)$

Outline

Conclusion

To summarize, this work on Dilithium signature:

- First exploitation of a zero value leakage on w_0 during signature execution
- In turn, allows to recover s_1, and then forge signatures
- Shows that the leakage can be exploited in practice through experimentations
- Discusses Filtering, Resolution and Error Management steps for efficiency
- Highlights simple known countermeasures

Future work on evaluating the impact of noise on error management tools

Thank you

Questions?

Bibliography

[ACNS2019] V. Migliore, B. Gérard, M. Tibouchi, and PA. Fouque, *Masking Dilithium: Efficient implementation and side-channel evaluation*.

[CHES2023] M. Azouaoui, O. Bronchain, G. Cassiers, et al., *Protecting Dilithium against Leakage: Revisited Sensitivity Analysis and Improved Implementations*.

[CHES2023] JS. Coron, F. Gérard, M. Trannoy, and R. Zeitoun, *Improved Gadgets for the High-Order Masking of Dilithium*.