A Closer Look at the Belief Propagation Algorithm in Side-Channel-Assisted Chosen-Ciphertext Attacks

Kexin Qiao[1], Siwe Sun[2], Zhaoyang Wang[1], Zehan Wu[1], Junjie Cheng[1], An Wang[1], and Liehuang Zhu[1]

[1] School of Cyberspace Science and Technolgy, Beijing Institute of Technology, Beijing 100081, China
[2] School of Cryptology, University of Chinese Academy of Sciences, Beijing 100089, China
{qiao.kexin, wanganl, liehuangz}@bit.edu.cn, sunsiwei@ucas.ac.cn

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 KYBER512 as a study case to construct bilateral inequalities on key variables with substantially narrower intervals using a side-channel-assisted oracle. The number of such inequalities required to recover the key with probability 1 utilizing the BP algorithm is reduced relative to previous unilateral inequalities. Furthermore, we introduce strategies aimed at further refining the interval of inequalities. Diving into the BP algorithm, we discover a measure metric named JSD-metric that can gauge the tightness of an inequality. We then develop a heuristic strategy and a machine learning-based strategy to utilize the JSD-metrics to contract boundaries of inequalities even with fewer inequalities given, thus improving the information 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· CCA · belief propagation · contraction strategy · machine learning.

1 Introduction

CRYSTALS-KYBER[ABD+21] is selected by the US National Institute of Standards and Technology (NIST) [Nat] 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 nears completion, ensuring the implementation security of PQC algorithms is a top priority in the cryptographic community.

The concretization of chosen ciphertext attack (CCA) security in KYBER's KEM variant is realized through the application of the Fujisaki-Okamoto (FO) transformation [FO13] during the decapsulation process, 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 instances. However, recent investigations 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 secret keys in KYBERare the coefficients of polynomials, which are taken from $\mathbb{Z}_q, q = 3329$. The number of such secret values varies from 512 to 1024 for different versions.

This paper aims at constructing linear inequalities on secret key components taken from $\mathbb{Z}_{3329}$ with narrower intervals from side-channel-based *decryption-failure* oracle. Additionally, the research endeavors to progressively refine these intervals during the solution procedure, thereby augmenting the informational content derived from the inequalities. We use KYBER512 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 [REB+21,RCDB22]. Major oracles employed in

side-channel-assisted CCA attacks on lattice-based KEMs include *plaintext-checking* oracle [RSRCB20,SCZ+22,DTVV19,BDHD+19,UXT+21,RRD+23,TUX+23], *decryption-failure* oracle [GJN20,BDH+21,DHP+22] and *full-decryption* oracle [XPR+22,RBRC22,NDGJ21]. These oracles can be constructed even for protected implementations (e.g. Bhasin *et al.* 's decryption-failure oracle [BDH+21] for protected implementation of ciphertext comparison [BPO+20,OSPG18]). However, maliciously chosen ciphertexts used for plaintext-checking oracle and full-decryption oracle attacks are very sparse with several zero coefficients and thus easy to be detected by ciphertext sanity check [RCDB22]. Decryption-failure oracle can also be constructed by fault-injection approaches [PP21,VOGR18,Cla07,HPP21,Del22], which are generally perceived as having a higher cost. Therefore, we will utilize the decryption-failure oracle build with side-channel approaches in this work.

Solving system of inequalities. For KYBER, though inequalities within interval $q/2^{d_v}$ have already been constructed by side-channel approaches in [BDH+21] (also mentioned in [DHP+22]), they only approximated the inequalities by equations and fed to the LWE framework [DSDGR20] 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} [BDH+21, Figure 4a]. So such kind of system of inequalities has not been practically solved yet. Pessl *et al.* [PP21] developed a belief-propagation technique to solve for secret variables practically. Hermelink *et al.* [HPP21] 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.* [DHP+22]. Delvaux [Del22] calculated the large number of summation distributions in the BP algorithm according to the central limit theorem (CLT) instead of Fast Fourier Transformation (FFT). Recently, Hermelink *et al.* [HMS+23] integrated the BP processed information in lattice reduction algorithms to make use of the advantages of both statistic and algebraic approaches. Qin *et al.* [QCZ+21] investigated the lower bound on the number of inequalities needed to recover the key. Currently, all practically solved inequalities are unilateral inequalities.

1.2 Contributions

Assuming the presence of a decryption-failure oracle constructed through side-channel methodologies (as exemplified in [BDH+21,DHP+22]), 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. Subsequently, we employ the BP algorithm to effectively address and resolve these bilateral inequalities. The approximate number of inequalities needed to recover the key with success rate 1 is reduced from state-of-the-art 9500 to 8600, and the query complexity is reduced from 9500×4 to 8600×2.5 assuming reasonably perfect reliability.

Moreover, we introduce strategies 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. When the measure is lower than a threshold the inequality can be contracted by an amount. Then we develop two contraction strategies to improve the information carried by the system of inequalities. The first one is a heuristic contraction strategy in which thresholds for the JSD-metric along with the corresponding contraction amount are enumerated to maximize the gain. The second one is a machine-learning-based strategy where the JSD-metrics and known proximities are used as the training set for a random forest model, that will predict the proximities for JSD-metrics collected under an unknown key. Note that these contraction strategies can be applied to a system with fewer inequalities, having the potential to solve the secret variables with lower query complexity.

2 Preliminary

In this section, we first give notations and then describe KYBER CCA-secure KEM, followed by the BP algorithm in solving a system of inequalities.

Table 1: Notations

Notation	Description
$a\|\|b$	concatenation of a and b
R_q	the ring $\mathbb{Z}_q[x]/(x^n+1)$,
	n, q fixed to $n = 256, q = 3329$ through this paper
regular font letters v, e	element in R_q
bold letters e.g. $\mathbf{v}$ (or $\mathbf{A}$)	column vector (or matrix) with components in R_q
$\mathbf{v}^T$ (or $\mathbf{A}^T$)	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.* $r' = r \mod q$
$r \bmod^{+} q$	positive modulo $r' \in [0, q)$ *s.t.* $r' = r \mod q$
$r \bmod^{*} q$ (r^* for short)	biased modulo $r' \in (-\frac{q}{4}, \frac{3q}{4})$ *s.t.* $r' = r \mod q$
$\lceil x \rfloor$	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 random seed
B_η	a centered binomial distribution:
	$\sum_{i=1}^{\eta}(a_i - b_i)$ where $(a_1, \cdots, a_\eta, b_1, \cdots, b_\eta) \leftarrow \{0,1\}^{2\eta}$
$G(\cdot), H(\cdot)$	hash function instantiated with SHA3-256 and SHA3-512
$KDF(\cdot)$	a key-derivation function

2.1 Notations

When we write that a polynomial $f \in R_q$ or a vector of such polynomials is sampled from B_η, we mean that each coefficient is sampled from B_η. Vectors of polynomials sometimes are serialized to byte arrays in a deterministic and straightforward way implicitly and vise verser.

Coefficients conversion. Multiplication on $R_q = \mathbb{Z}_q[x]/(x^n+1)$ a.k.a. modular multiplication of polynomials on $\mathbb{Z}_q$ can be converted to matrix-vector product on $\mathbb{Z}_q$. Define the conversion function

$$\begin{aligned} &Conv : \mathbb{Z}^n \to \mathbb{Z}^{n\times n} \\ &\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \mapsto \begin{pmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix} \end{aligned} \tag{1}$$

that applies on an element in R_q. For $a = (a_0, \cdots, a_{n-1})^T = \sum_{i=0}^{n-1} a_i x^i \in R_q, b = (b_0, \cdots, b_{n-1})^T = \sum_{i=0}^{n-1} b_i x^i \in R_q$ and $c = (c_0, \cdots, c_{n-1})^T = a \cdot b = \sum_{i=0}^{n-1} c_i x^i \in R_q$ where $a_i, b_i, c_i \in \mathbb{Z}_q, i = 0, \cdots, n-1$, we have

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} = Conv(a) \cdot \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \mod q,$$

where the j-th column of the conversion matrix is the coefficients of polynomial ax^j defined on R_q. We denote the i-th row of the matrix $Conv(a)$ by $Conv(a)[i]$.

Compression and Decompression. Function Comp_q and Decomp_q take in $x \in \mathbb{Z}_q$ and positive integer d and produce an integer:

$$\begin{aligned} \mathsf{Comp}_q(x, d) &= \lceil (2^d/q) \cdot x \rfloor \bmod^{+} 2^d, \\ \mathsf{Decomp}_q(x, d) &= \lceil (q/2^d) \cdot x \rfloor. \end{aligned}$$

When Comp_q or Decomp_q is used with $x \in R_q$ or $\mathbf{x} \in R_q^k$, the procedure is applied to each coefficient individually. When input d equals 1, we use $\mathsf{Encode}(\cdot)$ and $\mathsf{Decode}(\cdot)$ to represent the conversion from a polynomial in R_q to a bitstream of length n and vice-versa.

2.2 Description of KYBER.CCAKEM

KYBER[ABD+21] is an IND-CCA2-secure KEM that relies on the hardness of Module Learning with Errors (MLWE) problem [LS15]. 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 2. The parameters are shown in Table 2.

Table 2: Parameter sets for KYBER

	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}

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 with her public key to get ciphertext c'. The key is established if c' and the received c 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 26 in Figure 2) is

$$d = \mathbf{e}^T\mathbf{r} - \mathbf{s}^T(\Delta_{\mathbf{u}} + \mathbf{e}_1) + e_2 + \Delta_v. \tag{2}$$

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 $v'' - \mathbf{s}^T\mathbf{u}''$ is still within the hemisphere around the original value of m as is shown in Figure 1.

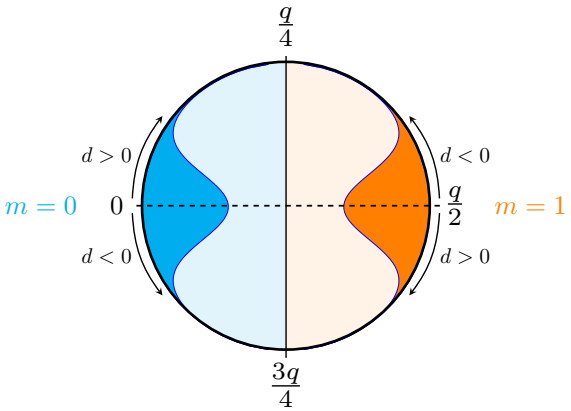


Fig. 1: Decoding of m with noise d.

2.3 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.* [PP21,HPP21]. It

Alice **Bob**

KYBER.CCAKEM.KeyGen()

1: $z \leftarrow \{0,1\}^{256}$
// CPA-secure KeyGen: ln 2-8
2: $\rho \leftarrow \{0,1\}^{256}$
3: $\mathbf{A} \in R_q^{k \times k}$ determined by ρ
4: $\mathbf{s} \leftarrow B_{\eta_1}$ ▷ $\mathbf{s} \in R_q^k$
5: $\mathbf{e} \leftarrow B_{\eta_1}$ ▷ $\mathbf{e} \in R_q^k$
6: $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \in R_q^k$
7: $pk = \mathbf{t} \parallel \rho$
8: $sk' = \mathbf{s}$
9: $sk = (sk'||pk||H(pk)||z)$
10: **return** (pk, sk)

$pk \longrightarrow$

KYBER.CCAKEM.Enc(pk)

11: $m \leftarrow \{0,1\}^{256}$
12: $m \leftarrow H(m)$
13: $(\bar{K}, r) := G(m||H(pk))$
// CPA-secure Enc: ln 14-20
14: $\mathbf{A} \in R_q^{k \times k}$ recovered from pk
15: $(\mathbf{r}, \mathbf{e}_1, e_2) \xleftarrow{r} (B_{\eta_1}, B_{\eta_2}, B_{\eta_2})$
▷ $(\mathbf{r}, \mathbf{e}_1, e_2) \in R_q^k \times R_q^k \times R_q$
16: $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \in R_q^k$
17: $v = \mathbf{t}^T\mathbf{r} + e_2 + \mathsf{Decode}(m)$
18: $c_1 = \mathsf{Comp}_q(\mathbf{u}, d_u) \triangleq \mathbf{u}'$
19: $c_2 = \mathsf{Comp}_q(v, d_v) \triangleq v'$
20: $c = c_1||c_2$
21: $K = KDF(\bar{K}||H(c))$
22: **return** (c, K)

$c \longleftarrow$

KYBER.CCAKEM.Dec(c, sk)

23: Recover $\mathbf{s}, pk, H(pk), z$ from sk
// CPA-secure Dec: ln 24-26
24: $\mathbf{u}'' = \mathsf{Decomp}_q(c_1, d_u)$
25: $v'' = \mathsf{Decomp}_q(c_2, d_v)$
26: $m' = \mathsf{Encode}(v'' - \mathbf{s}^T\mathbf{u}'')$
27: $(\bar{K}', r') = G(m'||H(pk))$
// Re-encrytion by CPA-secure Enc: ln 28-33
28: $(\mathbf{r}', \mathbf{e}_1', e_2') \xleftarrow{r'} B_{\eta_1} \times B_{\eta_2} \times B_{\eta_2}$
▷$(\mathbf{r}', \mathbf{e}_1', e_2') \in R_q^k \times R_q^k \times R_q$
29: $\mathbf{u}_{re} = \mathbf{A}^T\mathbf{r}' + \mathbf{e}_1' \in R_q^k$
30: $v_{re} = \mathbf{t}^T\mathbf{r}' + e_2' + \mathsf{Decode}(m')$
31: $c_1' = \mathsf{Comp}_q(\mathbf{u}_{re}, d_u)$
32: $c_2' = \mathsf{Comp}_q(v_{re}, d_v)$
33: $c' = c_1'||c_2'$
34: **if** $c' = c$ **then**
35: $\quad K = KDF(\bar{K}'||H(c))$
36: **else**
37: $\quad K = KDF(z||H(c))$
38: **return** K

Fig. 2: KYBER.CCAKEM procedure

is an error-tolerant and efficient method to solve systems with possible errors. The w linear inequalities deduced in previous work [PP21,HPP21,Del22,DHP+22,HMS+23] 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.

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

$$z[j] = \sum_{j' \in [0,\psi-1]\setminus\{j\}} A[i,j']x[j'], \tag{3}$$

i.e. the linear combination of all variables except the target variable $x[j]$. This can be done by FFT [PP21,HPP21] or by the CLT [Del22]. Then enumerate all the guesses for $x[j]$. For each guess $\epsilon \in B_{\eta_1}$, calculate the probability that the i-th inequality is satisfied

$$\begin{aligned} &\Pr[x[j]=\epsilon \text{ regarding the } i\text{-th inequality}] \\ =&\Pr[\sum_{j' \in [0,\psi-1]\setminus\{j\}} A[i,j']x[j'] \gtrless b[i]-A[i,j]\cdot\epsilon] \\ =&\sum_{t \gtrless b[i]-A[i,j]\cdot\epsilon} Pr[z[j]=t]. \end{aligned} \tag{4}$$

Then the i-th check node returns the probability vector to the j-th variable node according to Equation (4). 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".

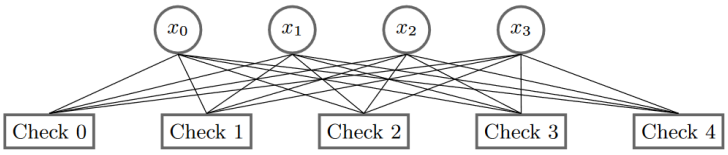


Fig. 3: A complete bipartite graph with four variable nodes and five check nodes corresponding to four variables and five inequalities [HPP21].

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 [HMS+23].

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 [BDH+21,DHP+22]. Such oracles predicts in the CCA attack wether decrypted plaintext m' (line 26 in Figure 2) from faulty

ciphetext during decapsulation equals the right plaintext m (line 12 in Figure 2). If they don't match, it is said that a decryption failure occurs. The leakage during equality checking operation (line 34 in Figure 2) or calculation of function G (line 27 in Figure 2) can all be utilized to construct such 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 Equation (2)). 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 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 Equation (2) to

$$d[j] = \mathbf{e}^T\mathbf{r}[j] - \mathbf{s}^T(\Delta^*_{\mathbf{u}} + \mathbf{e}_1)[j] + e_2[j] + \Delta^*_v[j], \tag{5}$$

so that all additions (and also subtractions) are performed on the integer ring, and the coefficients in multiplication with the unknown $\mathbf{e}$ and $\mathbf{s}$'s components can be converted to matrix on $\mathbb{Z}$ by the function $Conv(\cdot)$ defined in Equation (1). In this way, any hints on $d[j]$ will be converted to linear hints without modulo on secrete coefficients of $\mathbf{e}$ and $\mathbf{s}$.

To get hints on $d[j]$, after generating $c = (c_1||c_2) \triangleq (\mathbf{u}', v')$, Bob constructs faulty ciphertext $v'_{\texttt{fault}}$ 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 $d_{\texttt{fault}N} = d[j] + \Delta^*_N$, which means the faulty $d[j]$ increases by $q/2^{d_v} \times N$. When N is small, faulty $d_{\texttt{fault}N}$ 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 $d_{\texttt{fault}N}$ 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

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

which implies

$$(\lceil q/4 \rceil - \Delta^*_{N+1})^* \le d[j] \le (\lfloor q/4 \rfloor - \Delta^*_N)^*. \tag{6}$$

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

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

which implies

$$(\lceil 3q/4 \rceil - \lceil q/2 \rfloor - \Delta^*_{N+1})^* \leq d[j] \leq (\lfloor 3q/4 \rfloor - \lceil q/2 \rfloor - \Delta^*_N)^*. \tag{7}$$

The ranges of $d[j]$ in both Equation (6) and Equation (7) 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

$$B_l \leq d[j] \leq B_u, B_u - B_l = q/2^{dv}. \tag{8}$$

Combing Equation (8) and Equation (5), the inequality constructed on secrets $(\mathbf{e}, \mathbf{s})$ is

$$B_l - e_2[j] - \Delta^*_v[j] \leq \mathbf{e}^T\mathbf{r}[j] - \mathbf{s}^T(\Delta^*_{\mathbf{u}} + \mathbf{e}_1)[j] \leq B_u - e_2[j] - \Delta^*_v[j]. \tag{9}$$

Quering 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 (5)). According to an empirical simulation in [RCDB22], d follows a Gaussian distribution with a standard deviation $\sigma = 79$. 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 is $2 \times 0.4957 + 3 \times 0.4957 + 4 \times 0.0042 + 5 \times 0.0042 = 2.5163$. So the query complexity of the attack is estimated as 2.5 times the number of inequalities required. The attack procedure is depicted in Algorithm 1.

Table 3: Parameters used in the query trick.

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

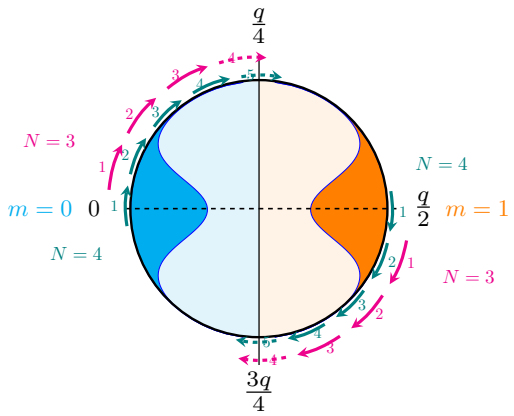


Fig. 4: Get a range of size $q/2^{d_v}$ ($q/16$ for KYBER512) for noise d.

Ciphertext filtering. Ciphertext filtering is an effective way to reduce the number of inequalities needed [PP21,HPP21,HMS+23]. The honest noise d obeys a $e_2 + \Delta_v$-centered distribution and when the secret

Algorithm 1: Inequalities generation procedure with decryption-failure oracle.

Input: list $N = [3, 4, 5, 2, 6, 1, 7]$, 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 // $s.t.\ \ c \leq Ax \leq b$

empty A, c, b ;

$n_{each} = w/256$ // collect n_{each} inequalities for each index

$j = 0, n_{collected} = 0$;

while $j < 256$ and $n_{collected} < n_{each}$ **do**

  $m \leftarrow \{0,1\}^{256}$;

  $(\mathbf{r}, \mathbf{e}_1, e_2) \leftarrow (B_{\eta_1}, B_{\eta_2}, B_{\eta_2})$;

  $v = \mathbf{t}^T\mathbf{r} + e_2 + \mathsf{Decode}(m, 1)$;

  $v' = \mathsf{Comp}_q(v, d_v)$;

  $\Delta_v^* = (\mathsf{Decomp}_q(v') - v) \bmod^* q$; // apply biased modular to remove possible extra q

  $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \in R_q^k$;

  $\mathbf{u}' = \mathsf{Comp}_q(\mathbf{u}, d_u)$;

  $\Delta_{\mathbf{u}}^* = (\mathsf{Decomp}_q(\mathbf{u}') - \mathbf{u}) \bmod^* q$; // apply biased modular to remove possible extra q

  $r =$ concatenation of $Conv(a)[j]$ 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 // $T[i] = \mathcal{B}(\mathbf{u}'||v'_{\mathtt{fault}})$

    $i++$;

  $N = N_T$;

  $\Delta_N^* = (\mathsf{Decomp}_q((v'[j] + N) \bmod^+ 2^{d_v}) - \mathsf{Decomp}_q(v'[j])) \bmod^* q$;

  $\Delta_{N+1}^* = (\mathsf{Decomp}_q((v'[j] + N + 1) \bmod^+ 2^{d_v}) - \mathsf{Decomp}_q(v'[j])) \bmod^* q$;

  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

    $B_l = (\lceil 3q/4 \rceil - \lceil q/2 \rfloor - \Delta_{N+1}^*) \bmod^* q$;

    $B_u = (\lfloor 3q/4 \rfloor - \lceil q/2 \rfloor - \Delta_N^*) \bmod^* q$;

  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**

    $n_{collected} = n_{collected} + 1$;

    $j = 0$;

return A, c, b

key changes it only oscillates within a small range. For the faulty ciphertexts, the center of $d_{\texttt{faultN}}$ 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 Equation (6) and Equation (7) is to constrain faulty $d_{\texttt{faultN}}$ in an interval between two adjacent division points and exclude $(\mathbf{e},\mathbf{s})$s that leaves $d_{\texttt{faultN}}$ out of this region. So averagely the closer the faulty $d_{\texttt{faultN}}$ 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 $d_{\texttt{faultN}}$ to division points. So to obtain more informative inequalities, Bob filters ciphertexts such that $|e_2+\Delta_v| \leq \epsilon$ to mount the attack. In Hermelink and Pessl *et al.* 's work for building unilateral inequalities [PP21,HPP21,HMS+23], 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 $|e_2[j]+\Delta_v^*[j]| \leq 10$. 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 [HPP21] to solve bilateral inequalities. The inequalities deduced from Algorithm 1 are of the form

$$\forall i \in [0, w-1], \quad c[i] \leq \sum_{j=0}^{\psi-1} A[i][j]x[j] \leq b[i]. \tag{10}$$

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 Equation (4) to

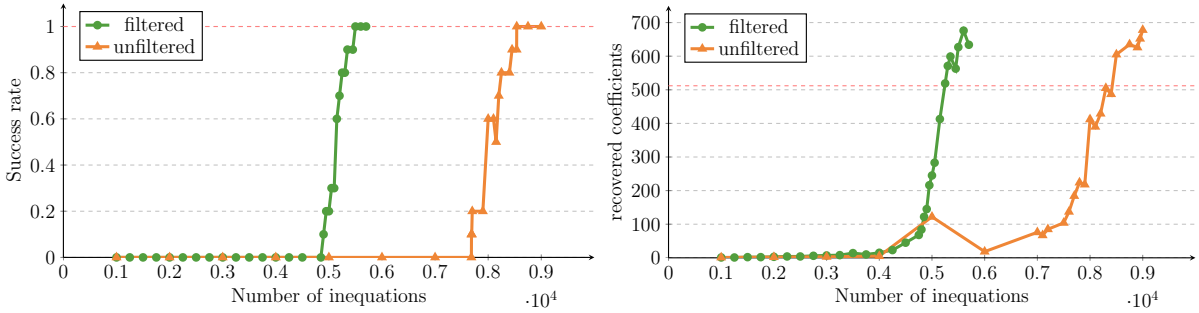
$$\begin{aligned} &\Pr[x[j] = \epsilon \text{ regarding the } i\text{-th inequality}] \\ =&\Pr[c[i] - A[i,j]\cdot\epsilon \leq \sum_{j' \in [0,\psi-1]\setminus\{j\}} A[i,j']x[j'] \leq b[i] - A[i,j]\cdot\epsilon] \\ =&\sum_{c[i]-A[i,j]\cdot\epsilon \leq t \leq b[i]-A[i,j]\cdot\epsilon} \Pr[z[j] = t]. \end{aligned} \tag{11}$$

The updation of the distribution in Equation (11) 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 [HMS+23] framework with the modified version described in section 3.3. For KYBER512, 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 as that in [HMS+23]. The approximate number of inequalities needed to recover the key with success rate 1 with filtered ciphertexts is about 5500, which is smaller than 6000 to 9000 for previous unilateral inequalities [HMS+23, Table 2]. In [HMS+23], 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 [Del22, Figure 4] and 9500 [HMS+23, Figure 8]. 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, the query factor is 2.5 assuming perfect side-chanel reliability as is indicated in [BDH+21,DHP+22]. The query factor of [PP21] is $1/0.17 \approx 5.88$ according to their practical implementation. The query factor of both [HPP21] and [HMS+23] are 4 assuming perfect fault injection reliability and the targeted value is boolean-masked. The number of fault injections of [Del22] increases by roughly one or two orders of magnitude compared to [HPP21], so we estimate the query complexity as 40 to get error free inequalities . The query complexity is the multiplication of the number of inequalities and the query factor.

The comparison is shown in Table 4. 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 [BDH+21, Figure 4a] with the same type of inequalities.



(a) Success rate

(b) Number of recovered coefficients

Fig. 5: Success rate and average number of recovered coefficients with respect to the number of bilateral inequalities for KYBER512.

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

Filtered		Unfiltered		QF	Solving Method	Ref.
#Eqs.	#Q.	#Eqs.	#Q.			
9000	3.6×10^5	13000	5.2×10^5	40	BP (CLT)	[Del22]
8000	4.704×10^4	-	-	5.88	BP (FFT v1)	[PP21]
6000	2.4×10^4	-	-	4	BP (FFT v2)	[HPP21]
5500	2.2×10^4	9500	3.8×10^4	4	BP (FFT v2) + BKZ	[HMS+23]
5500	1.375×10^4	8600	2.15×10^4	2.5	BP (FFT v2) + BKZ	This

#Eqs. : number of inequalities.
#Q. : number of queries.
QF: query factor.
BP (FFT v1): BP algorithm using FFT with clustering structure.
BP (FFT v2): BP algorithm using FFT with binary tree structure.

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 Equation (10) as

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

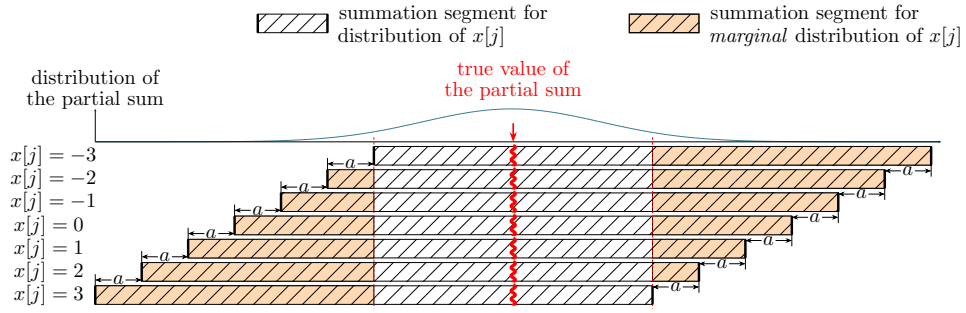
where $i \in [0, w-1]$ and $x[j]$s take the true values. In the rest of the paper, when we say that inequality can be *contracted* by amount m, it means that the upper bound of the inequality can be decreased by m and the lower bound of the inequality can be increased by m simultaneously.

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.

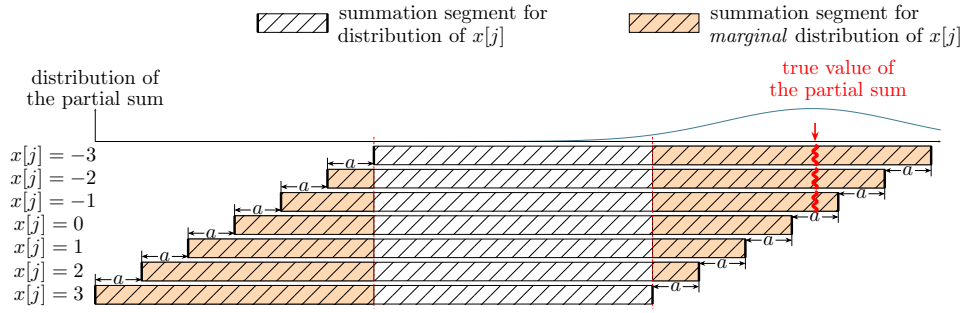
4.1 Metric for contracting inequalities

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 Equation (3). 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 Equation (11). Referring to Figure 6a, 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 6b, 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.



(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.

Fig. 6: Marginal distribution calculation with $x[j]$'s coefficient $A[i][j] \triangleq a > 0$.

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

$$P_j^{(i)} : \forall \epsilon \in [-\eta_1, \eta_1], Pr[x[j] = \epsilon \text{ in the margin}] = \sum_{t \in M} \Pr[z[j] = t], \tag{12}$$

where

$$M = [c[i] - A[j] \cdot \epsilon, c[i] + |A[j]| \cdot \eta_1 - 1] \cup [b[i] - |A[i][j]| \cdot \eta_1 + 1, b[i] - A[i][j] \cdot \epsilon],$$

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.* .

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

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.

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] [3] as a metric for each inequality. For each inequality $i \in [0, w-1]$,

$$JSD\text{-}metric[i] = \frac{\sum_{a\in\{-3,-2,-1,1,2,3\}} \mathsf{mean}(\{JSD(P_j^{(i)}, \mathcal{U}) | A[i][j] = a\})}{6}. \tag{13}$$

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

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 7, 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. The number of inequalities used here, *i.e.* , 5200 and 8000, is not exceptional. The statistical patterns of the JSD-metric for 4500 or more filtered inequalities and 8000 or more unfiltered inequalities are all similar to those in Figure 7. Note that the JSD-metric of each inequality is calculated without knowing the secret key and thus can be calculated entirely by the attacker. Suppose an appropriate threshold is chosen, for example, we chose 0.275 as a threshold in the 11-th iteration, for inequalities whose JSD-metric is lower than 0.275 in the 11-th iteration, the upper bound can be decreased and the lower bound be increased by 41 without introducing errors. This motivates us to draw a contraction strategy utilizing the JSD-metric.

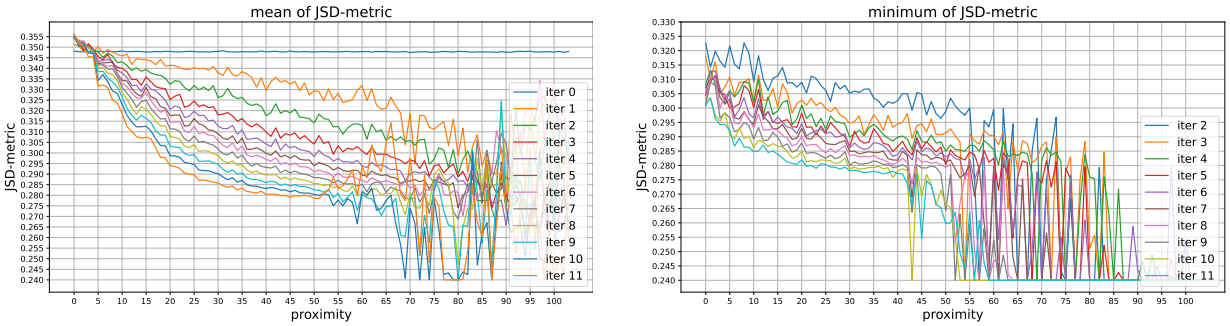
However, the proximity-related parameters 0.275 and 44 are derived with a known secret key since the proximity is calculated with the known secret key. So the parameters should be *trained* in the profiling phase with systems of inequalities under known keys, and then use the trained parameters for inequalities under an unknown key in attacking phase. The specific training procedure is discussed in the following.

4.2 Ad-hoc Contraction strategy

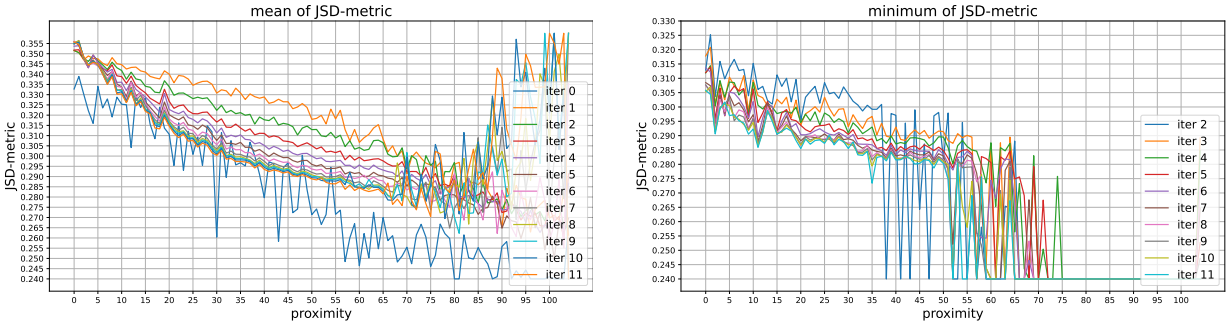
Overview of the ad-hoc contraction strategy. Given a system of inequalities, we first need to train the JSD-metric thresholds and contraction amounts suitable for this system so that the amount of information in this system can be maximized, *i.e.* , more equations contracted by more amounts as much as possible. This algorithm is given explicitly in Algorithm 2 below. We use the same algorithm to train the JSD-metric thresholds and contraction amounts for a large number of inequality systems, and then take their lower bound values as the final parameters to be used in the attacking phase. The minimum value is taken in order to try to make the newly generated inequalities under the unknown key not introduce errors when contracting accordingly. We assume that the distribution of JSD-metrics in each inequality class is some kind of normal distribution, and their expectation decreases with increasing proximity (Figure 8 confirms this assumption later). Taking a JSD-metric threshold that is on the small side will cause inequalities with JSD-metrics smaller than it to have greater proximity than the contraction amount, and thus the contracting does not tend to introduce errors.

Training parameters in profiling phase. Given a system of inequalities, we want to contract the boundaries by a fixed amount for inequalities whose JSD-metric is below a certain threshold. The overall contraction is defined as the product of the number of contracted equations and the amount of contraction. We use a heuristic enumeration method to find the parameters that maximize the overall contraction. Select the minimum JSD-metric value in the 0-proximity equation class as the upper bound, and then find the minimum JSD-metric of all inequalities as the lower bound. Traverse the threshold in small steps (0.002 in our algorithm) within that interval. With the threshold candidate fixed, the amount of

[3] We omit ± 4 coefficient classes since there are very few such coefficients. 0 coefficient class provide trivial JSD value.



(a) For 5200 filtered inequalities



(b) For 8000 unfiltered inequalities

Fig. 7: Mean and minimum values of JSD-metrics in each proximity class of filtered and unfiltered inequalities of KYBER512.

contraction is traversed from $[0, q/2^{d_v+1})$, which is the range of proximities of all inequalities. The number of inequalities to be contracted at a fixed threshold candidate are those whose JSD-metric is lower than the candidate. The larger the contraction amount the larger the gain, but it is also possible to introduce erroneous inequalities. Therefore when an error inequality is present, the gain is defined as the negative amount by which the true value deviates from the error bound. We take the contraction amount that maximizes the gain with the number of erroneous inequalities no more than a given threshold as the corresponding contraction amount of a fixed JSD-metric threshold candidate. It is possible to ensure that no error inequality is introduced by setting the error number threshold to 0. The threshold that maximizes the gain and its corresponding contraction amount is what we are looking for. The process of determining the JSD-metric threshold and the contraction amount is shown in Algorithm 2.

From the second iteration onwards ($i \geq 2$), we run a i-step BP algorithm and produce the JSD-metrics for all inequalities in the i-th iteration and acquire the JSD-metric threshold and contraction amount by Algorithm 2. Then repeat the $(i+1)$-step BP algorithm by applying contractions to the first i iterations according to parameters already trained, and produce the JSD-metrics for all inequalities in the $(i+1)$-th iteration and continue the process until no inequalities can be contracted.

Empirical settings. This algorithm has the following considerations and settings.

Firstly, in our experiments, the fact that contraction does not introduce errors is a first consideration. Although the BP algorithm is fault-tolerant, the presence of erroneous inequalities substentially increases the number of inequalities needed to recover the key [HMS+23]. Moreover, the side-channel approach already has the potential to introduce erroneous inequalities, and we do not want to continue to introduce errors at the algorithmic level.

Secondly, as the distribution of partial sum $z[j]$ is deduced by FFT, negative values occur at locations far from the center in the convolution series. Probabilities in the marginal distributions calculated in Equation (12) sometimes take negative values in the first few iterations while this does not happen in the calculation by Equation (11). Removing the common segment in summation (referring to Figure 6)

Algorithm 2: Determining JSD-metric threshold and contraction amount.

Input: secret key x (coefficients in $(\mathbf{e}, \mathbf{s})$), the coefficient matrix $A_{w\times\psi}$, lower bound vector c, upper bound vector b *s.t.* $c \le Ax \le b$, JSD-metric of all inequalities m_{JSD} in current iteration, bound for the number of error inequalities N_{error}

Output: Optimal threshold T_{opt}, optimal contracting amount C_{opt}

$T_{upper} = \min\{m_{JSD}[i] | Ax[i] = c[i] \text{ or } Ax[i] = d[i], i \in [0, w-1]\}$ // the minimum JSD of inequalities in 0-proximity class

$T_{lower} = \min\{m_{JSD}\}$;

$t_{step} = \min\{0.002, \frac{1}{2}(T_{upper} - T_{lower})\}$;

$T_{list} = [T_{lower} : t_{step} : T_{upper}]$ // JSD-metric threshold candidates

$P = [0 : 1 : q/2^{d_v+1}]$ // all proximity values

$C_{list} =$ empty list // contraction amount wrpt to JSD-metric candidates

$G_{list} =$ empty list // overall gain wrpt to JSD-metric candidates

for t *in* T_{list} **do**

  $g = [0 \text{ for } i \text{ in range}(0, q/2^{d_v+1})]$;

  $n_{error} = [0 \text{ for } i \text{ in range}(0, q/2^{d_v+1})]$;

  for $i \in [0, w-1]$ **do**

    if $m_{JSD}[i] < t$ **then**

      // the i-th inequality can be contracted

      for $c \in P$ **do**

        if $b[i] - c < Ax[i]$ *or* $c[i] + c > Ax[i]$ **then**

          // introduce erroneous inequality

          $n_{error}[c] = n_{error}[c] + 1$;

          $g[c] = g[c] + \min\{Ax[i] - (c[i] + c), (b[i] + c) - Ax[i]\}$// add a negative value to g

        $g[c] = g[c] + c$;

  $g_{opt} = \max\{g[i] | n_{error}[i] \le N_{error}, i \in [0, q/2^{d_v+1})\}$;

  $c_{opt} = P[\arg\max\{g[i] | n_{error}[i] \le N_{error}, i \in [0, q/2^{d_v+1})\}]$;

  $C_{list}.append(c_{opt})$;

  $G_{list}.append(g_{opt})$;

$i = \arg\max(G_{list})$;

return $T_{list}[i], C_{list}[i]$

results to such singular probabilities. So in our algorithm, if there were negative values in calculating the marginal distribution regarding some inequality, we treat it as a *raw* inequality that the true value of the linear combination of variables has not come to the fore, and do not contract the current inequality.

In addition, we empirically make each inequality contract only once. This is because after an inequality switches to the low-proximity class after contraction, their JSD-metric does not get an immediate boost, which tends to break the law that the JSD-metric decreases with increasing proximity, making the overall gain in the next iteration not large enough.

Illustration of Algorithm 2 results. In Figure 8, we demonstrate the thresholds, contraction amount and the number of inequalities that are contracted in the first 5 iterations in a test case of 8200 inequalities constructed from unfiltered ciphertexts of KYBER512. In each iteration, we computed the JSD-metric for all inequalities according to Equation (13), and then categorized the inequalities according to their proximity. Note that the inequalities counted are only those that are non-raw inequality and have not been contracted yet. To reflect the distribution of JSD-metric values in each class, we plot the distribution of JSD-metrics in each class with violin plots labeled medians, upper quartiles, and lower quartiles. The thresholds of JSD-metric found under Algorithm 2 are marked with red dashed lines. The inequalities whose JSD-metric is below the red line can be contracted. With no erroneous inequality introduced, the contraction amount is the minimum proximity value of the inequalities under the red line. The number of inequalities contracted at each iteration step is also given at the top of each figure. For the first iteration step (iteration 0), all variables take the initial distribution B_{η_1}, and in each class there are inequalities with small JSD-metric values, so they cannot be used for contraction. From the second iteration (iteration ≥ 1), the correlation between the JSD-metric and the proximity starts to become

obvious and the contraction strategy can be applied. In the second iteration (iteration 1), the upper and lower boundaries of 1153 inequalities can be contracted by 31, and in the third iteration (iteration 2), the upper and lower boundaries of 1461 equations can be contracted by 27, *etc.* .

Experiments show that when the contraction applies from the very beginning of the iteration (iteration 1), after four iterations, both the amount of contraction and the number of contractible equations decrease dramatically. Therefore our contraction strategy is performed in the first five steps to train parameters for each given system of inequalities. It can be seen from Figure 8 that as iterations and contractions proceed, the inequalities of high proximity are *swept* into the low-proximity classes, thus increasing the information carried by the system.

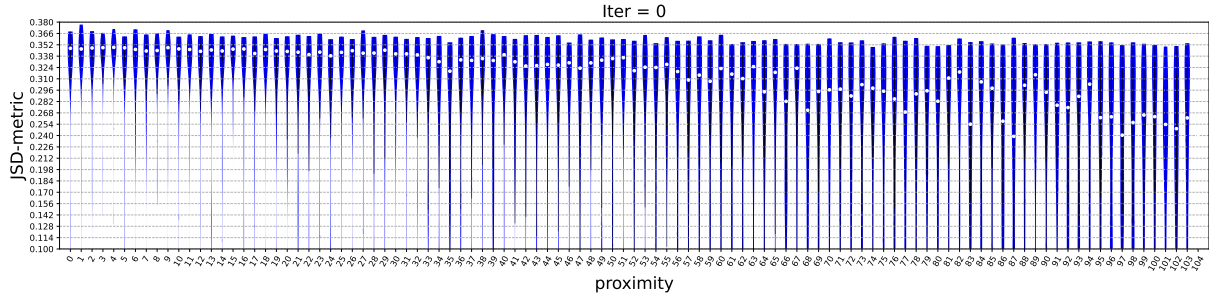
We tested on 10 sample systems with randomly generated public keys, secret keys, plaintexts and ciphertexts in KYBER512. The JSD-metric thresholds and contraction amounts trained for the first 5 iterations are shown in Table 5. The minimum values in bold are used as the parameters for contracting in the attacking phase.

Table 5: JSD-metric threshold and contraction amount trained in iteration 1-4 for 10 cases of unfiltered and filtered inequalities.

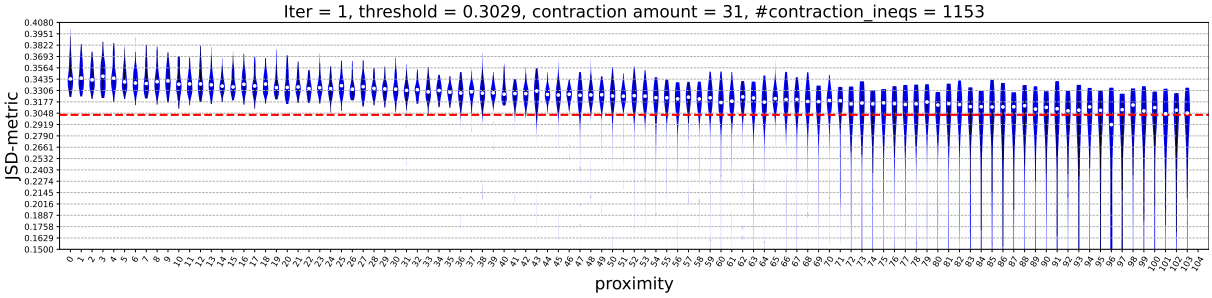
	cases	8200 unfiltered ineq.				5200 filtered ineq.			
		iter 1	iter 2	iter 3	iter 4	iter 1	iter 2	iter 3	iter 4
JSD-metric threshold	1	0.3078	0.3021	0.3062	0.3032	0.2372	0.2277	0.2296	0.2283
	2	0.3142	0.3029	0.3014	0.299	0.2198	0.2271	0.2271	0.2256
	3	0.3108	0.2961	0.3013	0.3044	0.2343	0.2281	0.233	0.2319
	4	0.305	0.2966	0.2934	0.3014	0.2353	0.2248	0.2272	0.2309
	5	0.308	0.3014	0.3004	0.3092	0.2162	0.2273	0.227	0.2395
	6	0.2941	0.3019	0.3001	0.3065	0.2404	0.2362	0.2363	0.2352
	7	0.3029	0.3023	0.2994	0.3009	0.24	0.2306	0.2289	0.2317
	8	0.3121	0.3058	0.3012	0.3149	0.2383	0.2295	0.2274	0.2249
	9	0.3074	0.2997	0.3019	0.2967	0.2357	0.2255	0.2268	0.2294
	10	0.3173	0.3056	0.3124	0.3114	0.2366	0.2276	0.2264	0.2274
	min	**0.2941**	**0.2961**	**0.2934**	**0.2967**	**0.2162**	**0.2248**	**0.2264**	**0.2249**
contraction amount	1	32	23	13	3	19	31	9	9
	2	12	27	14	15	29	28	22	19
	3	11	50	20	7	13	29	9	4
	4	18	47	30	7	23	43	15	7
	5	15	28	19	5	34	29	18	1
	6	20	32	17	4	14	8	3	3
	7	31	27	26	14	14	18	13	4
	8	20	16	15	1	17	28	13	12
	9	15	38	20	14	30	50	15	4
	10	9	19	4	5	28	32	19	4
	min	**9**	**16**	**4**	**1**	**13**	**8**	**3**	**1**

Contracting in attacking phase. Utilizing the JSD-metric and the associated contraction amount derived from the profiling phase for the first 5 steps in BP algorithm, during the attacking phase, we execute a 5-step BP algorithm on the inequalities formulated based on unknown secret keys. In this process, inequalities for which the JSD-metric falls below the specified threshold are subjected to contraction, with an amount corresponding to the current iteration. Subsequently, we proceed to apply the BP algorithm once more to the refined system of inequalities. This completes the contraction strategy.

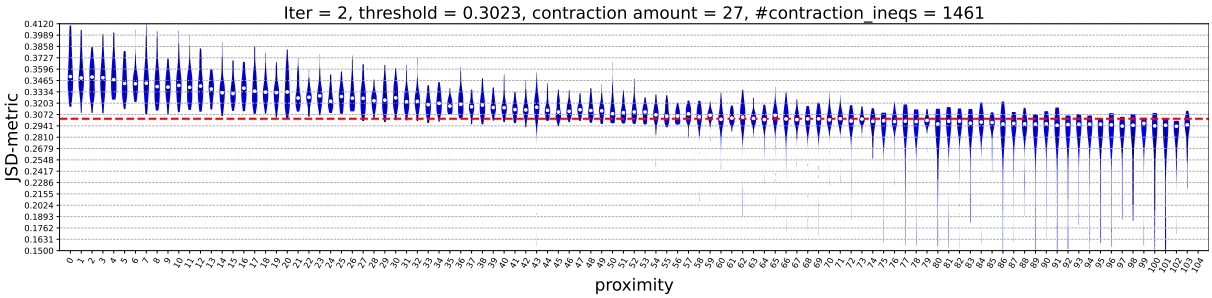
We tested the parameters trained for 8200 unfiltered inequalities and those for 5200 filtered inequalities as is shown in Table 5 on newly generated 10 systems of inequalities for KYBER512. The number of inequalities that are subjected to contraction during the first 5 steps are given in Table 6. Taking case $1'$ as an example, out of 8200 unfiltered equations, 996 equations in the first iteration have both upper and lower bounds contracted by 9, and 672 equations in the second iteration have both upper and lower



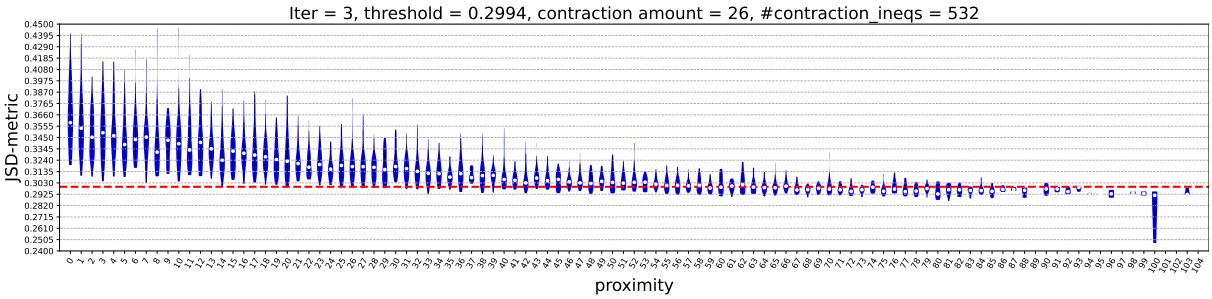
(a) iteration 0



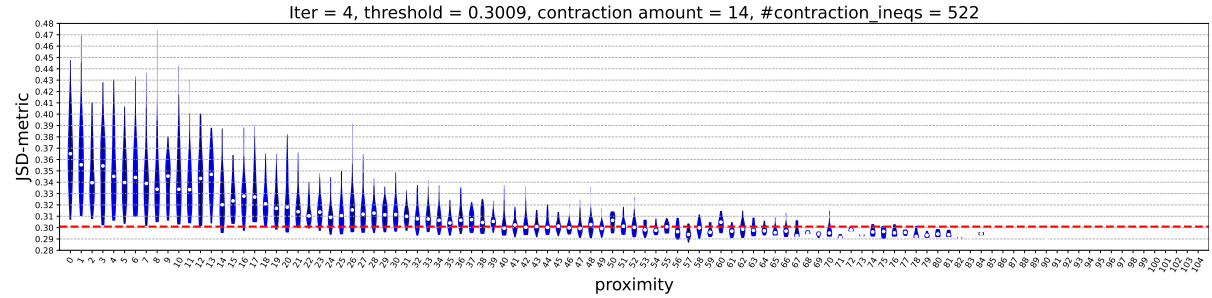
(b) iteration 1



(c) iteration 2



(d) iteration 3



(e) iteration 4

Fig. 8: JSD-metric thresholds and contraction amount in BP iterations for 8200 unfiltered inequalities of KYBER512. Inequalities with high proximity are *swept* to the low proximity classes.

bounds contracted by 16, etc. The contracted equations clearly provide a greater amount of information about the key.

Table 6: Number of contracted inequalities tested on another 10 cases for unfiltered and filtered inequalities and the number of erroneous inequalities introduced. Bold values are the contraction amount.

cases	8200 unfiltered inequalities					5200 filtered inequalities				
	iter 1	iter 2	iter 3	iter 4		iter 1	iter 2	iter 3	iter 4	
	9	**16**	**4**	**1**	# err	**13**	**8**	**3**	**1**	# err
$1'$	996	672	569	832	0	228	229	233	370	0
$2'$	1008	541	516	738	0	189	244	210	443	0
$3'$	954	724	651	794	0	216	250	251	394	0
$4'$	1024	652	543	820	0	198	248	244	395	0
$5'$	1007	625	451	725	0	217	212	241	473	1
$6'$	1002	690	600	811	0	220	211	276	462	1
$7'$	1030	654	567	808	0	215	201	228	408	0
$8'$	1032	640	569	775	0	217	244	246	378	0
$9'$	994	670	560	744	0	196	242	239	384	0
$10'$	972	725	560	856	0	205	238	252	385	0

Despite our algorithm's best efforts to make the parameters trained in the profiling phase not introduce error inequalities in the attacking phase, in experiments on filtered inequalities we encountered cases where error inequality was introduced, but only on a relatively small number of counts with only 1 error inequality. It is considered to be insignificant in the number of inequalities of several thousand magnitudes. We do not encounter erroneous inequalities for the unfiltered inequalities.

4.3 Machine Learning-based Contraction Strategy

Figure 7 tells us that there is an obvious correlation between JSD-metric and proximity, but it is not easy to accurately represent this relationship in mathematics. 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 improve the correctness of the classification, we build a binary classification model with a proximity cutoff equal to m (typically $m = 50$). Once the model has been trained, to introduce as few or no error equations as possible, the upper and lower bounds of the equation can be contracted by m only if the samples are predicted to have a high probability of proximity larger than m. In the following, we give details of the machine learning experiments for both filtered and unfiltered inequalities.

Training data and test data. For filtered inequalities, the target is to perform contractions on chosen inequalities from a total of 4500. Running BP algorithm on 4500 inequalities, JSD-metric sequences are collected for each inequality. The inequalities with proximity smaller than and equal to 50 are labeled 0, and those with proximity larger than 50 are labeled 1. Twenty groups of such samples are generated and a total of $20 \times 4500 = 9 \times 10^4$ training data are prepared. Inequalities in each two groups are generated randomly under the same secret key. Then with another key, a group of 4500 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.

Model, accuracy and prediction. For both filtered and unfiltered inequalities, we chose the random forest model for training. The number of trees is set to 300 and the max depth is set to 30. After

training, the out-of-bag accuracies are 88.75% and 92.04% respectively. When applying to test data, the classification accuracies are 87.82% and 78.39% respectively. If we contract the inequalities by 50 whose predicted label is 1, for filtered inequalities, 1163 will be contracted among which 112 (9.63%) are erroneous inequalities. For unfiltered inequalities, 2462 inequalities can be contracted by 50 but 24 (0.97%) are erroneous. To reduce the number of erroneous inequalities, we can require that only samples whose predicted 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.

Table 7: Number of contracted inequalities.

contraction amount	threshold	Among 4500 filtered		Among 8000 unfiltered	
		#correct	#err	#correct	#err
50	0.5	1051	112	2438	24
	0.6	962	85	2089	11
	0.7	863	49	1754	4
	0.8	754	31	1539	1
	0.9	563	11	1221	1

4.4 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. Since the BP algorithm is a statistical algorithm, given a certain number of inequalities, a reliable figure for the number of variables that can be recovered must be obtained based on a large number of experiments, i.e., it is necessary to obtain, based on a large number of experiments, how many of the variables that are in the top of the list at the end of BP are indeed the correctly recovered variables. And it is normal for individual experimental data to have wobbles.

Several hyperparameters can be set in our method. The current experiments are trained from the first iteration. If the correlation between JSD-metric and proximity becomes stronger after several iterations, applying the contraction strategy from an intermediate iteration will be better. Therefore starting iteration to apply the contraction strategy is a hyperparameter. Treating the updated system of inequalities as a fresh system, the contraction strategy can be applied again until there are no more contractions available. The number of overall applications is also a hyperparameter. In the machine learning-based contraction strategy, the random forest model is used to solve a binary classification problem to ensure robustness. The extreme case is to classify the equations strictly according to the number of categories of proximity. We can also build compromise multiclassification problems to design the contraction strategy. Also, whether there is a better metric than JSD-metric to reflect the proximity of the inequalities can be investigated.

Our approach gives an effective way to enhance the informativeness of inequality systems at the algorithmic level and can be employed in any attack endeavoring to establish a system of inequalities concerning key variables.

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 key variables 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.

We also proposed contraction strategies to further reduce the interval of inequalities to lead to a more informative system of inequalities.

Future work includes applying our proposed contraction strategy multiple times to the updated system of inequalities to further enhance the information of the system of inequalities. In addition, the current experiments have only been explored when all the inequalities are correct, and it remains to be investigated how error affects the JSD-metric and the contraction strategy.

References

ABD+21. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber algorithm specifications and supporting documentation (version 3.02). 2021. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

BDH+21. Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):334–359, Jul. 2021.

BDHD+19. Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse Attacks on Post-quantum Cryptosystems. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 747–776, Cham, 2019. Springer International Publishing.

BPO+20. Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. High-Speed Masking for Polynomial Comparison in Lattice-based KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):483–507, Jun. 2020.

Cla07. Christophe Clavier. Secret External Encodings Do Not Prevent Transient Fault Analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 181–194, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

Del22. Jeroen Delvaux. Roulette: A Diverse Family of Feasible Fault Attacks on Masked Kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):637–660, Aug. 2022.

DHP+22. Jan-Pieter D'Anvers, Daniel Heinz, Peter Pessl, Michiel Van Beirendonck, and Ingrid Verbauwhede. Higher-Order Masked Ciphertext Comparison for Lattice-Based Cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):115–139, Feb. 2022.

DSDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with Side Information: Attacks and Concrete Security Estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 329–358, Cham, 2020. Springer International Publishing.

DTVV19. Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. Timing Attacks on Error Correcting Codes in Post-Quantum Schemes. In *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*, TIS'19, page 2–9, New York, NY, USA, 2019. Association for Computing Machinery.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of cryptology*, 26:80–101, 2013.

GJN20. Qian Guo, Thomas Johansson, and Alexander Nilsson. A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 359–386, Cham, 2020. Springer International Publishing.

HMS+23. Julius Hermelink, Erik Møartensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(4):287–317, Aug. 2023.

HPP21. Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. Fault-Enabled Chosen-Ciphertext Attacks on Kyber. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, *Progress in Cryptology – INDOCRYPT 2021*, pages 311–334, Cham, 2021. Springer International Publishing.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

Nat. National Institute of Standards and Technology. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

NDGJ21. Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):676–707, Aug. 2021.

OSPG18. Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical CCA2-Secure and Masked Ring-LWE Implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):142–174, Feb. 2018.

PP21. Peter Pessl and Lukas Prokop. Fault Attacks on CCA-secure Lattice KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):37–60, Feb. 2021.

QCZ+21. Yue Qin, Chi Cheng, Xiaohan Zhang, Yanbin Pan, Lei Hu, and Jintai Ding. A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 92–121, Cham, 2021. Springer International Publishing.

RBRC22. Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. On Exploiting Message Leakage in (Few) NIST PQC Candidates for Practical Message Recovery Attacks. *IEEE Transactions on Information Forensics and Security*, 17:684–699, 2022.

RCDB22. Prasanna Ravi, Anupam Chattopadhyay, Jan Pieter D'Anvers, and Anubhab Baksi. Side-channel and Fault-injection attacks over Lattice-based Post-quantum Schemes (Kyber, Dilithium): Survey and New Results. Cryptology ePrint Archive, Paper 2022/737, 2022. https://eprint.iacr.org/2022/737.

REB+21. Prasanna Ravi, Martianus Frederic Ezerman, Shivam Bhasin, Anupam Chattopadhyay, and Sujoy Sinha Roy. Will You Cross the Threshold for Me? Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):722–761, Nov. 2021.

RRD+23. Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):418–446, Mar. 2023.

RSRCB20. Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):307–335, Jun. 2020.

SCZ+22. Muyan Shen, Chi Cheng, Xiaohan Zhang, Qian Guo, and Tao Jiang. Find the Bad Apples: An efficient method for perfect key recovery under imperfect SCA oracles – A case study of Kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(1):89–112, Nov. 2022.

TUX+23. Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):473–503, Jun. 2023.

UXT+21. Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):296–322, Nov. 2021.

VOGR18. Felipe Valencia, Tobias Oder, Tim Güneysu, and Francesco Regazzoni. Exploring the vulnerability of R-LWE encryption to fault attacks. In *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems*, pages 7–12, 2018.

XPR+22. Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems With Chosen Ciphertexts: The Case Study of Kyber. *IEEE Transactions on Computers*, 71(9):2163–2176, 2022.