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

Alexandre Berzati\textsuperscript{1}, Andersson Calle Viera\textsuperscript{1,2}, Maya Chartouny\textsuperscript{1,3}, Steven Madec\textsuperscript{1}, Damien Vergnaud\textsuperscript{2}, David Vigilant\textsuperscript{1}

Journées C2, 19 october 2023

\textsuperscript{1} Thales DIS, France
\textsuperscript{2} Sorbonne Université, France
\textsuperscript{3} Université Paris-Saclay, France
Outline

1 Introduction
   - Context
   - Dilithium

2 Our Profiling Attack on Dilithium
   - Exploited attack path
   - Template Attack

3 Countermeasures

4 Conclusion
1 Introduction
   - Context
   - Dilithium

2 Our Profiling Attack on Dilithium
   - Exploited attack path
   - Template Attack

3 Countermeasures

4 Conclusion
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
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

ML-DSA draft specification is derived from Version 3.1 of CRYSTALS-Dilithium (Dilithium)

CRYSNTALS-Dilithium is the main PQC signature algorithm, selected in 2022 by the NIST
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

ML-DSA draft specification is derived from Version 3.1 of CRYSrALS-Dilithium (Dilithium)
CRYSrALS-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:

\[ R_q = \mathbb{Z}_q[X]/(X^n + 1) \]

where \( n = 2^8 \) and \( q = 2^{23} - 2^{13} + 1 \)

1. \( A \in R_q^{k \times l} := \text{ExpandA}(\rho) \)
2. \( (s_1, s_2) \in S^l_\eta \times S^k_\eta \)
3. \( t := A s_1 + s_2 \in R_q^k \)
4. \( (t_1, t_0) := \text{Power2Round}_q(t, d) \)
5. return \( \text{pk} = (\rho, t_1), \text{sk} = (\rho, s_1, s_2, t_0, H(\text{pk})) \)
KeyGen:

\[ R_q = \mathbb{Z}_q[X]/(X^n + 1) \]

where \( n = 2^8 \) and

\[ q = 2^{23} - 2^{13} + 1 \]

1. \( A \in R_q^{k \times l} := \text{Expand}A(\rho) \)
2. \( (s_1, s_2) \in S_{\eta}^l \times S_{\eta}^k \)
3. \( t := A s_1 + s_2 \in R_q^k \)
4. \( (t_1, t_0) := \text{Power}2\text{Round}_q(t, d) \)
5. return \( \text{pk} = (\rho, t_1), \text{sk} = (\rho, s_1, s_2, t_0, H(\text{pk})) \)
Sign($M$, $sk$):

1. $A \in \mathcal{R}_{q}^{k \times l} := \text{ExpandA}(\rho)$
2. $\mu := H(H(pk) || M)$, $(z, h) := \bot$
3. while $(z, h) = \bot$ do
   4. $y \in \tilde{S}_{\gamma_1}^l$
   5. $w := Ay$
   6. $w_1, w_0 := \text{Decompose}_q(w, 2 \gamma_2)$
   7. $c \in B_\tau := H(\mu || w_1)$
   8. $z := y + cs_1$
   9. $r_0 := w_0 - cs_2$
10. if $||z||_{\infty} \geq \gamma_1 - \beta$ or $||r_0||_{\infty} \geq \gamma_2 - \beta$, then $(z, h) := \bot$
11. else
12. $h := \text{MakeHint}_q(w_1, r_0 + ct_0, 2 \gamma_2)$
13. if $||ct_0||_{\infty} \geq \gamma_2$, then $(z, h) := \bot$
14. return $\sigma = (c, z, h)$
Sign($M, sk$):

1. $A \in \mathcal{R}_{q}^{k \times l} := \text{ExpandA}(\rho)$
2. $\mu := H(H(pk) || M), (z, h) := \bot$
3. while $(z, h) = \bot$ do
   4. $y \in \tilde{S}_{\gamma_1}^l$
   5. $w := Ay$
   6. $w_1, w_0 := \text{Decompose}_q(w, 2\gamma_2)$
   7. $c \in B_{\tau} := H(\mu || w_1)$
   8. $z := y + c s_1$
   9. $r_0 := w_0 - c s_2$
10. if $\|z\|_\infty \geq \gamma_1 - \beta$ or $\|r_0\|_\infty \geq \gamma_2 - \beta$, then $(z, h) := \bot$
    else
    12. $h := \text{MakeHint}_q(w_1, r_0 + c t_0, 2\gamma_2)$
    13. if $\|c t_0\|_\infty \geq \gamma_2$, then $(z, h) := \bot$
14. return $\sigma = (c, z, h)$
Sign($M, sk$):

1. $A \in \mathcal{R}_q^{k \times l} := \text{Expand}_A(\rho)$
2. $\mu := H(H(pk) || M), (z, h) := \bot$
3. while $(z, h) = \bot$ do
   4. $y \in \tilde{S}^l_{\gamma_1}$
   5. $w := Ay$
   6. $w_1, w_0 := \text{Decompose}_q(w, 2\gamma_2)$
   7. $c \in B_{\tau} := H(\mu || w_1)$
   8. $z := y + cs_1$
   9. $r_0 := w_0 - cs_2$
   10. if $\|z\|_{\infty} \geq \gamma_1 - \beta$ or $\|r_0\|_{\infty} \geq \gamma_2 - \beta$, then $(z, h) := \bot$
   11. else
      12. $h := \text{MakeHint}_q(w_1, r_0 + ct_0, 2\gamma_2)$
      13. if $\|ct_0\|_{\infty} \geq \gamma_2$, then $(z, h) := \bot$
14. return $\sigma = (c, z, h)$
**Sign(\(M, sk\)):**

1. \(A \in \mathcal{R}_{q}^{k \times l} := \text{Expand}_{A}(\rho)\)
2. \(\mu := H(H(pk) || M), (z, h) := ⊥\)
3. while \((z, h) = ⊥\) do
   4. \(y \in \tilde{S}_{\gamma_1}^l\)
   5. \(w := Ay\)
   6. \(w_1, w_0 := \text{Decompose}_{q}(w, 2 \gamma_2)\)
   7. \(c \in B_{\tau} := H(\mu || w_1)\)
   8. \(z := y + cs_1\)
   9. \(r_0 := w_0 - cs_2\)
10. if \(||z||_\infty \geq \gamma_1 - \beta\ or \ ||r_0||_\infty \geq \gamma_2 - \beta\), then \((z, h) := ⊥\)
11. else
   12. \(h := \text{MakeHint}_{q}(w_1, r_0 + ct_0, 2 \gamma_2)\)
   13. if \(||ct_0||_\infty \geq \gamma_2\), then \((z, h) := ⊥\)
14. return \(\sigma = (c, z, h)\)
Sign($M, sk$):

1. $A \in \mathcal{R}^{k \times l}_q := \text{ExpandA}(\rho)$
2. $\mu := H(H(pk) || M), (z, h) := \bot$
3. while $(z, h) = \bot$ do
   4. $y \in \tilde{S}^l_{\gamma_1}$
   5. $w := A y$
   6. $w_1, w_0 := \text{Decompose}_q(w, 2 \gamma_2)$
   7. $c \in B_{\tau} := H(\mu || w_1)$
   8. $z := y + c s_1$
   9. $r_0 := w_0 - c s_2$
10. if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) := \bot$
11. else
   12. $h := \text{MakeHint}_q(w_1, r_0 + c t_0, 2 \gamma_2)$
   13. if $||c t_0||_\infty \geq \gamma_2$, then $(z, h) := \bot$
14. return $\sigma = (c, z, h)$
Sign($M$, $sk$):

1. $A \in \mathcal{R}^{k \times l}_q := \text{ExpandA}(\rho)$
2. $\mu := H(H(pk) || M)$, $(z, h) := \bot$
3. while $(z, h) = \bot$ do
4. $y \in \tilde{S}^l_{\gamma_1}$
5. $w := Ay$
6. $w_1, w_0 := \text{Decompose}_q(w, 2 \gamma_2)$
7. $c \in B_{\tau} := H(\mu || w_1)$
8. $z := y + cs_1$
9. $r_0 := w_0 - cs_2$
10. if $\|z\|_\infty \geq \gamma_1 - \beta$ or $\|r_0\|_\infty \geq \gamma_2 - \beta$, then $(z, h) := \bot$
11. else
12. $h := \text{MakeHint}_q(w_1, r_0 + ct_0, 2 \gamma_2)$
13. if $\|ct_0\|_\infty \geq \gamma_2$, then $(z, h) := \bot$
14. return $\sigma = (c, z, h)$
Verify\((pk, M, \sigma)\):  

1. \(\mu := H(H(pk) || M)\)  
2. \(w'_1 := \text{UseHint}_q(h, Az - c t_1 2^d, 2 \gamma_2)\)  
3. if \(||z||_\infty < \gamma_1 - \beta\) and \(c == H(\mu || w'_1)\) and \# 1's in \(h \leq \omega\) then return True  
4. else  
5. return False
A (brief) Note on Side Channel Attacks

Instead of directly attacking a cryptosystem, we can infer secret data on an implementation.
A (brief) Note on Side Channel Attacks

Instead of directly attacking a cryptosystem, we can infer secret data on an implementation.

19 October 2023
Exploiting Intermediate, Value Leakage in Dilithium: A Template-Based Approach
A (brief) Note on Side Channel Attacks

- Instead of directly attacking a cryptosystem, we can infer secret data on an implementation.
A (brief) Note on Side Channel Attacks

- Instead of directly attacking a cryptosystem, we can infer secret data on an implementation.
A (brief) Note on Side Channel Attacks

- Instead of directly attacking a cryptosystem, we can infer secret data on an implementation.
A (brief) Note on Side Channel Attacks

- Instead of directly attacking a cryptosystem, we can infer secret data on an implementation.
Outline

1 Introduction
   - Context
   - Dilithium

2 Our Profiling Attack on Dilithium
   - Exploited attack path
   - Template Attack

3 Countermeasures

4 Conclusion
Attack path

From the verification algorithm: \( 2w'_1 := \text{UseHint}_q(h, Az - c t_1 2^d, 2 \gamma_2) \)

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

\[
Az - c t_1 2^d = A(y + cs_1) - c(As_1 + s_2 - t_0) \\
= Ay - cs_2 + ct_0 \\
= w_1 2 \gamma_2 + w_0 + c(t_0 - s_2)
\]

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

\[
(Az - c t_1 2^d)_i = (w_1)_i 2 \gamma_2 + cst + (c(t_0 - s_2))_i \tag{1}
\]

Repeat for all the \( k \times n \) coefficients
Attack path

From the verification algorithm: \( 2w'_1 := \text{UseHint}_q(h, Az - ct_12^d, 2\gamma_2) \)

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

\[
Az - ct_12^d = A(y + cs_1) - c(As_1 + s_2 - t_0)
\]
\[
= Ay - cs_2 + ct_0
\]
\[
= w_1 2\gamma_2 + w_0 + c(t_0 - s_2)
\]

• Assuming an attacker is able to distinguish when \((w_0)_i = 0\) then

\[
(Az - ct_12^d)_i = (w_1)_i 2\gamma_2 + 0 + (c(t_0 - s_2))_i \quad (1)
\]

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

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

- 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

1. $A \in \mathcal{R}_q^{k \times l} := \text{ExpandA}(\rho)$
2. $\mu := \mathcal{H}(\mathcal{H}(pk) \ || \ M)$, $(z, h) := \bot$
3. while $(z, h) = \bot$ do
   4. $y \in \tilde{S}_1^l$
   5. $w := A y$
   6. $w_1, w_0 := \text{Decompose}_q(w, 2 \gamma_2)$
   7. $c \in B_\tau := \mathcal{H}(\mu \ || \ w_1)$
   8. $z := y + c s_1$
   9. $r_0 := w_0 - c s_2$
10. if $\|z\|_\infty \geq \gamma_1 - \beta$ or $\|r_0\|_\infty \geq \gamma_2 - \beta$, then $(z, h) := \bot$
11. else
   12. $h := \text{MakeHint}_q(w_1, r_0 + c t_0, 2 \gamma_2)$
   13. if $\|c t_0\|_\infty \geq \gamma_2$, then $(z, h) := \bot$
14. return $\sigma = (c, z, h)$
Highlighting potential leakage spots

1. $A \in \mathcal{R}_q^{k \times l} := \text{ExpandA}(\rho)$
2. $\mu := \mathcal{H}(\mathcal{H}(pk) || M)$, $(z, h) := \perp$
3. while $(z, h) = \perp$ do
4. \hspace{1cm} $y \in \tilde{S}_{\gamma_1}^l$
5. \hspace{1cm} $w := Ay$
6. \hspace{1cm} $w_1, w_0 := \text{Decompose}_q(w, 2 \gamma_2)$
7. \hspace{1cm} $c \in B_\tau := \mathcal{H}(\mu || w_1)$
8. \hspace{1cm} $z := y + cs_1$
9. \hspace{1cm} $r_0 := w_0 - cs_2$
10. if $||z||_\infty \geq \gamma_1 - \beta$ or $||r_0||_\infty \geq \gamma_2 - \beta$, then $(z, h) := \perp$
11. \hspace{1cm} else
12. \hspace{2cm} $h := \text{MakeHint}_q(w_1, r_0 + ct_0, 2 \gamma_2)$
13. \hspace{2cm} if $||ct_0||_\infty \geq \gamma_2$, then $(z, h) := \perp$
14. 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

1. \( A \in \mathcal{R}_q^{k \times l} := \text{Expand}_A(\rho) \)
2. \( \mu := H(H(pk) \| M), (z, h) := \perp \)
3. while \((z, h) = \perp\) do
4. \( y \in \tilde{S}_\gamma^l \)
5. \( w := Ay \)
6. \( w_1, w_0 := \text{Decompose}_q(w, 2\gamma_2) \)
7. \( c \in B_\tau := H(\mu \| w_1) \)
8. \( z := y + cs_1 \)
9. \( r_0 := \begin{cases} w_0 - cs_2 & \text{if } ||z||_\infty \geq \gamma_1 - \beta \text{ or } ||r_0||_\infty \geq \gamma_2 - \beta \end{cases} \)
10. if \(||z||_\infty \geq \gamma_1 - \beta \text{ or } ||r_0||_\infty \geq \gamma_2 - \beta\), then \((z, h) := \perp\)
11. else
12. \( h := \text{MakeHint}_q(w_1, r_0 + ct_0, 2\gamma_2) \)
13. if \(|ct_0||_\infty \geq \gamma_2\), then \((z, h) := \perp\)
14. 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
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 \( w_0 \)
- Leakage model: HW of each of the 4 bytes of a \( w_0 \)

Goal: Differentiate efficiently for a \( w_0 = 0 \)
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 → 18 hours
- 700 000 power traces on the ChipWhisperer → 24 hours

Figure: CW traces for \((w_0)_0\)
Learning Phase (Step 1 and 2):

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

Figure: CW traces for \((w_0)_0\)
Learning Phase (Step 1 and 2):

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

Figure: CW traces for \( w_0 \)

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

- 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 $\approx 100$ 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

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

Discard $\approx 70\%$ of the $k \times n$ coeffs where we might not have $(w_0)_i = 0$ (impact on fp)

However $\approx 5\%$ 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

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

Solve for \( t_0 - s_2 \) (LSM)

Error management
(Majority Vote)

19 October 2023

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

Matching phase
min. 1 trace per msg

Solve for \(t_0 - s_2\)
(LSM)

Error management
(Majority Vote)
Dilithium Secret Key Retrieval

**Learning phase**
- 700 K traces

**Matching phase**
- min. 1 trace per msg

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

**Solve for** $t_0 - s_2$
- (LSM)

**Error management**
- (Majority Vote)

**Solve for** $s_1$

---

19 October 2023

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

2 Our Profiling Attack on Dilithium
- Exploited attack path
- Template Attack

3 Countermeasures

4 Conclusion
**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 $w_0 = \text{LowBits}_q(w - c s, 2^\gamma)$
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 $r_0 = \text{LowBits}_q(w - c \cdot s_2, 2 \gamma_2)$
Outline

1. Introduction
   - Context
   - Dilithium

2. Our Profiling Attack on Dilithium
   - Exploited attack path
   - Template Attack

3. Countermeasures

4. Conclusion
Conclusion

To summarize, this work on Dilithium:

> First exploitation of a zero value leakage on $w_0$ during signature execution
> 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?

ia.cr/2023/050
Bibliography


Least squares method (LSM)

If \((\tilde{w}_0)^m_{i,j} = 0\) but \((w_0)_{i,j} \neq 0\),
\[
\begin{align*}
(A z - c t_1 2^d)_{j} - (w_1)_j 2 \gamma^2 &= \hat{c} (t_0 - s_2)_j + e \\
L &= \hat{c}\langle t_0 - s_2 \rangle + e
\end{align*}
\]

with \(\|e\| < \varepsilon\) thanks to the filter
\[\|c (t_0 - s_2) + e\| < q \implies \text{no modular reduction}\]

We get a candidate by using the LSM

\[
(t_0 - s_2) = (\tilde{C}^T \tilde{C})^{-1} \tilde{C}^T L
\]

If \(\|(t_0 - s_2) - (t_0 - s_2)\|_\infty < \frac{1}{2}\) then \(\lfloor (t_0 - s_2) \rfloor = (t_0 - s_2)\)