When NTT Meets SIS: Efficient Side-channel Attacks on Dilithium and Kyber

Zehua Qiao[1,2], Yuejun Liu[3], Yongbin Zhou[1,3], Mingyao Shao[1,2] and Shuo Sun[4]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{qiaozehua,shaomingyao}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
liuyuejun@njust.edu.cn
[4] China Mobile Internet

Abstract. In 2022, NIST selected Kyber and Dilithium as post-quantum cryptographic standard algorithms. The Number Theoretic Transformation (NTT) algorithm, which facilitates polynomial multiplication, has become a primary target for side-channel attacks. Among these, Correlation Power Analysis (CPA) attacks against NTT have received much attention, which aims to recover all the coefficients of the private key in NTT domain. The necessity to recover all these coefficients not only limits efficiency but also directly impacts the feasibility of such attacks. Thus, a crucial question emerges: can the remaining coefficients be recovered using only a subset of known ones? In this work, we respond affirmatively by introducing overdetermined system-based and SIS-assisted key recovery methods for both Dilithium and Kyber, tailored for scenarios with incomplete NTT domain private keys. The SIS-assisted method, by embedding NTT transform matrix into the SIS search problem, offers a complete key recovery with the minimum known coefficients in NTT domain. For Kyber512 and Dilithium2, only 64 and 32 coefficients are enough to recover a subset of the private key with 256 coefficients, respectively. Furthermore, we propose a parameter-adjustable CPA scheme to expedite the recovery of a single coefficient in NTT domain. Combining this CPA scheme with the SIS-assisted approach, we executed practical attacks on both unprotected and masked implementations of Kyber and Dilithium on an ARM Cortex-M4. The results demonstrate that we can recover a subset of 256 private key coefficients for Dilithium2 using 2,000 power traces in 0.5 minutes, while Kyber512 requires 0.4 minutes and 500 power traces. These attacks achieve a 400× speedup compared to the best-known attacks against Dilithium. Moreover, we successfully break the first-order mask implementations and explore the potential applicable to higher-order implementations.

Keywords: Lattice-based Cryptography · Number Theoretic Transformation · Side-channel Attacks · Short Integer Solution · Dilithium · Kyber

1 Introduction

Traditional public key cryptography relies on the computational intractability of problems such as integer factorization and discrete logarithms. However, the emergence of quantum computing has raised concerns, as it promises polynomial-time solutions to these problems, thereby compromising the security of our existing cryptographic algorithms [Sho94]. Recognizing this threat, the National Institute of Standards and Technology (NIST) initiated a Post-Quantum Cryptography (PQC) standardization process in 2016, aiming to identify

quantum-resistant cryptographic algorithms. By July 2022, NIST published its first set of post-quantum cryptographic standard algorithms [ACD+22], comprising three signature algorithms: Dilithium, FALCON, and SPHINCS+; and one key encapsulation mechanism (KEM) algorithm, Kyber. In August 2023, NIST released three draft standards except for FALCON. This paper mainly focuses on Dilithium and Kyber.

While the fundamental security of these algorithms has been widely recognized in the cryptographic community, Side-channel Attacks (SCAs) emerged as formidable adversaries, underscoring the imperative of securing cryptographic implementations against these non-traditional threats [KJJ99]. This was not a nuance lost on NIST, which duly emphasized side-channel security during its PQC evaluations [ACD+22].

Preliminary SCAs targeting Dilithium and Kyber have emerged. For the Kyber, a significant type of key recovery attack combines Chosen Ciphertext Attacks (CCA) with SCAs [RRCB20, XPR+22, RRD+23, TUX+23], in which adversaries constructed the Plaintext-Checking (PC) oracle with the help of SCAs to determine whether the message is successfully recovered, then extracted the information of private key. Recently, Shen *et al.* [SCZ+23] developed a method that adapts to imperfect PC oracles constructed via SCAs and is still capable of recovering private keys. For the Dilithium, a typical class of attack is the randomness leakage attack, initially addressed by Liu *et al.* [LZS+21, QLZ+23]. They demonstrated that even a single-bit leakage of random polynomial per signature can be devastating for lattice-based Fiat-Shamir signatures, including Dilithium. Later, SCAs based on similar mathematical tools were proposed by Marzougui *et al.* [MUTS22] and Berzati *et al.* [BVC+23]. These attacks require an in-depth understanding of Kyber and Dilithium. However, simpler methods are typically desirable, such as classical SCAs targeting operations in which the private key is directly involved.

Both Dilithium and Kyber operate over the cyclotomic ring $\mathbb{Z}_q[x]/(x^n+1)$, leveraging the Number Theoretic Transformation (NTT) to accelerate the polynomial multiplication. As a fundamental module of operation, the private key will almost certainly perform NTT polynomial multiplication, thereby making it vulnerable to side-channel attacks. These algorithms primarily employ two categories of operations: NTT and Inverse NTT (INTT) operations, large number multiplication and reduction (Montgomery or Barrett) operations. For the former, existing work mainly exploited profiled side-channel methods. In 2017, Primas *et al.* [PPM17] pioneered a single-trace attack targeting the NTT operation using the Template Attacks (TAs) and belief propagation algorithms. Building on this foundational work, Hamburg *et al.* [HHP+21] successfully recovered the private key of masked Kyber. In a related vein, Xu *et al.* [XPR+22] introduced a Simple Power Analysis (SPA) method targeting the INTT operation during Kyber's decapsulation phase. Correspondingly, Han *et al.* [HLK+21] executed the first practical attack using machine learning-based TA aimed at the NTT operation within Dilithium's signing procedure. While potent, such attacks typically usually require additional control privileges over the target system.

In contrast to the aforementioned operations, research on attacks against multiplication and reduction operations has predominantly utilized Correlation Power Analysis (CPA) methods. Regarding Kyber, Karlov *et al.* [KdG21] conducted practical attacks on Kyber's pqm4 open-source implementation [KPR+]. Enhancing this approach, Yang *et al.* [YWY+23] optimized the attack by filtering ciphertexts, thereby diminishing the enumeration space and enhancing the Signal-to-Noise (SNR). In the context of Dilithium, Fournaris *et al.* [FDK20] demonstrated the potential of CPA on its NTT polynomial multiplication procedure. Further refining this method, Chen *et al.* [CKA+21] optimized this method by employing the divide and conquer strategy, successfully reducing the time required to recover a single NTT domain coefficient from 6,357 to 818 seconds. While these attacks are conceptually simpler, they entail a substantial computational overhead.

The private keys for Kyber and Dilithium are represented as high-dimensional polynomials with independent coefficients. The ultimate objective of attacks is to deduce the

entire private key, which entails significant computational costs. An emerging strategy is to recover partial coefficients via SCAs instead and recover the remaining ones using other methods, such as enumeration and lattice reduction. This strategy has paid off, for example, the combination of lattice reduction and misuse attacks reduces the cost of attacking Kyber512 by 34% [MJZ22]. However, this strategy seems to fail in CPAs on NTT polynomial multiplication because all the coefficients in NTT domain are necessary to recover the normal domain private key through INTT transform. What's worse, the overhead of enumeration is substantial even if only a tiny number of coefficients in NTT domain are unknown. Intuitively, if only 2 of the 256 coefficients are unknown and their positions are uncertain, the enumeration space required to recover Dilithium's private key is $\binom{256}{1}q + \binom{256}{2}q^2 \approx 2^{61}$, with $q = 8380417$. Thus, Chen *et al.* [CKA+21] sacrificed significant efficiency to ensure a 100% success rate in SCAs.

Hence, a pivotal motivation of our work is: *can we quickly recover the complete normal domain private key for algorithms like Dilithium and Kyber if only a partial NTT domain private key is known?*

The transformation of the private key from the normal domain to the NTT domain is injective and doesn't lead to an expansion of the key space. Theoretically, recovering a complete set of 256 coefficients becomes plausible if the number of coefficients in the NTT domain private key, symbolized as m, satisfies $8380417^m > 5^{256}$, a condition pertaining to Dilithium2. When a method for recovering an incomplete NTT domain private key is available, the requirements and constraints on the attacker's ability to obtain the NTT domain private key are substantially reduced. This facilitation allows for the development of more efficient analysis strategies and minimizes the time overhead necessary for private key recovery of the respective algorithm.

Our contributions are as follows:

- By exploiting the mathematical properties of NTT inherent to both Dilithium and Kyber, we propose key recovery methods based on overdetermined system techniques and SIS-assisted approaches. These proposed methods can swiftly recover the complete private key even when the NTT domain private key is partially known. In instances where the positions of unknown coefficients are identified, our overdetermined system techniques markedly reduce the enumeration space. Specifically, for Dilithium2, the enumeration space is reduced from 8380417^m to 5^m, and for Kyber512, it shrinks from 3329^m to 7^m, with m representing the number of unknown coefficients. Our SIS-assisted method has better performance in the case of large unknown coefficients. For Dilithium2, a minimum of 32 NTT domain private key coefficients are essential to recover the private key. For Kyber512, obtaining 64 known NTT domain private key coefficients is possible for full private key recovery.
- Drawing inspiration from Tunstall *et al.* [THM+07], we have developed a parameter-adjustable CPA-based SCA scheme that targets the numerous multiplications fundamental to Dilithium. Utilizing this scheme, a coefficient of Dilithium's NTT domain private key can be recovered in under a second—achieving a speed that is 40 times faster than the methods presented by Chen *et al.* [CKA+21].
- We apply SCA combined with SIS-assisted key recovery methods and successfully conduct practical attacks on unprotected Dilithium2 and Kyber512 on the ARM Cortex-M4 platform. For recovering a set of private keys with 256 coefficients, Dilithium2 takes 0.5 minutes using 2,000 power traces, and Kyber512 takes 0.4 minutes using 500 power traces.
- We conduct practical attacks on first-order masked Dilithium2 and Kyber512. For masked Dilithium2, a set of 256 coefficients private keys can be recovered in 3.2 minutes with 2,000 traces. For the assembly-optimized masked Kyber512, we can recover a set of 256 coefficients for the private key in 6 minutes with 200 traces using the SIS-assisted method.

Discussions. Albrecht *et al.* [ADP18] and Hamburg *et al.* [HHP+21] have also explored key recovery methods for lattice-based cryptographic schemes when keys in NTT domain are (partially) known. Albrecht *et al.* [ADP18] introduced a method for recovering keys from noisy NTT keys with bit flips. Hamburg *et al.* [HHP+21] proposed a side-channel attack against (CCA2-secure masked) Kyber, focusing on inverse NTT operation involving the ciphertext and the private key. Their method required generating a ciephertext with sparse NTT coefficients, and this can be achieved via the BKZ lattice reduction algorithm. Partial private key coefficients in NTT domain were then recovered using the BP algorithm, followed by an enumeration process for complete key recovery. While the final step involves partial NTT coefficient recovery, it only works when the known NTT coefficients are consecutive. Therefore, despite both work investigate side-channel attacks on NTT operations in Kyber, our work significantly differs from *et al.* [ADP18] in side-channel analysis techniques and strategies for recovering incomplete NTT coefficients.

2 Preliminaries

2.1 Dilithium

Dilithium, a digital signature scheme based on the Module Learning with Errors (MLWE) and Module Short Integer Solution (MSIS) problems, offers different security levels through its adjustable specific parameters, thus accommodating a variety of application scenarios and device constraints. Detailed information about these parameters is presented in Tab.1.

Table 1: Dilithium parameters at different NIST security levels

NIST Security Level	2	3	4
d [dropped bits from $\boldsymbol{t}$]	13		
α[# of non-zero coefficients in $\boldsymbol{c}$]	39	49	60
γ_1 [cofficient range of $\boldsymbol{y}$]	131,072	524,288	
γ_2 [low-order rounding range]	95,232	261,888	
$(m \times n)$ [dimensions of $\mathbf{A}$]	(4,4)	(6,5)	(8,7)
η [private key range]	2	4	2

The computations involved in the Dilithium algorithm are carried out within the cyclotomic ring $\mathbb{R}_q^n$, wherein all coefficients are elements of the finite field $\mathbb{Z}_q$. The values of $q = 8380417$ and $n = 256$ are invariant at all security levels. Dilithium comprises three procedures: key generation, signing, and verification. Our study specifically concentrates on the signing process. Alg.1 provides an exhaustive outline of the signing procedure.

During the signing phase, the algorithm takes the private key and a message as inputs. Following this, the Expand function generates a matrix, denoted as $\mathbf{A}$, and a masking vector of polynomials, $\boldsymbol{y}$, whose coefficients are less than γ_1. To expedite the computations, NTT operations are applied to the matrix $\mathbf{A}$ and the private keys $\boldsymbol{s}_1$. The signer then computes $\mathbf{A}\boldsymbol{y}$ and designates the higher-order bits of the coefficients of this vector to $\boldsymbol{w}_1$. Using the message M and $\boldsymbol{w}_1$, the challenge $\boldsymbol{c}$ is constructed, which subsequently aids in the generation of the signature $\boldsymbol{z}$. The algorithm also incorporates a rejection sampling loop that checks if the generated challenge $\boldsymbol{c}$ and signature $\boldsymbol{z}$ meet the prescribed output conditions. If these conditions, as detailed in lines 13-16, are satisfied, the algorithm outputs the result and the signature process is completed. If not, the algorithm revisits line 6 to regenerate the signature until a valid one is obtained.

Algorithm 1 Dilithium Sign(sk,M)

Input: $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0), M$
Output: *signature*
1: $\mathbf{A} \in R_q^{m \times n} := \text{ExpandA}(\rho)$
2: $\mu \in \{0,1\}^{384} := \text{CRH}(tr||M)$
3: $\kappa := 0, (\boldsymbol{z}, \boldsymbol{h}) := \perp$
4: $\rho' \in \{0,1\}^{384} := \text{CRH}(K||\mu)$ (or $\rho' \leftarrow \{0,1\}^{384}$)
5: $\hat{\mathbf{A}} = \text{NTT}(\mathbf{A}), \hat{\boldsymbol{s}}_1 = \text{NTT}(\boldsymbol{s}_1)$
6: $\boldsymbol{y} \in S^n_{\gamma_1 - 1} := \text{ExpandMask}(\rho', \kappa)$
7: $\boldsymbol{w} := \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\boldsymbol{y}))$
8: $\boldsymbol{w}_1 := \text{HighBits}_q(w, 2\gamma_2)$
9: $\tilde{\boldsymbol{c}} \in \{0,1\}^{256} := \mathbf{H}(\mu||\boldsymbol{w}_1)$
10: $\hat{\boldsymbol{c}} := \text{NTT}(\text{SampleInBall}(\tilde{\boldsymbol{c}}))$
11: $\boldsymbol{z} := \boldsymbol{y} + \text{NTT}^{-1}(\hat{\boldsymbol{c}} \circ \hat{\boldsymbol{s}}_1)$
12: $\boldsymbol{r}_0 := \text{LowBits}_q(\boldsymbol{w} - \boldsymbol{c}\boldsymbol{s}_2, 2\gamma_2)$
13: **if** $||\boldsymbol{z}||_\infty \geqslant \gamma_1 - \beta$ **or** $||\boldsymbol{r}_0||_\infty \geqslant \gamma_2 - \beta$
then $\kappa := \kappa + l$, goto 6
14: **else**
15: $\boldsymbol{h} := \text{MakeHint}_q(-\boldsymbol{c}\boldsymbol{t}_0, \boldsymbol{w} - \boldsymbol{c}\boldsymbol{s}_2 + \boldsymbol{c}\boldsymbol{t}_0, 2\gamma_2)$
16: **if** $||\boldsymbol{c}\boldsymbol{t}_0||_\infty \geqslant \gamma_2$ **or** the # of $1^{'}$s in $\boldsymbol{h}$ is greater than ω
then $\kappa := \kappa + l$, goto 6
17: **return** *signature* $= (\boldsymbol{z}, \boldsymbol{h}, \tilde{\boldsymbol{c}})$

2.2 Kyber

Kyber is a KEM that achieves IND-CCA security. Its security depends on the complexity of the MLWE problem. Kyber offers three distinct security levels: Kyber512, Kyber768, and Kyber1024, each affording different tiers of cryptographic strength. The parameter choices for each of these security levels are outlined in Tab.2. The values of $q = 3329$ and $n = 256$ remain unchanged at all security levels.

Table 2: Kyber parameters at different NIST security levels

NIST Security Level	1(Kyber512)	3(Kyber768)	5(Kyber1024)
k[dimension of polynomial ring]	2	3	4
η_1[noise of $\boldsymbol{s}$,$\boldsymbol{e}$ in KeyGen() and r in Enc()]	3	2	2
η_2[noise of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ in Enc()]	2		
(d_u, d_v)[compression function parameters]	(10,4)	(10,4)	(11,5)

Kyber offers a construction known as Kyber.CCAKEM, which is derived from Kyber.CPAPKE (this CPA stands for Chosen Plaintext Attack) through a variation of the FO (Fujisaki–Okamoto) transform. The Kyber.CCAKEM includes three key phases: key generation, encapsulation, and decapsulation. Both secret keys and error vectors are sampled from a centered binomial distribution represented as $\mathbf{B}_\eta$. This distribution, denoted as $\mathbf{B}_\eta$, is expressed as $\sum_{i=1}^{\eta}(a_i - b_i)$, where each a_i and b_i are independently and randomly sampled from the set $\{0,1\}$. This paper primarily explores the decapsulation process, illustrated in Alg.2.

This algorithm takes the ciphertext and private key as input for decapsulation. The message is computed following decompression, and re-encryption is employed to verify the ciphertext's validity and deliver the final result. The algorithm's implementation also necessitates a significant number of polynomial multiplications over a finite field, with the NTT utilized to enhance the efficiency of the implementation.

Algorithm 2 Kyber.CCAKEM.Dec($\boldsymbol{c}$,sk)

Input: $\boldsymbol{c} = (c_1, c_2), sk$
Output: *shared key* K
1: $pk := sk + 12 \cdot k \cdot n/8$
2: $h := sk + 24 \cdot k \cdot n/8 + 32$
3: $z := sk + 24 \cdot k \cdot n/8 + 64$
4: $\boldsymbol{u} := \text{Decompress}_q(\text{Decode}_{d_u}(c_1), d_u)$
5: $v := \text{Decompress}_q(\text{Decode}_{d_v}(c_2), d_v)$
6: $\hat{\boldsymbol{s}}_k := \text{Decode}_{12}(sk)$
7: $m' := \text{Encode}_1(\text{Compress}_q(v - \text{NTT}^{-1}(\hat{\boldsymbol{s}}_k^T \circ \text{NTT}(\boldsymbol{u})), 1))$
8: $(\bar{K}', r') := \text{G}(m'||h)$
9: $c' := \text{Kyber.CPAPKE.Enc}(pk, m', r')$
10: **if** $\boldsymbol{c} = \boldsymbol{c}'$ **then**
  return $K := \text{KDF}(\bar{K}'||\text{H}(\boldsymbol{c}))$
11: **else**
  return $K := \text{KDF}(z||\text{H}(\boldsymbol{c}))$
12: **end if**
13: **return** K

2.3 Number Theoretic Transformations

The Number Theoretic Transform (NTT) is pivotal in enhancing the efficiency of lattice-based cryptography. Serving as the finite field counterpart to the Fast Fourier Transform, the NTT is instrumental in optimizing polynomial multiplications. To elaborate, consider the multiplication of polynomials $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$. The multiplication process of these polynomials can be broadly divided into three stages:

1) Transforming $\boldsymbol{x}$ and $\boldsymbol{y}$ into NTT domain, *i.e.*, $\hat{\boldsymbol{x}} = \text{NTT}(\boldsymbol{x})$ and $\hat{\boldsymbol{y}} = \text{NTT}(\boldsymbol{y})$.
2) Perform point-wise multiplication of $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, *i.e.*, $\hat{\boldsymbol{z}} = \hat{\boldsymbol{x}} \circ \hat{\boldsymbol{y}}$.
3) Apply the INTT to $\hat{\boldsymbol{z}}$ to transform it back to the normal domain, *i.e.*, $\boldsymbol{z} = \text{INTT}(\hat{\boldsymbol{z}})$.

Applying NTT reduces the computational complexity of polynomial multiplication from $O(n^2)$ to $O(n \log n)$ when compared to the conventional schoolbook method. Different polynomial rings, such as $\mathbb{Z}_q[x]/(x^n-1)$ and $\mathbb{Z}_q[x]/(x^n+1)$, necessitate positive and negative convolutions respectively to accomplish NTT operations [PG12]. For the polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$, which is utilized in Dilithium and Kyber, a negative convolution is employed. The NIST reference implementation of Dilithium applies $2n$-th primitive root of unity in $\mathbb{Z}_q$ $\phi = 1753$ and $\omega = \phi^2 = 3073009$ for all security levels. As such, once the NTT domain private key is obtained, it can be transformed into the normal domain using INTT. The detailed calculation process is provided below.

$$\Phi \equiv \begin{pmatrix} \phi^0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \phi^{n-1} \end{pmatrix} \pmod q \qquad \Omega \equiv \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \cdots & \omega^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)^2} \end{pmatrix} \pmod q$$

$$\hat{\boldsymbol{x}} = \text{NTT}(\boldsymbol{x}) = \Omega\Phi\boldsymbol{x} \qquad \hat{\boldsymbol{y}} = \text{NTT}(\boldsymbol{y}) = \Omega\Phi\boldsymbol{y}$$

$$\hat{\boldsymbol{z}} = \hat{\boldsymbol{x}} \circ \hat{\boldsymbol{y}} \pmod q \qquad \boldsymbol{z} = \text{INTT}(\hat{\boldsymbol{z}}) \pmod q \tag{1}$$

Kyber also employs the NTT algorithm, however, its finite field $q = 3329$ contains merely a 256-th primitive root, denoted as ζ. Consequently, this necessitates distinct

computational nuances in its implementation. The polynomial can be represented as

$$X^{256}+1=\prod_{i=0}^{127}(X^2-\zeta^{2i+1})=\prod_{i=0}^{127}(X^2-\zeta^{2br_7(i)+1}) \tag{2}$$

where $br_7(i)$ represents the bit-reversed order of the 7-bit unsigned integer i. The transformation process is detailed in [ABD+20]. In practice, Kyber performs seven butterfly operations on 256 coefficients in the polynomial, resulting in the final NTT domain polynomial as follows:

$$\hat{\boldsymbol{x}}_{2i}=\sum_{j=0}^{127}\boldsymbol{x}_{2j}\zeta^{(2br_7(i)+1)j} \qquad \hat{\boldsymbol{x}}_{2i+1}=\sum_{j=0}^{127}\boldsymbol{x}_{2j+1}\zeta^{(2br_7(i)+1)j} \tag{3}$$

$$\boldsymbol{Z}=\begin{pmatrix}(\zeta^1)^0 & 0 & \cdots & (\zeta^1)^{127} & 0\\ 0 & (\zeta^1)^0 & \cdots & 0 & (\zeta^1)^{127}\\ (\zeta^{129})^0 & 0 & \cdots & (\zeta^{129})^{127} & 0\\ 0 & (\zeta^{129})^0 & \cdots & 0 & (\zeta^{129})^{127}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ (\zeta^{255})^0 & 0 & \cdots & (\zeta^{255})^{127} & 0\\ 0 & (\zeta^{255})^0 & \cdots & 0 & (\zeta^{255})^{127}\end{pmatrix} \qquad \mathrm{NTT}(\boldsymbol{x})=\boldsymbol{Z}\boldsymbol{x} \tag{4}$$

It is also possible to split the Equation (4) into odd and even positions according to the coefficient subscript number and calculate the value of the NTT domain separately. Let $\boldsymbol{x}^{(e)}=(x_0,x_2,\cdots,x_{n-2})$ and $\boldsymbol{x}^{(o)}=(x_1,x_3,\cdots,x_{n-1})$, which can be obtained:

$$\boldsymbol{Z}_{\setminus\{0\}}=\begin{pmatrix}(\zeta^1)^0 & (\zeta^1)^1 & \cdots & (\zeta^1)^{127}\\ (\zeta^{129})^0 & (\zeta^{129})^1 & \cdots & (\zeta^{129})^{127}\\ \vdots & \vdots & \ddots & \vdots\\ (\zeta^{255})^0 & (\zeta^{255})^1 & \cdots & (\zeta^{255})^{127}\end{pmatrix} \qquad \begin{aligned}\mathrm{NTT}(\boldsymbol{x}^{(e)})&=\boldsymbol{Z}_{\setminus\{0\}}\boldsymbol{x}^{(e)}\\ \mathrm{NTT}(\boldsymbol{x}^{(o)})&=\boldsymbol{Z}_{\setminus\{0\}}\boldsymbol{x}^{(o)}\end{aligned} \tag{5}$$

In the NIST reference implementation, Kyber maintains $\zeta=17$ at all security levels. In summary, NTT offers an efficient framework for polynomial multiplication, a process integral to the performance of lattice-based cryptographic algorithms like Dilithium and Kyber. While both algorithms utilize NTT, it is important to acknowledge that their implementation details and parameters differ, with each being meticulously tailored to accommodate their unique algebraic structures.

2.4 Correlation Power Analysis

Correlation Power Analysis (CPA), introduced by Brier *et al.* [BCO04], stands as a potent and refined SCA method. Since its inception, it has been adeptly leveraged to break the security of cryptographic algorithms, including DES [LCC+06] and AES [CFG+11]. The canonical procedure of CPA entails an adversary determining the Pearson Correlation Coefficient (PCC) between the observed side-channel leakage and the estimated intermediate values. Let $f(x_1,\ldots,x_p,k^*)$ represent a deterministic function indicative of the intermediate value of interest, where x_i symbolize known fluctuating parameters (such as plaintext or ciphertext), and k^* is the concealed secret key. Accumulating n traces, denoted by L, the adversary elects a pertinent leakage model M (e.g., Identity or Hamming Weight) to ascertain the hypothetical intermediate values $H_{k_j}=M(f(x_1,\ldots,x_p,k_j))$ for every conceivable key k_j within the set $\mathbb{K}$. The PCC, symbolized as $\rho(L,H_{k_j})$, is subsequently computed for each key hypothesis. The formula for PCC is given by:

$$\rho(L, H_{k_j}) = \frac{\sum_{i=1}^{n}(l_i - \bar{l})(h_i - \bar{h})}{\sqrt{\sum_{i=1}^{n}(l_i - \bar{l})^2}\sqrt{\sum_{i=1}^{n}(h_i - \bar{h})^2}} \tag{6}$$

The candidate key k_{cpa} that provides the highest correlation is chosen as the recovered key. The attack is considered successful if k_{cpa} matches the secret key k^*.

3 Incomplete NTT domain recovery methods

If an attacker obtains all NTT domain private keys, executing INTT would yield the complete private key. However, actual analysis seldom guarantees a 100% success rate due to specific implementation and the attack environment. For Dilithium, even if an attacker can recover 99% of the 256 coefficients, the search space required to recover the private key is still approximately $8380417^{2.56} \approx 2^{59}$. To address this challenge, we introduce two methods for recovering incomplete NTT domain private keys: one approach is based on overdetermined systems of equations, while the other is assisted by the SIS search problem.

3.1 Overdetermined system-based method

During polynomial multiplication in Dilithium and Kyber, the private key transitions from the normal domain to the NTT domain. Although each coefficient's value range expands from $2\eta + 1$ to q, an injective transformation maintains the private key space at $(2\eta + 1)^{256}$, facilitated by a static, publicly accessible transformation matrix. Intuitively, when there are m coefficients of NTT domain private key unknown, the search space should be $(2\eta + 1)^m$ instead of q^m. This goal is achievable by establishing and resolving overdetermined equations. Consider a vector $\boldsymbol{s}$ and take the NTT in Dilithium as an example. If the unknown NTT domain coefficients are the first m (this analysis applies to unknown coefficients in other positions), the specific analysis process unfolds as follows:

$$\Omega\Phi \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_{n-1} \end{pmatrix} \equiv \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \hat{s}_{k_m} \\ \vdots \\ \hat{s}_{k_{n-1}} \end{pmatrix} + \begin{pmatrix} \hat{s}_{un_0} \\ \vdots \\ \hat{s}_{un_{m-1}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \pmod{q} \tag{7}$$

Equation (7) represents the NTT process, where $\hat{\boldsymbol{s}}_k$ is a vector consisting of known coefficients with unknown position 0, while $\hat{\boldsymbol{s}}_{un}$ is a vector of unknown coefficients with known position 0. Equation (7) can be considered as an overdetermined system with m unknown coefficients and n equation relations regarding $\hat{\boldsymbol{s}}_{un}$ over a finite field q. Solving the overdetermined system within the finite field is inherently complex. However, given that the Dilithium private key lies within the integer range of $[-\eta, \eta]$ and the system must have a solution, we propose a comparatively straightforward method. Firstly, select m fixed positions within $\boldsymbol{s}$ and propose a candidate value. Then, use Gaussian elimination to solve for the corresponding $\hat{\boldsymbol{s}}_{un}$. After substituting $\hat{\boldsymbol{s}}_{un}$ into Equation (7) to obtain $\boldsymbol{s}$, verify whether each coefficient of $\boldsymbol{s}$ lies within the $[-\eta, \eta]$. If all coefficients satisfy this condition, it is presumed that the correct private key has been acquired.

Using this method, the space of the enumeration can be reduced from q^m to $(2\eta + 1)^m$. Although the time complexity of this approach remains exponential, it proves efficient when the number of unknown coefficients, denoted as m, is relatively small. During practical analysis, given a maximum acceptable enumeration space of 2^l, this method is applicable under conditions where $m < l \log_{2\eta+1} 2$. For instance, consider Dilithium2, where the

key's normal domain range is $[-2, 2]$, and suppose the acceptable enumeration space is 2^{32}. Under these conditions, the maximum m will be 13, as opposed to 1. This adjustment means that an adversary would need to recover only 243 out of the 256 NTT domain coefficients to recover the complete private key.

3.2 SIS-assisted method

The Small Integer Solution (SIS) problem was proposed by Ajtai in 1996 [Ajt96]. It aims to find a sufficiently short integer vector that when multiplied by a randomly selected integer matrix under an upper bound, results in a zero vector. The specific definitions are as follows:

$\mathrm{SIS}_{q,m,n,d}$ distribution: *Choose a random matrix* $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$ *and a vector* $\boldsymbol{s} \leftarrow \{-d, \ldots, 0, \ldots, d\}^n$ *and output* $(\boldsymbol{A}, \boldsymbol{As})$.

$\mathrm{SIS}_{q,m,n,\beta}$ search problem: *Given a pair* $(\boldsymbol{A}, \boldsymbol{t})$ *from the* $\mathrm{SIS}_{q,m,n,d}$ *distribution, find a* $\boldsymbol{s} \in \{-d, \ldots, 0, \ldots, d\}^n$ *such that* $\boldsymbol{As} = \boldsymbol{t}$.

When $d \ll q^{(m/n)}$, there is a high probability that a unique vector $\boldsymbol{s}$ exists such that $\boldsymbol{As} = \boldsymbol{t}$, where q is the range of the finite field, m is the number of unknown polynomial coefficients, and n is the total number of polynomial coefficients. For the SIS search problem $\boldsymbol{As} = \boldsymbol{t}$, where $\boldsymbol{As} - \boldsymbol{t} = 0$, we define $\boldsymbol{A'} = \boldsymbol{A}|\boldsymbol{t}$ and solve the SIS problem $\boldsymbol{A'x} = 0$. If the solution $\boldsymbol{x}$ satisfies the form $(\boldsymbol{s}, -1)$, then we obtain the solution $\boldsymbol{s}$ to the SIS search problem.

We observe that the incomplete NTT domain recovery can be viewed as an SIS search problem. Assume that the first m coefficients in NTT domain are unknown. Then $\boldsymbol{A}$ is the $(n-m)\times(n-m)$ matrix obtained from the transform matrix ($\Omega\Phi$ in Equation (1) for Dilithium, $\boldsymbol{Z}$ or $\boldsymbol{Z}_{\setminus\{0\}}$ in Equation (4,5) for Kyber) by removing the first m columns, $\boldsymbol{s}$ is the private key which is exactly a short integer vector, and $\boldsymbol{t}$ is the recovered NTT domain coefficients. Taking Dilithium2 as an example, when $d = 2$, $m > 128$ satisfies the condition $d \ll q^{(m/n)}$, which means that the attacker only needs to recover half of the NTT domain coefficients and the remaining one can be recovered using the SIS-assisted method.

According to the definition of lattice, the SIS problem can be trivially reduced to the SVP problem. The entire solution set of an SIS problem instance constitutes a lattice $\Lambda^{\perp}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{Ax} = 0 \pmod q\}$. Therefore, solving an SIS problem instance is equivalent to solving the corresponding SVP problem on the lattice $\Lambda^{\perp}(\boldsymbol{A})$. The LLL algorithm and the BKZ algorithm are commonly used to solve the SIS problem.

The LLL algorithm was proposed by H. Lenstra, A. Lenstra, and Lovasz in 1982 [LLL82]. The algorithm outputs a short vector with an approximate factor of $\delta_0 = (4/3)^{(m-1)/4} \approx 1.075$ in polynomial time. The BKZ algorithm was introduced by Schnorr *et al.*. [SE91] as an improvement to the LLL algorithm using the idea of blockwise reduction. The LLL algorithm can be viewed as a special case of the BKZ algorithm with block size $\beta = 2$. The BKZ algorithm requires calling the LLL algorithm and a subroutine that solves the SVP problem on a lower dimensional lattice. In practice, the SVP solver is often implemented using enumeration or sieving algorithms. The key parameter for evaluating the BKZ algorithm is the block size β. As β increases, the quality of the reduced basis and the length of the shortest vector output by BKZ improve, but the running time also increases. In 2011, Chen and Nguyen proposed the BKZ 2.0 algorithm [BDGL16], which significantly improved the efficiency of the basis reduction algorithm in practice. Currently, in the core-SVP evaluation model, the complexity of the BKZ algorithm with block size β is approximately $O(2^{0.292\beta})$ in the classical computing model, and the approximate factor for the shortest vector output is

$$\lim_{n\to\infty} \delta_0 \approx \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}} \tag{8}$$

For the specific SIS search problem instance corresponding to Dilithium and Kyber, we can use the .LLL and .BKZ functions in the fpylll library in the Sage software to solve the transformed SVP problem.

We experimentally found that, as detailed in Sec.5.3, for Dilithium or Kyber, the SIS-assisted method can recover the complete normal domain private key when the attacker gets more than half of the NTT domain private key. Such a result extends the method to a more realistic scenario where an attacker can recover the NTT domain private key with a high success rate but cannot determine whether it is correct or not. Assuming that the minimum number of coefficients required to successfully execute the SIS-assisted method is x, and the number of correct NTT domain private keys obtained by the attacker is m. Randomly selecting x out of the n coefficients, when all x coefficients are correct the recovery of the private key can be accomplished by the SIS-assisted method, and the expected number of executions of the attack is $\binom{n}{x}/\binom{m}{x}$.

3.3 The divide-and-conquer strategy in SIS-assisted method

In the practical analysis, we find that the SIS-assisted method suffers from large time overhead in the high-dimensional case as well as the gap between the required NTT domain coefficients and the theoretical lower bound is too large, while the above problems are alleviated when the dimensionality is low. We note that the negative convolutions NTT, utilized by both Dilithium and Kyber, can be rewritten from a 2^n-dimensional NTT to a $2^{(n-1)}$-dimensional NTT [ADP18] using a divide and conquer strategy. To facilitate the subsequent formulation, let $\boldsymbol{s}^{(e)} = (s_0, s_2, \cdots, s_{n-2})$ and $\boldsymbol{s}^{(o)} = (s_1, s_3, \cdots, s_{n-1})$. In Dilithium, the transformation process is shown as follows:

$$\begin{aligned} 2\mathbf{NTT}_{n/2}(\boldsymbol{s}^{(e)})_i &= \mathbf{NTT}_n(\boldsymbol{s})_i + \mathbf{NTT}_n(\boldsymbol{s})_{i+n/2} \\ 2\phi\omega^i\mathbf{NTT}_{n/2}(\boldsymbol{s}^{(o)})_i &= \mathbf{NTT}_n(\boldsymbol{s})_i - \mathbf{NTT}_n(\boldsymbol{s})_{i+n/2} \end{aligned} \tag{9}$$

which $i \in \{0, 1, \ldots, n/2-1\}$. Let W_{256} be the product of Ω and Φ in Equation (1) for example, the transformation results are shown below:

$$W_{256}^{(+)} = \begin{pmatrix} 2 & 0 & 2\phi^2 & 0 & \cdots & 2\phi^{254} & 0 \\ 2 & 0 & 2\phi^6 & 0 & \cdots & 2\phi^{250} & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 2 & 0 & -2\phi^{254} & 0 & \cdots & -2\phi^2 & 0 \end{pmatrix}, W_{128} = \begin{pmatrix} 1 & \phi^2 & \cdots & \phi^{254} \\ 1 & \phi^6 & \cdots & \phi^{250} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & -\phi^{254} & \cdots & -\phi^2 \end{pmatrix}.$$

W_{128} is derived by dividing the matrix into two parts, front and back, each consisting of 128 rows. These parts are then added, all-zero columns are removed, and divide the remaining elements by 2.

Since the q used in Kyber exists only as a 256-th root of unity ζ, its conversion process is slightly different from that of Dilithium, analyzing the odd or even positions of Kyber's NTT domain private key coefficients, specifically:

$$\begin{aligned} 2\mathbf{NTT}_{n/2}(\boldsymbol{s}^{(e)})_i &= \mathbf{NTT}_n(\boldsymbol{s})_{2i} + \mathbf{NTT}_n(\boldsymbol{s})_{2i+1} \\ 2\zeta^{2br_7(2i)+1}\mathbf{NTT}_{n/2}(\boldsymbol{s}^{(o)})_i &= \mathbf{NTT}_n(\boldsymbol{s})_{2i} - \mathbf{NTT}_n(\boldsymbol{s})_{2i+1} \end{aligned} \tag{10}$$

which $i \in \{0, 1, \ldots, n/2-1\}$. Taking the NTT matrix of $\boldsymbol{Z}_{\setminus\{0\}}$ in Equation (5) as an example, the result is shown below:

$$W_{128}^{(+)} = \begin{pmatrix} 2 & 0 & 2\zeta^2 & 0 & \cdots & 2\zeta^{126} & 0 \\ 2 & 0 & -2\zeta^2 & 0 & \cdots & -2\zeta^{126} & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 2 & 0 & -2\zeta^{126} & 0 & \cdots & -2\zeta^2 & 0 \end{pmatrix}, W_{64} = \begin{pmatrix} 1 & \zeta^2 & \cdots & \zeta^{126} \\ 1 & -\zeta^2 & \cdots & -\zeta^{126} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & -\zeta^{126} & \cdots & -\zeta^2 \end{pmatrix}.$$

W_{64} is derived by splitting the matrix into two parts based on the parity of the rows. After adding these parts, all-zero columns are removed, and divide the remaining elements by 2.

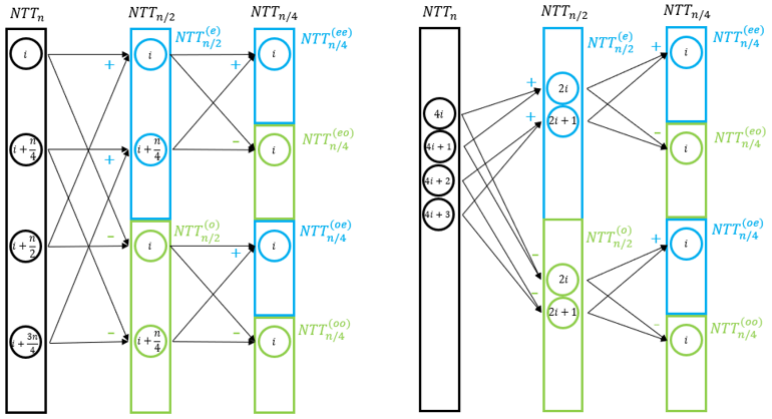


Figure 1: Transform process of the values under NTT domain (left: Dilithium; right:Kyber).

With the help of this transformation, the vectors under the normal domain can be iteratively split into several parts according to the parity position, which converts the original high-dimensional NTT matrix into several known and fixed low-dimensional matrices. In order to understand more intuitively, Fig.1 shows the transformation process of the values under NTT domain, when the value of a specific position under NTT domain is obtained, the value corresponding to a specific position of the low-dimensional matrix can be calculated. At this time, the dimension of the SIS search problem that the attacker needs to solve is reduced to $n/4$, and the smaller dimension makes the SIS-assisted scheme faster while the attacker needs to obtain fewer values in the NTT domain. If the minimum coefficients required to solve the SIS search problem corresponding to $\mathrm{NTT}_{n/4}$ is k, then the normal domain private key can be recovered after obtaining $4k$ NTT domain coefficients that satisfy the specific coefficients.

Table 3: Theoretical minimum coefficients required for each dimension of Dilithium

	NTT_{256}	NTT_{128}	NTT_{64}	NTT_{32}	NTT_{16}	NTT_{8}
Dilithium2,5	26	13	7	4	2	1
Dilithium3	36	18	9	5	3	2

For any dimensional NTT_i computation process, it is necessary to ensure that the known NTT domain coefficients m satisfy the condition $q^m > 5^i$ in order to perform the inverse transformation correctly. Tab.3 gives the theoretical minimum coefficients required for each dimension to be able to perform the inverse transformation correctly under different partitioning strategies of Dilithium, and the NTT domain coefficients required are multiplied by $2^{256/i}$ for NTT_i scenarios. In practice, we find (see Sec.5.3 for details of the results) that the results of the SIS-assisted method for high dimensional scenarios are far from the theoretical lower bound of 26, using Dilithium2 as an example. However, even though that rounding leads to an increase in the minimum required coefficients to 32 for $\mathrm{NTT}_{32,16,8}$ in the low-dimensional scenario, the SIS-assisted method is able to reach this lower bound, indicating that the divide and conquer strategy is effective.

4 Recovering NTT domain private keys using SCAs

Attackers may extract sensitive information from a cryptographic system during the execution through various means, including SCAs, cache attacks, and cold-boot attacks.

In this section, we will discuss the application of SCAs to swiftly recover the NTT domain private keys for Dilithium and Kyber.

4.1 CPA for Dilithium and Kyber

Dilithium and Kyber employ point-wise operations using NTT domain private keys, as illustrated in Alg.1 line 11 and Alg.2 line 7, respectively. The reference implementation of $\hat{\boldsymbol{c}} \circ \hat{\boldsymbol{s}}_1$ for Dilithium, submitted to NIST, is depicted in Fig.2, Kyber's implementation is fundamentally identical. Conducting SCAs on these operations using CPA is theoretically possible. In fact, Yang *et al.* [YWY+23], working with Kyber where $||\mathbb{K}|| = 3329$, successfully recovered the private key in a matter of minutes using CPA. However, the task becomes computationally very difficult for Dilithium, where $||\mathbb{K}|| = 8380417$, as this renders exhaustive enumeration of the entire key space impractical.

```
1 int32_t montgomery_reduce(int64_t a) {
2   int32_t t;
3   t = (int32_t)a*qinv;
4   t = (a - (int64_t)t*q) >> 32;
5   return t;}
6 void poly_point-wise_montgomery(poly *c, const poly *a, const poly *b) {
7   unsigned int i;
8   for(i = 0; i < N; ++i)
9     c->coeffs[i] = montgomery_reduce((int64_t)a->coeffs[i] * b->coeffs[i]);}
```

Figure 2: Dilithium $\hat{\boldsymbol{c}} \circ \hat{\boldsymbol{s}}_1$ reference implementation.

Tunstall *et al.* [THM+07] outlined the use of CPA to attack large word sizes, successfully extracting the DES key implemented on a 32-bit platform. The approach involves segmenting large byte intermediate values, and then using CPA to recover each segment independently. Given an intermediate value $f(x_1, \ldots, x_p, k)$, where k is the l-bit unknown fixed value and x_i is a known change value, we divide k into consecutive blocks $b_{n-1}, \ldots, b_0$ from the most significant the least significant bit, with each block being l_{b_i} bits. Consequently, the PCC for each block can be computed as follows:

$$\rho(L, H_{k^*/b_i}) = \rho(L, H_{k^*})\sqrt{\frac{l_{b_i}}{l}} \tag{11}$$

The practical attack procedure is as follows:

(1) Compute the PCCs for all possible values of b_0, rank them in descending order, and select the top h_0 as potential candidates.
(2) Merge the h_0 candidate values with b_1 to generate a new set of candidates, and then calculate the PCCs and sort them in descending order. Finally, pick the top h_1 candidate values.
(3) Recursively perform step (2) for the remaining b_i $(2 \leq i < n-2)$.
(4) Combine the h_{n-2} candidate values with b_{n-1}, calculate all possible PCCs, and choose the candidate that corresponds to the highest value as the result.

This strategy reduces the enumeration requirement from 2^l values to $2^{b_0} + h_0 \cdot 2^{b_1} + \cdots + h_{n-2} \cdot 2^{b_{n-1}}$. For instance, with $l = 32$, $h_i = 8$, and each b_i being 8-bit, the enumeration space is trimmed from 2^{32} to under 2^{13}. This approach can also be employed for analyzing large number multiplication operations. Through blocking, the private key can be recovered from low to high blocks in turn, while the carry operation within the multiplication makes it easier to distinguish the correct candidate value of the high block. While choosing smaller blocks might seem to expedite the attack, Equation (11) indicates that this also results in a smaller PCC calculated in step (1). Various factors, including the SNR and

the length of candidate values, influence the probability of identifying the correct among the h_0 candidates, ultimately affecting the successful recovery of k^*. It is crucial to avoid selecting excessively small blocks, as this could compromise the attack's accuracy.

Chen *et al.* [CKA+21] noted that the output from the Montgomery reduction (as seen in Fig.2, line 5) constitutes a distinct leakage. Utilizing CPA with this leakage, dubbed the Conservative scheme, they achieved a 100% success rate using 157 power traces. Since this calculation process involves a shift operation, it precludes the direct application of the method proposed by Tunstall *et al.*. Consequently, Chen *et al.* opted to divide the NTT domain into high and low two parts two blocks, likely aiming to ensure a high success rate. They employed CPA on the Montgomery reduction input (Fig.2, line 9) to recover the lower part, then used the output to recover the NTT domain private key and verify the accuracy of the results. When the results are not accurate, the Conservative scheme is enacted to guarantee 100% accuracy in the NTT domain results, leading to the naming of this approach as the Hybrid scheme. Nonetheless, the commitment to a 100% success rate inevitably compromises some degree of attack efficiency in Chen *et al.*'s approach, resulting in a still substantial time overhead, as evident from their outcomes.

4.2 Efficient SCA scheme for NTT domain private keys

The introduction of an incomplete NTT domain recovery method allows us to relax the requirement for correctness in exchange for efficient attacks. We design efficient SCA schemes against Dilithium and Kyber. Our scheme selects linear operations as the target, giving lower-order bits candidates with the help of Tunstall *et al.* [THM+07]'s method, and later selects nonlinear operations with more significant leakage in order to recover the complete private key.

Algorithm 3 CPA in NIST Reference Implementations of Dilithium and Kyber

Require: th_{sel}, th_{fin}, mcn, num_{blocks},L_m
1: Initialize CS to an empty set of size mcn ▷ Candidate Set
2: **for** $i \leftarrow 0$ **to** $num_{blocks} - 1$ **do**
3: $CS \leftarrow \text{combine}(CS, [2^{i \cdot m}, 2^{(i+1) \cdot m}])$) ▷ Combine lower-order bits
4: $temp \leftarrow \text{sort}(\text{CPA}(H_{middle}(CS)), L_m)$ ▷ Sort by CPA of middle part
5: $CS \leftarrow temp \geq th_{sel} \cdot \max(temp)$
6: $CZS \leftarrow \text{combine}(CS, 0)$ ▷ Candidate with Zero Set
7: **end for**
8: **for each** k **in** (CS, CZS) **do**
9: **if** $\text{CPA}(H_{final}(k), L_m) > th_{fin}$ **then**
10: **return** k and **break**
11: **end if**
12: **end for**
13: **return** -1 ▷ Indicate failure in key recovery

Alg.3 gives the attack process. Designed for practical attack scenarios, this algorithm introduces selection thresholds (th_{sel}), a maximum candidate number (mcn) for the Candidate Set (CS) size, and specifies the number of lower-order bit blocks (num_{blocks}), each block containing m bits. We establish a (CS) for the key by applying CPA to these values. The candidate key is iteratively computed by selecting the intermediate value of the linear transformation in the Montgomery reduction, though false positives can occur in this process. For instance, when the lowest block is zero ($b_0 = 0$), as in 0x23450, the result would erroneously be $b_{3\sim0} =$ 0x2345. This issue is addressed by zero-padding the existing candidate values during low-order bit attacks and storing the data in a Candidate with Zero Set (CZS). For the higher-order bits, the CS and CZS are merged, and the

most distinct leakage is identified through CPA to accurately reconstruct the entire key. Practical situations may necessitate a balance between success rate and attack speed, adjustable through mcn and th_{sel}. If traces are limited, increasing mcn or decreasing th_{sel} enhances the success rate. Alternatively, abundant traces allow for attack speed optimization by reducing mcn or increasing th_{sel}. During practical attacks, we noticed that the Montgomery reduction's shift and data storage operations made the PCC of the correct key substantially higher than that of incorrect candidates (see line 5 in Fig.2). This characteristic enables the setting of a threshold, th_{fin}, to expedite computation. It also serves as a criterion for determining the success of the current coefficient attack, facilitating efficient private key recovery using the incomplete NTT domain recovery method.

5 Experiments and results

5.1 Masked implementation of Dilithium and Kyber

There has been some work proposing protected implementations for Dilithium and Kyber. For Dilithium, Azouaoui *et al.* [ABC+23] proposed a protection strategy for intermediate computations, which considers the physical security requirements and classifies intermediate values into three categories: resist Differential Power Analysis (DPA), resist SPA, and publicly known. Different protection strategies were applied for each category. Fig.3 depicts a first-order masking scheme, where each coefficient of the $\boldsymbol{s}$ is split into two arithmetic shares, $\boldsymbol{s}^0$ and $\boldsymbol{s}^1$, during the key generation procedure. The NTT operation is then independently applied to $\boldsymbol{s}^0$ and $\boldsymbol{s}^1$. This is followed by point-wise multiplication with the signature $\boldsymbol{y}$ in the NTT domain and subsequent operations. Coron *et al.* [CGTZ23] proposed some gadgets to improve the efficiency of the algorithm based on the work of Azouaoui *et al.* and gave open source code. In this experiment, we use the open source code given by Coron *et al.* as the target of the attack.

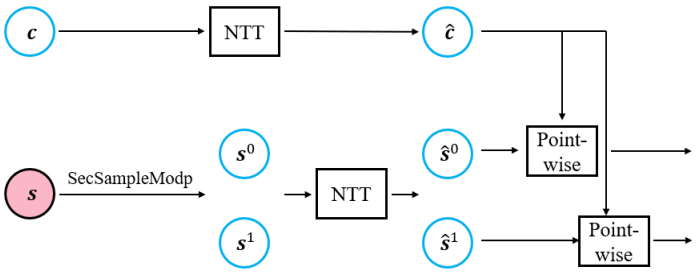


Figure 3: Procedure for masked protection implementation of polynomial multiplication in $\boldsymbol{cs}$. Red: DPA protection is required. Blue: No side-channel protection is needed.

For Kyber, since the scheme proposed by Bos *et al.* [BGR+21] is not open-source yet, this experiment utilizes the implementation by Heinz *et al.* [HKL+22]. They provided a mask-protected complete implementation of Kyber768, based on the unprotected Cortex-M4 optimized implementation in the PQM4 project [KPR+]. For the polynomial multiplication of the private key $\boldsymbol{s}_k$ with the ciphertext $\boldsymbol{c}$ during the decapsulation process, the implementation follows the same structure as shown in Fig.3.

5.2 Set up

The experiments conducted on unprotected algorithms employed the open-source, C-based implementation that was submitted to NIST. Notably, the open-source implementation of Kyber, optimized for the ARM Cortex-M4 platform using assembly language, does not

have separate experimental results presented herein due to its polynomial multiplication process being nearly identical to that of masked Kyber. The compiled code was executed on a ChipWhisperer UFO development board fitted with an STM32F405GTx microcontroller.

Our experimental setup includes a WR610Zi oscilloscope connected to a BLP1.9+ low-pass filter and a PA303 preamplifier, enabling the capture of power traces at a sampling rate of 100 MSa/s. For data analysis, we used a desktop computer equipped with an Intel i7-12700H processor and 16GB of DDR4 RAM. Analytical tools were implemented using Python 3.9 and SageMath 9.3. Each experiment was repeated ten times, with the average value of these repetitions taken as the result.

5.3 Results of incomplete NTT domain recovery methods

The full private key can be recovered using overdetermined equation-based and SIS-assisted methods when incomplete NTT domain coefficients are known. While the time overhead for the overdetermined equation-based method is exponential, it is efficient when there are few unknown coefficients. Fig.4 illustrates the time overhead associated with this key recovery method. The graph also includes results for Kyber when split by parity. For schemes with a normal domain $\eta = 2$ (e.g., Dilithium2, 5, and Kyber768, 1024), recovery of the private key takes under 10 minutes when there are no more than 10 unknown coefficients and less than 1 second when the number is below 6. For schemes with a normal domain $\eta = 3$ (e.g., Dilithium3 and Kyber512), the time overhead ranges between 2 to 5 minutes for 7 unknown coefficients, escalating exponentially as the number increases.

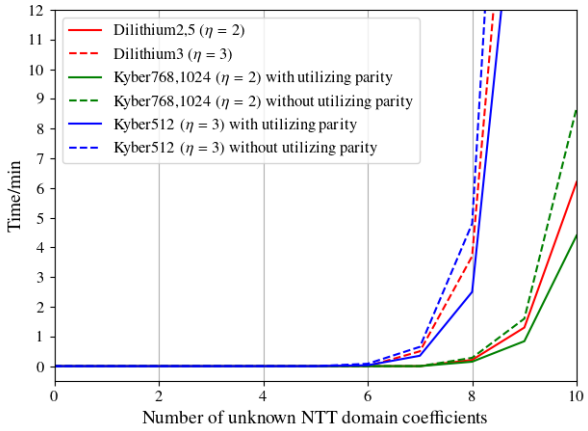


Figure 4: Time cost of the overdetermined equation-based key recovery.

Table 4: NTT domain coefficients and time required for SIS-assisted method in Dilithium

		NTT_{256}	NTT_{128}	NTT_{64}	NTT_{32}
Coefficients	Dilithium2,5	108(116)	60(70)	40(48)	32(40)
	Dilithium3	117(122)	70(80)	52(56)	40(48)
Time (s)	Dilithium2,5	1093.7(1152.5)	9.1(9.6)	3.9(4.6)	2.0(1.6)
	Dilithium3	1112.9(1186.2)	9.3(10.2)	3.9(4.0)	1.9(2.4)

() corresponds to the result of achieving a 100% success rate.

Tab.4 displays the minimum NTT domain coefficients required and the associated time overhead to recover a complete set of private keys for various Dilithium security levels using the SIS-assisted approach, both directly and with different divide and conquer strategies.

Using the SIS-assisted method directly, it is possible to recover a subset of private key by obtaining 108 NTT domain coefficients for Dilithium2,5 and 117 for Dilithium3. Given the large data dimension, this direct approach requires around 20 minutes to compute. As more coefficients become known, the computation time reduces, stabilizing at about 20 seconds beyond 220 coefficients. Employing a divide and conquer strategy optimizes this even further, particularly due to reduced data dimensions. For instance, after three splits using the NTT_{32} approach, only 32 NTT domain coefficients it is possible to recover the private key in 2 seconds, which has reached the theoretical lower bound of the scheme.

Table 5: NTT domain coefficients and time required for SIS-assisted method in Kyber

		NTT_{128}	NTT_{64}	NTT_{32}	NTT_{16}
Coefficients	Kyber512	58(66)	36(42)	32(36)	32(32)
	Kyber768,1024	54(61)	32(40)	28(32)	32(32)
Time (s)	Kyber512	4.5(4.8)	2.0(2.1)	1.0(0.8)	0.6(0.6)
	Kyber768,1024	4.6(5.1)	1.9(2.0)	0.9(0.9)	0.7(0.7)

() corresponds to the result of achieving a 100% success rate.

Shifting focus to Kyber, Sec.2.3 delineates that we can analyze NTT domain private key parity positions separately, subsequently merging them to retrieve the complete private key. Tab.5 presents the outcomes of recovering Kyber NTT domain parity position coefficients employing various divide and conquer strategies. Using the direct SIS-assisted approach, Kyber512 requires a minimum of 58 NTT domain coefficients, while Kyber768 and Kyber1024 require a slightly lower number of 54 coefficients. Utilizing a divide and conquer strategy, such as obtaining NTT_{32} after executing two splits, minimizes the required coefficient count to 28. This reduction enables the recovery of Kyber512 NTT domain odd position private keys within a mere second. For recovering the results of the full Kyber counterpart, double the data in the Tab.5.

5.4 SCAs of unprotected Dilithium and Kyber

In this experiment, we conduct SCAs on unprotected Dilithium2 and Kyber512 implementations. We then present the time overhead associated with recovering the complete private keys using the SIS-assisted method. The Montgomery reduction operation, crucial in both algorithms, is executed similarly in their respective C reference implementations. For clarity, Fig.5 illustrates the specific process of Dilithium and is used as an example.

```
temp1 = int64(NTT(c)*NTT(s1))
temp2 = int32(temp1*qinv)
temp3 = int64(temp2*q)
temp4 = (temp1-temp3)>>32
```

Figure 5: Intermediate values of Montgomery reduction operation in Dilithium.

Utilizing the Hamming Weight model, we examine the PCCs between intermediate values and power traces, illustrated in Fig.6. With a sample of 5,000 traces, both temp1 and temp2 show PCC peaks at the sample index 300 due to sharing the same lower 12 bits. However, temp1's actual leakage peak is slightly earlier and smaller. Since the last 32 bits of temp3 are identical to temp1, they are not displayed in Fig.6. The highest PCC value corresponding to temp4 is close to 1 and occurs at sample index 380. This substantial leakage is likely caused by the nonlinear shift operation and the subsequent STR operation storing the result in memory. Using Alg.3, temp2 is selected to compute the last 20 bits, and temp4 is employed to recover the final result. In this experiment, we designate 4 bits per block, set $th_{fin} = 0.8$, and choose 20 feature points near the highest PCC.

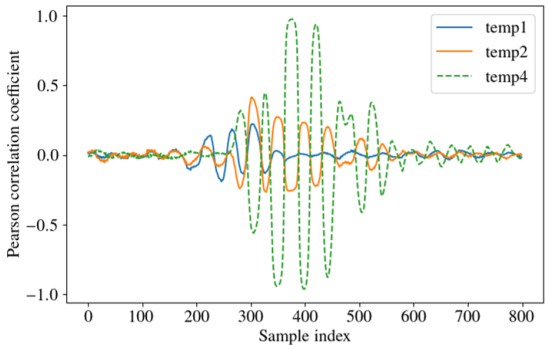


Figure 6: PCCs for the intermediate values temp1 and temp2 in Dilithium2.

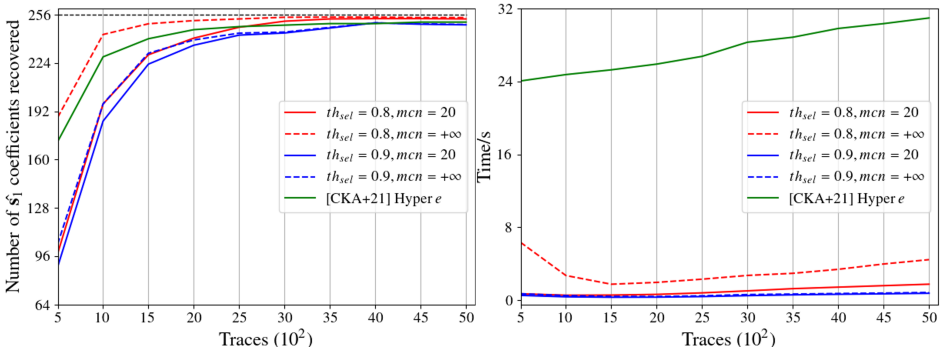


Figure 7: $\hat{\boldsymbol{s}}_1$ coefficients recovered and single coefficient time cost for unprotect Dilithium2.

Following the CPA scheme detailed in Sec.4.2, we set the parameters $th_{sel} = 0.8$ or 0.9, and $mcn = 20$ or $+\infty$. We also replicate Chen *et al.*'s optimal Hyper e scheme for comparison purposes. Fig.7 depicts the number of recovered $\hat{\boldsymbol{s}}_1$ coefficients and the single coefficient time cost for different trace amounts. Under $th_{sel} = 0.9$, the influence of different mcn values on the number of recovered coefficients is marginal. With 1,000 traces, around 180 coefficients are recovered, while 4,000 traces yield approximately 250. For $th_{sel} = 0.8$ and $mcn = 20$, the recovery trend with fewer traces mirrors that of $th_{sel} = 0.9$, but it allows for the recovery of more coefficients with over 2,000 traces, stabilizing at 253 coefficients with 4,000 traces. When $mcn = +\infty$, recovery is notably efficient with smaller trace amounts—about 190 coefficients with 500 traces and over 250 with 2,000 traces. Chen *et al.*'s method recovering around 170 coefficients with 500 traces and 251 with 5,000. The success rate of all schemes spikes with fewer than 2,000 traces and plateaus beyond 4,000. In terms of time cost, our approach significantly surpasses that of Chen *et al.*. With $th_{sel} = 0.9$, the time cost per recovered coefficient remains below one second for up to 5,000 traces. For $th_{sel} = 0.8$ and $mcn = 20$, the attack completes within a second for up to 3,000 traces. With $mcn = +\infty$, the time cost initially declines before rising as the number of traces increases, taking six seconds for 500 traces and 1.75 seconds for 1,500 traces.

Table 6: Comparison of experimental results of Dilithium2

Method	#Traces	Success Rate	Time(s)
$th_{sel} = 0.8$ $mcn = 20$	5,000	99.13	1.75
$th_{sel} = 0.8$ $mcn = \infty$		99.22	4.44
$th_{sel} = 0.9$ $mcn = 20$		97.66	0.75
$th_{sel} = 0.9$ $mcn = \infty$		97.66	0.84
[CKA+21] Hyper e		98.05	30.31
[CKA+21] Enumerate	200	100	483.66

Tab.6 contrasts our suggested method with the Chen *et al.*'s [CKA+21] optimized enumeration and Hyper e strategies. Utilizing 5,000 traces, our technique yields success rates that are either equivalent to or better than those of the other methods. Notably, our approach slashes the time requirement, delivering a speed boost ranging from 7 to 40 times. While the enumeration method can guarantee a 100% success rate with a mere 200 traces, its time cost is prohibitively high, making it unsuitable for practical attack.

```
1 void basemul(int16_t r[2], const int16_t a[2], const int16_t b[2], int16_t
      zeta){
2   r[0]  = fqmul(a[1], b[1]);
3   r[0]  = fqmul(r[0], zeta);
4   r[0] += fqmul(a[0], b[0]);
5   r[1]  = fqmul(a[0], b[1]);
6   r[1] += fqmul(a[1], b[0]);}
```

Figure 8: Kyber $\hat{\boldsymbol{c}} \circ \hat{\boldsymbol{s}}_k$ reference implementation.

Kyber undergoes seven rounds of butterfly transformations, resulting in five Montgomery reductions for each coefficient pair, as depicted in Fig.8. Shen *et al.* [SCZ+23] analyzed the impact of selecting different operations for attack and the number of traces required. For consistency in evaluating the efficiency of the attack under the same experimental setup, we opted to target the operation corresponding to coefficient multiplication for SCAs, specifically, lines 2 and 4 in Fig.8. The parameter choices for the CPA scheme are identical to those used for Dilithium, with the modification of determining the lower 8 bits.

The results of our experiments indicate that our scheme is also effective for Kyber. With $th_{sel} = 0.9$ or 0.8, approximately 175 and 210 coefficients are correctly recovered at 500 traces, respectively. This number increases with more traces and stabilizes at 4,000 traces. We replicated Shen *et al.*'s [SCZ+23] experiment, which involved randomly selecting ciphertexts under identical conditions, and compared the results in Tab.7. Although Shen *et al.*'s method can recover a coefficient in 0.4 seconds, our approach doubles the recovery speed while maintaining a high success rate.

Table 7: Comparison of experimental results for Kyber512

Method	#Traces	Success Rate	Time(s)
$th_{sel} = 0.8$ $mcn = 20$	5,000	98.44	0.28
$th_{sel} = 0.8$ $mcn = \infty$		99.21	0.32
$th_{sel} = 0.9$ $mcn = 20$		95.29	0.17
$th_{sel} = 0.9$ $mcn = \infty$		95.70	0.16
[SCZ+23] Random ciphertext	200	100	0.41

Our methodology prioritizes feature points tied to the most pronounced intermediate values of leakage, significantly cutting down the time needed for a single CPA execution. The parameters th_{sel} and mcn provide flexibility, allowing for optimization between the number of coefficients to be recovered and the number of traces at hand, ensuring optimal success rates. Importantly, our approach demonstrates heightened efficiency, especially in the context of larger finite fields, as seen with Dilithium.

If only SCAs are used, even though our CPA scheme can get a high success rate when traces are sufficient, the coefficients remaining for recovery still need to be obtained by the enumeration scheme proposed by Chen *et al.* [CKA+21]. In our experimental setup, the optimal scheme of Chen *et al.* takes over 3 hours to recover a set of normal domain private keys, and the introduction of the SIS-assisted method allows us to stop SCA after obtaining a small number of NTT domain values. Tab.8 gives the time required to recover a set of normal domain private keys for Dilithium2 using 2,000 traces. If an enumeration

Table 8: Time to recover a subset of normal domain private keys for Dilithium (min)

	without SIS	NTT_{256}	NTT_{128}	NTT_{64}	NTT_{32}
$th_{sel} = 0.8, mcn = 20$	131.5	2.6	1.1	0.9	0.8
$th_{sel} = 0.8, mcn = +\infty$	40.5	7.5	2.8	2.3	2.2
$th_{sel} = 0.9, mcn = 20$	170.4	1.5	0.7	0.6	0.5
$th_{sel} = 0.9, mcn = +\infty$	138.5	1.8	0.8	0.7	0.6
[CKA+21] Hyper e	191.3	95.6	31.2	22.5	19.8

The result includes the time overhead of SCAs and SIS-assisted methods. Where NTT_{256} uses SCAs to recover 220 coefficients and the $\text{NTT}_{128,64,32}$ are as few coefficients as possible.

scheme is used to recover the error coefficients, the minimum time required to complete the attack is 40 minutes for the four sets of parameters provided. When using the SIS-assisted scheme, the time overhead is significantly reduced and the attack time is reduced to less than 2 minutes. With the divide-and-conquer strategy, a group of normal domain private keys can be recovered in less than 1 minute.

For the Kyber algorithm, the smaller finite domain makes it possible to recover the complete private key in 104 seconds by enumerating all the candidate values during the SCA process, even if only 200 traces are used. At this point, combining with the SIS-assisted method can also increase the attack speed by three times, and the attack can be completed in about 30 seconds.

5.5 SCAs of masked Dilithium

Masked Dilithium implementation [CGTZ23] divides the private key into multiple shares in the key generation process, which are used for NTT operation respectively in the signature process. The adversary can first recover the shares separately, and then modulo adds the coefficients of successful attacks in the same position and finally uses the SIS-assist method to recover the private key.

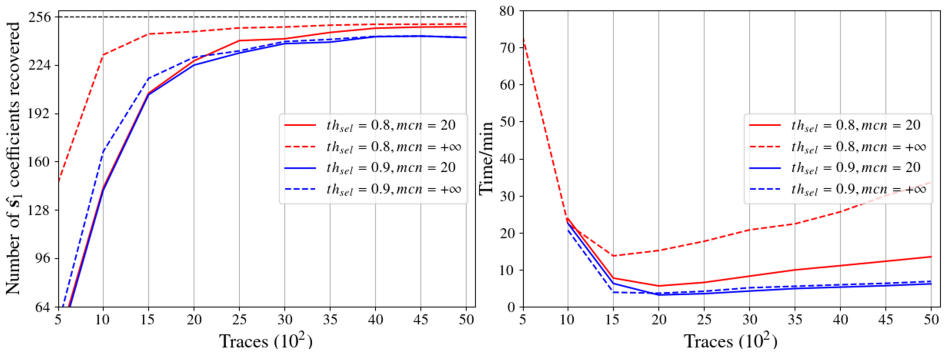


Figure 9: Number of $\hat{\boldsymbol{s}}_1$ coefficients recovered and time cost of masked Dilithium2.

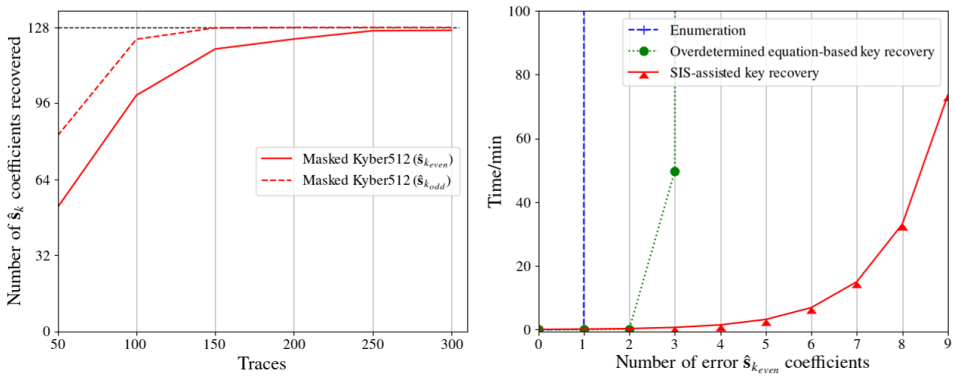
We performed an SCA against the first-order masked Dilithium. Fig.9 displays the number of $\hat{\boldsymbol{s}}_1$ coefficients recovered and the time cost for recovering a subset of 256 coefficients of $\boldsymbol{s}_1$ using the SIS-assisted method (without divide-and-conquer strategy). When $th_{sel} = 0.8$ and $mcn = +\infty$, $\boldsymbol{s}_1$ can be recovered using 500 traces, taking 73 minutes. When $th_{sel} = 0.9$ and $mcn = +\infty$, we obtain the minimum time required to recover $\boldsymbol{s}_1$ of 3.2 minutes using 2,000 traces. When using a divide-and-conquer SIS-assisted method, the attack can even be completed in just 1 minute.

For the [CGTZ23] scheme of arbitrary k^{th} order mask Dilithium, our method needs to recover all coefficients in the same position accurately to complete the subsequent attack. Assuming that the minimum number of coefficients required for the SIS-assisted method to recover the complete private key is m. Considering the worst case, the attacker only

needs to ensure that the SCA success rate (sr) satisfies $sr > \frac{256k+m}{256(k+1)}$. Taking the 5^{th} order mask scheme as an example, the success rate just needs to be greater than 91%, which is entirely feasible based on our practical SCA.

5.6 SCAs of masked Kyber

Masked Kyber has assembly optimizations for polynomial multiplication and Montgomery reduction. A pair of coefficients is changed from requiring 5 Montgomery reduction executions to 3, which makes it necessary for an attacker to get the correct even position coefficients before recovering the odd position coefficients. In addition, the optimized Montgomery reduction requires only two assembly instructions. All the intermediate values the previous schemes can use to recover the low bit are difficult to utilize. So, we cannot use the CPA scheme to determine whether the coefficients are recovered correctly.



(a) Number of $\hat{s}_k$ coefficients recovered (b) Time cost to recover a set of private key

Figure 10: SCAs result of Masked Kyber512.

In the practical attack of first-order masked Kyber512, we opt to utilize the outcome of the Montgomery reduction as the intermediate value and employ CPA. Considering that the implementation of Heinz *et al.* [HKL+22] was optimized using assembly. The entire attack process is systematically divided into four stages. 1) Employing SCAs to recover both $\hat{\boldsymbol{s}}^0_{k_{odd}}$ and $\hat{\boldsymbol{s}}^1_{k_{odd}}$; 2) Calculating $\boldsymbol{s}_{k_{odd}}$; 3) Utilizing SCAs to recover both $\hat{\boldsymbol{s}}^0_{k_{even}}$ and $\hat{\boldsymbol{s}}^1_{k_{even}}$; 4) Computing to determine $\boldsymbol{s}_{k_{even}}$. Fig.10(a) showcases the results of the SCA corresponding to stages 2) and 4). Overall, attacking coefficients in odd positions works better, requiring only 200 traces to recover all coefficients accurately. However, only 123 coefficients can be retrieved for the even positions at this juncture. For this attack, it is possible to keep increasing the number of traces until all coefficients can be recovered correctly, and this experiment achieves that goal at 500 traces in 3 minutes.

However, in practice, the attacker may face limitations on the number of traces that can be captured, or the SNR of the captured traces may be very low due to platform characteristics. In such cases, the success rate of SCA may not reach 100%. When it is not possible to determine whether the coefficients are wrong or not, according to the discussion in Sec.3.2 we can still try to use the SIS-assisted method to recover the private key. Fig.10(b) depicts the time overhead required by the direct enumeration, the overdetermined equation-based, and the SIS-assisted method to recover the complete private key in this experimental setting for varying SCA result errors. The direct enumeration method only applies to the case of one error in the SCA result. The overdetermined equation-based method relax to 3 error coefficients, which takes about 1 hour to recover the private key. The SIS-assisted method significantly outperforms the others. Even with 9 error

coefficients, it can complete the attack in 1.4 hours. The above results show that the theoretical success rate of the SCA is more than 93%, which theoretically allows the recovery a subset of 256 normal domain private keys of masked Kyber512.

6 Conclusion and future work

In this work, for the post-quantum algorithms Dilithium and Kyber, we introduce overdetermined systems and SIS-assisted methods to efficiently recover the normal domain private key after obtaining an incomplete NTT domain private key. These methods address the challenge of efficiently recovering the normal domain private key when only a subset of the NTT domain private key is available. Combining the NTT characteristics, we propose a SIS-assisted method based on the divide-and-conquer strategy. For Dilithium2 it is possible to recover a set of private keys after obtaining at least 32 NTT domain coefficients. After obtaining 64 NTT domain coefficients, it is possible to recover a set of private keys for Kyber512. Additionally, we present an efficient SCAs scheme for large number multiplications based on CPA. When paired with the SIS-assist methods, this allows for the recovery of a complete set of normal domain private keys for unprotected Dilithium2 and Kyber512 within a minute on the ARM Cortex-M4 platform. The recovery time increases to less than five minutes for first-order masking schemes.

Theoretically, the multi-dimensional nature of the lattice ensures the security of the lattice-based cryptosystem. With current computing power, it is almost impossible to brute force a private key through an exhaustive search. However, each coefficient is calculated independently when the algorithm is executed on a cryptographic platform. From an SCA perspective, this independence means that as the dimensionality increases, attackers will only face repeated challenges rather than more complex ones. A significant difficulty in performing SCA on lattice-based algorithms such as Dilithium is the requirement of recovering all the coefficients correctly, which is often difficult to guarantee. However, due to the introduction of NTT operations to improve efficiency, the requirement on the SCA success rate can be greatly relaxed using our proposed SIS-assisted method, and it is sufficient to recover approximately 1/6 of the NTT domain coefficients. This further emphasizes the importance of implementing these algorithms securely.

While our SIS-assist techniques have proven impactful, experimental outcomes suggest there's room for improvement to approach the theoretical lower bound of required NTT domain coefficients. Moving forward, our focus will be on delving deeper into the arithmetic properties of NTT in Dilithium and Kyber, striving to achieve private key recovery with even fewer NTT domain coefficients.

References

[ABC+23] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. Protecting dilithium against leakage revisited sensitivity analysis and improved implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):58–79, 2023.

[ABD+20] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation (version 3.0), 2020.

[ACD+22] Gorjan Alagic, David Cooper, Quynh Dang, Thinh Dang, John M. Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl A. Miller, Dustin Moody, Rene Peralta, Ray

Perlner, Angela Robinson, Daniel Smith-Tone, and Daniel Apon. Status report on the third round of the nist post-quantum cryptography standardization process, 2022-07-05 04:07:00 2022.

[ADP18] Martin R. Albrecht, Amit Deo, and Kenneth G. Paterson. Cold boot attacks on ring and module LWE keys under the NTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):173–213, 2018.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. *Electron. Colloquium Comput. Complex.*, TR96, 1996.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *IACR Cryptology ePrint Archive*, 2016.

[BGR+21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):173–214, 2021.

[BVC+23] Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting intermediate value leakage in dilithium: A template-based approach. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):188–210, 2023.

[CFG+11] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Improved collision-correlation power analysis on first order protected AES. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2011.

[CGTZ23] Jean-Sébastien Coron, François Gérard, Matthias Trannoy, and Rina Zeitoun. Improved gadgets for the high-order masking of dilithium. *IACR Cryptol. ePrint Arch.*, page 896, 2023.

[CKA+21] Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma, and Jiwu Jing. An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In *39th IEEE International Conference on Computer Design, ICCD 2021, Storrs, CT, USA, October 24-27, 2021*, pages 583–590. IEEE, 2021.

[FDK20] Apostolos P. Fournaris, Charis Dimopoulos, and Odysseas G. Koufopavlou. Profiling dilithium digital signature traces for correlation differential side channel attacks. In Alex Orailoglu, Matthias Jung, and Marc Reichenbach, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation - 20th International Conference, SAMOS 2020, Samos, Greece, July 5-9, 2020, Proceedings*, volume 12471 of *Lecture Notes in Computer Science*, pages 281–294. Springer, 2020.

[HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen ciphertext k-trace attacks on masked CCA2 secure kyber. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):88–113, 2021.

[HKL+22] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. First-order masked kyber on ARM cortex-m4. *IACR Cryptol. ePrint Arch.*, page 58, 2022.

[HLK+21] Jaeseung Han, Taeho Lee, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Jihoon Cho, Dong-Guk Han, and Bo-Yeon Sim. Single-trace attack on NIST round 3 candidate dilithium using machine learning-based profiling. *IEEE Access*, 9:166283–166292, 2021.

[KdG21] Alexandre Karlov and Natacha Linard de Guertechin. Power analysis attack on kyber. *IACR Cryptol. ePrint Arch.*, page 1311, 2021.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KPR+] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4.

[LCC+06] Thanh-Ha Le, Jessy Clédière, Cécile Canovas, Bruno Robisson, Christine Servière, and Jean-Louis Lacoume. A proposition for correlation power analysis enhancement. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2006.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Miklós Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[LZS+21] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage. *IEEE Trans. Inf. Forensics Secur.*, 16:1868–1879, 2021.

[MJZ22] Ruiqi Mi, Haodong Jiang, and Zhenfeng Zhang. Lattice reduction meets key-mismatch: New misuse attack on lattice-based nist candidate kems. Cryptology ePrint Archive, Paper 2022/1064, 2022. https://eprint.iacr.org/2022/1064.

[MUTS22] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on dilithium: A small bit-fiddling leak breaks it all. *IACR Cryptol. ePrint Arch.*, page 106, 2022.

[PG12] Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, volume 7533 of *Lecture Notes in Computer Science*, pages 139–158. Springer, 2012.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017.

[QLZ+23] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Jingdian Ming, Chengbin Jin, and Huizhong Li. Practical public template attack attacks on crystals-dilithium with randomness leakages. *IEEE Trans. Inf. Forensics Secur.*, 18:1–14, 2023.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[RRD+23] Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the limits of generic side-channel attacks on lwe-based kems - parallel PC oracle attacks on kyber KEM and beyond. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):418–446, 2023.

[SCZ+23] Muyan Shen, Chi Cheng, Xiaohan Zhang, Qian Guo, and Tao Jiang. Find the bad apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of kyber. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):89–112, 2023.

[SE91] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1991.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.

[THM+07] Michael Tunstall, Neil Hanley, Robert P. McEvoy, Claire Whelan, Colin C. Murphy, and William P. Marnane. Correlation power analysis of large word sizes. In *IETIrish Signals and Systems Conf - ISSC*, pages 145–150, 2007.

[TUX+23] Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):473–503, 2023.

[XPR+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David F. Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Trans. Computers*, 71(9):2163–2176, 2022.

[YWY+23] Yipei Yang, Zongyue Wang, Jing Ye, Junfeng Fan, Shuai Chen, Huawei Li, Xiaowei Li, and Yuan Cao. Chosen ciphertext correlation power analysis on kyber. *Integr.*, 91:10–22, 2023.