SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2024, Vol. 67, Iss. 11, 212302:1–212302:18
https://doi.org/10.1007/s11432-024-4150-3

A closer look at the belief propagation algorithm in side-channel attack on CCA-secure PQC KEM

Kexin QIAO[1,2*], Zhaoyang WANG[1], Heng CHANG[1], Siwei SUN[3*], Zehan WU[1], Junjie CHENG[1], Changhai OU[4*], An WANG[1*] & Liehuang ZHU[1*]

[1]*School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*State Key Laboratory of Cryptology, P. O. Box 5159, Beijing* 100878*, China;*
[3]*School of Cryptology, University of Chinese Academy of Sciences, Beijing* 100089*, China;*
[4]*School of Cyber Science and Engineering, Wuhan University, Wuhan* 430072*, China*

Received 9 February 2024/Revised 29 July 2024/Accepted 12 September 2024/Published online 23 October 2024

Abstract The implementation security of post-quantum cryptography (PQC) algorithms has emerged as a critical concern with the PQC standardization process reaching its end. In a side-channel-assisted chosen-ciphertext attack, the attacker builds linear inequalities on secret key components and uses the belief propagation (BP) algorithm to solve. The number of inequalities leverages the query complexity of the attack, so the fewer the better. In this paper, we use the PQC standard algorithm CRYSTALS-Kyber as a study case to construct bilateral inequalities on key variables with substantially narrower intervals using a side-channel-assisted oracle. For Kyber512, Kyber768, and Kyber1024, the average Shannon entropy carried by such inequality is improved from the previous 0.6094, 0.4734, and 0.8544 to 0.6418, 0.4777, and 1.2007. The number of such inequalities required to recover the key utilizing the BP algorithm for Kyber512 and Kyber1024 is reduced by 5.32% and 40.53% in theory and experimentally the reduction is even better. The query complexity is reduced by 43%, 37%, and 48% for Kyber512, 768, and 1024 assuming reasonably perfect reliability. Furthermore, we introduce a strategy aimed at further refining the interval of inequalities. Diving into the BP algorithm, we discover a measure metric named JSD (Jensen-Shannon distance)-metric that can gauge the tightness of an inequality. We then develop a machine learning-based strategy to utilize the JSD-metrics to contract boundaries of inequalities even with fewer inequalities given, thus improving the entropy carried by the system of linear inequalities. This contraction strategy is at the algorithmic level and has the potential to be employed in all attacks endeavoring to establish a system of inequalities concerning key variables.

Keywords Kyber, chosen-ciphertext attack, side-channel, belief propagation, contraction strategy, machine learning

1 Introduction

CRYSTALS-Kyber [1] is selected by the US National Institute of Standards and Technology (NIST) [2] as the standard post-quantum cryptographic algorithm for key-establishment to address the quantum computing challenge to classical public-key cryptographic systems like RSA. It is a lattice-based key-encapsulation mechanism (KEM) known for its small key and ciphertext sizes and high computational speed, making it well-suited for resource-constrained embedded devices which are more vulnerable to implementation attacks. As standardization completed, ensuring the implementation security of PQC algorithms is a top priority in the cryptographic community.

The concretization of the chosen ciphertext attack (CCA) security in Kyber's KEM variant is realized through the application of the Fujisaki-Okamoto (FO) transformation [3] during the decapsulation process. It serves as a mechanism to detect any alterations or anomalies within the ciphertexts, leading to the expedient abandonment of the shared cryptographic key in such cases.

* Corresponding author (email: qiao.kexin@bit.edu.cn, sunsiwei@ucas.ac.cn, ouchanghai@whu.edu.cn, wanganl@bit.edu.cn, liehuangz@bit.edu.cn)

However, recent investigations [4–11] have brought to light the vulnerability of the FO transformation to side-channel leakage. This susceptibility has been exploited within the chosen-ciphertext attacks, giving rise to equalities or inequalities concerning the secret keys and ultimately resulting in the compromise of the full key. The inequalities can be viewed as coding of the secret values and solving such a system is a decoding process. A sufficient number of inequalities provide sufficient information or Shannon entropy of secret values so that an effective decoding method can be applied. Lowering the number of inequalities is of great relevance. As is mentioned by Hermelin et al. [12], fewer inequalities mean fewer queries of faulty ciphertexts to the vulnerable device and thus fewer measurements in side-channel attacks. From a countermeasure perspective, an accurate estimation of the number of faulty queries needed to launch an attack determines how many failed decapsulations are allowed to tolerate transmission errors as well as to prevent attacks. In the realm of certification of security products, the number of side-information measurements needed to break the security is a key factor in determining the score in a test.

This paper proposed a method to construct linear inequalities on secret key components with narrower intervals and thus higher Shannon entropy by side-channel-assisted oracle so that the number of inequalities needed is reduced. Additionally, a strategy to further refine the intervals to augment the entropy carried by the inequalities is proposed. We use CRYSTALS-Kyber as a study case.

1.1 Related work

Oracle construction. Side-channel information is used to construct oracles to determine whether intermediate decrypted plaintext during the decapsulation with handcrafted ciphertexts matches some preassumed values [13, 14]. Major oracles employed in side-channel-assisted CCA attacks on lattice-based KEMs include plaintext-checking oracle [4, 6, 8, 9, 15–17], decryption-failure oracle [7, 18, 19] and full-decryption oracle [10, 11, 20]. These oracles can be constructed even for protected implementations (e.g., Bhasin et al.'s decryption-failure oracle [19] for protected implementation of ciphertext comparison [21,22]). Recently, Mondal et al. [23] presented an improved version of the plaintext checking oracle, which reduces the number of Kyber768 queries by about 23 percent. Maliciously chosen ciphertexts used in plaintext-checking oracle and full-decryption oracle attacks are very sparse in that most coefficients are zeros. Such ciphertexts are of extremely low probability for honest users thus are easy to be detected by ciphertext sanity check [14]. However, the number of queries of such ciphertexts needed in an attack is the fewest. For example, in Qin et al.'s work [5] of investigating the lower bound on the number of queries needed to recover the key, only one component of queried ciphertexts is non-zero. To avoid such a malicious characteristic, we chose to use decryption-failure oracles in which queried ciphertexts are random. Decryption-failure oracle can also be constructed by fault-injection approaches [24–28], which are generally perceived as having a higher cost. Recently, Kundu et al. [29] proposed a new fault attack on side-channel secure masked implementation of LWE-based KEMs exploiting fault propagation to build decryption-failure oracle.

Solving system of inequalities. For Kyber, though inequalities within interval $q/2^{d_v}$ have already been constructed by side-channel approaches in [19] (also mentioned in [7]), they only approximated the inequalities by equations and fed to the LWE framework [30] to estimate the remaining security level instead of solving the system practically. Furthermore, with the same amount of inequalities as those that will be solved practically in minutes in this work, their estimated security level is still above 2^{70} [19, Figure 4a]. So such kind of system of inequalities has not been practically solved yet. Pessl et al. [24] developed a belief-propagation technique to solve for secret variables practically. Hermelink et al. [27] modified the solving algorithm and formally introduced the belief propagation (BP) algorithm to solve such erroneous linear inequalities, which was also applied by D'Anvers et al. [7]. Delvaux [28] calculated the large number of summation distributions in the BP algorithm according to the central limit theorem (CLT) instead of Fast Fourier Transformation (FFT), which is recently applied by Kundu et al. [29] to solve inequalities obtained from the fault on Kyber512. Recently, Hermelink et al. [12] integrated the BP processed information in lattice reduction algorithms to make use of the advantages of both statistic and algebraic approaches. Currently, all practically solved inequalities are unilateral inequalities. Guo et al. [31] used BP algorithm to decode leakage in soft analytical side-channel attacks on symmetric ciphers.

1.2 Contributions

Assuming the presence of a decryption-failure oracle constructed through side-channel methodologies (as exemplified in [7,19]), we concretize the approach to build bilateral inequalities on secret keys with smaller intervals in chosen-ciphertext attack scenario and solve the system practically. Notably, the inequalities exhibit intervals of size $q/2^{d_v}$ (e.g., $q/16$ for Kyber512), which is more refined than the previously established unilateral inequalities. The Shannon entropy carried by such inequality for Kyber512, Kyber768, and Kyber1024 is about 0.6418, 0.4777, and 1.2007, which is higher than the previous 0.6093, 0.4734, and 0.9943. The increase for Kyber1024 is as high as 40.53%. Subsequently, the approximate number of inequalities needed to recover the key with success rate 1 with BP algorithm is reduced from state-of-the-art by 10.47% for Kyber512 without ciphertext filtering technique and 25.93% for Kyber1024 with ciphertext filtering technique. The number of inequalities required for an attack on Kyber1024 is even fewer than that for the Kyber768 version. The full key components can be recovered with 7000 inequalities in practical experiments. We explain this phenomenon from the perspective of information theory. The query complexity of Kyber512, Kyber768, and Kyber1024 is reduced by 43%, 37.5%, and 48.4% assuming reasonably perfect reliability.

Moreover, we introduce a strategy for further narrowing down the intervals of these inequalities. Firstly, we discover a quantitative measure, specifically the Jensen-Shannon distance (JSD) metric computed between a *marginal distribution* generated during BP iterations and a uniform distribution. This measure serves as an indicator of the proximity of the true value of the linear combination of secret variables to the established bounds. Once the proximity value is known, the inequality can be contracted by this amount so that the entropy it carries is improved. We develop a machine-learning model to predict the proximities of inequalities. In the profiling phase, the JSD-metrics and known proximities are used as the training set for a random forest model, and then in the attacking phase, the model will predict the proximities for JSD-metrics collected under an unknown key. Note that this contraction strategy can be applied to a system with fewer inequalities, having the potential to solve the secret variables with lower query complexity.

In summary, existing literature has considered how to implement oracles at the hardware level, but has not fully investigated how to make the number of inequalities required for the attack and the resulting query complexity lower, which is in fact an essential factor in the security evaluation of the implementation of PQC algorithms. In this paper, we are motivated to reduce the number of inequalities required for the attack by designing a side-channel attack method against CRYSTALS-Kyber, which sets a new record for the lowest number of inequalities and query complexity. Meanwhile, our method invokes the decryption-failure oracle, which has the advantages of being less likely to be detected by sanity check and having a lower device cost than the fault-injection approach. The results in this paper are important for the security evaluation of the implementations of PQC algorithms in the future, and the revealed discovery that the attack complexity of the Kyber1024 is lower than that of the Kyber768 also inspires the community to re-exam the choice of Kyber parameters.

2 Preliminaries

Notations used in this paper are in Table 1. When the functions Comp_q or Decomp_q is applied on $x \in R_q$ or $\mathbf{x} \in R_q^k$, the procedure is applied to each coefficient individually.

2.1 Description of KYBER.CCAKEM

Kyber [1] is an IND-CCA2-secure KEM that relies on the hardness of module learning with errors (MLWE) problem [32]. The elements are defined on ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ where $n = 256, q = 3329$. The parameters for the three versions are shown in Table 2. The CCA-secure is derived by using FO transformation of a CPA-secure publick-key encryption scheme (PKE). This transformation facilitates post-decryption re-encryption of the plaintext, followed by a rigorous equality check comparing the received ciphertext with the resultant re-encrypted output. The key establishment procedure is shown in Figure 1.

The ciphertext consists of compressing results of a vector of polynomials $\mathbf{u}$ (to get c_1) and a polynomial v (to get c_2). In the decapsulation, Alice decompresses both c_1 and c_2, retrieves approximate values of $\mathbf{u}$ and v, and retrieves the message m' using her secret key. Then she re-encrypts the retrieved message

Table 1 Notations

Notation	Description
$a\|\|b$	Concatenation of a and b
R_q	The ring $\mathbb{Z}_q[x]/(x^n+1)$, $n=256, q=3329$
Regular font letters e.g., v, e	Element in R_q
Bold letters e.g., $\mathbf{v}$ (or $\mathbf{A}$)	Column vector (or matrix) with components in R_q
$\mathbf{v}^{\mathrm{T}}$ (or $\mathbf{A}^{\mathrm{T}}$)	Transpose of $\mathbf{v}$ (or $\mathbf{A}$)
$v[i]$	Coefficient of monomial x^i in $v \in R_q$
$\mathbf{v}[i]$	i-th entry of $\mathbf{v}$
$\mathbf{A}[i][j]$	Entry in row i, column j of $\mathbf{A}$
$r \bmod^{\pm} q$	Centered modulo $r' \in [-\frac{q-1}{2}, \frac{q-1}{2}]$ s.t. $r' \equiv r \mod q$
$r \bmod^{+} q$	Positive modulo $r' \in [0, q)$ s.t. $r' \equiv r \mod q$
$r \bmod^{*} q$ (r^* for short)	Biased modulo $r' \in (-\frac{q}{4}, \frac{3q}{4})$ s.t. $r' \equiv r \mod q$
$\lceil x \rfloor$	Rounding of $x \in \mathbb{Q}$ to the closest integer
$\leftarrow$	Sample randomly from a distribution
$\xleftarrow{r}$	Sample from a distribution using r as the random seed
B_η	A centered binomial distribution: $\sum_{i=1}^{\eta}(a_i - b_i)$ where $(a_1, \cdots, a_\eta, b_1, \cdots, b_\eta) \leftarrow \{0,1\}^{2\eta}$
$\mathsf{Comp}_q(\cdot,\cdot)$	$\mathsf{Comp}_q(x,d) = \lceil (2^d/q)\cdot x \rfloor \bmod^{+} 2^d$
$\mathsf{Decomp}_q(\cdot,\cdot)$	$\mathsf{Decomp}_q(x,d) = \lceil (q/2^d)\cdot x \rfloor$
$\mathsf{Encode}(\cdot)$	$\mathsf{Encode}(x) = \mathsf{Comp}_q(x,1)$
$\mathsf{Decode}(\cdot)$	$\mathsf{Decode}(x) = \mathsf{Decomp}_q(x,1)$
$G(\cdot), H(\cdot)$	Hash function instantiated with SHA3-256 and SHA3-512
$\mathrm{KDF}(\cdot)$	A key-derivation function

Table 2 Kyber parameters

	n	k	q	η_1	η_2	(d_u, d_v)	δ
Kyber512	256	2	3329	3	2	(10,4)	2^{-139}
Kyber768	256	3	3329	2	2	(10,4)	2^{-164}
Kyber1024	256	4	3329	2	2	(11, 5)	2^{-174}

with her public key to get ciphertext $c_1'||c_2'$. The key is established if $c_1'||c_2'$ and the received $c_1||c_2$ are equal; otherwise, a random number is returned.

The compression performed by Bob and decompression performed by Alice introduce offsets on $\mathbf{u}$ and v, i.e., $\Delta_v = v'' - v = \mathsf{Decomp}_q(\mathsf{Comp}_q(v)) - v$ and $\Delta_{\mathbf{u}} = \mathbf{u}'' - \mathbf{u} = \mathsf{Decomp}_q(\mathsf{Comp}_q(\mathbf{u})) - \mathbf{u}$. The noise introduced in computing m' (line 3.3 in Figure 1) is

$$d = \mathbf{e}^{\mathrm{T}}\mathbf{r} - \mathbf{s}^{\mathrm{T}}(\Delta_{\mathbf{u}} + \mathbf{e}_1) + e_2 + \Delta_v. \tag{1}$$

To avoid decryption errors by honest users, the parameters of Kyber are chosen such that each component of d satisfies $d[j]^* \in (-q/4, q/4), \forall j \in [0, n-1]$ with approximately probability 1. So even with noise d, during the compression to recovered m', the $v'' - \mathbf{s}^{\mathrm{T}}\mathbf{u}''$ is still within the hemisphere around the original value of m as is shown in Figure 2.

2.2 Belief propagation algorithm

The belief propagation (BP) algorithm was first introduced to solve a system of linear inequalities with variables sampling from a finite centered distribution by Hermelink and Pessl et al. [24,27]. It is an error-tolerant and efficient method to solve systems with possible errors. The w linear inequalities deduced in previous work [7,12,24,27,28] are of the form

$$\forall i \in [0, w-1], \quad \sum_{j=0}^{\psi-1} A[i][j]x[j] \gtrless b[i],$$

where $\psi = 2nk$ and $x[0:\psi/2-1]$ are coefficients of $\mathbf{e}$ and $x[\psi/2:\psi-1]$ are coefficients of $\mathbf{s}$ in the attack scenerio. As $\mathbf{e}, \mathbf{s}$ satisfy the public key generation expression $\mathbf{As} + \mathbf{e} = \mathbf{t}$, any nk known values of x are enough to deduce the other unknowns.

Alice		**Bob**
1. ▷ KYBER.CCAKEM.KeyGen()		2. ▷ KYBER.CCAKEM.Enc(pk)
1.1 $\mathbf{A} \xleftarrow{\rho} R_q^{k\times k}$		2.1 $m \leftarrow\$ \{0,1\}^{256}$
1.2 $\mathbf{s},\mathbf{e} \leftarrow\$ B_{\eta_1}^k$		2.2 $(\bar{K}, r) := G(m\|\|H(pk))$
1.3 $\mathbf{t} = \mathbf{As} + \mathbf{e}$	$\xrightarrow{pk}$	2.3 $(\mathbf{r}, \mathbf{e}_1, e_2) \xleftarrow{r} (B_{\eta_1}^k, B_{\eta_2}^k, B_{\eta_2})$
1.4 **return** $(pk := (\rho, \mathbf{t}), \mathbf{s})$		2.4 $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
		2.5 $v = \mathbf{t}^T\mathbf{r} + e_2 + \mathsf{Decode}(m)$
3. ▷ KYBER.CCAKEM.Dec($c_1, c_2, \mathbf{s}$)		2.6 $c_1 = \mathsf{Comp}_q(\mathbf{u}, d_u) \triangleq \mathbf{u}'$
3.1 $\mathbf{u}'' = \mathsf{Decomp}_q(c_1, d_u)$		2.7 $c_2 = \mathsf{Comp}_q(v, d_v) \triangleq v'$
3.2 $v'' = \mathsf{Decomp}_q(c_2, d_v)$		2.8 $K_B = KDF(\bar{K}\|\|H(c_1\|\|c_2))$
3.3 $m' = \mathsf{Encode}(v'' - \mathbf{s}^T\mathbf{u}'')$	$\xleftarrow{c_1, c_2}$	2.9 **return** (c_1, c_2, K_B)
3.4 $(\bar{K}', r') = G(m'\|\|H(pk))$		
3.5 $(\mathbf{r}', \mathbf{e}_1', e_2') \xleftarrow{r'} (B_{\eta_1}^k, B_{\eta_2}^k, B_{\eta_2})$		
3.6 $\mathbf{u}_{re} = \mathbf{A}^T\mathbf{r}' + \mathbf{e}_1'$		
3.7 $v_{re} = \mathbf{t}^T\mathbf{r}' + e_2' + \mathsf{Decode}(m')$		
3.8 $c_1' = \mathsf{Comp}_q(\mathbf{u}_{re}, d_u)$		
3.9 $c_2' = \mathsf{Comp}_q(v_{re}, d_v)$		
3.10 **if** $c_1'\|\|c_2' = c_1\|\|c_2$ **then**		
3.11 $K_A = KDF(\bar{K}'\|\|H(c_1'\|\|c_2'))$		
3.12 **else**		
3.13 random K_A		
3.14 **return** K_A		

Figure 1 (Color online) KYBER.CCAKEM procedure (simplified).

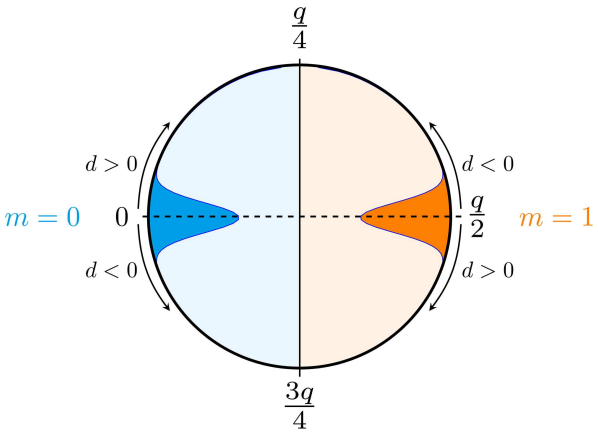

Figure 2 Decoding of m with noise d.

The mechanism of the BP algorithm to solve a system of linear inequalities is as follows. Referring to Figure 3, variables and inequalities constitute nodes of two subsets in a complete bipartite graph. In each variable note, store a vector of length $2\eta_1 + 1$ of the probabilities for taking values in $[-\eta_1, \eta_1]$. The original distribution of the unknown $x[j]$s are B_{η_1} for $j \in [0, \psi - 1]$. Then the probability vectors propagate to check nodes. Each check node uses one inequality to update the probabilities of all variables. Without loss of generality, to update the probability vector of $x[j]$ by the i-th inequality, compute the probability distribution of

$$z[j] = \sum_{j' \in [0,\psi-1]\setminus\{j\}} A[i, j']x[j'], \tag{2}$$

i.e., the linear combination of all variables except the target variable $x[j]$. This can be done by FFT [24,27] or by the CLT [28]. Then enumerate all the guesses for $x[j]$. For each guess $\epsilon \in B_{\eta_1}$, calculate the

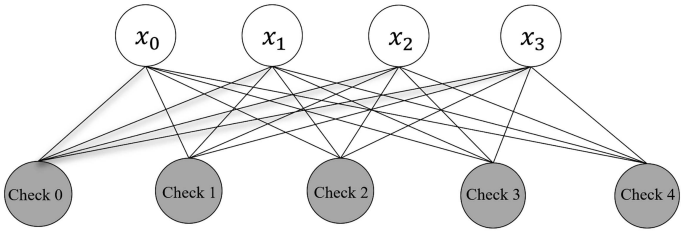

Figure 3 A complete bipartite graph with four variable nodes and five check nodes corresponding to four variables and five inequalities.

probability that the i-th inequality is satisfied

$$\begin{aligned} &\Pr[x[j]=\epsilon \text{ regarding the } i\text{-th inequality}] \\ =&\Pr[\sum_{j'\in[0,\psi-1]\setminus\{j\}} A[i,j']x[j'] \gtrless b[i]-A[i,j]\cdot\epsilon] \\ =&\sum_{t\gtrless b[i]-A[i,j]\cdot\epsilon} \Pr[z[j]=t]. \end{aligned} \tag{3}$$

Then the i-th check node returns the probability vector to the j-th variable node according to Equation (3). For each variable node, after receiving probability vectors from all check nodes, perform a normalization and use it as an updated distribution. This completes one iteration. Given enough inequalities, the distributions are expected to converge to the correct solution. When the entropy of the variables is low enough or is no longer decreasing, the iteration stops and outputs the suggested solutions, or states "fail".

When the BP algorithm can not fully recover the secrets, the partially recovered keys can be integrated into lattice reduction algorithms to solve the LWE equation system $\mathbf{As}+\mathbf{e}=\mathbf{b}$ and further estimate the remaining security [12].

3 Side-channel attack

3.1 Side-channel based decryption failure oracle

Decryption-failure oracles can be constructed by side-channel approaches as exemplified in [7, 19]. Such oracles predict in the CCA attack whether decrypted plaintext m' (line 3.3 in Figure 1) from faulty ciphertext during decapsulation equals the right plaintext m (line 2.1). If they don't match, it is said that a decryption failure occurs. The leakage during equality checking operation (line 3.10) or calculation of function G (line 3.4) can all be utilized to construct such an oracle. In this paper, we assume such oracle $\mathcal{B}$ is already presented and focus on how to choose ciphertexts in queries to deduce inequalities concerning secret coefficients in $\mathbf{e}$ and $\mathbf{s}$ with interval $q/2^{d_v}$. Oracle $\mathcal{B}$ takes $c=\mathbf{u}'||v'$ as input and outputs 1 if $m'=m$ and 0 otherwise.

$$\mathcal{B}(c)=\begin{cases}1, & \text{if } m'=m \text{ during decapsulation},\\ 0, & \text{otherwise.}\end{cases}$$

3.2 Attack scenario

Malicious Bob aims to derive biliteral inequalities on Alice's secret key $(\mathbf{e},\mathbf{s})$ with interval $q/2^{d_v}$ by obtaining the interval for decryption noise d (recall (1)). Hints on some $d[j]$ should be converted to linear hints on coefficients in $\mathbf{e}$ and $\mathbf{s}$. When the coefficients in v lie in $(q-1-q/2^{d_v+1}, q-1]$, besides an offset within $q/2^{d_v+1}$ an extra offset q will also be introduced in Δ_v and then to the decryption error d. Specifically, for KYBER512, $v[j]\in[3225,3328]$ are compressed to $v'[j]=0x0000$ and then decompressed to $v''[j]=0$. Then $\Delta_v[j]=v''-v=\mathsf{Decomp}_q(\mathsf{Comp}_q(v))-v\in[-3328,-3225]=[1-q, q/2^{d_v+1}-q]$. This is the same case for $\mathbf{u}$. Considering that the decryption error d lies in interval $(-q/4, q/4)$ after the biased modulo, to construct perfect hints on the secret keys, a biased modulo or centered modulo should

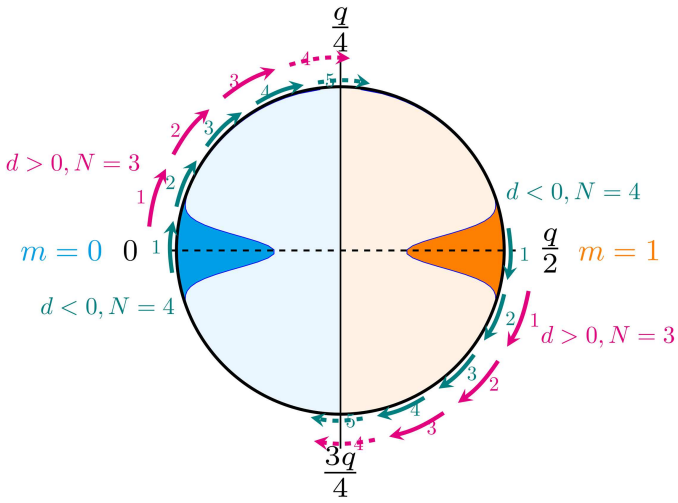

Figure 4 Get a range of size $q/2^{d_v}$ ($q/16$ for Kyber512) for noise d.

be applied to Δ_v and $\Delta_{\mathbf{u}}$ to remove the extra offset q when calculating d. So we alter the expression of the j-th component of d from (1) to

$$d[j] = \mathbf{e}^{\mathrm{T}}\mathbf{r}[j] - \mathbf{s}^{\mathrm{T}}(\Delta^*_{\mathbf{u}} + \mathbf{e}_1)[j] + e_2[j] + \Delta^*_v[j], \tag{4}$$

so that all calculations are performed on the integer ring. Multiplication on $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, also known as modular multiplication of polynomials on $\mathbb{Z}_q$, can be converted to matrix-vector multiplication on $\mathbb{Z}_q$. In this way, any hints on $d[j]$ will be converted to linear hints on secrete coefficients of $\mathbf{e}$ and $\mathbf{s}$.

To get hints on $d[j]$, after generating $(c_1||c_2) \triangleq (\mathbf{u}', v')$, Bob constructs faulty ciphertext $v'_{\mathtt{fault}}$ by increasing $v'[j]$ by N (N starts from 1). The faulty ciphertext is sent to Alice and Bob observes whether decryption failure occurs from oracle. When $v'[j]$ increases by 1, $v''[j] = \lceil (q/2^{d_v}) \cdot v'[j] \rfloor$ increases by roughly $q/2^{d_v}$ in the decompression procedure, except when faulty $v'[j]$ increases from $2^{d_v} - 1$ to 0 such that v'' increases by $q/2^{d_v} - q$.

Use Δ^*_N to denote the effective increase in $d[j]$ when increasing the j-th component of ciphertext v' by N. Bob can calculate

$$\begin{aligned}\Delta^*_N =& (\mathsf{Decomp}_q((v'[j] + N) \bmod^+ 2^{d_v}) - \mathsf{Decomp}_q(v'[j])) \bmod^* q \\ =& q/2^{d_v} \times N.\end{aligned}$$

Denote $d_{\mathtt{fault}N} = d[j] + \Delta^*_N$, which means the faulty $d[j]$ increases by $q/2^{d_v} \times N$. When N is small, faulty $d_{\mathtt{fault}N}$ is still within $(-q/4, q/4)$, so no decryption failure is observed. Bob continues the query by adding another 1 on the same ciphertext coefficient, so that faulty $d[j]$ increases by another $q/2^{d_v}$. When $d_{\mathtt{fault}N}$ steps across the hemisphere border as is illustrated in Figure 4, decryption failure is observed. Bob obtains the faulty boundary that enables decryption failure so that a range of size $q/2^{dv}$ for $d[j]$ is derived. Suppose no decryption failure occurs when increasing the ciphertext by N and a decryption failure first appears with fault $(N + 1)$. When $m'[j]$ changes from 0 to 1, the attacker has

$$\begin{cases} d[j] + \Delta^*_N \leqslant \lfloor q/4 \rfloor, \\ d[j] + \Delta^*_{N+1} \geqslant \lceil q/4 \rceil, \end{cases}$$

which implies

$$(\lceil q/4 \rceil - \Delta^*_{N+1})^* \leqslant d[j] \leqslant (\lfloor q/4 \rfloor - \Delta^*_N)^*. \tag{5}$$

Similarly, when $m'[j]$ changes from 1 to 0, the attacker has

$$\begin{cases} d[j] + \Delta^*_N + \lceil q/2 \rfloor \leqslant \lfloor 3q/4 \rfloor, \\ d[j] + \Delta^*_{N+1} + \lceil q/2 \rfloor \geqslant \lceil 3q/4 \rceil, \end{cases}$$

Algorithm 1 Inequalities generation procedure with decryption-failure oracle

Input: list N', N_T in Table 3, public key $\mathbf{t}$ and $\mathbf{A}$, KYBER parameters, w // totally collect w inequalities;
Output: coefficent matrix A, lower bound vector c, upper bound vector b // s.t. $c \leqslant Ax \leqslant b$;

Empty A, c, b;
$n_{\text{each}} = w/256$ // collect n_{each} inequalities for each index;
$j = 0, n_{\text{collected}} = 0$;
while $j < 256$ and $n_{\text{collected}} < n_{\text{each}}$ **do**
  $m \leftarrow \{0,1\}^{256}$;
  $(\mathbf{r}, \mathbf{e}_1, e_2) \leftarrow (B_{\eta_1}, B_{\eta_2}, B_{\eta_2})$;
  $v = \mathbf{t}^{\mathrm{T}}\mathbf{r} + e_2 + \mathsf{Decode}(m)$;
  $v' = \mathsf{Comp}_q(v, d_v)$;
  $\Delta_v^* = (\mathsf{Decomp}_q(v') - v) \bmod^* q$; // apply biased modular to remove possible extra q
  $\mathbf{u} = \mathbf{A}^{\mathrm{T}}\mathbf{r} + \mathbf{e}_1 \in R_q^k$;
  $\mathbf{u}' = \mathsf{Comp}_q(\mathbf{u}, d_u)$;
  $\Delta_{\mathbf{u}}^* = (\mathsf{Decomp}_q(\mathbf{u}') - \mathbf{u}) \bmod^* q$; // apply biased modular to remove possible extra q
  $r =$ concatenation of $\mathrm{Conv}(a)[j]$ for each compoment a in $\mathbf{r}||(-\mathbf{e}_1 - \Delta_{\mathbf{u}}^*)$;
  Append r to A as a row;
  $i = 0$;
  $T = []$;
  while not success(T) **do**
    $v'_{\mathtt{fault}} = v'$;
    $v'_{\mathtt{fault}}[j] = (v'[j] + N'[i]) \bmod^+ 2^{d_v}$;
    Append $\mathcal{B}(\mathbf{u}'||v'_{\mathtt{fault}})$ to T // $T[i] = \mathcal{B}(\mathbf{u}'||v'_{\mathtt{fault}})$
    $i++$;
  end while
  $N = N_T$;
  $\Delta_N^* = (\mathsf{Decomp}_q((v'[j] + N) \bmod^+ 2^{d_v}) - \mathsf{Decomp}_q(v'[j])) \bmod^* q$;
  $\Delta_{N+1}^* = (\mathsf{Decomp}_q((v'[j] + N + 1) \bmod^+ 2^{d_v}) - \mathsf{Decomp}_q(v'[j])) \bmod^* q$;
  if $m[j] = 0$ **then**
    $B_l = (\lceil \frac{q}{4} \rceil - \Delta_{N+1}^*) \bmod^* q$;
    $B_u = (\lfloor \frac{q}{4} \rfloor - \Delta_N^*) \bmod^* q$;
  else
    $B_l = (\lceil 3q/4 \rceil - \lceil q/2 \rfloor - \Delta_{N+1}^*) \bmod^* q$;
    $B_u = (\lfloor 3q/4 \rfloor - \lceil q/2 \rfloor - \Delta_N^*) \bmod^* q$;
  end if
  Append $B_l - e_2[j] - \Delta_v^*[j]$ to c;
  Append $B_u - e_2[j] - \Delta_v^*[j]$ to b;
  $j = j + 1$;
  if $j = n$ **then**
    $n_{\text{collected}} = n_{\text{collected}} + 1$;
    $j = 0$;
  end if
end while
return A, c, b;

which implies

$$(\lceil 3q/4 \rceil - \lceil q/2 \rfloor - \Delta_{N+1}^*)^* \leqslant d[j] \leqslant (\lfloor 3q/4 \rfloor - \lceil q/2 \rfloor - \Delta_N^*)^*. \tag{6}$$

The ranges of $d[j]$ in both (5) and (6) are of size roughly $q/2^{dv}$, which is $q/16$ for KYBER512 and KYBER768 and $q/32$ for KYBER1024. In both cases we get

$$B_l \leqslant d[j] \leqslant B_u, B_u - B_l = q/2^{dv}. \tag{7}$$

Combining (7) and (4), the inequality constructed on secrets $(\mathbf{e}, \mathbf{s})$ is

$$B_l - e_2[j] - \Delta_v^*[j] \leqslant \mathbf{e}^T\mathbf{r}[j] - \mathbf{s}^T(\Delta_{\mathbf{u}}^* + \mathbf{e}_1)[j] \leqslant B_u - e_2[j] - \Delta_v^*[j]. \tag{8}$$

We illustrate the attack process in Algorithm 1.

Querying Trick. In the experiments for KYBER512, the most frequent values of N are 3 and 4, and a few are 2 and 5. The N is determined by the distribution of the honest noise d (Equation (4)). According to our empirical simulation and also what is mentioned in [14], d follows a Gaussian distribution with a standard deviation of about $\sigma = 79$. So the attacker tries N from 3, and then 4 to derive the boundary for decryption failure. In Table 3, we present a query trick by giving the trial list for N said N' and the probability that the boundary can be deduced with the current number of queries. The output of oracle $\mathcal{B}$ by querying with faulty ciphertext corresponding to $N'[i]$ is appended to list T. When T takes the value illustrated in column 'success(T)=true', the attacker can deduce the boundary N_T. The function success($\cdot$) output $false$ with all other input Ts. The expected number of queries to deduce an inequality

Table 3 Parameters used in the query trick for Kyber512

N' list	#Queries	Probability	Range of d	success(T)=true	N_T
$N'[0]=3$ $N'[1]=4$	2	0.4957	$(0, q/16)$	$T=[1,0]$	3
$N'[2]=5$	3	0.4957	$(-q/16, 0)$	$T=[1,1,0]$	4
$N'[3]=2$	4	0.0042	$(q/16, q/8)$	$T=[0,0,0,1]$	2
$N'[4]=6$	5	0.0042	$(-q/8, -q/16)$	$T=[1,1,1,1,0]$	5
$N'[5]=1$	6	6.9182×10^{-8}	$(q/8, 3q/16)$	$T=[0,0,0,0,0,1]$	1
$N'[6]=7$	7	6.9182×10^{-8}	$(-3q/16, -q/8)$	$T=[1,1,1,1,1,1,0]$	6

is $2\times0.4957+3\times0.4957+4\times0.0042+5\times0.0042=2.5163$. So the query complexity of the attack is estimated as 2.5 times the number of inequalities required. Similarly, for Kyber768 and Kyber1024, the expected number of queries to deduce an inequality is 2.5163 and 2.5743. Details are given in the Appendix A.

Ciphertext filtering. Ciphertext filtering is an effective way to reduce the number of inequalities needed [12, 24, 27]. The honest noise d obeys a $e_2+\Delta_v$-centered distribution and when the secret key changes it only oscillates within a small range. For the faulty ciphertexts, the center of $d_{\texttt{fault}N}$ moves a distance Δ_N^* which are multiples of $q/2^{d_v}$ from $e_2+\Delta_v$ and we call them division points. The inequality of (5) and (6) is to constrain faulty $d_{\texttt{fault}N}$ in an interval between two adjacent division points and exclude $(\mathbf{e},\mathbf{s})$s that leaves $d_{\texttt{fault}N}$ out of this region. So averagely the closer the faulty $d_{\texttt{fault}N}$ to the division points, the more $(\mathbf{e},\mathbf{s})$s (half of them in the ideal case) are excluded so that the inequality is more informative. The honest noise is around the center $e_2+\Delta_v\in[-\eta_2-q/2^{d_v+1},\eta_2+q/2^{d_v+1}]$, so the closer $e_2+\Delta_v$ to division points (primarily 0), the closer the $d_{\texttt{fault}N}$ to division points. So to obtain more informative inequalities, Bob filters ciphertexts such that $|e_2+\Delta_v|\leqslant\epsilon$ to mount the attack. In Hermelink and Pessl et al.'s work for building unilateral inequalities [12,24,27], ϵ is set to 10, which is also applied in building bilateral inequalities in this work. To generate inequalities for filtered ciphertexts, after line 9 in Algorithm 1, test whether $|e_2[j]+\Delta_v^*[j]|\leqslant10$. If so, continue the algorithm; otherwise, go to line 6 to reproduce v randomly.

3.3 BP algorithm for solving systems of bilateral inequalities

We modify the BP method in [27] to solve bilateral inequalities. The inequalities deduced from Algorithm 1 are of the form:

$$\forall i\in[0,w-1],\quad c[i]\leqslant\sum_{j=0}^{\psi-1}A[i][j]x[j]\leqslant b[i]. \tag{9}$$

Given the distribution of $z[j]=\sum_{j'\in[0,\psi-1]\setminus\{j\}}A[i,j']x[j']$, the distribution for the $x[j]$ indicated by the i-th inequality is modified from (3) to

$$\begin{aligned}&\Pr[x[j]=\epsilon \text{ regarding the } i\text{-th inequality}]\\ =&\Pr[c[i]-A[i,j]\cdot\epsilon\leqslant\sum_{j'\in[0,\psi-1]\setminus\{j\}}A[i,j']x[j']\leqslant b[i]-A[i,j]\cdot\epsilon]\\ =&\sum_{c[i]-A[i,j]\cdot\epsilon\leqslant t\leqslant b[i]-A[i,j]\cdot\epsilon}\Pr[z[j]=t].\end{aligned} \tag{10}$$

The updation of the distribution in (10) enables the BP algorithm to solve faster than when treating bilateral inequalities as two unilateral inequalities. In experiments where secrets are recovered with probability 1, the efficiency is improved by roughly 3 times.

3.4 Results

We replace the BP algorithm in Hermelink's BP meets LWE [12] framework with the modified version described in Section 3.3. For Kyber512, Kyber768 and Kyber1024, the success rate and the average number of recovered coefficients concerning the number of bilateral inequalities are shown in Figure 5. We ran 10 samples per number of inequalities. All experiments ran up to 50 iterations which is the same

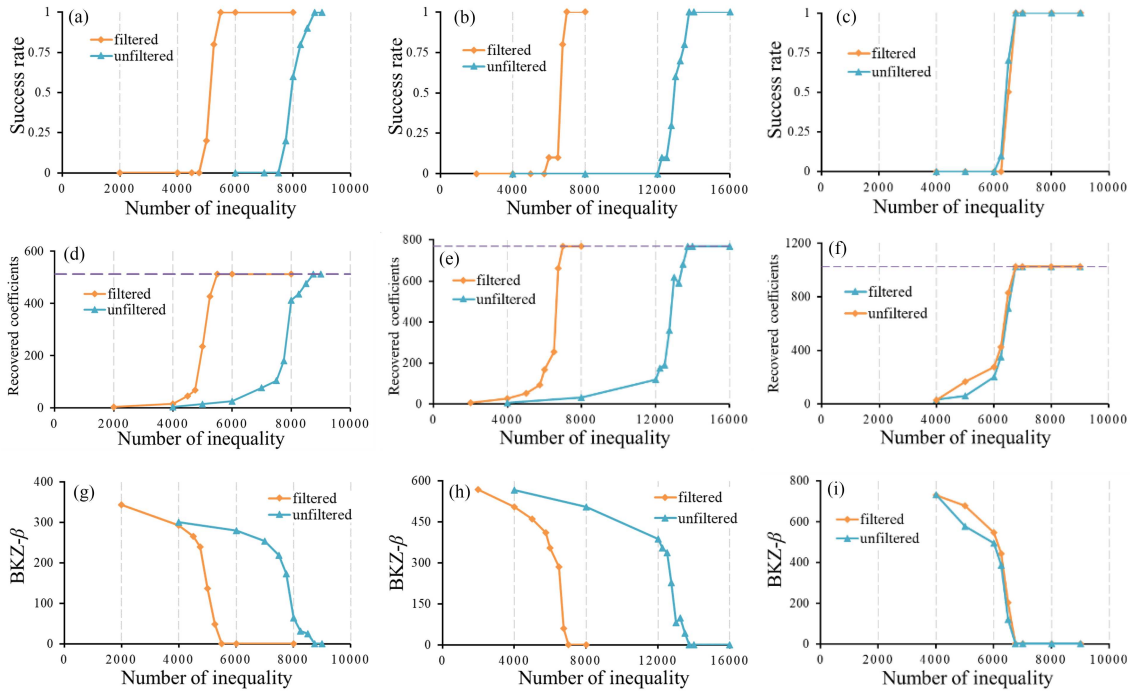

Figure 5 Success rate, average number of recovered coefficients, and remaining security with respect to the number of bilateral inequalities for Kyber512, Kyber768, and Kyber1024. (a) Success rate on Kyber512; (b) success rate on Kyber768; (c) success rate on Kyber1024; (d) average number of recovered coefficients on Kyber512; (e) average number of recovered coefficients on Kyber768; (f) average number of recovered coefficients on Kyber1024; (g) remaining security on Kyber512; (h) remaining security on Kyber768; (i) remaining security on Kyber1024.

as that in [12]. The running time for 9000 unfiltered inequalities is about 10 minutes on a 64-thread server, which is superior to the estimated security level 2^{70} previously [19, Figure 4a] with the same type of inequalities. When the BP algorithm terminates with success, the number of recovered coefficients is truncated to half the number of key components. Otherwise, the remaining security is measured by the parameter β in the BKZ algorithm for solving a reduced LWE problem [12].

The comparison with previous work is shown in Tables 4 and 5. For Kyber512, the approximate number of inequalities needed to recover the key with success rate 1 with filtered ciphertexts is about 5500 with our construction, which is smaller than 6000 to 9000 for previous unilateral inequalities [12, Table 2]. In [12], this number is also reduced to 5500. For inequalities constructed with unfiltered ciphertexts, the number of inequalities needed to recover coefficients with success rate 1 is reduced to 8600 by our construction, which is substantially smaller than previous 13000 [27, Figure 4] and 9500 [12, Figure 8]. The number of inequalities on filtered ciphertexts for Kyber1024 is reduced from state-of-the-art 8500 to 6750 by our method. There was no significant reduction on Kyber768. The unfiltered version of Kyber768 and Kyber1024 lacks previous comparative experiments.

Regarding the query complexity, we use the notation *query factor* to denote the average number of oracle calls to derive one inequality. In our attack scenario, by using the query trick, Kyber512, Kyber768 and Kyber1024 have a query factor of 2.5, 2.5 and 2.6 assuming perfect side-chanel reliability as is indicated in [7,19]. The query factor of [24] is $1/0.17 \approx 5.88$ according to their practical implementation. The query factor of both [27] and [12] are 4 assuming perfect fault injection reliability and the targeted value is boolean-masked. The number of fault injections of [28] increases by roughly one or two orders of magnitude compared to [27], so we estimate the query factor as 40 to get error-free inequalities. In [29], they empirically repeat the fault injection 2.67 times for each ciphertext, so we estimate the query factor to be 2.67. The query complexity is the multiplication of the number of inequalities and the query factor. Compared with state-of-the-art results, the query complexity of our method is reduced by 37.5%, 37.5% and 48.4% for filtered inequalities for Kyber512, 768, 1024 and 43.4% for unfiltered inequalities for Kyber512.

Table 4 Approximate number of filtered and unfiltered inequalities needed to recover the key with success rate 1 with different methods

	#Eqs. (Kyber512)		#Eqs. (Kyber768)		#Eqs. (Kyber1024)		Solving method
	Filtered	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered	
Pessl et al. [24]	8000	-	10500	-	13000	-	BP (FFT v1)
Hermelink et al. [27]	6000	-	7000	-	8500	-	BP (FFT v2)
Delvaux [28]	9000	13000	9300	-	12100	-	BP (CLT)
Kundu et al. [29]	30000	-	-	-	-	-	BP (CLT)
Hermelink et al. [12]	5500	9500	-	-	-	-	BP (FFT v2) + BKZ
This work	**5500**	**8600**	**7000**	**13750**	**6750**	**7000**	BP (FFT v2)+BKZ

Note: #Eqs.:number of inequalities. BP (FFT v1), BP (FFT v2), and BP (CLT) represent three different BP algorithms that utilize FFT with clustering structure, binary tree structure, and central limit theorem, respectively. BKZ: the correctly solved variables are brought into the LWE problem defined by the public and private keys and solved by BKZ.

Table 5 Comparison of the query complexity to recover keys with success rate 1 with different methods

	#Q. (Kyber512)		#Q. (Kyber768)		#Q. (Kyber1024)		QF
	Filtered	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered	
Pessl et al. [24]	4.704×10^4	-	6.174×10^4	-	7.644×10^4	-	5.88
Hermelink et al. [27]	2.4×10^4	-	2.8×10^4	-	3.4×10^4	-	4
Delvaux [28]	3.6×10^5	5.2×10^5	3.72×10^5	-	4.84×10^5	-	40
Kundu et al. [29]	8.01×10^4	-	-	-	-	-	2.67
Hermelink et al. [12]	2.2×10^4	3.8×10^4	-	-	-	-	4
This work	$\mathbf{1.375 \times 10^4}$	$\mathbf{2.15 \times 10^4}$	$\mathbf{1.75 \times 10^4}$	$\mathbf{3.4375 \times 10^4}$	$\mathbf{1.755 \times 10^4}$	$\mathbf{1.82 \times 10^4}$	2.5,2.5,2.6

Note: #Q.:number of queries. QF: query factor.

3.5 Analysis from an information theory perspective

The number of bilateral inequalities needed to recover the key is significantly reduced for unfiltered ciphertexts but shows no advantage over previous unilateral inequalities for filtered ciphertexts. This can be explained from the perspective of information theory. An inequality tells the interval to which d belongs. The unilateral inequality built in previous work is of the form $d > 0$ or $d < 0$, which divides the range $(-q/4, q/4)$ into two intervals. For a bilateral inequality, the range $(-q/4, q/4)$ is divided into 2^{d_v-1} intervals of length $q/2^{d_v}$. We simulate the distribution of d under a fixed set of parameters (including public keys and $\mathbf{e}_1, e_2, m, \mathbf{r}$) by sampling 2^{15} random (e, s) values. Then fit the distribution of d with a normal distribution $\mathcal{N}(\mu, \sigma)$, whose cumulative distribution function is denoted by $\Phi_{\mu,\sigma}(\cdot)$. The entropy of a unilateral inequality is estimated as

$$H_u = -(p\log_2(p) + (1-p)\log_2(1-p)), \text{ where } p = \Phi_{\mu,\sigma}(0).$$

The entropy of a bilateral inequality is estimated as

$$H_b = -\sum_{i=-2^{d_v-2}+1}^{2^{d_v-2}} p(i)\log_2(p(i)), \text{ where } p(i) = \Phi_{\mu,\sigma}(\frac{iq}{2^{d_v}}) - \Phi_{\mu,\sigma}(\frac{(i-1)q}{2^{d_v}}).$$

We generate 50 sets of parameters and calculate the entropy of inequalities of d in all 256 components. Figure 6 shows the information distribution of unilateral and bilateral inequalities generated under Kyber512, Kyber768 and Kyber1024, and Table 6 shows the mean of specific information distribution. For filtered ciphertexts of Kyber512 and both cases of Kyber768, the information carried by bilateral and unilateral inequalities is very close. However, for unfiltered ciphertexts of Kyber512 and both cases of Kyber1024, the information or Shannon entropy carried by a bilateral inequality is an average of 0.6418, 1.2198 and 1.2007, larger than 0.6094, 0.9943, and 0.8544 - the information carried by a unilateral inequality. Theoretically, the number of inequalities providing the same amount of entropy can be reduced by $(0.6418 - 0.6094)/0.6094 = 5.32\%$, $(1.2198 - 0.9943)/0.9943 = 22.68\%$, and $(1.2007 - 0.8544)/0.8544 = 40.53\%$. The experiment shows that the reduction percentage is $(9500 - 8600)/8600 = 10.47\%$ and $(8500 - 6750)/6750 = 25.93\%$ for the first two cases, better than theoretical predictions. Increasing the amount of information an equation carries is the primary advantage of bilateral inequalities over unilateral inequalities.

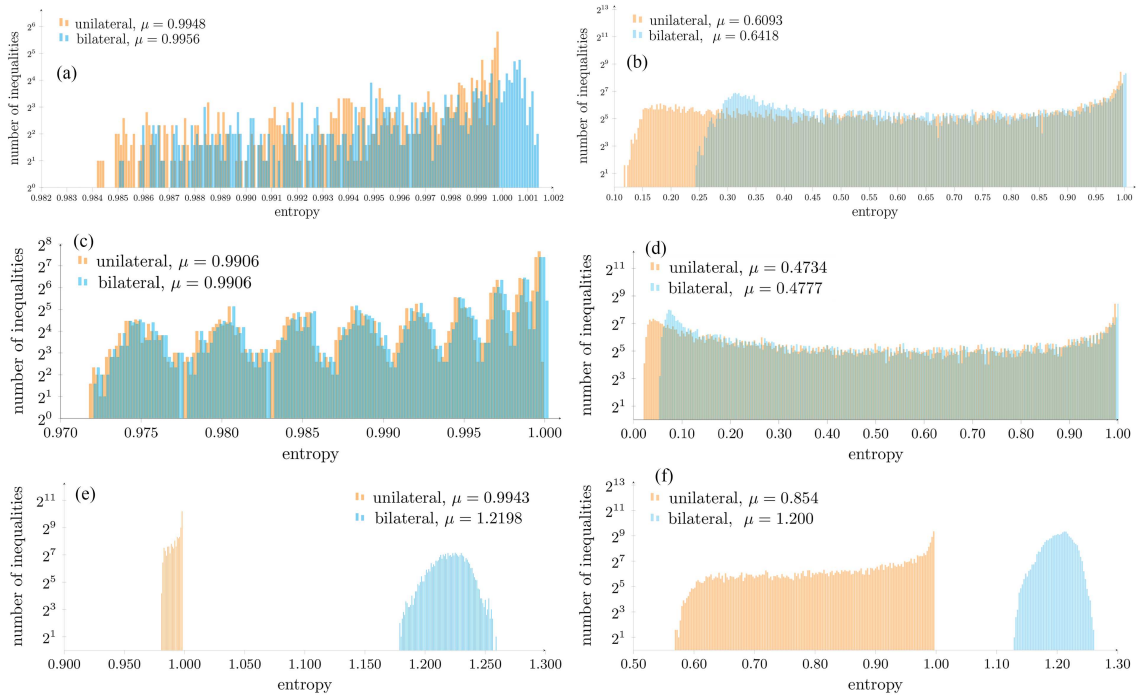

Figure 6 Distributions of entropy carried by inequalities. (a) Filtered inequalities-Kyber512; (b) unfiltered inequalities-Kyber512; (c) filtered inequalities-Kyber768; (d) unfiltered inequalities-Kyber768; (e) filtered inequalities-Kyber1024; (f) unfiltered inequalities-Kyber1024.

Table 6 Comparison of average entropy carried by inequalities

Inequality type	Kyber512		Kyber768		Kyber1024	
	Filtered	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered
Unilateral inequality	0.9948	0.6094	0.9906	0.4734	0.9943	0.8544
Bilateral inequality	0.9956	0.6418	0.9906	0.4777	1.2198	1.2007

The question that why attacking Kyber1024 requires fewer inequalities than attacking Kyber768 can be answered based on the entropy data. Compared to Kyber768, the total amount of unknown information in Kyber1024 increases by one-third. In the case of ciphertext filtering, the information of the bilateral inequality in Kyber1024 is higher than that in Kyber768 by $(1.2198–0.9906)/0.9906 = 23.13\%$. Considering that the pratical experimental results are better than the theoretical predictions, this increase compensates for the increased amount of unknown information. Thus, the number of inequalities required for the BP algorithm to successfully solve the unknowns remains about the same. In the case of unfiltered ciphertexts, the increase is as high as $(1.2007–0.4777)/0.4777 = 151.35\%$, greatly exceeding the increase in the total amount of unknown information. Based on the proportion of the increase in the information carried by the inequality and the unknowns, as well as the practical number of bilateral inequalities required for Kyber768, we can estimate the number of inequalities required to attack Kyber1024 to be $13750 \times 0.4777 \times (1 + 1/3)/1.2007 = 7293$, which is roughly consistent with the experimental result 7000. In this way, from an information theory perspective, we can explain why Kyber1024 is not more secure than Kyber768 under the attack method described in this paper.

4 Modifying system of inequalities

In this section, we aim to reduce the intervals of deduced inequalities further. We formally use the definition *proximity* to measure the tightness of inequalities in (9) as

$$\text{proximity}[i] = \min\{b[i] - \sum_{j=0}^{\psi-1} A[i][j]x[j], \sum_{j=0}^{\psi-1} A[i][j]x[j] - c[i]\},$$

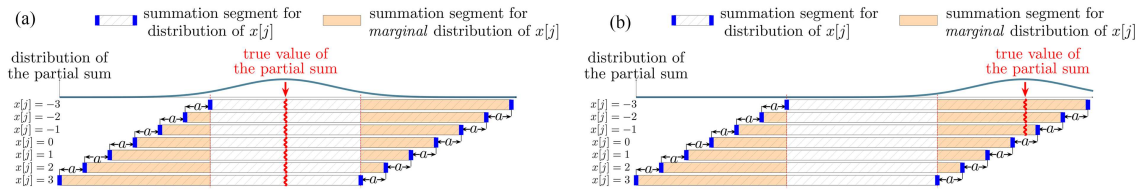

Figure 7 Marginal distribution calculation with $x[j]$'s coefficient $A[i][j] \triangleq a > 0$. (a) For inequality with large proximity, marginal distribution is close to uniform distribution; (b) for inequality with small proximity, marginal distribution is unlike uniform distribution.

where $i \in [0, w-1]$ and $x[j]$s take the true values. In the rest of the paper, when we say that inequality can be *contracted* by amount m, it means that the upper bound of the inequality can be decreased by m and the lower bound of the inequality can be increased by m simultaneously.

From an intuitive standpoint, a decrease in the proximity of the inequalities signifies a more informative system of inequalities. However, to judge whether inequality can be contracted and by how much without knowing the secret key is not easy. Inspired by the BP algorithm, we discover a metric for this judgment and propose a contraction strategy that can further contract inequalities at the algorithmic level.

4.1 Metric for contracting inequalities

We find that the potential for inequality to undergo contraction can be discerned through certain metric assessments during BP iterations. We begin by giving an intuitive analysis of a single variable in a single inequality in the BP algorithm. In an iteration, for each inequality, the distribution of $z[j]$ - partial sum (linear combinations) of variables excluding the current variable $x[j]$, is firstly calculated as in (2). It is a basis for determining the distribution of $x[j]$. Assuming that at some point the true values of the other variables have come to the fore, then the true value of the partial sum has also come to the fore to some degree, and the probability in the distribution $z[j]$ is higher around this true value. As the value of $x[j]$ traverses through all the possible values that could be taken, the upper and lower bounds on the inequality determine which segment of the probability in the distribution $z[j]$ is selected for summation in (10). Referring to Figure 7(a), if the true value of the partial sum is far from the boundary of the inequality, then high probabilities around the true value are included in each summation for each possible value taken by $x[j]$, making the probabilities under different values taken by $x[j]$ less distinguishable and more like a uniform distribution. In this case, there is room for the boundaries of inequality to contract. Conversely, if the true value of the partial sum is close to the inequality boundary as is shown in Figure 7(b), it will result in high probabilities being included in only some of the summations for the values taken by $x[j]$. The probabilities of $x[j]$ taking different values in this case are highly differentiated and more unlike the uniform distribution. In this case, the inequality boundaries do not contract. The equations with true values close to the boundary already provide a high level of information.

Now we need a metric to distinguish the above two cases. To improve the SNR of the distributions of $x[j]$ of the above two cases, we dismiss the common segments in the summation and only extract the *marginal distribution*. Formally, with coefficient $A[i][j]$, the marginal distribution of $x[j]$, say $P_j^{(i)}$, is

$$\begin{aligned} P_j^{(i)} : \forall \epsilon \in [-\eta_1, \eta_1], Pr[x[j] = \epsilon \text{ in the margin}] = \sum_{t \in M} \Pr[z[j] = t], \\ \text{where } M = [c[i] - A[j] \cdot \epsilon, c[i] + |A[j]| \cdot \eta_1 - 1] \cup [b[i] - |A[i][j]| \cdot \eta_1 + 1, b[i] - A[i][j] \cdot \epsilon], \end{aligned} \tag{11}$$

for $j \in [0, \psi-1]$ and $i \in [0, w-1]$. Note that each summation only sums over $2 \cdot A[i][j] \cdot \eta$ elements.

We use Jensen-Shannon distance (JSD) to measure the difference between the marginal distribution $P_j^{(i)}$ and uniform distribution $\mathcal{U}$, i.e.,

$$\mathrm{JSD}(P_j^{(i)}, \mathcal{U}) = \sqrt{H(\frac{1}{2}(P_j^{(i)} + \mathcal{U})) - \frac{1}{2}(H(P_j^{(i)}) + H(\mathcal{U}))},$$

for $j \in [0, \psi-1], i \in [0, w-1]$, where $H(P) = -\sum_{x \in \chi} p(x) \log p(x)$ is the Shannon entropy for distribution P. If the JSD regarding a variable in an inequality is lower, it is a hint that the current inequality may be contracted.

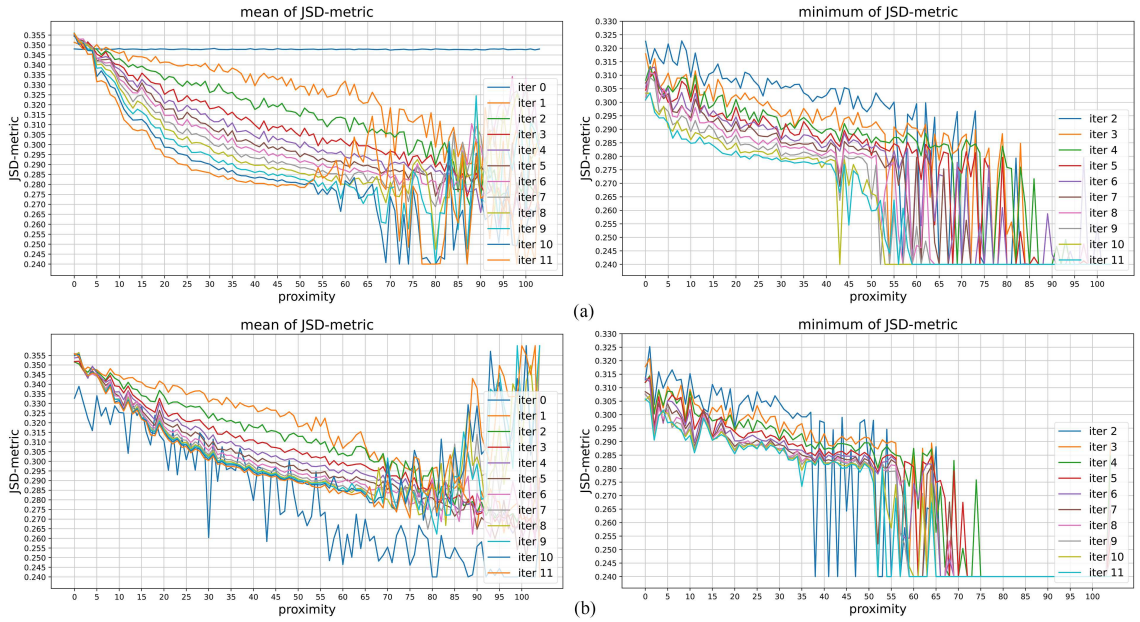

Figure 8 Mean and minimum values of JSD-metrics in each proximity class of filtered and unfiltered inequalities of Kyber512. (a) For 5200 filtered inequalities; (b) for 8000 unfiltered inequalities.

An inequality is to give an update of the distributions for all $2nk$ variables. In order to synthesize the JSD regarding all the variables, using Kyber512 as an example where coefficients of the inequalities only take $[-4, 4]$, we classify the variables into 9 categories by their coefficients in the inequality, compute the mean of the JSDs of the variables within each category, and then take the mean again for categories with coefficients $[-3,-2,-1,1,2,3]$ [1)] as a metric for each inequality. For each inequality $i \in [0, w-1]$,

$$\text{JSD-metric}[i] = \frac{\sum_{a\in\{-3,-2,-1,1,2,3\}} \mathsf{mean}(\{\text{JSD}(P_j^{(i)}, \mathcal{U}) | A[i][j] = a\})}{6}. \tag{12}$$

We use the term JSD-metric to represent this metric on individual inequalities to distinguish from the definition on individual variables.

Illustration. To show the effectiveness of the JSD-metric in reflecting the tightness of equations, we take 5200 inequalities deduced by filtered ciphertexts and 8000 inequalities deduced by unfiltered ciphertexts for Kyber512 as an example. In Figure 8, we classify all inequalities by their proximity and calculate the mean and minimum values of JSD-metrics regarding all inequalities in each class. We clip the data to be above 0.24. It can be seen that the average JSD-metric decreases as proximity increases. Besides, from the lines of the minimum of JSD-metrics, taking the line of the 11th iteration for 5200 filtered inequalities for example, we know that if the JSD-metric in the 11th iteration of an inequality is below 0.275, the proximity of this inequality is by no means smaller than 40. Note that the JSD-metric sequence of each inequality is calculated without knowing the secret key and thus can be calculated entirely by the attacker. Suppose the relation reflected in Figure 8 is so strong that the proximity of an inequality can be predicted with a given sequence of JSD-metric calculated in each iteration (as will be proved in the next subsection), the upper bound and the lower bound of an inequality can be increased and decreased by an amount equal to its proximity value without introducing errors. This motivates us to draw a contraction strategy utilizing the JSD-metric sequences.

Since the proximity is calculated according to the known key and bounds of the inequality, the contraction strategy can be divided into the profiling phase and the attacking phase. In the profiling phase, the attacker captures the relation between the JSD-metric sequence and the proximity of inequalities. In the attacking phase, the attacker calculates JSD-metric sequences of inequalities generated under an unknown key and predicts the proximity of inequalities. Then the inequalities can be contracted so that the entropy carried by them is improved.

1) We omit ± 4-coefficient classes since there are very few such coefficients. 0-coefficient class provides trivial JSD value.

Table 7 Numbers of contracted inequalities

Contraction amount	Threshold	Among 5200 filtered		Among 8000 unfiltered	
		#Correct	#Error	#Correct	#Error
50	0.5	1126	36	2438	24
	0.6	1017	15	2089	11
	0.7	918	7	1754	4
	0.8	769	1	1539	1
	0.9	594	0	1221	1
30	0.5	2261	105	4299	46
	0.6	2115	53	4064	30
	0.7	1974	27	3746	17
	0.8	1768	14	3394	6
	0.9	1482	1	2918	1

4.2 Machine learning-based contraction strategy

The correlation between JSD-metric and proximity reflected in Figure 8 is not easy to be explicitly expressed mathematically. The problem of obtaining proximity from JSD-metric sequences of an inequality is a classification problem, and a natural idea is to use machine learning to solve this classification problem hoping to capture this relationship better than mankind. Once the proximity of an inequality is determined, this means that the upper and lower bounds of the inequality can be contracted by this amount.

In the machine learning-based contraction strategy, to acquire a higher accuracy of classification, we build a binary classification model with proximity$\leqslant m$ labeled by 0 and proximity$> m$ by 1 (choose $m = 50$ and 30 as examples). Once the model has been trained, to introduce error inequalities as few as possible, the bounds of an inequality can be contracted by m only if the predicted probability is higher than an increased threshold. In the following, we give details of the machine learning experiments for both filtered and unfiltered inequalities.

4.2.1 *Training data and test data*

For filtered inequalities, the target is to perform contractions on chosen inequalities from a total of 5200. Running BP algorithm with 50 steps on 5200 inequalities, a JSD-metric sequence of length 50 is collected for each inequality as training samples. The JSD-metric sequences whose corresponding inequalities are with proximity smaller than and equal to m are labeled 0, and those with proximity larger than m are labeled 1. Seven groups of such samples are generated and a total of $7 \times 5200 = 3.64 \times 10^4$ training data are prepared. Then with another key, a group of 5200 inequalities is generated randomly and run with BP algorithm to produce the JSD-metric sequences as test data.

For unfiltered inequalities, the data set is similar. Ten groups of 8000 inequalities are generated randomly each corresponding to a random secret key. 8×10^4 samples are used as training data and another group is used as test data.

4.2.2 *Model, accuracy and prediction*

For both filtered and unfiltered inequalities, we train a random forest model. The number of trees is set to 300 and the max depth is set to 30. After training, the out-of-bag accuracies are 91.72%, 92.04% for $m = 50$ and 89.81%, 92.11% for $m = 30$. When applying to test data, the classification accuracies are 88.06%, 78.39% for $m = 50$ and 85.69%, 82.31% for $m = 30$. To reduce the number of erroneous inequalities, we can require that only samples whose predicted classification probability is larger than an improved threshold can be contracted. Table 7 shows some thresholds for contraction and the number of correctly and erroneously contracted inequalities. For example, by using the trained model for 5200 filtered inequalities with $m = 50$ and setting the classification probability threshold to 0.9, among 5200 filtered inequalities, 594 can be contracted by an amount of 50 (the interval decreases by 100) without introducing erroneous inequalities.

Our machine learning model is trained specifically for the following setups: (1) a specific Kyber version, (2) the inequality collection method described in Section 3, (3) a specific number of inequalities (usually with a success probability lower than 1), (4) a specific number of iterations in the BP algorithm that

determines the length of the JSD-metric, and (5) a specific classification definition of the proximities (e.g., in the binary classification problem, the threshold for labels 0 and 1). As long as these settings do not change, the machine learning model does not need to be retrained.

4.3 Discussion

Although the above methods can reduce the proximity of the inequalities, we are still missing the last piece of the puzzle in improving the number of unknowns recovered by the BP algorithm, that is, the number of unknowns recovered by the BP algorithm and the overall tightness of the inequalities are not strictly monotonically related. That is, there may be a case where the proximity of inequalities decreases thus the system becomes more informative, but the number of correctly recovered unknowns does not improve. As is mentioned by Hermelink et al. [12], the BP algorithm as a decoding method is not ideal and redundancy in entropy is required. When the contraction of inequality intervals can not improve the entropy by a significant amount, the result of the BP algorithm can not be improved. Nevertheless, the JSD-metric and contraction strategies present a possibility to further improve the entropy carried by inequalities from the algorithmic level and can be employed in any attack endeavoring to establish a system of inequalities concerning secret variables.

Open problems. First, the current random forest model is used to solve a binary classification problem to ensure reliability. A multi-classification problem is also possible by dividing the proximity range (104 for Kyber512) into several classes. What is the optimal way for classification is a problem. Secondly, in the attack phase, the classification probability threshold used for contraction can be varied to strike a balance between entropy improvement and the number of errors. The upper bound of the number of error inequalities that do not affect the overall performance of the BP algorithm remains unclear. Third, the contraction strategy can be repeated on refined inequality systems until no further contraction can be performed. We believe that multiple applications of this contraction strategy can significantly improve the entropy carried by the system and thus have the potential to further reduce the number of required inequalities. Fourth, the hyperparameters in the machine learning model may have better options to achieve high accuracy. We leave all these problems for future studies.

5 Conclusion

In this paper, we focus on the implementation security of the post-quantum cryptographic algorithm Kyber, and present a method for constructing bilateral inequalities about the secret variables carrying higher Shannon entropy using the side-channel approach in the CCA attack. This is the first time that systems of bilateral inequalities are practically solved. The number of inequalities needed for recovering the secret key with probability 1 is substantially lower than that of unilateral inequalities built from fault-injection approaches previously, as well as the query complexity. We also proposed contraction strategies to further reduce the interval of inequalities to lead to a more informative system of inequalities.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2022YFB3103800), National Natural Science Foundation of China (Grant No. 62102025), Beijing Natural Science Foundation (Grant No. 4222035), and Open Project Program of the State Key Laboratory of Cryptology (Grant No. MMKFKT202212). We thank the anonymous reviewers for their helpful comments.

References

1 National Institute of Standards and Technology (US). Module-lattice-based key-encapsulation mechanism standard. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS), 2024, NIST FIPS 203. https://doi.org/10.6028/NIST.FIPS.203
2 National Institute of Standards and Technology. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization
3 Fujisaki E, Okamoto T. Secure integration of asymmetric and symmetric encryption schemes. J Cryptol, 2013, 26: 80–101
4 Ravi P, Sinha Roy S, Chattopadhyay A, et al. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. IACR Trans Cryptogr Hardw Embed Syst, 2020, 3: 307–335
5 Qin Y, Cheng C, Zhang X, et al. A systematic approach and analysis of key mismatch attacks on lattice-based NIST candidate KEMs. In: Proceedings of Advances in Cryptology–ASIACRYPT 2021, Cham: Springer International Publishing, 2021. 92–121
6 Shen M, Cheng C, Zhang X, et al. 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
7 D'Anvers J P, Heinz D, Pessl P, et al. Higher-order masked ciphertext comparison for lattice-based cryptography. IACR Trans Cryptogr Hardw Embed Syst, 2022, 2: 115–139
8 Rajendran G, Ravi P, D'Anvers J P, et al. 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

9 Tanaka Y, Ueno R, Xagawa K, et al. Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. IACR Trans Cryptogr Hardw Embed Syst, 2023, 3: 473–503
10 Xu Z, Pemberton O, Roy S S, et al. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: the case study of Kyber. IEEE Trans Comput, 2022, 71: 2163–2176
11 Ravi P, Bhasin S, Roy S S, et al. On exploiting message leakage in (few) NIST PQC candidates for practical message recovery attacks. IEEE Trans Inform Forensic Secur, 2022, 17: 684–699
12 Hermelink J, Møartensson E, Samardjiska S, et al. Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. IACR Trans Cryptogr Hardw Embed Syst, 2023, 4: 287–317
13 Ravi P, Ezerman M F, Bhasin S, et al. Will you cross the threshold for me? IACR Trans Cryptogr Hardw Embed Syst, 2022, 1: 722–761
14 Ravi P, Chattopadhyay A, D'Anvers J P, et al. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (Kyber, Dilithium): survey and new results. ACM Trans Embed Comput Syst, 2024, 23: 1–54
15 D'Anvers J P, Tiepelt M, Vercauteren F, et al. Timing attacks on error correcting codes in post-quantum schemes. In: Proceedings of Proceedings of ACM Workshop on Theory of Implementation Security Workshop, New York: Association for Computing Machinery, 2019. 2–9
16 Băetu C, Durak F B, Huguenin-Dumittan L, et al. Misuse attacks on post-quantum cryptosystems. In: Proceedings of Advances in Cryptology–EUROCRYPT 2019, Cham: Springer, 2019. 747–776
17 Ueno R, Xagawa K, Tanaka Y, et al. Curse of re-encryption: a generic power/EM analysis on post-quantum KEMs. IACR Trans Cryptogr Hardw Embed Syst, 2022, 1: 296–322
18 Guo Q, Johansson T, Nilsson A. A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. In: Proceedings of Advances in Cryptology – CRYPTO 2020, Cham: Springer, 2020. 359–386
19 Bhasin S, D'Anvers J P, Heinz D, et al. Attacking and defending masked polynomial comparison for lattice-based cryptography. IACR Trans Cryptogr Hardw Embed Syst, 2021, 3: 334–359
20 Ngo K, Dubrova E, Guo Q, et al. A side-channel attack on a masked IND-CCA secure Saber KEM implementation. IACR Trans Cryptogr Hardw Embed Syst, 2021, 4: 676–707
21 Bache F, Paglialonga C, Oder T, et al. High-speed masking for polynomial comparison in lattice-based KEMs. IACR Trans Cryptogr Hardw Embed Syst, 2020, 3: 483–507
22 Oder T, Schneider T, Pöppelmann T, et al. Practical CCA2-secure and masked ring-LWE implementation. IACR Trans Cryptogr Hardw Embed Syst, 2018, 1: 142–174
23 Mondal P, Kundu S, Bhattacharya S, et al. A practical key-recovery attack on LWE-based key-encapsulation mechanism schemes using rowhammer. In: Proceedings of the Applied Cryptography and Network Security - ACNS 2024, 2024. 271–300
24 Pessl P, Prokop L. Fault attacks on CCA-secure lattice KEMs. IACR Trans Cryptogr Hardw Embed Syst, 2021, 2: 37–60
25 Valencia F, Oder T, Güneysu T, et al. Exploring the vulnerability of R-LWE encryption to fault attacks. In: Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems, 2018. 7–12
26 Clavier C. Secret external encodings do not prevent transient fault analysis. In: Proceedings of the 9th International Workshop on, Cryptographic Hardware and Embedded Systems-CHES 2007, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. 181–194
27 Hermelink J, Pessl P, Pöppelmann T. Fault-enabled chosen-ciphertext attacks on Kyber. In: Proceedings of Progress in Cryptology – INDOCRYPT 2021, Cham: Springer, 2021. 311–334
28 Delvaux J. Roulette: a diverse family of feasible fault attacks on masked Kyber. IACR Trans Cryptogr Hardw Embed Syst, 2022, 4: 637–660
29 Kundu S, Chowdhury S, Saha S, et al. Carry your fault: a fault propagation attack on side-channel protected LWE-based KEM. IACR Trans Cryptogr Hardw Embed Syst, 2024, 2: 844–869
30 Dachman-Soled D, Ducas L, Gong H, et al. LWE with side information: attacks and concrete security estimation. In: Proceedings of Advances in Cryptology – CRYPTO 2020, Cham: Springer, 2020. 329–358
31 Guo Q, Grosso V, Standaert F X, et al. Modeling soft analytical side-channel attacks from a coding theory viewpoint. IACR Trans Cryptogr Hardw Embed Syst, 2020, 4: 209–238
32 Langlois A, Stehlé D. Worst-case to average-case reductions for module lattices. Des Codes Cryptogr, 2015, 75: 565–599

Appendix A Query factors for KYBER768 and KYBER1024

For KYBER768, the interval of an inequality is the same as KYBER512 and d follows a Gaussian distribution with a standard deviation of about $\sigma = 76$. The parameters in query tricks are the same as KYBER512, shown in Table A1. The expected number of queries to deduce an inequality is $2 \times 0.4957 + 3 \times 0.4957 + 4 \times 0.0042 + 5 \times 0.0042 = 2.5163$.

For KYBER1024, the most frequent values of N are 7 and 8, and a few are 6 and 9. According to our empirical simulation, d follows a Gaussian distribution with a standard deviation of about $\sigma = 58$. The expected number of queries to deduce an inequality is $2 \times 0.4812 + 3 \times 0.4812 + 4 \times 0.0187 + 5 \times 0.0187 = 2.5743$. The parameters are shown in Table A2.

Table A1 Parameters used in the query trick for KYBER768

N' list	#Queries	Probability	Range of d	success(T)=true	N_T
$N'[0] = 3$ $N'[1] = 4$	2	0.4957	$(0, q/16)$	$T = [1, 0]$	3
$N'[2] = 5$	3	0.4957	$(-q/16, 0)$	$T = [1, 1, 0]$	4
$N'[3] = 2$	4	0.0042	$(q/16, q/8)$	$T = [0, 0, 0, 1]$	2
$N'[4] = 6$	5	0.0042	$(-q/8, -q/16)$	$T = [1, 1, 1, 1, 0]$	5
$N'[5] = 1$	6	6.9182×10^{-8}	$(q/8, 3q/16)$	$T = [0, 0, 0, 0, 0, 1]$	1
$N'[6] = 7$	7	6.9182×10^{-8}	$(-3q/16, -q/8)$	$T = [1, 1, 1, 1, 1, 1, 0]$	6

Table A2 Parameters used in the query trick for Kyber1024

N' list	#Queries	Probability	Range of d	success(T)=true	N_T
$N'[0]=7$ $N'[1]=8$	2	0.4812	$(0, q/32)$	$T=[1,0]$	7
$N'[2]=9$	3	0.4812	$(-q/32, 0)$	$T=[1,1,0]$	8
$N'[3]=6$	4	0.0187	$(q/32, q/16)$	$T=[0,0,0,1]$	6
$N'[4]=10$	5	0.0187	$(-q/16, -q/32)$	$T=[1,1,1,1,0]$	9
$N'[5]=5$	6	1.5825×10^{-8}	$(q/16, 3q/32)$	$T=[0,0,0,0,0,1]$	5
$N'[6]=11$	7	1.5825×10^{-8}	$(-3q/32, -q/16)$	$T=[1,1,1,1,1,1,0]$	10
$N'[7]=4$	8	2.1617×10^{-10}	$(3q/32, q/8)$	$T=[0,0,0,0,0,0,0,1]$	4
$N'[8]=12$	9	2.1617×10^{-10}	$(-q/8, -3q/32)$	$T=[1,1,1,1,1,1,1,1,0]$	11