Bit Security Analysis of Lattice-Based KEMs under Plaintext-Checking Attacks

Ruiqi Mi[1,2][0000−0003−2123−8491] Haodong Jiang[3] and Zhenfeng Zhang[1,2]

[1] University of Chinese Academy of Sciences, Beijing 100049, China
[2] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{ruiqi2017,zhenfeng}@iscas.ac.cn
[3] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, Henan, China.
hdjiang13@163.com

Abstract. Plaintext-checking attack (PCA) is a type of attack where an adversary recovers the secret key with the help of a plaintext-checking (PC) oracle that decides if a given ciphertext decrypts to a given plaintext. In particular, PCA exists in both the key misuse attacks for IND-CPA-secure lattice-based KEMs and generic side-channel attacks for IND-CCA-secure lattice-based KEMs. The query number of PC-oracle is a vital criterion for evaluating a PCA attack. Recently, Qin et al. [ASIACRYPT 2021] gave a systematic approach to finding the theoretical lower bound of PC-oracle query numbers for NIST-PQC lattice-based KEMs. Most of the prior works consider the substantial Oracle queries needed to recover the entire key. However, the adversary often has inadequate access to PC Oracles to fully recover the secret key. The concrete bit security loss with arbitrary PC Oracle access is unknown.

In this paper, we give a unified method to analyze the bit security loss with arbitrary PC Oracle access for lattice-based KEMs. First, we model the information leakage in the PC Oracle by PC-hint, and give a generic transformation from PC-hints to the perfect inner-product hint, which allows the adversary to integrate PC-hints progressively. Then, following the security analysis for LWE with the perfect inner-product hint given in Dachman-Soled et al. [CRYPTO 2020], we give a concrete relationship between the PC Oracle query number and the bit-security of the lattice-based KEM under PCA. Our proposed method is applicable to all CCA-secure NIST candidate lattice-based KEMs. Applying our methods to NIST-PQC lattice-based KEMs, we get the bit-security loss of the lattice-based KEM under PCA. Take Kyber768 (original 182-bit-security) as an example, the bit security of Kyber768 is reduced to 128 after 444 PC-oracle queries and reduced to 64 after 998 PC-oracle queries, while in Qin et al. [ASIACRYPT 2021] 1774 queries are required to recover the whole secret key. Our analysis also demonstrates the possibility of reducing the Oracle queries needed in PCA. The adversary may stop querying plaintext-checking oracle and solves the remaining part of reused secret offline with the help of lattice reduction algorithms when the cost of lattice reduction algorithms becomes acceptable.

Keywords: Lattice-based cryptography · Plaintext-checking Attacks · KEM · Learning With Errors · Kyber.

1 Introduction

Current Diffie-Hellman key exchange and other widely used public key cryptography based on factoring or discrete logarithm problems will no longer be secure if large-scale quantum computers become available. According to the roadmap released by the US National Institute of Standards and Technology (NIST) and the Department of Homeland Security [19], the transition to post-quantum standards should be completed by 2030.

NIST began the call for post-quantum cryptography algorithms from all over the world in February 2016 [18]. In the third round, there are 4 finalists and 5 alternative candidates for Public Key Encryption (PKE) or Key Encapsulation Mechanism (KEM). There are 3 lattice-based KEMs among the 4 finalists. After careful analysis, NIST has selected one finalist and four alternate candidates to move on to the fourth round. Crystals-Kyber [3] is the first PKE/KEM candidate to be standardized, which is based on the lattice assumption [20].

The construction of CPA-secure PKE usually follows the design pattern given in [16]. Most of the lattice-based NIST candidate CPA-secure KEMs are designed in such a pattern (e.g. Crystals-Kyber [3], Saber [8], FrodoKEM [17], NewHope [22]), and their hardness comes from the Learning With Errors (LWE) problem [27]. All LWE-based KEMs in Rounds 2 and 3 of the NIST standardization use a Fujisaki-Okamoto (FO) transformation [10] to achieve IND-CCA security.

The ongoing standardization process raises an important question: a plaintext-checking attack may happen when the public key is reused, thus there is no security guarantee on both IND-CPA and IND-CCA KEMs. For IND-CPA secure KEM, the plaintext-checking attack runs as follows. Suppose Alice reuses her public key pk_A. The adversary $\mathcal{A}$ impersonates Bob and tries to recover each coefficient of Alice's reused secret key sk_A with the help of the plaintext-checking oracle (PC Oracle) $\mathcal{O}$. $\mathcal{A}$ crafts ciphertext ct and shared secret K and sends ct, K to $\mathcal{O}$. $\mathcal{O}$ determines if the two shared keys match or not. For each coefficient $sk_A[i]$ of sk_A, $\mathcal{A}$ determines the subset to which $sk_A[i]$ belongs based on $\mathcal{O}$'s reply. For IND-CCA secure KEMs, a plaintext-checking attack can also be launched with the help of side-channel information. According to [26], FO transformation can be bypassed by accessing physical decapsulation devices and collecting useful match or mismatch information.

Practically, the adversary $\mathcal{A}$ often has limitations in gathering sufficient perfect side-channel information and constructing a PC Oracle. In plaintext-checking attacks, users may stop misusing their public key in a short time. Thus, the adversary has restricted time to query the PC Oracle and fully recover the secret key. For example, the adversary $\mathcal{A}$ can only access a PC Oracle that is constructed from a USB key for online banking service before the users report the loss. PC Oracle constructed by reusing the KEM's public key cannot be accessed when users stop reusing the public key. Thus, an optimal plaintext-checking attack has the least number of plaintext-checking Oracle queries for successful key recovery. There are numerous works on reducing oracle queries in the plaintext-checking attack [9, 4, 23, 21, 24, 12, 29, 14, 25]. All these attacks aim to fully recover the reused secret with as few queries as possible. Qin et al. [25] gave a systematic approach to finding the theoretical lower bound of PC-oracle query numbers for all NIST-PQC lattice-based IND-CPA/IND-CCA secure KEMs. The calculation of their lower bounds is essentially the computation of a certain Shannon entropy. Thus, one cannot find a better attack with fewer queries on average for full key recovery. Their lower bounds are also confirmed by experiments.

Most of the prior works about plaintext-checking attacks to lattice-based KEMs focus on recovering the full reused secret key. However, the adversary may not have enough oracle access to construct a reliable plaintext-checking oracle for recovering the full secret. Thus, compared to recovering the full secret with substantial amounts of oracle queries, one may be interested in the following question: *How to analyze the concrete bit security loss of PKE/KEM after a limited number of oracle queries in plaintext-checking attacks?*.

Some works already analyzed the effects of information leakage on the LWE problem. For example, Dachman-Soled et al. [7] give a general framework to analyze the influence of side-channel information. They provide four types of side-channel information ("hint") and analyze the concrete security loss for each type of hint. However, the side-channel information leaked in the plaintext-checking attack ("plaintext-checking hint") has not been considered. Thus, it remains unclear how to analyze the influence of plaintext-checking attacks on the hardness of LWE information.

Let $\mathcal{S} = \{\boldsymbol{S_0}, \boldsymbol{S_1}, ..., \boldsymbol{S_{n-1}}\}$ be the set of all possible values for one coefficient block and its corresponding probabilities $\{P_0, P_1, ..., P_{n-1}\}$. For a single coefficient block $sk_A[i]$, $P_j = Pr(sk_A[i] = \boldsymbol{S_j} | sk_A \leftarrow \mathcal{S})$ for $j = 0, 1, ..., n-1$. Let $H(\mathcal{S})$ the Shannon entropy for $\mathcal{S}$, Typically, we have $H(\mathcal{S}|PChint) \leq H(\mathcal{S})$. In other words, each oracle query decreases the Shannon entropy of Alice's reused secret sk_A. $\mathcal{O}$ returns a bit b depending on whether the plaintext matches or not. Intrinsically, for each coefficient of reused secret key sk_A, the querying process can be described as a function $\boldsymbol{f}$ of reused secret key sk_A, plaintext pt, ciphertext ct, in which:

$$\boldsymbol{f}(sk_A, ct, pt) = b \in \{0, 1\}$$

We define such type of side-channel information as a plaintext-checking hint. Suppose the adversary $\mathcal{A}$ tries to recover the i-th coefficient of sk_A. Let v be a unit vector with $v[i] = 1$. Thus it is very natural to express $\boldsymbol{f}(sk_A, ct, pt)$ as:

$$\boldsymbol{f}(sk_A, ct, pt) := \boldsymbol{f}(\langle sk_A, v \rangle)$$

$\boldsymbol{f}(sk_A, ct, pt)$ is a general description of plaintext-checking hint. We find a solution to transform plaintext-checking hints to known hints for lattice-based KEMs.

Contributions. The main contributions of this paper include:

-We give a unified method to analyze the bit security loss even with very little PC Oracle access for all lattice-based NIST candidate KEMs. Our basic idea is to give a unified description of the information leakage in the PC-oracle (called PC-hint). PC-hint is described in the form of $\boldsymbol{f}(sk_A, ct, pt)$, which is suitable for analyzing a security loss when the adversary has limited Oracle access. Then, we give a generic transformation from the least number of PC-hints to the perfect inner-product hint for lattice-based KEMs in the form of $\langle sk_A, v\rangle = l$. The least number of PC-hints needed in such a transformation is the lower bound of oracle queries needed to recover a single coefficient block as analyzed in [25]. We show that PC-hints can be transformed into a perfect inner-product hint when a coefficient block of sk_A is recovered, and the adversary can integrate PC-hints progressively. Finally, by following the security analysis for LWE with the perfect inner-product hint given in Dachman-Soled et al. [7], we establish a concrete relationship between the PC-oracle query number and the bit-security of the lattice-based KEM under PCA in Section 4. Our proposed method is applicable to all CPA-secure and CCA-secure NIST candidate lattice-based KEMs.

-We analyze the bit security loss under the plaintext-checking attack for all lattice-based NIST candidate KEMs with the help of the toolbox given in [7]. We present the relationship between bit security and plaintext-checking oracle query times in Table 1. The number in parentheses is the Oracle query needed when classical bit security is 100 (original classical bit-security less than 128). Note that the classical bit security of Kyber512 is 118. The classical bit security of LightSaber is 118. The classical bit security of NewHope512 is 112. The bit security of Kyber768 is reduced to 128 after 444 PC-oracle queries and further reduced to 64 after 998 PC-oracle queries, whereas 1774 queries are required to recover the entire secret key, as indicated in [25]. We provide the concrete relationship between the number of oracle queries and classical/quantum bit security in Section 5. Such a result reminds us that the loss of security is non-negligible even when the adversary cannot fully recover the secret key.

-Based on the analysis above, the plaintext-checking attack can be further enhanced by combining Qin's plaintext-checking attack [25] with standard lattice reduction techniques. The adversary may stop querying the plaintext-checking oracle and solve the remaining part of the reused secret offline with the help of lattice reduction algorithms when the cost of lattice reduction algorithms becomes acceptable. We present a detailed analysis of dimension, volume, and lattice basis after each PC-hint integration in Section 4.3. These results can be directly used as input to lattice reduction algorithms when the cost becomes acceptable.

Organizations. We start with some preliminaries in Section 2. Section 3 models the secret leakage in plaintext-checking attack (plaintext-checking hint). Section 4 gives a concrete mathematical expression of plaintext-checking hint (Section 4.2), explains how to integrate plaintext-checking hint into the lattice (Section 4.3). Section 5 gives experimental results. It shows the concrete relationship between bit security and the number of oracle queries for NIST second-round KEM candidates: Kyber, Saber, Frodo and NewHope.

Table 1. Relationship between the number of queries and classical bit security of all lattice-based NIST KEMs, $E(\#Queries)$ denotes the theoretical lower bound for the number of queries given in [25]. The number in parentheses is the Oracle queries needed when classical bit security is 100 (original classical bit-security less than 128).

Bit Security	**Kyber512** (E(#Queries)=1312)	**Kyber768** (E(#Queries)=1774)	**Kyber1024** (E(#Queries)=2365)
128(100)	(80)	444	950
64	533	998	1459
Bit Security	**LightSaber** (E(#Queries)=1460)	**Saber** (E(#Queries)=2091)	**FireSaber** (E(#Queries)=2642)
128(100)	(228)	612	1181
64	631	1230	1782
Bit Security	**Frodo640** (E(#Queries)=18329)	**Frodo976** (E(#Queries)=26000)	**Frodo1344** (E(#Queries)=29353)
128	833	8177	14006
64	8005	15445	20234
Bit Security	**NewHope512** (E(#Queries)=1660)		**NewHope1024** (E(#Queries)=3180)
128(100)	(137)		1406
64	571		2140

Independent and Concurrent Work. Very recently, Guo and Mårtensson [13] showed an improved plaintext-checking attack that recovers multiple secret coefficients in a parallel way. The comparisons are summarized below:

1 Guo and Mårtensson showed how to recover partial information of multiple secret entries in each oracle call. The adversary split the two-dimensional plane for two secret coefficients and decides from the mismatch oracle call which part the two coefficients belong to. Compared to the lower bound given in [25], the attack given in [13] reduces the number of queries needed by 0.08%, 10.6%, 10.6% for Kyber512, Kyber768, Kyber1024, and 3.4%, 5.01%, 8.1% for LightSaber, Saber, FireSaber.

2 In the discussion part, they give a rough estimation of the query sample complexity for Kyber and Saber when post-processing is allowed. They employ the lattice estimator given in [2]. They did not give concrete relationship between the number of queries and the geometry of the lattice in theory.

In our work, we investigate the form of side information the adversary get from oracle queries and the volume and dimension change after integrating such information in Section 4.2. We give a quantitative and detailed analysis between query times and bit security in theory. Besides, we applied our theory to all NIST second-round KEM candidates except NTRU [6] and NTRU Prime [5].

2 Preliminaries

A lattice is a discrete additive subgroup of $\mathbb{R}^m$, denoted as Λ. Lattice Λ is generated by a set of linearly independent *basis* $\{\boldsymbol{b_j}\} \subset \mathbb{R}^m$, that is $\Lambda :=$

$\{\Sigma_j z_j \boldsymbol{b_j} : z_j \in \mathbb{Z}\}$. The i-th *successive minimum* of a lattice, $\lambda_i(\Lambda)$, is the radius of the smallest ball centered at the origin containing at least i linearly independent lattice vectors.

We denote the dimension of lattice Λ as m and the rank as n. If $n = m$, the lattice is full rank. Matrix $\boldsymbol{B}$ having all basis vectors as rows can be called a *basis*. The volume of the lattice is defined as $Vol(\Lambda) := \sqrt{det\,(\boldsymbol{BB^T})}$. The dual lattice of Λ in $\mathbb{R}^n$ is defined as:

$$\Lambda^* := \{\boldsymbol{y} \in Span(\boldsymbol{B}) \mid \forall x \in \Lambda, \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}\} \tag{1}$$

Definition 1 (search-LWE problem with short secrets). *Let n, m, q be positive integers, and let χ be a distribution over $\mathbb{Z}$. The search LWE problem (with short secrets) for parameters (n, m, q, χ) is:*

***Given** the pair $(\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}, \boldsymbol{b} = \boldsymbol{z}\boldsymbol{A^T} + \boldsymbol{e} \in \mathbb{Z}_q^m)$ where:*

1. $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ is sampled uniformly at random.

2. $\boldsymbol{z} \leftarrow \chi^n$, and $\boldsymbol{e} \leftarrow \chi^m$ are sampled with independent and identically distributed coefficients following the distribution χ.

***Find** $\boldsymbol{z}$.*

The complexity of solving (search-)LWE against primal attack consists of viewing the LWE as an instance of (Distorted-)Bounded Distance Decoding problem, reducing DBDD to uSVP(via Kannan's Embedding [15], and finally applying lattice reduction algorithm to solve the uSVP instance [1]. DBDD accounts for potential distortion in the distribution of the secret noise vector that is to be recovered, and the secret noise vector is found at a lower cost.

Definition 2 (γ-uSVP). *given a lattice Λ such that $\lambda_2(\Lambda) > \gamma\lambda_1(\Lambda)$, find a shortest nonzero vector in Λ.*

Definition 3 (Distorted Bounded Distance Decoding Problem, DBDD). *Let $\Lambda \subset \mathbb{R}^d$ be a lattice, $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ be a symmetric matrix and $\boldsymbol{\mu} \in Span(\Lambda) \subset \mathbb{R}^d$ such that $Span(\boldsymbol{\Sigma}) \subsetneq Span\left(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T\boldsymbol{\mu}\right) = Span(\Lambda)$*

The Distorted Bounded Distance Decoding Problem $\boldsymbol{DBDD_{\Lambda,\mu,\Sigma}}$ is:

***Given** $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and a basis of Λ.*

***Find** the unique vector $\boldsymbol{x} \in \Lambda \cap E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.*

Where $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the ellipsoid

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \{\boldsymbol{x} \in \boldsymbol{\mu} + Span(\boldsymbol{\Sigma}) \mid (\boldsymbol{x} - \boldsymbol{\mu}) \cdot \boldsymbol{\Sigma}^{\sim} \cdot (\boldsymbol{x} - \boldsymbol{\mu})^T \le rank(\boldsymbol{\Sigma})\} \tag{2}$$

The triple $I = (\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ will be referred to as the instance of the $DBDD_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ problem.

Definition 4 (Primitive Vectors). *A set of vector $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_k \in \Lambda$ is said primitive with respect to Λ if $\Lambda \cap Span(\boldsymbol{y}_1, \cdots, \boldsymbol{y}_k)$ is equal to the lattice generated by $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_k$. Equivalently, it is primitive if it can be extended to a basis of Λ. If $k = 1$, $\boldsymbol{y}_1$, this is equivalent to $\boldsymbol{y}_1/i \notin \Lambda$ for any integer $i \ge 2$.*

3 Side-Channel Information In Plaintext-Checking Attacks

3.1 The meta-structure of IND-CPA secure KEM

Suppose there exists six *additive Abelian groups* S_{sk}, S_A, S_B, S_t, S_U, S_V and four *bilinear mappings*(denoted as $\times$). The four bilinear mappings are $S_A \times S_{sk} \to S_B$, $S_U \times S_{sk} \to S_V$, $S_t \times S_A \to S_U$, $S_t \times S_B \to S_V$. The multiplication satisfies associativity in the sense that $(t \times A) \times sk = t \times (A \times sk)$ for all $t \in S_t$, $A \in S_A$, and $sk_A \in S_{sk}$. The multiplication works as **block 1** on Fig. 1.

We list the meta-structure of CPA-secure KEM in Algorithm 1, in which:

Algorithm 1 The meta-structure of IND-CPA secure KEM

```
1: function $setup(1^\lambda)$
2:     setup the algebra
3:     define public parameter $pp$
4:     return $pp$
1: function $Gen(pp; coin_A)$
2:     $A \xleftarrow{\$} S_A$
3:     $sk_A \xleftarrow{\$} S_{sk}$
4:     $d \xleftarrow{\$} s_B$
5:     randomness comes from $coin_A$
6:     $B \leftarrow A \times sk_A + d$
7:     $pk_A \leftarrow (A, B)$
8:     return $(sk_A, pk_A)$
1: function $Enc(pp, pk_A, pt; coin_B)$
2:     $parse\ pk_A = (A, B)$
3:     $t \xleftarrow{\$} S_t, e \xleftarrow{\$} S_U, f \xleftarrow{\$} S_V$
4:     randomness comes from $coin_B$
5:     $\bar{U} \leftarrow t \times A + e$
6:     $\bar{V} \leftarrow t \times B + f + encode(pt)$
7:     $U \leftarrow Compress(\bar{U})$
8:     $V \leftarrow Compress(\bar{V})$
9:     $K \leftarrow H(pt \| ct = (U, V))$
10:    return $K$
1: function $Dec(pp, sk_A, ct)$
2:     $Parse\ ct = (U, V)$
3:     $\bar{U} \leftarrow Decompress(U)$
4:     $\bar{V} \leftarrow Decompress(V)$
5:     $W \leftarrow \bar{V} - \bar{U} \times sk_A$
6:     $pt' \leftarrow decode(W)$
7:     $K' \leftarrow H(pt' \| ct = (U, V))$
8:     return $K'$
```

-For t, d, f, e, sk, such sparse elements are chosen to be sampled from discrete Gaussian distribution or central binomial distribution $\boldsymbol{B}_\eta$ whose sample is generated by $\Sigma_{i=1}^{\eta}(a_i - b_i)$, where $a_i, b_i \leftarrow \{0, 1\}$ and mutually independent. A sample is chosen according to $\boldsymbol{B}_\eta$ means every component is chosen randomly from $\boldsymbol{B}_\eta$.

-the *encode* : $\mathcal{M} \to S_V$, *decode* : $S_V \to \mathcal{M}$ is not necessary but usually employed. The encode is an injective function. A typical code is $D-v$ lattice code. Message bits are encoded by multiplication to $L = (q-1)/2$ and represented v times in $Y = encode(pt)$. NewHope [22] selects $v = 2$, thus $Y_i = Y_{i+256} = pt_i \cdot (q-1)/2$. The decoding process of $Y = encode(pt)$ is finding the value b that minimizes $|Y_i - b \cdot \frac{q-1}{2}| + |Y_{i+256} - b \cdot \frac{q-1}{2}|$.

-The Compress/Decompress is usually used to decrease the communication cost. Typically, a ciphertext $\bar{V}$ is replaced by $V = Compress(\bar{V}, p) = \lceil p/q \cdot$

$\bar{V}\rfloor \ mod\ p$ and the decompress operates in an opposite way $\bar{V} = Compress(V, p) = \lceil q/p \cdot V\rfloor$.

Almost all NIST candidate lattice-based CPA-secure KEM are designed as Alg. 1. We give two examples below: NewHope [22], Crystals-Kyber [3].

Example 1 Kyber defines $S_{sk} = \mathcal{R}_q^k$, $S_B = \mathcal{R}_q^k$, $S_U = \mathcal{R}_q^k$, $S_t = \mathcal{R}_q^k$, $S_V = \mathcal{R}_q$, $S_A = \mathcal{R}_q^{k\times k}$. Kyber does not use *encode/decode* algorithm. Elements in $\mathcal{R}_q$ are considered as polynomials in variable X modulo $X^n + 1$. Elements in $\mathcal{R}_q^k$ are considered as vector with components in $\mathcal{R}_q$. Elements in $\mathcal{R}_q^{k\times k}$ are considered as matrix with components in $\mathcal{R}_q$. In Kyber512-KEM, e, f, d is sampled sparsely from $\boldsymbol{B}_3$. In Kyber768-KEM abd Kyber1024-KEM, t, d, f, e, sk are sampled from $\boldsymbol{B}_2$. Other parameters for Kyber is $q = 3329, n = 256$. $k = 2/3/4$ for Kyber512/Kyber 768/Kyber 1024.

Example 2 NewHope-CPA-PKE defines $S_{sk} = S_A = S_B = S_t = S_U = S_V = \mathcal{R}_q$. Elements in R_q are considered as polynomials in variable X modulo $X^n + 1$. For t, d, f, e, sk, sparse elements are sampled from centered binomial distribution $\boldsymbol{B}_8$. For NewHope512/NewHope1024, the parameters are $n = 512, n = 1024$ and $q = 12289$. The *encode/decode* algorithm are described as above.

3.2 Model of Plaintext-Checking Attack

In a plaintext-checking attack, the adversary interacts with plaintext checking oracle $\mathcal{O}$, which works as shown in Algorithm 2. $\mathcal{O}$ is a plaintext checking oracle which receives ct and pt, returning one bit showing if ct decrypts to pt. IND-CPA secure public key encryption/key encapsulation mechanisms are vulnerable to plaintext-checking attack.

The plaintext-checking oracle exists in many cases. In the client-server protocol where the ciphertext is the encryption of some symmetric key k. The adversary can construct faulty ciphertexts that may or may not decode to k and deliver them to the server. Then the adversary can see if secure messaging works and hence simulates a plaintext-checking oracle $\mathcal{O}$. IND-CCA secure KEM may also suffer a plaintext-checking attack since the adversary can create oracle $\mathcal{O}$ by a test-based template approach as given in [11].

Algorithm 2 Plaintext-Checking Attack

```
1: $sk'_A \leftarrow \mathcal{A}^{\mathcal{O}}(pk_A)$
2: if $sk'_A = sk_A$ then
3:     return 1
4: else
5:     return 0
```

```
1: ORACLE $\mathcal{O}(ct = (U, V), K)$
2:     $K' \leftarrow KEM.Dec(ct)$
3:     if $K = K'$ then
4:         return 1
5:     else
6:         return 0
```

3.3 Secret Leakage Model in Plaintext-Checking Attack

Since many lattice-based IND-CPA secure KEMs use the same meta-structure, they may have similar plaintext-checking attack procedures. Suppose Alice reuses her public key $pk_A = (A, B)$. As described in Algorithm 2, the adversary $\mathcal{A}$ crafts different plaintext and ciphertext ct, pt to recover Alice's secret key sk_A with as fewer oracle access as possible. Each coefficient of sk_A is sampled independently from $\mathcal{S} \subset S_{sk}$.

When the adversary $\mathcal{A}$ tries to recover the i-th coefficient of sk_A, each oracle call leaks information about $\mathcal{S}$. Without loss of generality, let

$$\mathcal{S} = \{\boldsymbol{S}_0, \boldsymbol{S}_1, ..., \boldsymbol{S}_{n-1}\}$$

be the set of all possible values of the original sparse secret distribution. Let P_j be the probability that $sk_A[i] = \boldsymbol{S}_j$ where $sk_A[i]$ is generated from the distribution $\mathcal{S}$, that is, $P_j = Pr[sk_A[i] = \boldsymbol{S}_j | sk_A[i] \leftarrow \mathcal{S}]$ for $j = 0, 1, ..., n-1$. Denote the new secret distribution after the oracle query as $\mathcal{S}'$ after querying plaintext checking oracle $\mathcal{O}$. When the adversary gets a returned value from the Oracle, he can narrow the range of $sk_A[i]$ from $\mathcal{S}$ to $\mathcal{S}'$ until the exact value of $sk_A[i]$ is determined.

The change of secret distribution is shown in Fig. 1. As described in Section 3.1, block 1 (in a dashed rectangle) represents the multiplication of Abelian groups S_A, S_B, S_t, S_U, S_V (the yellow blocks) before the adversary $\mathcal{A}$ queries the PC Oracle $\mathcal{O}$. The blue block in block 1 represents the secret distribution $\mathcal{S} \subset S_{sk}$ before the adversary queries the PC Oracle $\mathcal{O}$. The green block in block 2 represents the new secret distribution $\mathcal{S}' \subset S_{sk}$ after $\mathcal{A}$ queries the PC Oracle $\mathcal{O}$. Other Abelian groups remain unchanged.

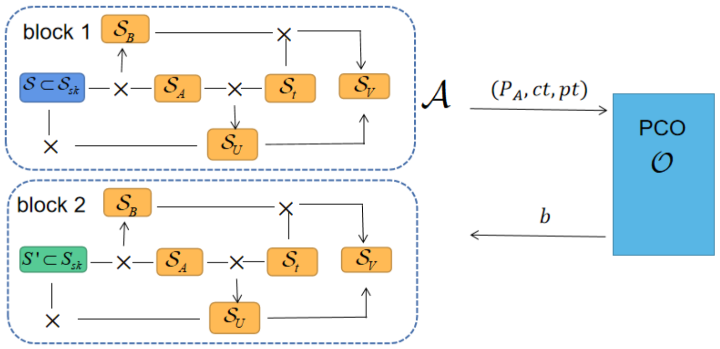

Fig. 1. Oracle query in Plaintext-Checking Attack and the change of secret distribution.

In plaintext-checking attack, the adversary $\mathcal{A}$ tries to recover the reused secret by accessing oracle $\mathcal{O}$ as few as possible. In other words, each oracle query decreases the Shannon entropy of reused secret sk_A. $\mathcal{A}$ tries to reduce the entropy of $\mathcal{S}$ as much as possible. Intrinsically, for each coefficient of Alice's reused secret key sk_A, the querying process can be described as a function $\boldsymbol{f}$ of reused secret key sk_A, secret distribution $\mathcal{S}$, ciphertext $ct \in S_U \times S_V, pt \in \mathcal{M}$. Formally, we can define the information leakage in plaintext-checking attack as:

Definition 5 (Plaintext-Checking Hint). *A plaintext-checking hint on the reused secret sk_A is the crafted plaintext pt, ciphertext ct, such that*

$$\boldsymbol{f}(sk_A, ct, pt) = b \in \{0, 1\}$$

Let v be a unit vector with $v[i] = 1$. The expression of plaintext-checking hint $\boldsymbol{f}(sk_A, ct, pt)$ can be simplified as:

$$\boldsymbol{f}(\langle sk_A, v \rangle) = b \in \{0, 1\}$$

Since the adversary tries to recover the i-th coefficient of sk_A.

Most of the prior works consider the Oracle access times for recovering the full reused key. Since the adversary may be prohibited from gathering sufficient side-channel information to build a plaintext-checking oracle, the adversary $\mathcal{A}$ does not have sufficient access to a plaintext-checking oracle to completely recover the reused secret. For example, PC Oracle constructed by reusing KEM's public key cannot be accessed when users stop reusing the public key. Thus one may be interested in a more precise analysis of $\boldsymbol{f}(sk_A, ct, pt)$ to learn security loss after certain times of Oracle access. To investigate security loss after limited times of Oracle queries, one possible way is to express $\boldsymbol{f}(sk_A, ct, pt)$ in the form that can be integrated into the lattice.

4 Reducing PC-Hint to Perfect Inner-Product Hint

This section presents how to reduce PC-Hint to a perfect inner-product hint. First, we describe a practical plaintext-checking attack that reaches the theoretical lower bound of oracle queries in section 4.1. We then analyze the message leakage in each PC Oracle query, which is described in the form of plaintext-checking hint in section 4.2. Finally, we present a method for integrating plaintext-checking hints into the lattice in section 4.3. We do this by transforming plaintext-checking hints into perfect inner-product hints and integrating perfect inner-product hints into the lattice. Additionally, we analyze the changes in lattice volume and dimension, which directly affect the bit security of KEMs.

4.1 Practical Plaintext-Checking Attack With Theoretical Lower Bound

We adopt the plaintext-checking attack given in [25]. They get their lower bounds for all lattice-based NIST KEM candidates by building the optimal Binary Recovery Tree (BRT), and they show that the calculation of these bounds becomes essentially the computation of a certain Shannon entropy, which means that on average one cannot find a better attack with fewer queries than their results in the full key recovery.

The lower bound of oracle access for recovering a single coefficient of reused secret sk_A has been analyzed in Theorem 1 in [25]. Let $min\ E(\mathcal{S})$ represent the lower bound for the minimum average number of queries. Moreover, let $H(\mathcal{S})$ represent the Shannon entropy for $\mathcal{S}$. Then, we have $H(\mathcal{S}) \leq min\ E(\mathcal{S}) < H(\mathcal{S}) + 1$. In the following section, we give a brief explanation of their attack.

Take plaintext-checking attack to IND-CPA secure Kyber512 as an example. The adversary selects proper ciphertext $ct = (U, V)$ as inputs to $\mathcal{O}$. Then, the adversary is able to recover Alice's reused secret sk_A from the oracle response $\hat{\boldsymbol{s}}$. The recovery of each coefficient is mutually independent. We give the approach to recover the first coefficient block $sk_A[0]$ of $\boldsymbol{sk}$, other coefficient blocks can be recovered similarly.

The attacker selects plaintext $pt = (1, 0, ..., 0)$ and $ct = (U, V)$, where $\bar{U} = (\lceil \frac{q}{16} \rfloor, 0, ..., 0)$, $U = Compress(\bar{U}, 2^{d_U})$ and $V = (h, 0, ..., 0)$. $d_U = 10, d_V = 4$ is the parameter selected by Kyber512. Then the attack query the plaintext-checking oracle $\mathcal{O}$ with ct. The oracle $\mathcal{O}$ calculates $\bar{U} = Decompress(U, 2^{d_U}), \bar{V} = Decompress(V, 2^{d_V}) = (\lceil \frac{q}{16} h \rfloor, 0, ..., 0)$.

Thus, the adversary constructs a relationship between $pt'[0]$ and $sk_A[0]$ after decryption as $pt'[0] = Compress((\bar{V} - sk_A^T \cdot \bar{U})[0], 2) = \lceil \frac{2}{q}(\bar{V}[0] - (sk_A^T \cdot \bar{U})[0]) \rfloor\ mod\ 2$.

Since $\bar{V}[0] = \lceil \frac{q}{16} h \rfloor$ and $(sk_A^T \cdot \bar{U})[0] = sk_A^T[0]\bar{U}[0] = sk_A[0]\lceil \frac{q}{16} \rfloor$, it holds that $pt'[0] = \lceil \frac{2}{q}([\frac{q}{16} h] - sk_A^T[0][\frac{q}{16}]) \rfloor\ mod\ 2$, where h is a parameter chosen by the attacker. Let $h = 4$, if $sk_A[0] \in [0, 3], pt'[0] = 0$, then the oracle $\mathcal{O}$ will output 0. Otherwise, $sk_A[0] \in [-3, -1], pt'[0] = 1$, the oracle $\mathcal{O}$ will output 1.

The attacker could adaptively choose h to recover $sk_A[0]$ based on the sequence $\hat{\boldsymbol{s}}$ from oracle $\mathcal{O}$. If the attacker uses well-selected h, he could recover $sk_A[0]$ with as few queries as possible. With the help of the optimal binary recovery tree, the adversary divides the range of the coefficient block in half each time and tries to recover $\boldsymbol{S_i}$ with the biggest probability as soon as possible. We list the selection of h in Table 2. In such a plaintext-checking attack, each query divides the possible range of $sk_A[0]$ into (nearly) half. [25] gives the selection of h and the corresponding changes of states in section 4.1.

Table 2. The choice of h and the States for Kyber512

	State1	**State2**	**State3**	**State4**	**State5**	**State6**
h	**4**	**3**	**9**	**12**	**13**	**7**
$\mathcal{O} \to 0$	**State4**	$sk_A[0] = -1$	$sk_A[0] = -3$	$sk_A[0] = 0$	$sk_A[0] = 1$	$sk_A[0] = 3$
$\mathcal{O} \to 1$	**State2**	**State3**	$sk_A[0] = -2$	**State5**	**State6**	$sk_A[0] = 2$

According to Theorem [25], The lower bound for Kyber512, Kyber768, and Kyber1024, in theory, is 1216, 1632, 2176. The expectation of queries needed to recover a single coefficient in sk_A is $\frac{5}{16} \times 2 + \frac{15}{64} \times (3+2) + \frac{3}{32} \times (4+3) + \frac{1}{32} \times 3 = 2.56$. The average number of queries needed in a plaintext-checking attack for Kyber512, Kyber768, and Kyber1024, in theory, is 1312, 1774, and 2365. The gap is less than 9%. Besides, Qin et al. also did an experiment to verify their theory. The experiment result shows that the number of queries is 1311, 1777, 2368 separately.

4.2 Message Leakage in Each Query

Suppose Alice reuses her public key, the corresponding secret key is sk_A. The adversary tries to recover the first coefficient of sk_A. The analysis is similar to other coefficients.

In Section 3.3, we give a plaintext-checking hint $\boldsymbol{f}(sk_A, ct, pt)$ to describe the change of secret distribution after each oracle query. For Kyber512, $\mathcal{A}$ sets $ct = (U, V)$. If $sk_A[0] \in [0, 3]$, set $h = 4$, $pt'[0] = \lceil \frac{2}{q}([\frac{q}{16}h] - sk_A^T[0][\frac{q}{16}]) \rfloor \ mod \ 2 = 0$, then $\mathcal{O} \to 0$. Otherwise $pt'[0] = 1$, $\mathcal{O} \to 1$. The plaintext-checking hint can be described as:

$$\boldsymbol{f}(sk_A, ct = (U, V), pt) = \lceil \frac{2}{q}(\bar{V}[0] - (sk_A^T \cdot \bar{U})[0]) \rfloor \ mod \ 2 \tag{3}$$

Let v be a unit vector with $v[0] = 1$. Since the adversary tries to recover a certain coefficient in each oracle query, it is very natural to express $\boldsymbol{f}(sk_A, ct, pt)$ as $\boldsymbol{f}(\langle sk_A, v\rangle)$. Specifically, we have:

-$h = 4$, after the first query, plaintext-checking hint $PCHint_1$: $\boldsymbol{f}(\langle sk_A, v\rangle) = \lceil \frac{2}{q}([\frac{q}{16} \cdot 4] - sk_A^T[0][\frac{q}{16}]) \rfloor \ mod \ 2 = 0$, and $\langle sk_A, v\rangle \in [0, 3]$.

-$h = 12$, after the second query, plaintext-checking hint $PCHint_2$: $\boldsymbol{f}(\langle sk_A, v\rangle) = \lceil \frac{2}{q}([\frac{q}{16} \cdot 12] - sk_A^T[0][\frac{q}{16}]) \rfloor \ mod \ 2 = 1$, and $\langle sk_A, v\rangle \in [1, 3]$.

-$h = 13$, after the third query, plaintext-checking hint $PCHint_3$: $\boldsymbol{f}(\langle sk_A, v\rangle) = \lceil \frac{2}{q}([\frac{q}{16} \cdot 13] - sk_A^T[0][\frac{q}{16}]) \rfloor \ mod \ 2 = 1$, and $\langle sk_A, v\rangle \in [2, 3]$.

-$h = 7$, after the fourth query, plaintext-checking hint $PCHint_4$: $\boldsymbol{f}(\langle sk_A, v\rangle) = \lceil \frac{2}{q}([\frac{q}{16} \cdot 7] - sk_A^T[0][\frac{q}{16}]) \rfloor \ mod \ 2 = 0$, and $\langle sk_A, v\rangle = 3$.

Now the adversary collect four plaintext-checking hints $PCHint_1$, $PCHint_2$, $PCHint_3$, $PCHint_4$. Then the adversary can transform these hints into "perfect hint" as described in [7]:

$$\langle sk_A, v\rangle = sk_A[0]$$

Let $\mathcal{S} = \{\boldsymbol{S}_0, \boldsymbol{S}_1, ..., \boldsymbol{S}_{n-1}\}$ be the set of all possible values of the original sparse secret distribution. Denote by H_i the number of plaintext-checking hints $\mathcal{A}$ needs to determine the coefficient block when it is exactly $\boldsymbol{S}_i$, which is actually the oracle access need to determine the coefficient block as analyzed at the beginning of Section 4. Let $E(\#PCHint)$ be the average number we needed to transform a plaintext-checking hint into a perfect hint. According to the analysis above, we have:

$$E(\#PCHint) = \Sigma_{i=0}^{n-1} P_i H_i$$

Intrinsically, the average number we needed to transform plaintext-checking hints into a perfect hint is the average number of oracle queries $\mathcal{A}$ needed to recover a single coefficient block. Thus the lower bound of $E(\#PCHint)$ can be derived from [25]. We list $E(\#PCHint)$ for all lattice-based NIST KEMs (both IND-CPA/IND-CCA) in Table 3.

Table 3. $E(\#PCHint)$ against lattice-based NIST KEMs

Schemes	$\mathcal{S}$	$E(\#PCHint)$	**Schemes**	$\mathcal{S}$	$E(\#PCHint)$
Kyber512	**[-3,3]**	**2.77**	**Frodo640**	**[-12,12]**	**3.59**
Kyber768	**[-2,2]**	**2.31**	**Frodo976**	**[-10,10]**	**3.34**
Kyber1024	**[-2,2]**	**2.31**	**Frodo1344**	**[-6,6]**	**2.73**
LightSaber	**[-5,5]**	**2.88**	**NewHope512**	**[-8,8]**	**3.24**
Saber	**[-4,4]**	**2.73**	**NewHope1024**	**[-8,8]**	**3.11**
FireSaber	**[-3,3]**	**2.56**	**-**	**-**	**-**

In the following parts, we describe how to predict security loss after collecting several PC hints and transformed these PC hints into a perfect hint.

4.3 Integrating Plaintext-Checking Hints into Lattice

The intuition behind estimating security loss (under primal attack) is to estimate the hardness of the underlying LWE problem (as defined in Definition 1) after integrating plaintext-checking hints. The adversary $\mathcal{A}$ collects plain LWE samples as shown in line 5, 6 in function *Enc* of Algorithm 1. Then $\mathcal{A}$ transforms the LWE problem to DBDD problem(Definition 3) and constructs a lattice basis. Then $\mathcal{A}$ integrates the plaintext-checking hints into the DBDD problem. Finally, $\mathcal{A}$ transforms the DBDD problem to uSVP problem(Definition 2). The uSVP problem can be solved by lattice reduction algorithm.

The solution of the LWE problem is (e, sk_A). It can be extended to a short vector $(e, sk_A, 1)$, which is an short vector of the lattice:

$$\Lambda = \{(\boldsymbol{x}, \boldsymbol{y}, w) \in \mathbb{Z}^{n+m+1} | \boldsymbol{x} + \boldsymbol{y}\boldsymbol{A^T} - \boldsymbol{b}w\} = 0 \; mod \; q$$

which is of full rank in $\mathbb{R}^d$ and has volume q^m. The row vectors of

$$\begin{bmatrix} q\boldsymbol{I_m} & 0 & 0 \\ \boldsymbol{A^T} & -\boldsymbol{I_n} & 0 \\ \boldsymbol{b} & 0 & 1 \end{bmatrix} \tag{4}$$

constitute a basis of Λ. For a single coefficient block $sk_A[b]$, $P_i = Pr(sk_A[b] = \boldsymbol{S_i} | sk_A \leftarrow \mathcal{S})$ for $i = 0, 1, ..., n-1$. We denote the average and variance of the

LWE original secret distribution $\mathcal{S}$ as μ and σ^2. Such a LWE instance can be converted to a $DBDD_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ instance with $\boldsymbol{\mu} = [\mu, ..., \mu, 1]$, $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma^2 \boldsymbol{I}_{m+n} & 0 \\ 0 & 0 \end{bmatrix}$.

To integrate plaintext-checking hints into $DBDD$ problem, the adversary $\mathcal{A}$ collect several plaintext-checking hints until they can be transformed to a perfect hint $\langle sk_A, v\rangle = sk_A[0]$ as shown in Section 4.2. Suppose $\mathcal{A}$ recovers the first coefficient of sk_A. v can be extended to $\bar{\boldsymbol{v}} := (\mathbf{0}; v; -l)$, where $\mathbf{0}$ is an all-zero vector of dimension m, $v = (1, 0, \cdots, 0)$ of dimension n, $l = sk_A[0]$. The adversary $\mathcal{A}$ can integrate $\bar{\boldsymbol{v}}$ by modifying $DBDD_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ to $DBDD_{\Lambda',\boldsymbol{\mu'},\boldsymbol{\Sigma'}}$.

Integrating $\bar{\boldsymbol{v}}$ into $DBDD_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ means finding an intersection between Λ and an hyperplane orthogonal to $\bar{\boldsymbol{v}}$. We denote the new lattice as Λ'. Intuitively, Λ' has lower dimension than Λ, meaning that solving $DBDD_{\Lambda',\boldsymbol{\mu'},\boldsymbol{\Sigma'}}$ is easier than $DBDD_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$.

The new mean $\boldsymbol{\mu'}$ and new covariance $\boldsymbol{\Sigma'}$ can be derived according to the equation given in equations (12) and (13) in [7]. Specifically, we have $\boldsymbol{\Sigma'} = \boldsymbol{\Sigma} - \frac{(\bar{\boldsymbol{v}}\boldsymbol{\Sigma})^T \bar{\boldsymbol{v}}\boldsymbol{\Sigma}}{\bar{\boldsymbol{v}}\boldsymbol{\Sigma}\bar{\boldsymbol{v}}^T}$, and $\boldsymbol{\mu'} = \boldsymbol{\mu} - \frac{\langle \bar{\boldsymbol{v}}, \boldsymbol{\mu}\rangle}{\bar{\boldsymbol{v}}\boldsymbol{\Sigma}\bar{\boldsymbol{v}}^T} \bar{\boldsymbol{v}}\boldsymbol{\Sigma}$. Since $\bar{\boldsymbol{v}}$ is an all-zero vector with dimension $m+n+1$ except that the $m+1$-th coefficient is 1 and the $m+n+1$-th coefficient is $-sk_A[0]$. Thus we have $\bar{\boldsymbol{v}}\boldsymbol{\Sigma}$ is an all-zero vector except that the $m+1$-th coefficient is σ^2.

Thus we have $\boldsymbol{\Sigma'} = \boldsymbol{\Sigma} - \frac{1}{\boldsymbol{\sigma^2}}\boldsymbol{M}$, where $\boldsymbol{M}$ is a $m+n+1$-dimension diagonal matrix with $\boldsymbol{M}_{m+1,m+1} = \sigma^4, \boldsymbol{M}_{m+n+1,m+n+1} = 0$ and all other diagonal elements 0. Thus $\boldsymbol{\Sigma'}$ is a $m+n+1$-dimension diagonal matrix with $\boldsymbol{M}_{m+1,m+1} = 0, \boldsymbol{M}_{m+n+1,m+n+1} = 0$ and all other diagonal elements σ^2. $\boldsymbol{\mu'} = [\mu, ..., \mu, 1] - (\mu - sk_A[0])\frac{1}{\sigma^2}\bar{\boldsymbol{v}}\boldsymbol{\Sigma}$. Thus $\boldsymbol{\mu'}$ is an all-μ vector except that $\boldsymbol{\mu'}_{m+1} = sk_A[0]$, $\boldsymbol{\mu'}_{m+n+1} = 1$. The volume of Λ' is analyzed in Theorem 1.

Theorem 1. *Given the LWE sample* $(\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}, \boldsymbol{b} = sk_A^T\boldsymbol{A} + \boldsymbol{e} \in \mathbb{Z}_q^m)$. *Suppose the adversary* $\mathcal{A}$ *collects* $\#PCHint$ *plaintext-checking hints* $f_1, ..., f_{\#PCHint}$ *when recovering* $sk_A[i]$. *The adversary* $\mathcal{A}$ *can transform* $f_1, ..., f_{\#PCHint}$ *into a perfect hint* $\bar{\boldsymbol{v}} := (\mathbf{0}; v; -l)$, *where* $\mathbf{0}$ *is an all-zero vector of dimension* m, v *is a unit vector with* $v[i] = 1$ *and dimension* n, $l = sk_A[i]$. *Including hint* $\bar{\boldsymbol{v}}$ *modifies* $DBDD_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ *to* $DBDD_{\Lambda',\boldsymbol{\mu'},\boldsymbol{\Sigma'}}$ *with dimension* $dim(\Lambda')$ *and volume* $Vol(\Lambda')$:

$$\begin{aligned} \dim(\Lambda') &= \dim(\Lambda) - 1 \\ Vol(\Lambda') &= \mathrm{Vol}(\Lambda) \cdot \sqrt{1 + sk_A[i]}^2 \cdot det(\boldsymbol{\Pi}_\Lambda) \end{aligned} \tag{5}$$

When $\bar{\boldsymbol{v}}$ *is a primitive vector, we have*

$$\mathrm{Vol}(\Lambda') = \mathrm{Vol}(\Lambda)\sqrt{1 + sk_A[i]^2} \tag{6}$$

Proof. When $\bar{\boldsymbol{v}}$ is a primitive vector($\bar{\boldsymbol{v}}$ can be extended to a basis of Λ), the volume of the lattice after integrating hint $\bar{\boldsymbol{v}}$ is $Vol(\Lambda) = \|\bar{\boldsymbol{v}}\| \cdot \mathrm{Vol}(\Lambda) = \mathrm{Vol}(\Lambda) \cdot \sqrt{1 + sk_A[i]^2}$(see Lemma 12 of [7]).

When $\bar{\boldsymbol{v}}$ is not in the span of Λ, we can also apply orthogonal projection $\bar{\boldsymbol{v}}' = \bar{\boldsymbol{v}} \cdot \boldsymbol{\Pi}_\Lambda$ of $\bar{\boldsymbol{v}}$ onto Λ. Replacing $\bar{\boldsymbol{v}}$ by $\bar{\boldsymbol{v}}'$ is still valid. The orthogonal projection matrix is $\boldsymbol{\Pi}_{\boldsymbol{\Lambda}} = \boldsymbol{\Pi}_{\boldsymbol{\Sigma'}} = \sqrt{\boldsymbol{\Sigma'}}^{\sim} \cdot \boldsymbol{\Sigma'} \cdot \sqrt{\boldsymbol{\Sigma'}}^{\sim T}$, where $\boldsymbol{\Sigma'} = \boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}$ is the covariance matrix after homogenization, $\sqrt{\boldsymbol{\Sigma'}}^{\sim}$ is the restricted inverse of $\sqrt{\boldsymbol{\Sigma'}}$(see definition 3 of [7]).

Thus we have $\mathrm{Vol}\,(\Lambda') = \mathrm{Vol}(\Lambda) \cdot \sqrt{1 + sk_A[i]^2} \cdot det(\boldsymbol{\Pi}_\Lambda))$, where $\boldsymbol{\Pi}_\Lambda$ is the orthogonal projection onto Λ .

It is predicted that the $BKZ - \beta'$ can solve a $uSVP_{\Lambda'}$ after $E(\#PCHint)$ queries s.t. $\sqrt{\beta'} \leq \delta_{\beta'}^{2\beta'-dim(\Lambda')-1} \cdot Vol\,(\Lambda')^{1/dim\left(\Lambda'\right)}$, where $dim(\Lambda'), Vol(\Lambda')$ are as described in equation (5, 6). The expectation of $\#PCHint$ ($E(\#PCHint)$) and the transformation from plaintext-checking hints to perfect hints has been described in Section 4.2.

5 Experiment Results

5.1 Kyber

Fig. 2 gives the concrete relationship between the number of queries and bit-security for all parameter sets of Kyber in the NIST third-round submission. In Table 4, we present the number of queries needed for Kyber512/ Kyber768/ Kyber1024 when the bit security of the underlying LWE reaches 128, 64, 48, 32, 24, 16 under primal and dual attacks.

Taking Kyber512 as an example, the adversary $\mathcal{A}$ can query PC Oracle $\mathcal{O}$ for only 867 times instead of 1312 times. The classical bit security of the LWE problem is decreased to 32. The adversary may stop querying the PC Oracle and solve the remaining part of the secret key using the lattice reduction algorithm.

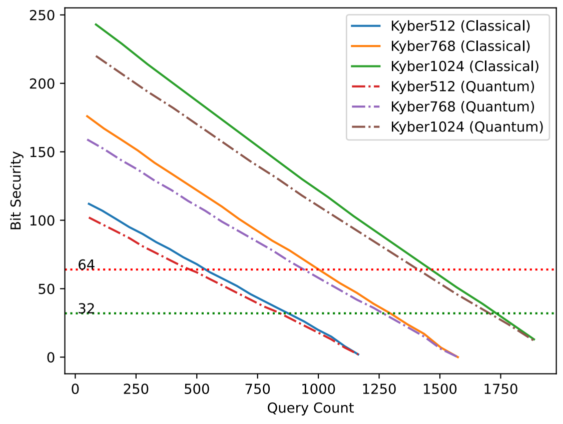

Fig. 2. Relationship between query and security under primal attack for Kyber.

For our experiment, we make use of the LWE estimator from [7]. Estimating the hardness needs the dimension of the lattice Λ and its volume only. According to Theorem 1, for Kyber512/Kyber768/ Kyber1024, every 2.77/2.31/2.31 queries reduces the dimension of the lattice by 1. After integrating short vectors into the lattice, we get the concrete dimension of the lattice Λ, which tells us the security of current LWE problem after certain times of queries. The number in parentheses is the Oracle queries needed when classical/quantum bit security is 100 (The original classical bit security of Kyber is 118).

Table 4. Classical&Quantum Query-Security For Kyber 512/768/1024.

Bit Security	Kyber512 (classical/quantum)		Kyber768 (classical/quantum)		Kyber1024 (classical/quantum)	
-	Primal	Dual	Primal	Dual	Primal	Dual
128(100)	(80)/(65)	(150)/(129)	444/333	562/543	950/848	1274/1205
64	533/464	657/624	998/938	1112/1096	1459/1404	1732/1673
48	699/646	761/733	1144/1098	1206/1176	1593/1550	1805/1773
32	867/831	865/838	1292/1259	1320/1302	1728/1699	1915/1893
24	953/925	922/906	1366/1343	1396/1361	1796/1775	1973/1959
16	1033/1016	981/962	1437/1423	1481/1476	1862/1849	2095/2029

5.2 Saber

Fig. 3 gives the relationship between the number of queries and security for all parameter sets of Saber in NIST third round submission. We list the number of queries needed for LightSaber/Saber/FireSaber when the bit security of the underlying LWE reaches 128, 64, 48, 32, 24, 16 in Table 5 under primal/dual attack.

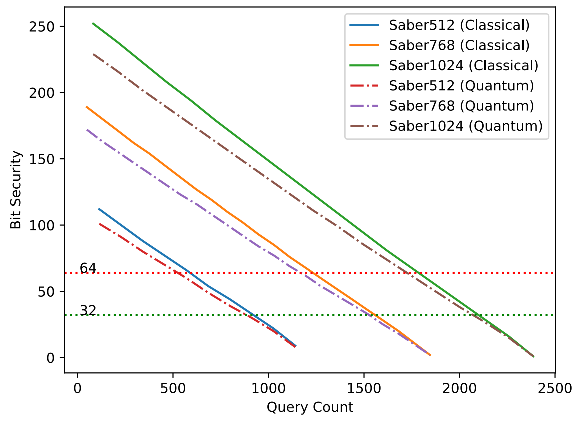

Fig. 3. Relationship between query and security under primal attack for Saber.

According to Theorem 1, for LightSaber/ Saber/ FireSaber, every 2.88/ 2.73/ 2.56 queries reduce the dimension of the lattice by 1. After integrating short vectors into the lattice, we get the concrete dimension of the lattice Λ, which tells us the security of current LWE problem after certain times of queries. The number in parentheses is the Oracle queries needed when classical/quantum bit security is 100 (The original classical bit security of LightSaber is 118).

Table 5. Classical&Quantum Query-Security For LightSaber/Saber/FireSaber.

Bit Security	LightSaber (classical/quantum)		Saber (classical/quantum)		FireSaber (classical/quantum)	
-	Primal	Dual	Primal	Dual	Primal	Dual
128(100)	(228)/(124)	(407)/(352)	612/487	832/817	1181/1063	1395/1364
64	631/553	844/799	1230/1162	1446/1415	1782/1718	1963/1929
48	839/772	933/902	1390/1339	1550/1517	1941/1890	2067/2029
32	1075/1023	1029/1003	1554/1521	1656/1625	2100/2066	2179/2165
24	1204/1164	1096/1066	1638/1611	1714/1699	2179/2153	2257/2223
16	1340/1311	1155/1138	1717/1701	1774/1759	2258/2243	2326/2289

5.3 Frodo

Fig. 4 gives the relationship between the number of queries and security for all parameter sets of Frodo in NIST second round submission. We list the number of queries needed for Frodo640/ Frodo976/ Frodo1344 when the bit security of the underlying LWE reaches 128, 64, 48, 32, 24, 16 in Table 6 under primal/dual attack.

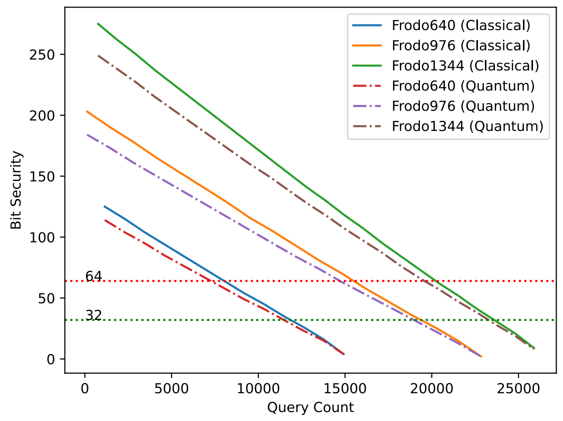

Fig. 4. Relationship between query and security under primal attack for Frodo.

According to Theorem 1, for Frodo640/ Frodo976/ Frodo1344, every 3.59/ 3.34/ 2.73 queries reduce the dimension of the lattice by 1. After integrating short vectors into the lattice, we get the concrete dimension of the lattice Λ, which tells us the security of current LWE problem after certain times of queries. The number in parentheses is the Oracle queries needed when classical/quantum bit security is 100 (The original quantum bit security of Frodo640 is 124).

Table 6. Classical&Quantum Query-Security For Frodo 640/976/1344

Bit Security	Frodo640 (classical/quantum)		Frodo976 (classical/quantum)		Frodo1344 (classical/quantum)	
-	Primal	Dual	Primal	Dual	Primal	Dual
128(100)	833/(2755)	2943/(4861)	8177/6734	12546/11986	14006/12761	17859/16758
64	8005/7230	10508/9961	15445/14670	19863/19384	20234/19556	23284/23009
48	9899/9296	12001/11474	17368/16754	21041/20792	21872/21370	24257/23812
32	11821/11419	13572/13230	19346/18918	22368/21994	23577/23205	25217/25124
24	12796/12509	14577/14235	20334/20014	23154/22773	24429/24145	25841/25581
16	13772/13571	15475/15162	21296/21109	23609/23527	25259/25084	26304/26174

5.4 NewHope

Fig. 5 gives the relationship between the number of queries and security for all parameter sets of NewHope in NIST second round submission. We list the number of queries needed for NewHope512/ NewHope1024 when the bit security of the underlying LWE reaches 128, 64, 48, 32, 24, 16 in Table 7 under primal/dual attack.

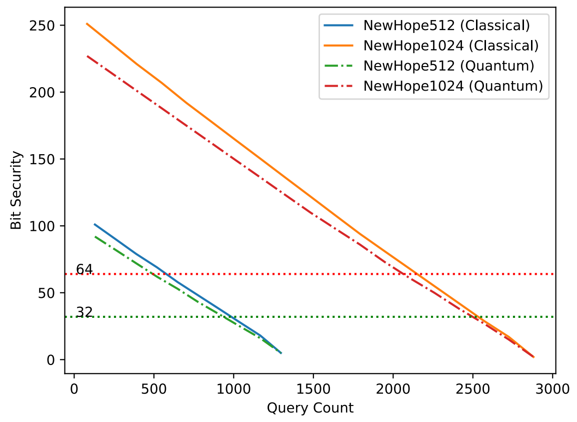

Fig. 5. Relationship between query and security under primal attack for NewHope.

According to Theorem 1, for NewHope512/ NewHope1024, every 3.24/ 3.11 queries reduces the dimension of the lattice by 1. After integrating short vectors into the lattice, we get the concrete dimension of the lattice Λ and its volume, which tells us the security of current LWE problem after certain times of queries. The number in parentheses is the Oracle queries needed when classical/quantum bit security is 100 (The original classical bit security of NewHope512 is 112).

Table 7. Classical&Quantum Query-Security For NewHope 512/1024

Bit Security	NewHope512 (classical/quantum)		NewHope1024 (classical/quantum)	
-	Primal	Dual	Primal	Dual
128(100)	(137)/(20)	(528)/(461)	1406/1260	1820/1756
64	571/490	915/866	2140/2062	2475/2438
48	772/710	1054/1011	2333/2274	2594/2555
32	979/934	1173/1148	2532/2488	2714/2678
24	1083/1050	1231/1224	2632/2600	2771/2738
16	1183/1164	1325/1309	2728/2709	2864/2837

6 Conclusions & Discussions

In this paper, we explicitly build the relationship between the number of Oracle queries and the security loss of the reused secrets for all NIST second-round lattice-based KEMs. Our analysis can be divided into three steps. First, we model the information leakage in the PC-oracle by PC-hint and give a generic transformation from PC-hints to the perfect inner-product hint, which allows the adversary to integrate PC-hints progressively. Then, we give a concrete relationship between the PC Oracle query number and the bit security of the lattice-based KEM under PCA. Our bit security analysis is inspired by the security analysis for LWE with the perfect inner-product hint given in [7], Our proposed method is applicable to all CCA-secure NIST candidate lattice-based KEMs.

We applied our methods to NIST-PQC lattice-based KEMs, we get the bit-security loss of the lattice-based KEM under PCA. Take Kyber768 (original 182-bit-security) as an example, the bit security of Kyber768 is reduced to 128 after 444 PC-oracle queries and reduced to 64 after 998 PC-oracle queries, while in Qin et al. [25] 1774 queries are required to recover the whole secret key. Our analysis also demonstrates the possibility of reducing the Oracle queries needed in PCA. The adversary may stop querying plaintext-checking oracle and solves the remaining part of reused secret offline with the help of lattice reduction algorithms when the cost of lattice reduction algorithms becomes acceptable.

Another strategy to describe security loss in PCA is to transform PC-hints into "modular hints" in the form of $\langle sk_A, \boldsymbol{v} \rangle = l \ mod \ k$ as described in [7].

Integrating such modular hints into the lattice reduces the volume of the lattice by k. When the range of secret and error of KEM is large, it requires more PC Oracle queries (e.g. Frodo640 requires 18227 times of queries to $\mathcal{O}$ to recover secret sk_A), such strategy performs may perform better than the strategy used in this work.

However, It is difficult to transform PC-hints into modular hints. Take Kyber512 as an example, the range of secret key $[-3, 3]$ should be separated into $\{-2, 0, 2\}$ and $\{-3, -1, 1, 3\}$, thus $\langle sk_A, \boldsymbol{v}\rangle = l \mod 2$ with $l = 1$ or $l = 0$. However, we have tried all possible h and we found that it is impossible to create such a modular hint. The situations are similar in other schemes. We leave this as an open problem: can we utilize modular hints in PCA to analyze the relationship between bit-security and arbitrary PC Oracle access?

The bit-security analysis in this paper works under perfect PC Oracle. Recently, Shen et al. [28] presents a new checking approach in the plaintext-checking attacks, which is preferable when the constructed PC Oracle is imperfect. Imperfect PC Oracle may occur due to environmental noises, or simply the measurement limitations in implementing the PC Oracle. Their basic idea is to design new detection codes that efficiently find the problematic entries in the recovered secret key and corrects problematic entries with a small number of additional traces. When the raw oracle accuracy is fixed, Their new attack requires only 41% of the EM traces needed in a majority-voting attack in our experiments. It is appealing to analyze the relationship between imperfect PC Oracle queries and bit-security of lattice-based KEMs.

Acknowledgements We thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. Ruiqi Mi is supported by the National Key R&D Program of China (Grant No. 2021YFB3100100). Haodong Jiang is supported by National Natural Science Foundation of China (Grant No. 62002385).

References

1. Albrecht, M.R., Fitzpatrick, R., Göpfert, F.: On the Efficacy of Solving LWE by Reduction to Unique-SVP. In: Lee, H., Han, D. (eds.) Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8565, pp. 293–310. Springer (2013). https://doi.org/10.1007/978-3-319-12160-4_18, https://doi.org/10.1007/978-3-319-12160-4_18
2. Albrecht, M.R., Player, R., Scott, S.: On The Concrete Hardness of Learning With Errors. J. Math. Cryptol. **9**(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml
3. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber :Algorithm Specifications And Supporting Documentation(2019) (2020), https://pq-crystals.org/kyber/index.shtml

4. Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the Key-Reuse Resilience of NewHope. In: Matsui, M. (ed.) Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11405, pp. 272–292. Springer (2019). https://doi.org/10.1007/978-3-030-12612-4_14, https://doi.org/10.1007/978-3-030-12612-4_14
5. Bernstein, D.J., Chitchanok, Chuengsatiansup, Lange, T., van Vredendaal: Ntru prime: round 2. submission to the nist post-quantum project (2020), https://ntruprime.cr.yp.to/
6. Chen, C., et al.: NTRU Algorithm Specifications and Supporting Documentation.Submission To The NIST Post-Quantum Project (2020), https://ntru.org/
7. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with Side Information: Attacks and Concrete Security Estimation. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 329–358. Springer (2020). https://doi.org/10.1007/978-3-030-56880-1_12, https://doi.org/10.1007/978-3-030-56880-1_12
8. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F., et al.: Saber: Mod-lwr based kem algorithm specification and supporting documentation. submission to the nist post-quantum project (2019) (2020), https://www.esat.kuleuven.be/cosic/pqcrypto/saber/
9. Ding, J., Fluhrer, S.R., RV, S.: Complete Attack on RLWE Key Exchange with Reused Keys, Without Signal Leakage. In: Susilo, W., Yang, G. (eds.) Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10946, pp. 467–486. Springer (2018). https://doi.org/10.1007/978-3-319-93638-3_27, https://doi.org/10.1007/978-3-319-93638-3_27
10. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. J. Cryptol. **26**(1), 80–101 (2013). https://doi.org/10.1007/s00145-011-9114-1, https://doi.org/10.1007/s00145-011-9114-1
11. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side channel resistance (2011)
12. Greuet, A., Montoya, S., Renault, G.: Attack on LAC Key Exchange in Misuse Situation. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12579, pp. 549–569. Springer (2020). https://doi.org/10.1007/978-3-030-65411-5_27, https://doi.org/10.1007/978-3-030-65411-5_27
13. Guo, Q., Mårtensson, E.: Do Not Bound to a Single Position: Near-Optimal Multi-Positional Mismatch Attacks Against Kyber and Saber. IACR Cryptol. ePrint Arch. p. 983 (2022), https://eprint.iacr.org/2022/983
14. Huguenin-Dumittan, L., Vaudenay, S.: Classical Misuse Attacks on NIST Round 2 PQC - The Power of Rank-Based Schemes. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12146, pp. 208–227. Springer (2020). https://doi.org/10.1007/978-3-030-57808-4_11, https://doi.org/10.1007/978-3-030-57808-4_11

15. Kannan, R.: Minkowski's Convex Body Theorem and Integer Programming. Math. Oper. Res. **12**(3), 415–440 (1987). https://doi.org/10.1287/moor.12.3.415, https://doi.org/10.1287/moor.12.3.415
16. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. J. ACM **60**(6), 43:1–43:35 (2013). https://doi.org/10.1145/2535925, https://doi.org/10.1145/2535925
17. Naehrig, M., Alkim, E., et al.: Frodokem learning with errors key encapsulation: algorithm specification and supporting documentation. submission to the nist post-quantum project(2019) (2020), https://frodokem.org/
18. NIST: Call For Proposals, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals
19. NIST: Preparing for Post-Quantum Cryptography:Informatic (May 2021), https://www.dhs.gov/sites/default/files/publications/post-quantum_cryptography_infographic_october_2021_508.pdf
20. NIST: Selected algorithms 2022 (2022), https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
21. Okada, S., Wang, Y., Takagi, T.: Improving Key Mismatch Attack on NewHope with Fewer Queries. In: Liu, J.K., Cui, H. (eds.) Information Security and Privacy - 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30 - December 2, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12248, pp. 505–524. Springer (2020). https://doi.org/10.1007/978-3-030-55304-3_26, https://doi.org/10.1007/978-3-030-55304-3_26
22. Poppelmann, T., Alkim, E., et al.: Newhope: algorithm specification and supporting documentation(2019) (2020), https://newhopecrypto.org/
23. Qin, Y., Cheng, C., Ding, J.: A Complete and Optimized Key Mismatch Attack on NIST Candidate NewHope. In: Sako, K., Schneider, S.A., Ryan, P.Y.A. (eds.) Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11736, pp. 504–520. Springer (2019). https://doi.org/10.1007/978-3-030-29962-0_24, https://doi.org/10.1007/978-3-030-29962-0_24
24. Qin, Y., Cheng, C., Ding, J.: An Efficient Key Mismatch Attack on the NIST Second Round Candidate Kyber. IACR Cryptol. ePrint Arch. p. 1343 (2019), https://eprint.iacr.org/2019/1343
25. Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., Ding, J.: A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13093, pp. 92–121. Springer (2021). https://doi.org/10.1007/978-3-030-92068-5_4, https://doi.org/10.1007/978-3-030-92068-5_4
26. Ravi, P., Roy, S.S., Chattopadhyay, A., Bhasin, S.: Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 307–335 (2020). https://doi.org/10.13154/tches.v2020.i3.307-335, https://doi.org/10.13154/tches.v2020.i3.307-335
27. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93. ACM (2005). https://doi.org/10.1145/1060590.1060603, https://doi.org/10.1145/1060590.1060603

28. Shen, M., Cheng, C., Zhang, X., Guo, Q., Jiang, T.: 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). https://doi.org/10.46586/tches.v2023.i1.89-112, https://doi.org/10.46586/tches.v2023.i1.89-112
29. Zhang, X., Cheng, C., Ding, R.: Small Leaks Sink a Great Ship: An Evaluation of Key Reuse Resilience of PQC Third Round Finalist NTRU-HRSS. In: Gao, D., Li, Q., Guan, X., Liao, X. (eds.) Information and Communications Security - 23rd International Conference, ICICS 2021, Chongqing, China, November 19-21, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12919, pp. 283–300. Springer (2021). https://doi.org/10.1007/978-3-030-88052-1_17, https://doi.org/10.1007/978-3-030-88052-1_17