

國立臺灣大學理學院應用數學科學研究所

碩士論文

Institute of Applied Mathematical Sciences

College of Science

National Taiwan University

Master Thesis



快速 Dilithium 於 Cortex-M4 平台實現的旁通道分析  
Side-Channel Analysis of Faster Dilithium on Cortex-M4

李昇峰

Sheng-Fong Li

指導教授: 陳君明 博士

Advisor: Jiun-Ming Chen Ph.D.

中華民國 112 年 6 月

June, 2023

國立臺灣大學碩士學位論文

口試委員會審定書



快速 Dilithium 於 Cortex-M4 平台實現的旁通道分析

Side-Channel Analysis of Faster Dilithium on Cortex-M4

本論文係李昇峰君 (R10246010) 在國立臺灣大學應用數學科學研究所完成之碩士學位論文，於民國 112 年 6 月 20 日承下列考試委員審查通過及口試及格，特此證明

口試委員： 陳君明

(指導教授)

陳榮傑

陳君明

楊和國

楊和國

謝致仁

所長： \_\_\_\_\_





## 摘要

隨者量子電腦的發展，後量子密碼演算法，將會取代現有的非對稱密碼系統。在 2022 年七月，美國國家標準暨技術研究院，公布了標準化的後量子數位簽章法，Crystal-Dilithium 是三個標準的其中一個，也是三個之中可以在合理時間內，於 Cortex-M4 上運行的後量子數位簽章。

2022 年一月，一種運行於 Cortex-M4 加速版本的 Dilithium 被研發出來，它在小係數多項式乘法有更快的運算，使得運行的時間被進一步地縮短，然而也使其對旁通道攻擊的弱點進一步地被放大。

本文使用了相關性能量分析攻擊 (Correlation Power Analysis) 和 T 檢定 (T-test)，將這兩種分析的方式結合，成功的攻擊了 Dilithium-2 的小係數多項式乘法，並且準確地還原其私鑰。Correlation Power Analysis 可以在短時間內，從 66049 種可能性中找出最有可能的私鑰組合，而 Profiling T-test，則可從少數的組合中找到正確的答案，形成一個快速又有效果的攻擊方式。如果沒有使用 masking 或 shuffling 進行防護，Dilithium 對於旁通道攻擊的防護是非常脆弱的。

**關鍵字：**相關性能量分析攻擊、電子簽章、快速數論變換、模板攻擊、後量子密碼、司徒頓 t 檢定





# Abstract

With the development of quantum computers, post-quantum cryptography (PQC) and its digital signatures will replace asymmetric cryptographic systems. In July 2022, the National Institute of Standards and Technology (NIST) announced the standardized Postquantum signatures. Crystal-Dilithium is one of the three digital signature standards, and it is also one of the three that can run on the Cortex-M4 in a reasonable time. In January 2022, a faster version of Dilithium was developed. It has faster operations in small coefficient polynomial multiplication, further shortening the running time and amplifying its vulnerability to side-channel attacks.

This article uses the combination of Correlation Power Analysis and Profiling T-test to successfully attack Dilithium-2's small coefficient polynomial multiplication to recover its sensitive information  $s_1$  and  $s_2$ . Correlation Power Analysis can find the most likely  $s_1$  and  $s_2$  coefficient pairs from 66049 possibilities quickly. In contrast, the Profiling T-test can find the correct answer from a few candidates, forming a fast and effective attack

method. Without the countermeasure of masking or shuffling for protection, Dilithium will be very vulnerable to side-channel attacks.



**Keywords:** Correlation Power Analysis, Digital Signature, Number Theoretic Transform, Online Template Attack, Post-quantum Cryptography, Welch' s T-test

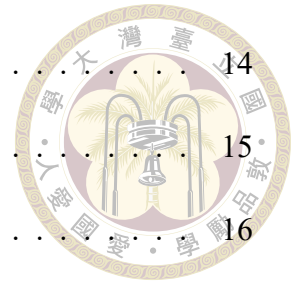


# Contents

	<b>Page</b>
<b>Verification Letter from the Oral Examination Committee</b>	<b>i</b>
<b>摘要</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>Denotation</b>	<b>xiii</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Background</b>	<b>3</b>
2.1 Notation . . . . .	3
2.2 Dilithium Signature . . . . .	3
2.3 Faster Dilithium on Cortex-M4 . . . . .	7
2.3.1 Dilithium2 and Dilithium5 . . . . .	7
2.3.2 Dilithium3 . . . . .	8
2.4 Side Channel Attack on Dilithium . . . . .	10
<b>Chapter 3 Power Side-Channel Leakage in Dilithium</b>	<b>13</b>
3.1 Hamming weight leakage attack . . . . .	13



3.1.1	Hamming weight leakage in Dilithium3 . . . . .	14
3.2	Profiling T-test . . . . .	15
3.3	Online Template Attack (OTA) . . . . .	16
3.4	Decision Tree . . . . .	17
<b>Chapter 4</b>	<b>Experiment setup and Results</b>	<b>19</b>
4.1	Experiment setup . . . . .	19
4.2	Hamming weight leakage attack Results . . . . .	19
4.2.1	CPA Attack Result of Dilithium3 . . . . .	21
4.3	Profiling T-test attack Results . . . . .	22
4.4	Online Template Attack Results . . . . .	23
<b>Chapter 5</b>	<b>Countermeasures</b>	<b>25</b>
5.1	Countermeasures on Faster Dilithium . . . . .	25
5.2	Countermeasure result . . . . .	26
<b>Chapter 6</b>	<b>Conclusions</b>	<b>29</b>
<b>References</b>		<b>31</b>





# List of Figures

2.1	Polynomial multiplication in Original Dilithium . . . . .	5
2.2	Small-coefficient polynomial multiplication in Dilithium2 and Dilithium5	7
2.3	Assembly code of small-coefficient polynomial multiplication in Dilithium2 and Dilithium5 . . . . .	9
2.4	Small-coefficient polynomial multiplication in Dilithium3 . . . . .	9
2.5	Assembly code of small-coefficient polynomial multiplication in Dilithium3	10
3.1	Decision tree of Dilithium attack . . . . .	17
4.1	Success case of attack odd and even coefficients (73,-86) . . . . .	19
4.2	Failure case of attack coefficients(-11,29) . . . . .	20
4.3	CPA attack of Dilithium3 . . . . .	21
4.4	Profiling t-test of (-46,57,-27,7)(blue), (-92,114,-54,14)(red), (-92,114,- 108,28)(green) and answer is (-46,57,-54,14) . . . . .	22
5.1	T-Test evaluation on unmasking in $s_1c$ (2000 traces) . . . . .	26
5.2	T-Test evaluation on first order masking in $s_1c$ (100000 traces) . . . . .	26





# List of Tables

2.1 Parameters of Dilithium in different security levels . . . . .	4
--	---





# Denotation

TLS	Transport Layer Security (傳輸層安全性協定)
PQC	Post-quantum cryptography (後量子密碼)
NIST	National Institute of Standards and Technology (美國國家標準暨技術研究院)
NTT	Number Theoretic Transform
INTT	inverse number theoretic transform
FNT	Fermat number transform
CPA	Correlation Power Analysis





# Chapter 1 Introduction

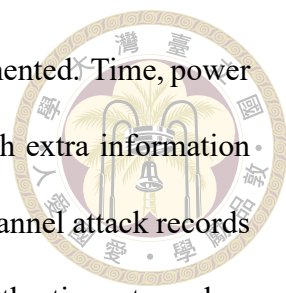
Key cryptography is a system that uses key pairs consisting of a public key and corresponding secret key, and it is fundamental security primitive in modern cryptosystems, such as Transport Layer Security (TLS), Secure Shell Protocol (SSH), digital signatures, and others.

However, conventional public key systems, like RSA or Elliptic Curve Cryptography (ECC), whose security is based on the complex problem of large integer factorization, will be broken by quantum computers. That is the reason why Post-quantum cryptography (PQC) is necessary. PQC refers to public key cryptographic algorithms that are secure against quantum and classical computers.

CRSTALS-Dilithium is a National Institute of Standards and Technology (NIST) recommended public key digital signature. The secret key and public key are matrixes that have many polynomials. It has many operations of polynomial multiplications in the signing operation. To accelerate the process's speed, [1] construct a faster number theoretic transform (NTT) and polynomial ring multiplication to improve efficiency. However, the paper only focuses on the efficiency of the Dilithium but ignores the weak point in the design of the algorithm, which can be side-channel attacked.

A side-channel attack can recover secret information by gathering extra information





because of the fundamental way the cryptographic algorithm is implemented. Time, power consumption, electromagnetic leaks, and sound are examples of such extra information which can be used for attack. For instance, in [5], this kind of side channel attack records many traces of the power consumption in Dilithium signing, using the time step when the sensitive information  $s_1$  and  $s_2$  are multiplying other values to recover the part of the secret key.

We construct a side-channel attack based on the paper [5] that attacks the polynomial multiplication of the sensitive information in the signing process to recover the secret key. A similar concept is used to attack the Faster Dilithium and get an 82.5% success rate. And the failure cases are related to the correct keys with a power of 2. This kind of case can be solved by generating the power traces with different possible key pairs and using Welch' s T-test or Online Template Attack (OTA) to find the correct key pair. And OTA can use fewer power traces (100 traces) than the Profiling T-test attack (about 1000 traces). Finally, the attacker can combine the two strategies to recover the small-coefficient  $s_1$  and  $s_2$ .

To prevent the attack, paper [12] has a masking countermeasure that divides the  $s_1$  and  $s_2$  into many arithmetic shares and then can against the power analysis attack. Although the countermeasure is time-consuming, this method is still a practical protection of faster Dilithium.



## Chapter 2 Background

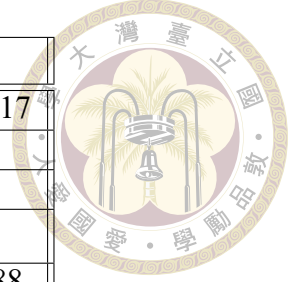
### 2.1 Notation

In latticed-based post-quantum cryptography, every element is executed in a polynomial ring  $R_q = Z_q[x]/(X^n + 1)$ . Every matrix and vector consists of numbers of polynomials. And  $\forall a, b \in R_q \rightarrow ab \in R_q$ .  $R_\eta^k$  means a matrix with  $k$  polynomials and the maximum absolute value of the coefficient doesn't exceed  $\eta$ . For  $z \in R_q$ ,  $z_i$ ,  $0 \leq i \leq 255$  implies the  $i$ th coefficient of  $z$  and  $z_i \in [-q/2, q/2]$ .  $\|z\|_\infty$  refers to the maximum absolute value of coefficient in  $z$ . The  $\hat{c}$  means the polynomial  $c$  in the NTT domain.

### 2.2 Dilithium Signature

Dilithium is a latticed-based post-quantum digital signature, whose security is based on the hardness of finding short vectors in lattices.[2] The scheme is based on the "Fiat-Shamir with Aborts" approach[9, 10] and is similar to the schemes proposed in [3, 8].

Dilithium depends on parameters  $q, d, \tau, \gamma_1, \gamma_2, k, l, \eta, \beta$  and  $w$ . Some parameters will be different because of the NIST Security Level. In the following section, we will talk about the processes of the key generation, signing, and verification of Dilithium.



NIST Security Level	II	III	V
$q$	8380417	8380417	8380417
$d$	13	13	13
$\tau$	39	49	60
$\gamma_1$	$2^{17}$	$2^{19}$	$2^{19}$
$\gamma_2$	95232	261888	261888
$(k, l)$	(4,4)	(6,5)	(8,7)
$\eta$	2	4	2
$\beta = \tau \cdot \eta$	78	196	120
$w$	80	755	75

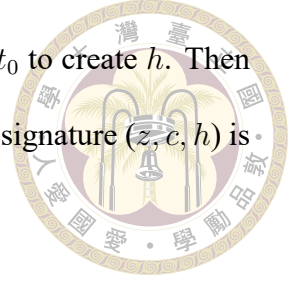
Table 2.1: Parameters of Dilithium in different security levels

### Key generation

In the key generation process at first, it constructs  $\rho, \rho'$ , and  $K$  by the hash value of initial value  $\zeta$ . And then it creates a  $A$  from 256 bits of  $\rho$  as part of the public key, and  $A$  is a matrix with  $k \times l$  size. And  $s_1 \in R_\eta^k, s_2 \in R_\eta^l$  from 256 bits of  $\rho'$  as part of the secret key.  $As_1 + s_2 = T$ , and use the function Power2round to divide  $t$  into vectors  $t_1$  and  $t_0$ .  $t_1$  is part of the public key and  $t_0$  is part of the secret key. Finally, the public key is the package of  $(\rho, t_1)$ , and the secret key is the package of  $(\rho, K, tr, s_1, s_2, t_0)$ .

### Signing

The signature masking process in Dilithium is not a time-constant process. At first, a matrix  $y \in R_{\gamma_1}^k$  is created.  $Ay = w$  and use the function Highbits to create  $w_1$ .  $w_1$  and hashed executed message  $\mu$  are hashed to create a binary polynomial, which means the coefficient can only be  $\pm 1$  and 0, and the numbers of  $\pm 1$  are  $\tau$ .  $y + cs_1 = z$  and  $\text{Lowbits}(w - cs_2) = r_0$ . And then the legitimacy of the signature will be tested. If the value of  $\|z\|_\infty$  and  $\|r_0\|_\infty$  do not meet the requirements. The process will go back to the  $y$  construction.



If passing the requirements, Makehint uses  $ct_0$  and  $w - cs_2 + ct_0$  to create  $h$ . Then test the legitimacy of the signature again. Finally, the valid Dilithium signature  $(z, c, h)$  is returned.

### Verification

In verification,  $A, z, c, t_1$ , and  $h$  are used to reproduce the value of  $w_1$ . The hashed message  $\mu$  and  $w_1$  are hashed to create another binary polynomial that is the same as  $c$  if the signature is valid. And the value of  $\|z\|_\infty$  and #'s of  $h$  which are tested in the signing process is also verified.

---

#### Algorithm 1 Key generation

---

```

 $\zeta = \{0, 1\}^{256}$ 
 $(\rho, \rho', K) \in \{0, 1\}^{256 \times 3} = H(\zeta)$ 
 $A \in R_q^{k \times l} = ExpandA(\rho)$ 
 $(s_1, s_2) \in S_\eta^l \times S_\eta^k = ExpandS(\rho')$ 
 $T = As_1 + s_2$ 
 $(t_1, t_0) = Power2Round(T, d)$ 
 $tr \in \{0, 1\}^{256} = H(\rho || t_1)$ 
return  $(pk = (\rho, t_1), sk = (\rho, K, tr, s_1, s_2, t_0))$ 

```

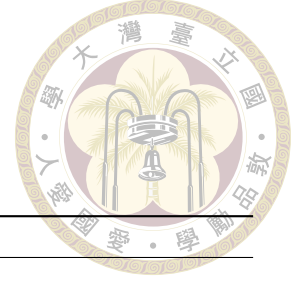
---

### Number theoretic transform (NTT) and polynomial multiplication



Figure 2.1: Polynomial multiplication in Original Dilithium

In every process of Dilithium, it has many multiplication operations. To improve efficiency, the Number theoretic transform (NTT) is used. In the “fully-splitting” Dilithium




---

**Algorithm 2 Sign**


---

$A \in R_q^{k \times l} = \text{Expand}A(\rho)$   
 $\mu \in \{0, 1\}^{512} = H(\text{tr}||M)$   
 $\kappa \leftarrow 0, (z, h) = \perp$   
 $\rho' \in \{0, 1\}^{512} = H(K||\mu)$  (or  $\rho' \leftarrow \{0, 1\}^{512}$  for randomized signing)  
**while**  $(z, h) = \perp$  **do**  
     $y \in S_{\gamma_1}^l = \text{ExpandMask}(\rho', \kappa)$   
     $w = Ay$   
     $w_1 = \text{Highbits}(w, 2\gamma_2)$   
     $c = H(\mu||w_1)$   
     $z = y + cs_1$   
     $r_0 = \text{Lowbits}(w - cs_2, 2\gamma_2)$   
    **if**  $\|z\|_\infty \geq \gamma_1 - \beta$  **or**  $\|r_0\|_\infty \geq \gamma_2 - \beta$  **then**  
         $(z, h) = \perp$   
    **else**  
         $h = \text{MakeHint}(-ct_0, w - cs_2 + ct_0, 2\gamma_2)$   
        **if**  $\|ct_0\|_\infty \geq \gamma_2$  **or**  $\#$  of 1's in  $h$  is greater than  $\omega$  **then**  
             $(z, h) = \perp$   
        **end if**  
    **end if**  
     $\kappa = \kappa + l$   
**end while**  
**return**  $(c, z, h)$

---



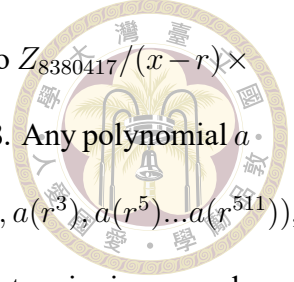
---

**Algorithm 3 Verify**


---

$A \in R_q^{k \times l} = \text{Expand}A(\rho)$   
 $\mu \in \{0, 1\}^{512} = H(H(\rho||t_1)||M)$   
 $w'_1 = \text{UseHint}(h, Az - ct_1 \cdot 2^d, 2\gamma_2)$   
**return**  $[\|z\|_\infty < \gamma_1 - \beta]$  **and**  $[c = H(\mu||w'_1)]$  **and**  $[\#$  of 1's in  $h \leq \omega]$

---



NTT algorithm, the polynomial ring  $Z_{8380417}/(x^{256} + 1)$  is computed to  $Z_{8380417}/(x - r) \times Z_{8380417}/(x - r^3) \times Z_{8380417}/(x - r^5) \times \dots Z_{8380417}/(x - r^{511})$ ,  $r = 1753$ . Any polynomial  $a$  in  $Z_{8380417}/(x^{256} + 1)$  can be transformed into NTT-domain  $\hat{a}$  like  $(a(r), a(r^3), a(r^5) \dots a(r^{511}))$ , the the multiplication of polynomial  $a$  and  $b$  in NTT-domain is coordinate-wise inner product  $(a(r) \cdot b(r), a(r^3) \cdot b(r^3), a(r^5) \cdot b(r^5) \dots a(r^{511}) \cdot b(r^{511}))$ . And finally, use the inverse-NTT algorithm to recover to the ordinary domain.

## 2.3 Faster Dilithium on Cortex-M4

### 2.3.1 Dilithium2 and Dilithium5

	[ int32 ]	[ int32 ]	
NTT(c)	C[0]	C[1]	C[2] C[3] Range:[-128,128]
Precompute(NTT(c))	CP[0] = C[0]	CP[1] = (C[1]*27)%257	CP[2] = C[2] CP[3] = (C[1]*-27)%257 Range:[-128,128]
NTT(s)	S[0]	S[1]	S[2] S[3] Range:[-128,128]
NTT(z)	Z[0]=S[0]*CP[0]+S[1]*CP[1]	Z[1]=S[0]*C[1]+S[1]*C[0]	Range:[-32768,32768]
	[ int32 ]		

Figure 2.2: Small-coefficient polynomial multiplication in Dilithium2 and Dilithium5

In latticed-based post-quantum cryptography, the multiplication of polynomial rings is often executed. To get high efficiency, the number-theoretic-transforms (NTT) is designed to deal with the polynomial for fastening the multiplication in the microcontroller like Arm Cortex-M4.

In [1], a faster NTT way is used to speed the process of  $cs_1$  and  $cs_2$ , due to the small absolute value coefficients of  $s_1, s_2$  and  $c$ . In Dilithium2 and Dilithium5, the absolute

value of coefficients in  $cs_1$  and  $cs_2$  do not exceed the  $\beta = \eta\tau$ , so the module  $q$  can be switched to  $q' = 257 > 2\beta$ , which is the Fermat prime that can use a special kind of NTT named the Fermat number transform (FNT). In Dilithium3, the  $q'$  is switched to 769 because of the bigger  $\beta$ , although the NTT calculation is still faster than the original module version.

In the original Dilithium NTT, 8 layers are computed and coefficients are multiplied point by point in polynomial multiplication to generate the NTT executed result. Then the result will be recovered to the ordinary by using Inverse-NTT (INTT). However, faster NTT in [1] computes 7 layers instead of 8. So the polynomial multiplication is composed of 128 2 schoolbook multiplications. In computing  $\hat{z}_{2i} + z_{2i+1}x = (\hat{c}_{2i} + c_{2i+1}x)(\hat{s}_{2i} + s_{2i+1}x) \bmod (x^2 - twiddles)$ . We have  $\hat{z}_{2i} = \hat{z}_{2i}\hat{z}_{2i} + z_{2i+1}z_{2i+1}twiddles$ ,  $z_{2i+1} = \hat{z}_{2i}z_{2i+1} + z_{2i+1}\hat{z}_{2i}$ .

### 2.3.2 Dilithium3

Dilithium3, different from Dilithium2 and Dilithium5, uses 769 as the modulus number. So as the way of NTT. But the storage of binomial polynomial and secret keys in the NTT domain is similar. There are two numbers stored in the 32-bit register. And in multiplication, the even and odd coefficients are counted by the SMUAD and SMUADX instructions. The only different way is that the result coefficients are stored in the 32-bit register with two numbers. And finally, the number  $2i$  and  $2i + 1$  coefficients are stored in the array at one time.



```

.align 2
.global __asm_asymmetric_mul_257_16
.type __asm_asymmetric_mul_257_16, %function
__asm_asymmetric_mul_257_16:
    push.w {r4-r11, lr}

    .equ width, 4

    add.w r12, r0, #256*width
    _asymmetric_mul_16_loop:

    ldr.w r7, [r1, #width]
    ldr.w r4, [r1], #2*width
    ldr.w r8, [r2, #width]
    ldr.w r5, [r2], #2*width
    ldr.w r9, [r3, #width]
    ldr.w r6, [r3], #2*width

    smuad r10, r4, r6    Z[0]=S[0]*CP[0]+S[1]*CP[1]
    smuadx r11, r4, r5  Z[1]=S[0]*C[1]+S[1]*C[0]

    str.w r11, [r0, #width]
    str.w r10, [r0], #2*width

    smuad r10, r7, r9    Z[2]=S[2]*CP[2]+S[3]*CP[3]
    smuadx r11, r7, r8  Z[3]=S[2]*C[3]+S[3]*C[2]

    str.w r11, [r0, #width]
    str.w r10, [r0], #2*width

    cmp.w r0, r12
    bne.w _asymmetric_mul_16_loop

    pop.w {r4-r11, pc}

```

Figure 2.3: Assembly code of small-coefficient polynomial multiplication in Dilithium2 and Dilithium5

	[	int32	]	[	int32	]			
NTT(c)		C[0]		C[1]		C[2]		C[3]	Range: [-7q', 7q']
Precompute(NTT(c))		CP[0] = C[0]		CP[1] = (C[1]*-23*9) %769		CP[2] = C[2]		CP[3] = (C[1]*23*9) %769	Range: [-7q', 7q']
NTT(s)		S[0]		S[1]		S[2]		S[3]	Range: [-7q', 7q']
NTT(z)		Z[0] = (S[0]*CP[0] + S[1]*CP[1])*9 %769		Z[1] = (S[0]*CP[1] + S[1]*CP[0])*9 %769		Z[2] = (S[2]*CP[2] + S[3]*CP[3])*9 %769		Z[3] = (S[2]*CP[3] + S[3]*CP[2])*9 %769	Range: [-384, 384]

Figure 2.4: Small-coefficient polynomial multiplication in Dilithium3





```

.align 2
.global small_asymmetric_mul_asm
.type small_asymmetric_mul_asm, %function
small_asymmetric_mul_asm:
    push.w {r4-r11, lr}

    movw r14, #769
    movt r14, #767
    .equ width, 4
    add.w r12, r0, #256*2
    _asymmetric_mul_16_loop:
    ldr.w r7, [r1, #width]
    ldr.w r4, [r1, #2*width]
    ldr.w r8, [r2, #width]
    ldr.w r5, [r2, #2*width]
    ldr.w r9, [r3, #width]
    ldr.w r6, [r3, #2*width]

    smuad r10, r4, r6
    montgomery r14, r14, r10, r6
    smuadx r11, r4, r5
    montgomery r14, r14, r11, r10

    pkhtb r10, r10, r6, asr#16

    str.w r10, [r0], #width

    smuad r10, r7, r9
    montgomery r14, r14, r10, r6
    smuadx r11, r7, r8
    montgomery r14, r14, r11, r10

    pkhtb r10, r10, r6, asr#16
    str.w r10, [r0], #width

    cmp.w r0, r12
    bne.w _asymmetric_mul_16_loop

```

$$Z[0] = (S[0] * CP[0] + S[1] * CP[1]) * 9\%769$$

$$Z[1] = (S[0] * C[1] + S[1] * C[0]) * 9\%769$$

$$Z[2] = (S[2] * CP[2] + S[3] * CP[3]) * 9\%769$$

$$Z[3] = (S[2] * C[3] + S[3] * C[2]) * 9\%769$$

Figure 2.5: Assembly code of small-coefficient polynomial multiplication in Dilithium3

## 2.4 Side Channel Attack on Dilithium

### Correlation Pearson Attack

The main idea of the attack on Dilithium is from the [5]. It provides an efficient side-channel attack on the polynomial multiplication process to recover the sensitive information  $s_1$ ,  $s_2$ , and  $t_0$  without invading the devices. It is called the Correlation Power Analysis(CPA). There are five steps:

(1) Find a good intermediate value in the point of interest that is a function of the secret information and a known variable.

(2) Record  $n$  power consumption traces each with  $m$  power samples to construct a



matrix  $T_{n \times m}$

(3) Construct the intermediate value matrix  $V_{n \times k}$ . Then fill the matrix according to the function of the known ciphertext and the possible hypothesis key.

(4) Use the matrix  $V$  to create a matrix  $H_{n \times k}$  with the same size, the value of  $H_{i,j}$  is corresponding to the power model (such as Hamming weight) of  $V_{i,j}$ .

(5) Calculate the correlation coefficients matrix  $R_{k \times m}$  by the column of  $H_{n \times k}$  and  $T_{n \times m}$ . Pearson correlation of columns  $H_i$  and  $T_j$  is computed below, in which  $\bar{H}_i$  and  $\bar{T}_j$  are the average of the column.

$$R_{i,j} = \frac{\sum_{x=1}^n (H_{x,i} - \bar{H}_i)(T_{x,j} - \bar{T}_j)}{\sqrt{\sum_{x=1}^n (H_{x,i} - \bar{H}_i)^2 \cdot \sum_{x=1}^n (T_{x,j} - \bar{T}_j)^2}}$$

The index of the maximum absolute value in  $R$  is the correct time in the target process with the correct key used in the device.

### Welch's t-test

In statistics, Welch's t-test is used to test whether two sets A and B are different. The null hypothesis is that the two sets are the same population. That is to say their means  $\mu_A$  and  $\mu_B$  are the same. Let  $L_A$  and  $L_B$  represent their means,  $s_A^2$  and  $s_B^2$  represent the sample variance,  $n_A$  and  $n_B$  represent the number of elements in each set. Then the t-test value is given by

$$t_u = \frac{L_A - L_B}{\sqrt{\frac{s_A^2}{n_A} + \frac{s_B^2}{n_B}}}$$

The null hypothesis is rejected if the absolute value of  $|t_u|$  exceeds some threshold. In cryptographic security, Welch's t-test is used to find the possible leakage of the device

under test by creating two sets with different kinds of input.[6] That is to say, it also can be used in side-channel attacks to find the weak point of the device.



### **Online Template Attack (OTA)**

Online Template Attack, presented by [4], the attack is characterized online because the attacker creates the template after getting the information about the victims (like power traces). The attack functions by getting a few power traces from the device of the victims and comparing it with the templates created from the devices of the attackers that are similar to the victims. Pattern matching occurs if the input key of the victims is the same as the guessing key by the attacker. This causes the power traces to be the most similar to other wrong keys.

This attack is originally used to attack the elliptic curve. But with a little change, it also can be used to attack other kinds of cryptography.



## Chapter 3 Power Side-Channel Leakage in Dilithium

In the side-channel attack, for recovering the secret key of Dilithium, the attack has to get the power traces of the victim's devices and the corresponding signature  $c$ .

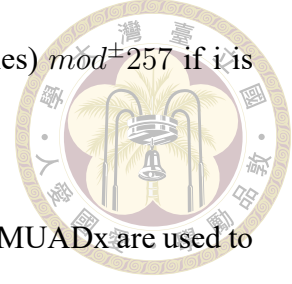
### 3.1 Hamming weight leakage attack

In this section, we analyze the code of faster Dilithium2 and Dilithium5 in [1] and find that it has similar leakage to the original Dilithium that can be attacked like [5].

The multiplication of small coefficient polynomial  $cs_1$  and  $cs_2$  is an appropriate point to be attacked by CPA. In the multiplication, the sensitive information  $s_1$  and  $s_2$  are multiplied by a public binary polynomial  $c$  just like [5]. Even though the coefficient values are executed in the incomplete NTT domain, they can be easily recovered to the normal polynomial with INTT.

After the NTT, the small coefficient polynomial with 256 coefficients is saved in 128 32-bit integers. A 32-bit integer is divided into 16-bit bottom halfword and top halfword to save the NTT-executed coefficients with range  $[-128, 128]$ . Furthermore, a new array with 256 16-bit integers named `cprime` is created. Its coefficient value depends on the

value of  $\hat{c}$ .  $cprime[i]=c[i]$  if  $i$  is even and  $cprime[i]=(c[i] \times twiddles) \bmod 257$  if  $i$  is odd. The range is also  $[-128, 128]$ .



In polynomial multiplication, the assembly code SMUAD and SMUADx are used to execute the schoolbook multiplication, SMUAD on  $cprime$  and  $\hat{s}$  is the same as  $\hat{z}_{2i} = s_{2i}c_{2i} + s_{2i+1}c_{2i+1}twiddles$ , and SMUADx on  $\hat{c}$  and  $\hat{s}$  is the same as  $\hat{z}_{2i+1} = s_{2i}c_{2i+1} + s_{2i+1}c_{2i}$ . And the result is finally saved in the 32-bit integer, which has the leakage message about Hamming weight. That can be used to recover the  $s_1$  and  $s_2$  coefficient by CPA attack.

In schoolbook multiplication, there are two coefficients calculated with two pairs of coefficients, so in the CPA attack, the key range of coefficients is  $[-128, 128] \times [-128, 128]$  with the  $257^2 = 66049$  key guessing table. This makes the CPA on Dilithium easier than [5] with  $2^{27}$  but harder than Advanced Encryption Standard (AES) with  $2^8$ .

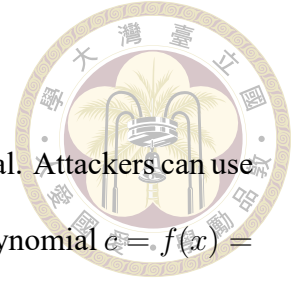
Furthermore, every pair of coefficients is used to make two result coefficients, so it has two leakage points. We call the leakage of  $\hat{z}_{2i}$  as even coefficients and  $\hat{z}_{2i+1}$  as odd coefficients.

### 3.1.1 Hamming weight leakage in Dilithium3

In Dilithium3, the polynomial multiplication is almost the same in Dilithium2 and Dilithium5. The only difference is the storage way of the result coefficient. In Dilithium3, it stores both the  $2i$ th and  $2i + 1$ th coefficients in the 32-bit register at the same time. So it leakage the hamming weight of two 16-bit numbers. And the searching space is  $[-384, 384] \times [-384, 384]$ . Because in INTT execution, the number that is greater than 384 or lower than -384 will be mod 769 to the range  $[-384, 384]$ . The number of the key

range is  $769^2 = 591361$ .

After getting 256 coefficients of the secret information polynomial. Attackers can use small-coefficient polynomial multiplication to times the binomial polynomial  $c = f(x) = 1$ . And finally, attackers can use the INTT execution to recover the secret information polynomial.



## 3.2 Profiling T-test

In this section, the attacker has to own a cryptographic device that is similar to the victim's device. In polynomial multiplication, the attacker knows the signatures' coefficients  $(\hat{c}_{2i}, \hat{c}_{2i+1})$  and the corresponding power trace but doesn't know the sensitive information  $s_1$  and  $s_2$  coefficients  $(\hat{s}_{2i}, \hat{s}_{2i+1})$ .

In the Profiling T-test attack, the attacker used his own device to do the polynomial multiplication with known signatures  $c$  and every possible sensitive information coefficient pair in  $[-128, 128] \times [-128, 128]$ , record the power traces, and classify them with the different sensitive information coefficient pairs. Then the attack gets a set of power traces  $A$  from the victim and 66049 other sets  $B_i$ , and the attacker can use Welch's T-test to create 66049 t-values between  $A$  and  $B_i$ .

We can say that a similar device will record two similar power traces if the two inputs are the same. That is to say, the t-value of the part of the traces will be small. We know that one of the  $B_i$  set is the correct sensitive information polynomial coefficients that is the same as  $A$ , the attack can use the part of the t-values and sum them with absolute values and find the smallest  $B_i$ , the corresponding coefficients is the correct sensitive information coefficients that the victim use.



### 3.3 Online Template Attack (OTA)

The attack of profiling T-test needs to construct many power traces by the attacker, and it costs many times and resources. However, the Online Template Attack (OTA), presented by [4] supplies a kind of side-channel attack that the attacker can construct less amount of power traces than before. It also needs that the device of the attacker is similar to the victim.

The Online Template Attack also focuses on polynomial multiplication. With a power trace recorded from the victim by known signature  $(\hat{c}_{2i}, \hat{c}_{2i+1})$  and unknown sensitive information  $(\hat{s}_{2i}, \hat{s}_{2i+1})$  coefficient pair. The attacker constructs many power traces with the corresponding  $(\hat{c}_{2i}, \hat{c}_{2i+1})$  and guessing keys. If the guessing key is the correct key that the victim uses, the power trace generated by the attacker will be the most similar to the victim's power trace to other wrong keys. The similarity between the power traces depends on the absolute value norm of the victim's power trace and the attacker's guessing key power trace.

---

**Algorithm 4** Online Template Attack on polynomial multiplication

---

Input: Numbers of signature coefficients  $(\hat{c}_{2i}, \hat{c}_{2i+1})_l$  and corresponding piece of power trace  $P_l, 1 \leq l \leq n$ , and guessing sensitive information coefficient  $S_j, 1 \leq j \leq m$ .

```
Rank[m]=0
for l in n do
  for j in m do
     $T_j$ = power trace of multiplication by  $(\hat{c}_{2i}, \hat{c}_{2i+1})_l$  and  $S_j$ 
  end for
   $x^* = \min(x | \|P_l - T_x\|)$ 
  Rank[ $x^*$ ] +=1
end for
return correct coefficients  $S_j$  that  $j = \max(x | \text{Rank}[x])$ 
```

---



### 3.4 Decision Tree

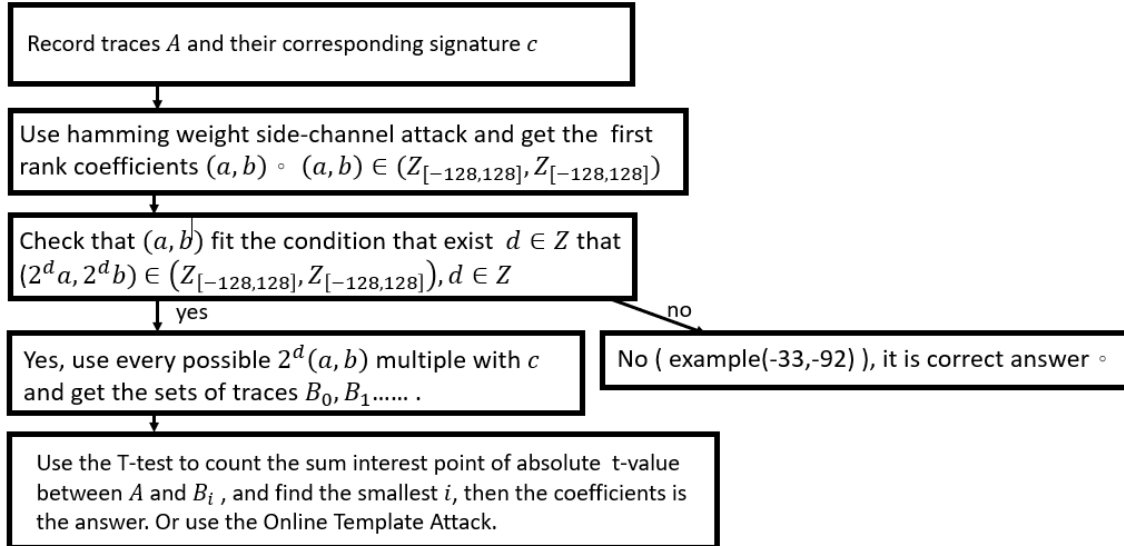


Figure 3.1: Decision tree of Dilithium attack

The two ways described before are both can be used to recover the sensitive information coefficients of  $s_1$  and  $s_2$ . However, they also have some kinds of flaws. In Hamming weight leakage attacks, the coefficients of the maximum absolute value of the correlation value may not be the correct answer, it may have some kind of relevance with the correct coefficients. Those things are shown in Sections 4-2.

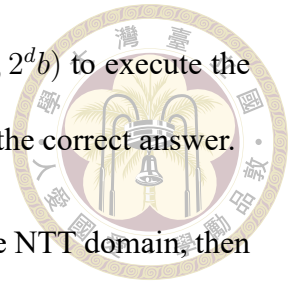
The profiling T-test, for using it to attack needs to generate a very large number of traces. For example, the number of recording traces from the victim’s device is 2000. Then the attack has to generate  $66049 \times 2000$  traces, which is very time-consuming and memory-consuming. For getting a practical attack, a kind of decision tree is constructed.

As shown in the Figure. 3.1, the attacker first collects the power traces and signatures and uses them to execute the hamming weight leakage side-channel attack to get the most possible first-rank coefficients  $(a, b) \in Z_{[-128,128]} \times Z_{[-128,128]}$ . Then what the attacker has to do depends on the value of  $(a, b)$ . If exist other nonzero integers  $d \in Z \setminus \{0\}$  that



$(2^d a, 2^d b) \in Z_{[-128,128]} \times Z_{[-128,128]}$  Then collect all the sets of  $(2^d a, 2^d b)$  to execute the profiling T-test or the online template attack. Otherwise, the  $(a, b)$  is the correct answer.

If the attacker collects 128 pairs of  $(a, b)$  of the polynomial in the NTT domain, then the attacker can use the INTT algorithm to recover the sensitive information polynomial to recover part of secret key  $s_1$  and  $s_2$ .





# Chapter 4 Experiment setup and Results

## 4.1 Experiment setup

The victim device is a CW308U board equipped with an STM32F415 chip. The chip runs the small-coefficient polynomial multiplication of round 3 faster Dilithium code. The data is delivered to the computer with a USB cable. The Correlation Power Analysis on data is executed by Python3 in the computer.

## 4.2 Hamming weight leakage attack Results

Attack z[0]				Attack z[1]			
[ 73.	-86.	284.	0.8271782 ]	[ 73.	-86.	269.	-0.96151122]
[ -73.	86.	284.	-0.81460753]	[ -73.	86.	269.	0.94880769]
[ 17.	-20.	284.	0.80546739]	[ 18.	-21.	269.	-0.9329573 ]
[ 34.	-40.	284.	0.80464334]	[ -18.	21.	269.	0.93242945]
[ 68.	-80.	284.	0.80367174]]	[ 36.	-42.	269.	-0.93234306]]

Figure 4.1: Success case of attack odd and even coefficients (73,-86)

The experiment uses the trace number  $N = 2000$  to attack the even and odd coefficients. They both have a success rate of 82.5% (33/40). For  $N = 10000$ , They both have a success rate of 85 % (17/20).

In the experiment, the maximum correlation value of the correct key set is always

positive on attacking even coefficients, on the other hand, the peak value on attacking odd result coefficients is minus.



[ [-44.	116.	305.	-0.98719936]
[ -22.	58.	305.	-0.98696398]
[ -11.	29.	305.	-0.986682 ]
[ 11.	-29.	305.	0.97440054]
[ 22.	-58.	305.	0.9740579 ]]

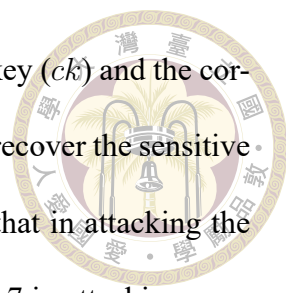
Figure 4.2: Failure case of attack coefficients(-11,29)

In the failure attacking case, we find that the top ranking (A, B) of the sensitive information coefficients set has a correlation with the answer (a,b). For example, if the answer coefficients set is (24, 12), the top rankings are (96, 48),(48, 24),(24, 12), (12, 6), (6, 3), the top rankings are 2 power of the answer.  $(A, B) = 2^d(a, b), d \in Z$ . And this problem can not be solved by ascending the number of traces or attacking odd or even coefficients.

It is because for sensitive information coefficients  $(\hat{s}_0, \hat{s}_1)$  and ciphertext coefficients  $(\hat{c}_0, \hat{c}_1)$ ,  $HW(2^k s_0 \hat{c}_0 + 2^k s_1 \hat{c}_1) = HW(2^k(\hat{s}_0 \hat{c}_0 + \hat{s}_1 \hat{c}_1)) = HW(\hat{s}_0 \hat{c}_0 + \hat{s}_1 \hat{c}_1)$  if  $\hat{s}_0 \hat{c}_0 + \hat{s}_1 \hat{c}_1 \geq 0$ . And  $HW(2^k s_0 \hat{c}_0 + 2^k s_1 \hat{c}_1) = HW(\hat{s}_0 \hat{c}_0 + \hat{s}_1 \hat{c}_1) - k$  if  $\hat{s}_0 \hat{c}_0 + \hat{s}_1 \hat{c}_1 < 0$ . It causes the result Hamming weight correlation of  $(\hat{s}_0, \hat{s}_1)$  and  $2^k(\hat{s}_0, \hat{s}_1)$  are very closed to 1.

But if the top ranking does not have any other possible, for example, (1,66), one of the coefficients is odd, and twice of the coefficients are over the range  $[-128, 128] \times [-128, 128]$ .

However, this still has some problems if the top rankings have correlations between 2 power. It is hard to confirm which coefficient set is the answer. We may have to think of another way to solve this problem.



In [11],  $\rho(ck, ct)$  refers to the Pearson correlation of the correct key ( $ck$ ) and the correct time ( $ct$ ) to estimate the number of traces that the attack needs to recover the sensitive information. We use 50000 traces in the experiment. And we find that in attacking the odd coefficients, the absolute value of  $\rho(ck, ct)$  is at least 0.95, and 0.7 in attacking even coefficients. According to [11], it only needs 40 and 16 traces to recover the sensitive information. Actually, the attack can be done with such a small number of traces. However, different from the paper, the Pearson correlation of the wrong keys does not converge to 0. It is because of the power of two problems. For example, in Figure 4.2, the number of rank 3 (18,-21) times 4 equal to (72,-84) is very close to rank 1 (73,-86). The little difference causes the correlation distance can not be ascended by adding traces number.

### 4.2.1 CPA Attack Result of Dilithium3

[ [ -20.	-287.	380.	0.96189323 ]
[ 20.	287.	380.	-0.90056965 ]
[ -196.	-20.	380.	0.56181412 ]
[ 196.	20.	380.	-0.52382375 ]
[ -287.	-249.	381.	0.45819591 ] ]
0			

Figure 4.3: CPA attack of Dilithium3

Different from the result of attacking Dilithium2, the problem of 2 power doesn't exist in attacking the polynomial multiplication of Dilithium3. Because in the result coefficients, there is one modulus execution done in the result coefficients. So when the pair of secret keys both times two, the value of the result is not necessarily timed two.

And about the correct answer, the correlation must be a positive number. Figure 4.3, shows that the first correlation has a big distance from the second and even third correlation. So in attacking Dilithium3, we do not have to do the later attack and we can

find the correct answer.



### 4.3 Profiling T-test attack Results

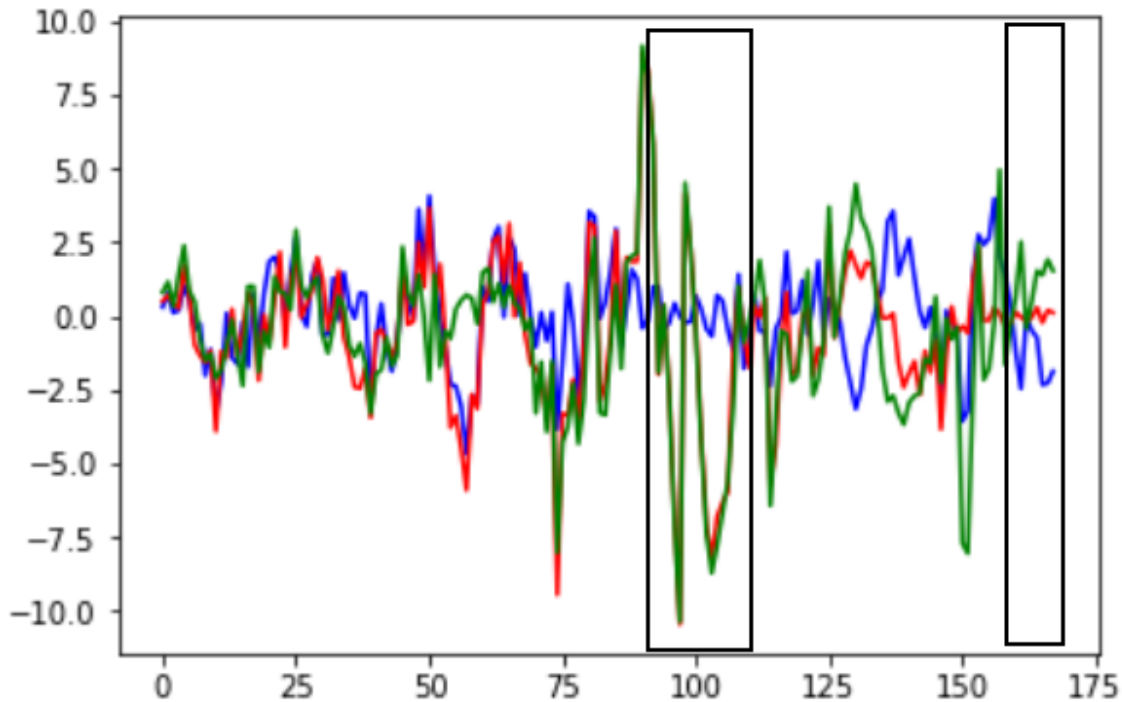
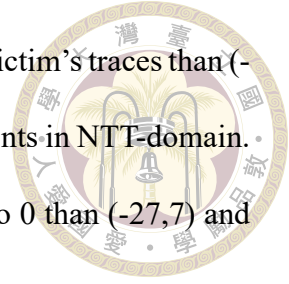


Figure 4.4: Profiling t-test of  $(-46,57,-27,7)$ (blue),  $(-92,114,-54,14)$ (red),  $(-92,114,-108,28)$ (green) and answer is  $(-46,57,-54,14)$

As shown in Figure 2.2, a loop uses two pairs of coefficients to generate four result coefficients. And every pair of coefficients may have different opportunities. For example, key pairs  $(-46,57)$  have another opportunity  $(-92,114)$ , and  $(-27,7)$  have other two opportunities  $(-54,14)$  and  $(-108,28)$ . So the attacker uses those key pairs and known signatures  $c$  from the victim to do the polynomial multiplication. In this section, the attacker uses the key pairs  $(-46,57,-27,7)$ ,  $(-92,114,-54,14)$ ,  $(-92,114,-108,28)$  and records their power traces. And uses three sets of power traces to do Welch's T-test with the set recorded from the victim. Figure 4.3 is the T-test value of the three key pairs in the loops.

As shown in Figure 4.3, the blue values from points 90 to 110 are closer to 0 than

green and red, then we know that the key pair  $(-46,57)$  is closer to the victim's traces than  $(-92,114)$ , and  $(-46,57)$  is the correct answer of the polynomial coefficients in NTT-domain. On the other hand, the red values from points 160 to 168 are closer to 0 than  $(-27,7)$  and  $(-108,28)$ , so the key pair  $(-54,14)$  is the correct answer.



## 4.4 Online Template Attack Results

For getting the correct coefficients, We use not only one power trace to do the online template attack but use at least 100 power traces to do the attack and statistics their result to find the most possible answer. However, the cost of constructing the power traces is still more less than the profiling T-test attack. For example, to find out the guessing secret coefficients of  $(6,3), (12,6), (24,12), (48,24), (96,48)$ . We use the 100 power traces and corresponding signature coefficients to do the online template attack. And there are 14 results that show that  $(6,3)$  is the correct coefficient pair. And 14 for  $(12,6)$ , 38 for  $(24,12)$ , 16 for  $(48,24)$ , 18 for  $(96,48)$ . Then, we can say that the coefficient pair  $(24,12)$  is the most possible sensitive information coefficient pair.

Using the combination of hamming weight leakage and profiling T-test or the faster online template attack, the attack can find all the correct polynomial coefficients in the NTT domain and use the INTT algorithm to get the small-coefficient polynomial  $s_1$  and  $s_2$ .





## Chapter 5 Countermeasures

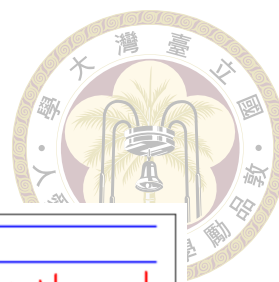
### 5.1 Countermeasures on Faster Dilithium

There are several ways that can be used to protect the signing process of Dilithium from our side-channel attack. For example, shuffling is a way that changes the order of coefficient polynomial execution, it can disrupt the execution order of polynomial multiplication while maintaining the result of multiplication, which is a feasible countermeasure method.

Masking Dilithium, described in [12], is to separate one coefficient value into two or more boolean or arithmetic shares like  $x = a_0 \oplus a_1 \cdots a_n$  and  $x = a_0 + a_1 \cdots a_n$ . Separating into  $n+1$  shares is called  $n$ -order masking. If the value of shares changes in every signing, it is hard to use the side-channel attack to recover the key.

In arithmetic masking, a secret key  $s_1$  is masking to  $s_{1,0}, s_{1,1}, s_{1,2}, \cdots s_{1,n}$  and multiple  $c$  respectively into  $z_0, z_1, \cdots z_n$ . The result polynomial  $s_1 c = z$  has a correlation that  $z \equiv z_1 + z_2 + \cdots z_n \pmod{257}$ . The result can use to generate the coefficients that are different in every signing process, so as the coefficients in the NTT domain.





## 5.2 Countermeasure result

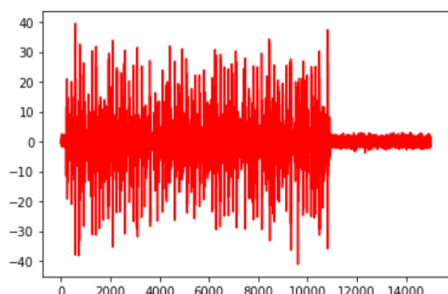


Figure 5.1: T-Test evaluation on unmasking in  $s_1c$ (2000 traces)

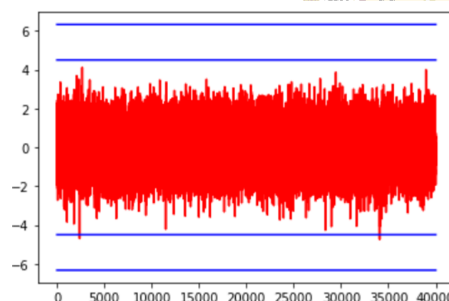


Figure 5.2: T-Test evaluation on first order masking in  $s_1c$ (100000 traces)

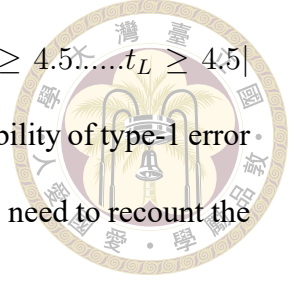
Figure 5.1 is the T-test evaluation on unmasking small-coefficient polynomial multiplication. Obviously, the t-value is over the value of 10. This shows that the unmasking multiplication is exposed to the side-channel attack.

Figure 5.2 is the T-test evaluation on first-order masking  $s_1c$  after 100000 traces. 50000 traces are generated by the fixed key  $s_1$  but with different values of shares, and the other is the random key  $s$  with shares.

In time cost, the first-order multiplication cost 37841 triggers, and the unmasking version cost 14461 triggers. The time drawn by first-order multiplication is 2.6 times that of unmasking. That is because after the polynomial multiplication, the first-order masking need to add the result of shares together to get the final result. So the time is two times longer than the original.

In Welch's T-test, there are only two points where the absolute value exceeds 4.5. The threshold value of 4.5 is computed by the type-1 error to be  $10^{-5}$ . The type-1 error is that the process leaks no information but is misjudged. However, shown in [7], when the number of points ascends, the threshold value should be changed to maintain the same probability of type-1 error. Because when the number of points ascends to  $L$  points, the probability

of at least one point being over the threshold 4.5 is  $P(t_1 \geq 4.5 \text{ or } t_2 \geq 4.5 \dots t_L \geq 4.5 |$   
leak no information)  $= 1 - (1 - 10^{-5})^L$ . When  $L = 40000$ , the probability of type-1 error  
is 0.32968129461. For maintaining the probability of type-1 error, we need to recount the  
threshold.



In [7], the threshold value is counted by

$$TH \approx CDF_{\mathcal{N}(0,1)}^{-1}\left(1 - \frac{1 - (1 - \alpha)^{\left(\frac{1}{L}\right)}}{2}\right)$$

$\alpha$  is the type-1 error probability and  $L$  is the number of points, then use  $\alpha = 10^{-5}$ ,  $L =$   
40000,  $TH \approx 6.327$ . And there is no point that the absolute value exceeds it. That is to  
say, the first-order masking on polynomial multiplication leaks no information.





## Chapter 6 Conclusions

The faster dilithium replaces the smaller the modulus number in the small-coefficient polynomial multiplication. It causes the speed of multiplication of  $cs_1$  and  $cs_2$  to become faster. However, the weakness of polynomial multiplication still exists which will cause it exposed to the side-channel attack.

In this paper, the faster Dilithium-2 digital signature has been attacked. Combining the Correlation Power Analysis and Profiling T-test or Online Template attack methods, the small-coefficient polynomials  $s_1$  and  $s_2$  have been recovered to recover the private key. The attack method also works in Dilithium-5 because it uses the same NTT and polynomial multiplication.

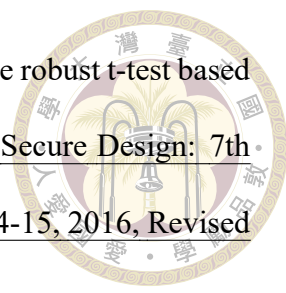
To prevent the attack, arithmetic masking is a good countermeasure. With first-order masking, although it takes more than twice the length of time. The t value is all less than the threshold of 6.32.





## References

- [1] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels. Faster kyber and dilithium on the cortex-m4. In Applied Cryptography and Network Security: 20th International Conference, ACNS 2022, Rome, Italy, June 20–23, 2022, Proceedings, pages 853–871. Springer, 2022.
- [2] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium: Algorithm specifications and supporting documentation (version 3.1). NIST Post-Quantum Cryptography Standardization Round, 3, 2021.
- [3] S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In Topics in Cryptology–CT-RSA 2014: The Cryptographer’s Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings, pages 28–47. Springer, 2014.
- [4] L. Batina, Ł. Chmielewski, L. Papachristodoulou, P. Schwabe, and M. Tunstall. Online template attacks. Journal of Cryptographic Engineering, 9:21–36, 2019.
- [5] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. Jing. An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In 2021 IEEE 39th International Conference on Computer Design (ICCD), pages 583–590. IEEE, 2021.

- 
- [6] A. A. Ding, C. Chen, and T. Eisenbarth. Simpler, faster, and more robust t-test based leakage detection. In Constructive Side-Channel Analysis and Secure Design: 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers 7, pages 163–183. Springer, 2016.
- [7] A. A. Ding, L. Zhang, F. Durvaux, F.-X. Standaert, and Y. Fei. Towards sound and optimal leakage detection procedure. In Smart Card Research and Advanced Applications: 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13–15, 2017, Revised Selected Papers, pages 105–122. Springer, 2018.
- [8] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Cryptographic Hardware and Embedded Systems—CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14, pages 530–547. Springer, 2012.
- [9] V. Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Advances in Cryptology—ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15, pages 598–616. Springer, 2009.
- [10] V. Lyubashevsky. Lattice signatures without trapdoors. In Advances in Cryptology—EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31, pages 738–755. Springer, 2012.
- [11] S. Mangard, E. Oswald, and T. Popp. Power analysis attacks: Revealing the secrets of smart cards, volume 31. Springer Science & Business Media, 2008.

- [12] V. Migliore, B. Gérard, M. Tibouchi, and P.-A. Fouque. Masking dilithium: efficient implementation and side-channel evaluation. In Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17, pages 344–362. Springer, 2019.

