Research Article

On Accuracy of Testing Decryption Failure Rate for Encryption Schemes under the LWE Assumption

Lin Wang 1, Yang Wang 2,3 and Huiwen Jia 4

1 Science and Technology on Communication Security Laboratory, Chengdu 610041, China
2 State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
3 School of Mathematics, Shandong University, Jinan 250100, China
4 School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006, China

Correspondence should be addressed to Lin Wang; wanglin4math@outlook.com

Received 7 June 2023; Revised 2 September 2023; Accepted 12 September 2023; Published 23 January 2024

Copyright © 2024 Lin Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Due to Shor’s algorithm [1, 2], quantum computing seriously threatens popular public key cryptosystems, including RSA [3], encryption and digital signature schemes based on discrete logarithm [4, 5], and elliptic curve cryptography [6–8]. Much research has been carried out to construct robust cryptography in the quantum era, referring to a survey [9]. Particularly, the National Institute of Standards and Technology (NIST) began its standardization project on post-quantum cryptography (PQC) in 2016 [10], selected four algorithms in July 2022 [11] and then advances further into the fourth round [12].

Lattice-based cryptography is the most promising and the most significant in PQC [13]. It occupies 7 seats among 15 in the third round of NIST PQC standardization [14]. Particularly, among the four post-quantum ciphers selected by NIST [11], three are lattice-based, particularly the unique key encapsulation mechanism (KEM) CRYSTALS-Kyber [15].

Dating back to knapsack cryptosystems [16] and successful NTRU [17], lattice-based cryptography has made great progress since Regev [18] proposed the learning with errors (LWE) problem. Let $q$ be a positive integer, $\mathbb{Z}_q$ the residue ring modulo $q$, and $a \mod q$ the unique representative of $a$ in the range $[-q/2, q/2]$. Let $D$ be a (discrete) distribution over $\mathbb{Z}_q$. A sample $X$ complies with $D$ if $\Pr[X = a] = D(a)$ for $D(a) \geq 0$, and we denote this by $X \leftarrow D$. For a set $S$, without ambiguity $X \leftarrow S$ means that $X$ is uniformly sampled over $S$. The LWE problem is to find the secret $s \in \mathbb{Z}_q^* \setminus \frac{\mathbb{Z}_q}{q}$ given (sufficiently many) pairs $(a, b)$, where $a \leftarrow \mathbb{Z}_q$, $e \leftarrow \mathbb{Z}_q$, and $b = a^T s + e$. The Lindner–Peikert encryption scheme [19] is grounded on the assumption that the LWE problem is intractable, and therefrom many lattice-based KEMs have developed along with other techniques, for example, structured lattices [20, 21], variants of LWE [22, 23], and compressing public keys/ciphertexts [21]. Figure 1 below shows a version of the Lindner–Peikert cryptosystem enciphering a message in $\mathbb{Z}_q$.

For the cryptosystem in Figure 1, decryption fails if the size of $e_1^T s_2 - s_1^T e_2 + e_3 \mod q$ is not as small as required. Explicitly, the decryption failure rate (DFR; the condition in Equation (1)) [24] is also interpreted as $|e_1^T s_2 - s_1^T e_2 + e_3 \mod q| > t$, for example, in D’Anvers et al. [23, 25, 26]. Whether $t$ is counted in does not exert substantial influence on computing $\delta_{\text{fail}}$, denoted by $\delta_{\text{fail}}$, is the following probability.
Therefore, it is meaningful and interesting to ef
figure boosting \( \delta \) lattice-based schemes with large DFR are vulnerable to the
evaluation of
and
m

\[ \text{key generation} \]

| 1: \( A \leftarrow Z_q^{m'} \) | 1: \( s_2 \leftarrow D_{s_2} \) |
| 2: \( s_1 \leftarrow D_{s_1} \) | 2: \( e_2 \leftarrow D_{e_2} \) |
| 3: \( e_1 \leftarrow D_{e_1} \) | 3: \( e_3 \leftarrow D_{e_3} \) |
| 4: \( b = A s_1 + e_1 \) | 4: \( c_i = A^T s_i + e_2 \) |
| 5: \( pk = (A, b) \) | 5: \( c_2 = b^T s_2 + e_3 + \text{Encode}(m) \) |
| 6: \( sk = s_1 \) | 6: \( ck = (c_1, c_2) \) |

\[ \text{Decryption} \]

\[ \text{Pr} \left[ |e_1^T s_2 - s_1^T e_2 + e_3 \mod q| \notin [-t, t) \right] : s_1 \leftarrow D_{s_1}, \quad s_2 \leftarrow D_{s_2}, \quad e_1 \leftarrow D_{e_1}, \quad e_2 \leftarrow D_{e_2}, \quad e_3 \leftarrow D_{e_3} \]

\[ \text{(1)} \]

where the critical positive value \( t \) depends on the specific scheme, for example, involving the modulus \( q \) and the num-
ber of bits in the plaintext \( m \).

Decimal failure is closely related to the security of lattice-
based cryptography. On the one hand, the DFR impacts the
tightness of constructing IND-CCA encryption/KEMs in the
(quantum) random oracle model [27–29]. On the other hand,
lattice-based schemes with large DFR are vulnerable to the “fail-
ure boosting” attack [26, 30, 31] and risk a loss of security level.
Therefore, it is meaningful and interesting to efficiently compute
the DFR \( \delta_{\text{fail}} \) with confident accuracy.

The key to obtaining \( \delta_{\text{fail}} \) is to characterize the distribution
of \( e_1^T s_2 - s_1^T e_2 + e_3 \mod q \) in Equation (1). At present
there are two approaches [24, 25]. One is a heuristic estimate via the central limit theorem. The other is to compute the
r-fold convolution of distributions via the “double-and-add” method. Specifically, let \( r \) have its binary representation
\( r_0 2^n + r_{n-1} 2^{n-1} + \cdots + r_0 \), and denote

\[
\begin{align*}
\mathcal{P}_i & \quad \text{the distribution of} \left[ \sum_{j=0}^{i-n} r_j 2^{-j} \right] \cdot (e_1 s_2 - s_1 e_2), \\
\mathcal{P}_{i,\text{dbl}} & \quad \text{the distribution of} \left[ \sum_{j=0}^{i-n} r_j 2^{-j} \right] \cdot (e_1 s_2 - s_1 e_2), \\
\mathcal{P}_{\text{fin}} & \quad \text{the distribution of} \left[ r \right] \cdot (e_1 s_2 - s_1 e_2) + e_3;
\end{align*}
\]

Fourth, we also analyze how the test of decryption failure
is influenced by algebraic lattices and the rounding compress-
and whether the proposed test is feasible for lattice-based
ID-based encryption (IBE) and attribute-based encryption
(ABE).
less than the machine precision of common floating-point arithmetic. According to available program scripts [14, 15, 24, 25], CRYSTALS-Kyber and SABER use double-precision floating-point while FrodoKEM employs float128 in the python numpy package.

To the best of our knowledge, an explicit relation between the precision of δ\(_{\text{fail}}\) and the trimming threshold β has not been given and there has not been a reasonable approach to choosing the trimming threshold β. Intuitively, the greater β is, the less computation time we need; and the smaller β is, the preciser our approximation is. At present practical trimming thresholds are used. According to available program scripts [14], during computation Equation (3), CRYSTALS-Kyber and SABER neglect probability not greater than 2\(^{-300}\) and give log δ\(_{\text{fail}}\) with three significant digits [15, 25], and FrodoKEM always removes probability less than 10\(^{-200}\) and gives log δ\(_{\text{fail}}\) with four significant digits [24].

1.3. Organization. The rest of this paper is organized as follows. In Section 2, we prepare some definitions and facts on floating-point arithmetic and discrete distributions. Section 3 includes the main result and consists of three subsections: Subsection 3.1 analyzes effectiveness of the “double-and-add” algorithm with floating-point errors and the trimming technique; Subsection 3.2 shows our method to select the floating-point datatype; Subsection 3.3 gives a new algorithm to estimate the DFR whose outcome is confirmed to be precise as required; and Subsection 3.5 analyzes the impact of structured lattices and the rounding compression on the “double-and-add” test, and also discuss its application in IBE/ABE cryptosystems. Finally, Section 4 concludes this paper with a summary.

2. Preliminaries

2.1. Floating-Point Arithmetic. Let \(\varepsilon_M\) (called unit round-off in Saad [32]) denote the upper bound of relative errors to represent real numbers by normalized floating-point numbers, and let \(\alpha_M\) be the minimal positive normalized floating-point number in machine. Both \(\varepsilon_M\) and \(\alpha_M\) highly depend on machine precision. For example, by IEEE Standard 754 [33], rounding a real number to its nearest 64 bit normalized floating-point number yields a relative error at most 2\(^{-53}\) [32], \(\alpha_M = 2^{-126}\) for single precision and \(\alpha_M = 2^{-1.022}\) for double precision.

In the sequel, for any variables (or functions) \(f\) and \(g\), let \(f \sim g(1 \pm \varepsilon_M)^m\) denote \(g \cdot (1 - \varepsilon_M)^m \leq f \leq g \cdot (1 + \varepsilon_M)^m\). (4)

The relative error of floating-point arithmetic is known to be bounded.

Lemma 1 (see [32]). Let \(a_1\) and \(a_2\) be non-negative normalized floating-point numbers, and \(s\) (resp. \(p\)) the sum (resp. product) of \(a_1\) and \(a_2\) in floating-point arithmetic. If no overflow occurs, then

\[ s \sim (a_1 + a_2)(1 \pm \varepsilon_M) \text{ and } p \sim (a_1 \cdot a_2)(1 \pm \varepsilon_M). \] (5)

2.2. Pseudo-Laws. A distribution completely characterizes a discrete random variable, and we introduce the term “pseudo-law” to describe part of a distribution.

Definition 1. (Pseudo-law). A function \(D\) on a set \(R\) is called a pseudo-law if \(D(a) \geq 0\) for any \(a \in R\) and

\[ \sum_{a \in R} D(a) \leq 1. \] (6)

In this paper, we only consider pseudo-laws over \(R = \mathbb{Z}_q\).

Let \(C_a\) denote the law distributed only at \(a \in \mathbb{Z}_q\), that is,

\[ C_a(x) = \begin{cases} 1, & \text{if } x = a; \\ 0, & \text{if } x \neq a. \end{cases} \] (7)

Let \(D_1\) and \(D_2\) be pseudo-laws. The convolution of \(D_1\) and \(D_2\), denoted by \(D_1 \circ D_2\) is

\[ D_1 \circ D_2(c) = \sum_{a \in \mathbb{Z}_q} D_1(a) \cdot D_2(c - a), \] (8)

and the product of \(D_1\) and \(D_2\), denoted by \(D_1 \odot D_2\), is

\[ D_1 \odot D_2(c) = \sum_{a \in \mathbb{Z}_q} D_1(a) \cdot D_2(c - a). \] (9)

Notice that \(C_{-1} \circ D\) is a mirror reflection of \(D\) over the \(Y\)-axis.

The convolution \(D_1 \circ D_2\) (partially) describes the sum of independent random variables complying with pseudo-laws \(D_1\) and \(D_2\).

Remark 1. A pseudo-law \(D\) can be represented by

\[ \chi_D(x) = \sum_{a \in \mathbb{Z}_q} D(a) \cdot x^a \] (10)

in the quotient ring \(\mathbb{R}[x]/(x^t - 1)\) of polynomials, where \((x^t - 1)\) denotes the principal ideal generated by \(x^t - 1\). Thereby, the convolution of \(D_1\) and \(D_2\) is implemented by \(\chi_{D_1} \cdot \chi_{D_2}\) and hence techniques for polynomial multiplication are admissible and helpful. This strategy was essentially employed by FrodoKEM [24] to accelerate convolutions, while CRYSTALS-Kyber [15] and SABER [25] used the definition Equation (8).

The statistical distance measures how far two random variables differ from each other [34], and it naturally extends to pseudo-laws.

Definition 2. (Statistical distance). Given two pseudo-laws \(D_1\) and \(D_2\), the statistical distance of \(D_1\) and \(D_2\), denoted by \(\Delta(D_1, D_2)\), is defined to be
Lemma 2. Let \( D_1, D_2, D_3 \) be pseudo-laws over \( \mathbb{Z}_q \). Then
\[
\Delta(D_1, D_2) = \sum_{a \in \mathbb{Z}_q} |D_1(a) - D_2(a)|.
\]

Lemma 3. Let \( D_1, D_2, E_1, E_2 \) be pseudo-laws over \( \mathbb{Z}_q \). Then
\[
\Delta(D_1 \oplus D_2, E_1 \oplus E_2) \leq \Delta(D_1, E_1) + \Delta(D_2, E_2).
\]

Definition 3. (Trim). Let \( S \subseteq \mathbb{Z}_q \). A trim of a pseudo-law \( D \) by \( S \), denoted as \( \text{Trim}_S(D) \), is a pseudo-law defined by
\[
\text{Trim}_S(D)(a) = \begin{cases} 
D(a), & a \in \mathbb{Z}_q \setminus S, \\
0, & a \in S.
\end{cases}
\]

Without ambiguity, \( \text{Trim}_S(D) \) is written as \( \text{Trim}_\beta(D) \) if \( S = \{ a \in \mathbb{Z}_q : D(a) \leq \beta \} \).

Input: the modulus \( q \); the distributions \( D_{e_1}, D_{e_2}, D_{s_1}, D_{s_2}, D_{t_1}, D_{t_2} \) of coordinates of \( e_1, e_2, s_1, s_2, t_1, t_2 \), respectively; the distribution \( D_{\omega} \) of \( \omega \); the dimension \( r \); the critical value \( t \) of decryption failure; the trimming threshold \( \beta \).

Output: an approximation of \( \delta_{\text{fail}} \).

1: Compute the distribution \( D = e_1 \cdot s_2 - s_1 \cdot e_2 \), where
\[
s_2 \leftarrow D_{s_2}, \quad s_1 \leftarrow D_{s_1}, \quad e_2 \leftarrow D_{e_2}, \quad e_1 \leftarrow D_{e_1},
\]
for example, \( D = (D_{s_1} \oplus D_{s_2}) \oplus (C_{1-i} \oplus D_{t_1} \oplus D_{t_2}) \).

2: \( D_{h} = \text{Trim}_\beta(D) \).

3: Get the binary representation \( r = \sum_{i=0}^{q-1} r_i 2^i \), where \( r_i \in \{0,1\} \) and \( r_n = 1 \).

4: for \( i = 1 \) to \( n \) do
5: \( D_{i}^{\text{dbl}} = \text{Trim}_\beta(D_{i-1} \oplus D_{i-1}) \).
6: if \( r_{n-i+1} = 1 \) then
7: \( D_i = \text{Trim}_\beta(D_{i}^{\text{dbl}} \oplus D_0) \).
8: else
9: \( D_i = D_i^{\text{dbl}} \).
10: end if
11: end for
12: \( D_{\text{fin}} = \text{Trim}_\beta(D_n \oplus D_0) \).
13: return \( \delta_{\text{alg}} = \sum_{a \in \mathbb{Z}_q, a \neq (-1, t)} D_{\text{fin}}(a) \).

Input: the modulus \( q \); the distributions \( D_{e_1}, D_{e_2}, D_{s_1}, D_{s_2}, D_{t_1}, D_{t_2} \) of coordinates of \( e_1, e_2, s_1, s_2, t_1, t_2 \), respectively; the distribution \( D_{\omega} \) of \( \omega \); the dimension \( r \); the critical value \( t \) of decryption failure; a number \( \beta \) for trimming.

Output: an approximation of \( \delta_{\text{fail}} \).

1: Compute the distribution \( D = e_1 \cdot s_2 - s_1 \cdot e_2 \), where
\[
s_2 \leftarrow D_{s_2}, \quad s_1 \leftarrow D_{s_1}, \quad e_2 \leftarrow D_{e_2}, \quad e_1 \leftarrow D_{e_1},
\]
for example, \( D = (D_{s_1} \oplus D_{s_2}) \oplus (C_{1-i} \oplus D_{t_1} \oplus D_{t_2}) \).

2: \( D_{0} = \text{Trim}_\beta(D) \), where
\[
\delta_0 = \{ a \in \mathbb{Z}_q : D(a) \leq \beta \}
\]
and \( D \) is given in Line 2 of Algorithm 1.

3: Get the binary representation \( r = \sum_{i=0}^{q-1} r_i 2^i \), where \( r_i \in \{0,1\} \) and \( r_n = 1 \).

4: for \( i = 1 \) to \( n \) do
5: \( D_{i}^{\text{dbl}} = \text{Trim}_\beta(D_{i-1} \oplus D_{i-1}) \), where
\[
\delta_{i}^{\text{dbl}} = \{ a \in \mathbb{Z}_q : (D_{i-1} \oplus D_{i-1})(a) \leq \beta \}
\]
and \( D_{i} \oplus D_{i-1} \) is given in Line 5 of Algorithm 1.

6: if \( r_{n-i+1} = 1 \) then
7: \( D_i = \text{Trim}_\beta(D_{i}^{\text{dbl}} \oplus D_0) \), where
\[
\delta_{i}^{\text{dbl}} = \{ a \in \mathbb{Z}_q : (D_{i}^{\text{dbl}} \oplus D_0)(a) \leq \beta \}
\]
and \( D_i \oplus D_0 \) is given in Line 7 of Algorithm 1.

8: else
9: \( D_i = D_i^{\text{dbl}} \).
10: end if
11: end for
12: \( D_{\text{fin}} = \text{Trim}_\beta(D_n \oplus D_0) \), where
\[
\delta_{\text{fail}} = \{ a \in \mathbb{Z}_q : (D_n \oplus D_0)(a) \leq \beta \}
\]
and \( D_n \oplus D_0 \) is given in Line 12 of Algorithm 1.
13: return \( \delta_{\text{alg}} = \sum_{a \in \mathbb{Z}_q, a \neq (-1, t)} D_{\text{fin}}(a) \).

Algorithm 1: Estimate DFR with floating-point arithmetic.

Algorithm 2: Estimate DFR with ideal precision.

3. Main Results

3.1. The "Double-and-Add" Algorithm. Algorithm 1 [15, 24, 25] below implements Equation (3) with trimming \( \beta \) and deterministically computes the DFR \( \delta_{\text{fail}} \).

In Algorithm 1, values of pseudo-laws are stored and operated in floating-point numbers.

Above all, using the "double-and-add" method, Algorithm 1 terminates in polynomial time \( O(q^2 \log q) \) since a convolution or a product cost at most \( \Theta(q^2) \). As in Remark 1, this complexity can be further reduced using Fourier transformation.

Theorem 1. Algorithm 1 runs in deterministic polynomial time \( O(q^2 \log q) \).

Now we analyze effectiveness of Algorithm 1, that is, how closely it approximates the DFR \( \delta_{\text{fail}} \).

Theorem 2. If \( \beta \geq \sqrt{\alpha_M} \) and each nonzero value of distributions \( D_{e_1}, D_{e_2}, D_{s_1}, D_{s_2}, D_{t_1} \) and \( D_{t_2} \) is not less than \( \sqrt{\alpha_M} \), then
\[
\delta_{\text{alg}} \geq (1 - \epsilon_M^{1+tr(q+1)}) \delta_{\text{fail}} - 2q \delta;
\]
\[
\delta_{\text{alg}} \leq \delta_{\text{fail}} (1 + \epsilon_M^{1+tr(q+1)}).
\]
Theorem 2 describes and connects the factors that exert influence on the effectiveness of Algorithm 1.

To prove Theorem 2, we show a variant of Algorithm 1 with ideal precision (Algorithm 2).

Notice that Algorithm 2 differs from Algorithm 1 in two aspects:

(i) The values of pseudo-laws are stored and processed as real numbers with ideal precision. To distinguish these notations, we add a superscript "\( \ast \)" to pseudo-laws in Algorithm 2.

(ii) Instead of a fixed trimming threshold \( \beta \), Algorithm 2 imposes zero value exactly at the same elements of \( \mathbb{Z}_q \) as Algorithm 1 does. This is feasible since, as Theorem 1 ensures, it is efficient to simulate Algorithm 1.

To show how closely \( \delta_{ab} \) approximates \( \delta_{thal} \), the proof contains four parts. Part I describes how pseudo-laws in Algorithm 1 approximates their counterparts in Algorithm 2. Part II quantifies the influence of trimming, and then Part III derives the fact that \( D_{\text{fin}} \) in Algorithm 2 is close to \( P_{\text{fin}} \). Part IV integrates the above to complete the proof.

Part I (Lemma 4): Using a power of \((1 + \epsilon_M)\) as multiplier, we relate the pseudo-laws in Algorithm 1 to their counterparts in Algorithm 2.

Lemma 4. Let \( E \) be any of the pseudo-laws \( D_{\text{fin}}, D_i \oplus D_{c_i}, D_{i-1} \oplus D_{c_{i-1}}, \) and \( D_i^{\text{d-bl}} \oplus D_0 \) (\( 1 \leq i \leq n \)) in Algorithm 1, and let \( E^\ast \) denote its ideally precise counterpart among \( D_{\text{fin}}^\ast, D_i^\ast \oplus D_{c_i}^\ast, D_{i-1}^\ast \oplus D_{c_{i-1}}^\ast, \) and \( D_i^{\text{d-bl}} \oplus D_0^\ast \) in Algorithm 2. Given the condition in Theorem 2, it holds that

\[
E \sim E^\ast(1 + \epsilon_M)^{1 + 4r(q+1)}.
\] (16)

Proof. Since only pseudo-laws are processed and we always have Equation (6), no overflow occurs in Algorithm 1. Furthermore, the condition in Theorem 2 ensures that all processed floating-point numbers are normalized and hence no underflow occurs.

By Lemma 1, to compute a convolution or a product of two pseudo-laws, the floating-point arithmetic yields a factor upper-bounded (resp. lower-bounded) by \((1 + \epsilon_M)^q\) (resp. \((1 - \epsilon_M)^q\)). Thus, comparing Algorithms 1 and 2, we add (possible) errors by floating-point representation of \( D_i, D_i^\ast, D_{c_i}, \) and \( D_{c_i} \), and count the number of convolutions, and then find that the following integers satisfy

\[
\begin{align*}
D &\sim D^\ast(1 + \epsilon_M)^m_i; \\
(D_i \oplus D_{c_i}) &\sim (D_i^\ast \oplus D_{c_i}^\ast)(1 + \epsilon_M)^m_i, \quad 1 \leq i \leq n; \\
(D_i^{\text{d-bl}} \oplus D_0) &\sim (D_i^{\text{d-bl}} \oplus D_0^\ast)(1 + \epsilon_M)^m_i, \quad r_{i-1} = 1, 1 \leq i \leq n; \\
(D_i \oplus D_{c_{i-1}}) &\sim (D_i^\ast \oplus D_{c_{i-1}}^\ast)(1 + \epsilon_M)^m_i; \\
D_{\text{fin}} &\sim D_{\text{fin}}^\ast(1 + \epsilon_M)^m_{\text{fin}}.
\end{align*}
\] (18)

Explicitly, these integers are

\[
\begin{align*}
m_0 &= 3q + 4; \\
m_i^\text{d-bl} &= -q + 4(q + 1) \cdot \sum_{j=0}^{i-1} r_{i-j} 2^j, \quad 1 \leq i \leq n; \\
m_i &= -q + 4(q + 1) \cdot \sum_{j=0}^{i-1} r_{i-j} 2^j, \quad 1 \leq i \leq n; \\
m_{\text{fin}} &= 1 + 4r(q + 1).
\end{align*}
\] (19)

Finally, if \( 0 \leq a < b \) and \( f \sim g(1 + \epsilon_M)^a \) then \( f \sim g(1 + \epsilon_M)^b \). Therefore, the proof of Lemma 4 is completed since \( m_{\text{fin}} > \max\{m_0, m_i, m_i^\text{d-bl} : 1 \leq i \leq n\} \).

Remark 2. In the proof of Lemma 4, we conservatively choose \( m_0 = 3q + 4, \) where the addend \( 3q \) is attributed to two \( \odot \)’s and one \( \oplus \) in Line 1 of Algorithm 1, and the addend \( 4 \) is attributed to representing \( D_{c_i}, D_{c_i}^\ast, D_{c_{i-1}} \), and \( D_{c_{i-1}}^\ast \) in floating-point numbers. Anyhow, practical cryptosystems are likely to allow smaller \( m_0 \). On the one hand, their secret and errors are distributed in a comparatively small interval rather than the whole range \( \mathbb{Z}_q \) and hence one \( \odot \) there contributes a relative error much tamer than \((1 + \epsilon_M)^q\). Taking Frodo640 [24] for example, the relative error of \( \odot \) is bounded by \((1 + \epsilon_M)^{625}, \) far tamer than \((1 + \epsilon_M)^{257}\). On the other hand, the input distributions can be exactly represented on a machine. Actually, in CRYSTALS-Kyber [15], SABER [25], and FrodoKEM [24], all input distributions are evaluated as fractions with a power-of-two denominator and are hence stored as accurate floating-point numbers.

Part II (Lemma 5): The changes of pseudo-laws by trimming in Algorithm 2 are upper-bounded.

Lemma 5. Let \( E \) be any of the pseudo-laws \( D_i \oplus D_{c_i}, D_i, D_i^\ast, \) and \( D_i^{\text{d-bl}} \oplus D_0 \), \( 1 \leq i \leq n \) in Algorithm 2, and let \( \text{Trim}(E) \) denote its corresponding trimming among \( D_{\text{fin}}, D_i, D_i^\ast \), and \( D_i^{\text{d-bl}} \) in Algorithm 2. Then

\[
\Delta(E, \text{Trim}(E)) \leq q\beta(1 - \epsilon_M)^{-1 + 4r(q+1)}.
\] (20)
\[ \Delta(\text{F, Trim}_S(\text{F})) = \sum_{a \in S} F(a) \leq |S| \cdot \max_{a \in S} F(a) \leq q \cdot \max_{a \in S} F(a). \] (21)

Similarly, we also have
\[
\Delta(D^0, D^1) = \Delta(\text{Trim}_{S_0}(D^0), D^1) \leq q \cdot \max_{a \in S_0} D^1(a) \quad \text{(using Equation (21))}
\leq q \cdot (1 - \epsilon_M)^{-1 - 4r(q+1)} \max_{a \in S_0} D(a) \quad \text{(using Lemma 4)}
\leq q \beta (1 - \epsilon_M)^{-1 - 4r(q+1)}. \quad \text{(using Algorithm 2)}
\] (22)

Part III (Lemma 6): Using Part II, we upper-bound the statistical distance between \( D^1_{\text{fin}} \) and \( P_{\text{fin}} \).

\[
\Delta(P_{\text{fin}}, D^1_{\text{fin}}) \leq \Delta(P_{\text{fin}}, D^1_n \oplus D^1_{\epsilon_1}) + \Delta(D^1_n \oplus D^1_{\epsilon_1}, D^1_{\text{fin}}) \quad \text{(by Lemma 2)}
\leq \Delta(P_n \oplus D^1_n, D^1_n \oplus D^1_{\epsilon_1}) + \Delta(D^1_n \oplus D^1_{\epsilon_1}, D^1_{\text{fin}}) \quad \text{(by Equation (3))}
\leq \Delta(P_n, D^1_n) + \Delta(D^1_n, D^1_{\epsilon_1}) + \Delta(D^1_n \oplus D^1_{\epsilon_1}, D^1_{\text{fin}}) \quad \text{(by Lemma 3)}
\leq \Delta(P_n, D^1_n) + \beta (1 - \epsilon_M)^{-1 - 4r(q+1)}. \quad \text{(by Lemma 5)}
\] (24)

Similarly, we also have
\[
\begin{align*}
\Delta(P, D^1) &\leq \Delta(P^\text{db}, D^1_{\text{db}}^e) + r_{n-1} \cdot 2q \beta (1 - \epsilon_M)^{-1 - 4r(q+1)}; \\
\Delta(P^\text{db}, D^1_{\text{db}}) &\leq 2 \cdot \Delta(P_{n-1}, D_{n-1}^1) + \beta (1 - \epsilon_M)^{-1 - 4r(q+1)}. \quad \text{(25)}
\end{align*}
\]

\[
\begin{align*}
\Delta(P_{\text{fin}}, D^1_{\text{fin}}) &\leq \beta (1 - \epsilon_M)^{-1 - 4r(q+1)} + \sum_{i=1}^{n} 2^{n-i} \cdot q \beta (1 - \epsilon_M)^{-1 - 4r(q+1)} \cdot 2^n + \sum_{i=1}^{n} 2^{n-i} \\
&= q \beta (1 - \epsilon_M)^{-1 - 4r(q+1)} (1 + 2r_{n-1}) \\
&= 2q \beta (1 - \epsilon_M)^{-1 - 4r(q+1)}. \quad \text{(26)}
\end{align*}
\]

Part IV: Using Part I and III, we characterize how the output \( \delta_{\text{alg}} \) returned by Algorithm 1 approximates the DFR \( \delta_{\text{gal}} \). The detailed proof of Equation (25) is included in Appendix C.

\[
\text{Lemma 7. If pseudo-laws } D_1, D_2, E_1, E_2 \text{ satisfy } D_1 \leq E_1 \text{ and } D_2 \leq E_2, \text{ then}
D_1 \oplus D_2 \leq E_1 \oplus E_2. \quad \text{(27)}
\]

\]
Proof of Theorem 2. Since \( \text{Trim}_{\delta}(E) \leq E \) for any pseudo-law \( E \), the pseudo-laws in Algorithm 2 satisfy

\[
\begin{align*}
D^0_o & \leq D^1 = P_o; \\
D^i_{\text{dbl}} & \leq D^i_{\text{rel}} \otimes D^i_{\text{rel}}, \quad 1 \leq i \leq n; \\
D^i_{\text{fin}} & \leq D^i_{\text{fin}} \otimes D^i_{\text{rel}}, \\
\end{align*}
\]

(28)

Comparing Equations (3) and (28), from Lemma 7 we derive that

\[
D^i_{\text{fin}} \leq P_{\text{fin}}.
\]

(29)

Since \( e^T s_2 - s_1^T e_2 \mod q \) is the sum of \( r \) independent random variables with the same distribution \( P_o \), it holds that

\[
\delta_{\text{fail}} = \sum_{a \in \mathbb{Z}_q} \Pr[a \in [-t, t)] P_{\text{fin}}(a).
\]

(30)

Recall that

\[
\delta_{\text{alg}} = \sum_{a \in \mathbb{Z}_q} \Pr[a \in [-t, t)] D^i_{\text{fin}}(a).
\]

(31)

Therefore, it follows from Equation (29) and Lemma 6 that

\[
0 \leq \delta_{\text{fail}} - \delta_{\text{alg}} = \Delta(P_{\text{fin}}, D^i_{\text{fin}}) \leq 2qr \beta (1 - \epsilon_M)^{1 + 4r(q+1)}.
\]

(32)

In addition, it follows from Lemma 4 that

\[
\delta_{\text{alg}}^i (1 - \epsilon_M)^{1 + 4r(q+1)} - \delta_{\text{alg}}^i \leq \delta_{\text{alg}}^i (1 + \epsilon_M)^{1 + 4r(q+1)}.
\]

(33)

Finally, the proof concludes by combining Equations (32) and (33).

\[ \square \]

Remark 3. In Algorithm 1 the trimming in Lines 2 and 12 are optional. Whether the two trimmings are skipped or not will affect the lower-bound in Equation (15), but the impact is not significant as implied by the proof of Theorem 2.

In the sequel, we always assume that each nonzero value of input distributions \( D_{i_1}, D_{i_2}, D_{i_3}, D_{i_4}, \) and \( D_{i_5} \), is not less than \( \sqrt{\alpha_M} \) since this condition is almost trivial for nowadays LWE-based encryption schemes.

Conventionally, the failure probabilities are expressed as powers of two, and their exponents are concerned and compared \([15, 23, 24]\). Therefore, we take \( \log_2(\delta_{\text{fail}}) \) as the final result we expect, and aim to control the absolute/relative error of \( \log_2(\delta_{\text{alg}}) \).

Corollary 1. Let \( \epsilon_{\text{abs}} > 0 \) and \( \epsilon_{\text{rel}} > 0 \). The statements below hold.

(i) If \( \epsilon_M \leq 1 - 2^{-\epsilon_{\text{abs}}/(1+4r(q+1))} \) and

\[
\sqrt{\alpha_M} \leq \beta \leq \frac{\delta_{\text{fail}}}{2qr} \cdot \left( (1 - \epsilon_M)^{1 + 4r(q+1)} - 2^{-\epsilon_{\text{abs}}} \right).
\]

then

\[
|\log_2(\delta_{\text{alg}}) - \log_2(\delta_{\text{fail}})| \leq \epsilon_{\text{abs}}.
\]

(35)

(ii) If \( \epsilon_M \leq 1 - \delta_{\text{fail}}/(1 + 4r(q+1)) \) and

\[
\sqrt{\alpha_M} \leq \beta \leq \frac{\delta_{\text{fail}}}{2qr} \cdot \left( (1 - \epsilon_M)^{1 + 4r(q+1)} - \delta_{\text{fail}} \right).
\]

then

\[
|\log_2(\delta_{\text{alg}}) - \log_2(\delta_{\text{fail}}) - 1| \leq \epsilon_{\text{rel}}.
\]

(37)

Corollary 1 is straightforward from Theorem 2, and its proof is included in Appendix D.

Corollary 2. Let \( \epsilon_{\text{abs}} > 0 \) and \( \epsilon_{\text{rel}} > 0 \). If

\[
\epsilon_M \leq 1 - 2^{-\epsilon_{\text{abs}}/(1+4r(q+1))}
\]

and

\[
\sqrt{\alpha_M} \leq \beta \leq \frac{\delta_{\text{alg}}}{2qr(1 + \epsilon_M)^{1 + 4r(q+1)}}
\]

then Equation (35) holds. If

\[
\epsilon_M \leq (1 - \epsilon_M)^{-\epsilon_{\text{rel}}} (\delta_{\text{alg}} + 2qr \beta)^{\epsilon_{\text{rel}}/(1 + 4r(q+1))}
\]

and

\[
\epsilon_M \leq 1 - \frac{(\delta_{\text{alg}} + 2qr \beta)^{\epsilon_{\text{rel}}}}{(1 - \epsilon_M)^{1 + 4r(q+1)}(1 + \epsilon_{\text{rel}})}
\]

(40)

then Equation (37) holds.

Combining Corollary 1 and the inequality Equation (15) derives Corollary 2, and it gives a sufficient condition to verify whether Algorithm 1 returns an approximation with required precision.
Table 1: Verify the precision of $\delta_{flg}$ in Schwabe [15], Alkim et al. [24], and D’Anvers et al. [25].

<table>
<thead>
<tr>
<th>Cipher</th>
<th>$\varepsilon_M$</th>
<th>r.h.s. Equation (38)</th>
<th>r.h.s. Equation (40)</th>
<th>$\beta$</th>
<th>r.h.s. Equation (39)</th>
<th>r.h.s. Equation (41)</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kyber512</td>
<td>2$^{-53}$</td>
<td>2$^{-31.20}$</td>
<td>2$^{-33.96}$</td>
<td>2$^{-300}$</td>
<td>2$^{-176.82}$</td>
<td>2$^{-179.59}$</td>
<td>Y</td>
</tr>
<tr>
<td>Kyber768</td>
<td>2$^{-53}$</td>
<td>2$^{-31.78}$</td>
<td>2$^{-34.31}$</td>
<td>2$^{-300}$</td>
<td>2$^{-203.47}$</td>
<td>2$^{-206.00}$</td>
<td>Y</td>
</tr>
<tr>
<td>Kyber1024</td>
<td>2$^{-53}$</td>
<td>2$^{-32.20}$</td>
<td>2$^{-34.64}$</td>
<td>2$^{-300}$</td>
<td>2$^{-213.84}$</td>
<td>2$^{-216.29}$</td>
<td>Y</td>
</tr>
<tr>
<td>LightSaber</td>
<td>2$^{-53}$</td>
<td>2$^{-32.17}$</td>
<td>2$^{-35.13}$</td>
<td>2$^{-300}$</td>
<td>2$^{-213.84}$</td>
<td>2$^{-162.50}$</td>
<td>Y</td>
</tr>
<tr>
<td>Saber</td>
<td>2$^{-53}$</td>
<td>2$^{-32.76}$</td>
<td>2$^{-35.55}$</td>
<td>2$^{-300}$</td>
<td>2$^{-175.93}$</td>
<td>2$^{-178.73}$</td>
<td>Y</td>
</tr>
<tr>
<td>FireSaber</td>
<td>2$^{-53}$</td>
<td>2$^{-33.17}$</td>
<td>2$^{-35.70}$</td>
<td>2$^{-300}$</td>
<td>2$^{-205.45}$</td>
<td>2$^{-207.97}$</td>
<td>Y</td>
</tr>
<tr>
<td>Frodo640</td>
<td>2$^{-64}$</td>
<td>2$^{-34.49}$</td>
<td>2$^{-35.58}$</td>
<td>2$^{-200}$</td>
<td>2$^{-502.79}$</td>
<td>2$^{-503.88}$</td>
<td>Y</td>
</tr>
<tr>
<td>Frodo976</td>
<td>2$^{-64}$</td>
<td>2$^{-36.10}$</td>
<td>2$^{-37.66}$</td>
<td>10$^{-200}$</td>
<td>2$^{-374.65}$</td>
<td>2$^{-376.21}$</td>
<td>Y</td>
</tr>
<tr>
<td>Frodo1344</td>
<td>2$^{-64}$</td>
<td>2$^{-36.56}$</td>
<td>2$^{-38.51}$</td>
<td>10$^{-200}$</td>
<td>2$^{-294.17}$</td>
<td>2$^{-296.12}$</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Table 2: Estimate machine precision for testing DFR of FrodoKEM.**

<table>
<thead>
<tr>
<th>$q$</th>
<th>$r$</th>
<th>$\varepsilon_{abs} = 5 \times 10^{-3}$</th>
<th>$\varepsilon_{rel} = 5 \times 10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2$^{15}$</td>
<td>640</td>
<td>$\varepsilon_M \leq 2^{-34.50}$</td>
<td>$\varepsilon_M \leq 2^{-44.47}$</td>
</tr>
<tr>
<td>2$^{16}$</td>
<td>976</td>
<td>$\varepsilon_M \leq 2^{-36.11}$</td>
<td>$\varepsilon_M \leq 2^{-46.07}$</td>
</tr>
<tr>
<td>2$^{16}$</td>
<td>1,344</td>
<td>$\varepsilon_M \leq 2^{-36.57}$</td>
<td>$\varepsilon_M \leq 2^{-46.55}$</td>
</tr>
</tbody>
</table>

For most nowadays lattice-based encryption schemes and KEMs, their parameters satisfy $q \leq 2^{16}$ and $r \leq 2^{14}$, and their DFRs are usually located in the range $2^{-256} < \delta_{fail} < 2^{-40}$. Hence, the required absolute (resp. relative) error $\varepsilon_{abs} = 5 \times 10^{-3}$ (resp. $\varepsilon_{rel} = 5 \times 10^{-6}$) suffices. Under such conditions, $\varepsilon_M \leq 2^{-43.82}$ and $\alpha_M \leq 2^{-623.94}$ satisfy Equations (42) and (43). Therefore, the double precision (64 bit) floating-point is sufficient to run Algorithm 1 on ciphers with such parameters.

We have to remark that (i) the above datatype selection is based on the practical range of $\delta_{fail}$, while Algorithm 3 in the next subsection selects datatype only dependent on cipher parameters; (ii) lattice-based cryptosystems in other scenarios, for example, fully homomorphic encryption, may use other parameters and hence require distinct machine precision.

**Experiment 2.** In Table 2, we set $\varepsilon_{abs} = 5 \times 10^{-3}$, $\varepsilon_{rel} = 5 \times 10^{-6}$, and list the parameters of FrodoKEM [24] and their corresponding $\varepsilon_M$ estimated in Equation (42) (here conservatively using $\log_2 \delta_{fail} \leq -1$). Neither Equations (42) nor (43) is satisfied for single precision floating-point numbers. Running Algorithm 1 in 32 bit floating-point arithmetic fails to approximate $\delta_{fail}$. Anyhow, it suffices to use double precision (64 bit) floating-point instead of float128 in the python numpy package to find the DFR $\delta_{fail}$ of FrodoKEM, and this is effective as confirmed by Experiment 3. This experiment suggests that Equation (42) is effective for selecting floating-point datatype.

3.3. A Hybrid Test of DFR with Progressive Trimming. Now we propose a new test (Algorithm 3) of DFR.

Algorithm 3 calls Algorithm 1 as its inner core subprocedure, and it selects the trimming threshold $\beta$ in a progressive way. Specifically, the heuristic estimate $\delta_{fit}$ through a continuous normal distribution helps to decide $\beta$ for a tentative test, denoted by $\beta_{fit}$ for the absolute error $\varepsilon_{abs}$ (resp. by $\beta_{rel}$ for the relative error $\varepsilon_{rel}$), and then an expected
**Input:** the modulus $q$; the distributions $D_{e_1}, D_{e_2}, D_{s_1}, D_{s_2}$ of coordinates of $e_1, e_2, s_1, s_2$, respectively; the distribution $D_{e_3}$ of $e_3$; the dimension $r$; the critical value $t$ of decryption failure and $e_{abs} > 0$ (resp. $e_{rel} > 0$).

**Output:** estimate the DFR $\delta_{fail}$.

1: Always select the floating-point datatype satisfying $\epsilon_M \leq 1 - 2^{-e_{abs}/(1+4r(q+1))}$ (resp. $\epsilon_M \leq 1 - 2^{-e_{rel}/(1+4r(q+1))}$).

2: Compute the distribution $D$ of $e_1 \cdot s_2 - s_1 \cdot e_2$, where $s_1 \leftarrow D_{s_1}, s_2 \leftarrow D_{s_2}, e_1 \leftarrow D_{e_1}$, and $e_2 \leftarrow D_{e_2}$. [This step is the same as Line 1 of Algorithm 1, and all the three tests below share $D$ as an input.]

3: **[A heuristic test]** Use the central limit theorem to approximate the DFR, for example, by Algorithm 4. Denote its returned value by $\delta_{clt}$.

4: **[A tentative test]** includes Lines 5–7.

5: Set

$$
\beta_{abscnt} = \frac{\delta_{cli} \cdot (1 - \epsilon_M)^{1+4r(q+1)} - 2^{-e_{as}}}{2^{pr}(1 + \epsilon_M)^{1+4r(q+1)}}
$$

6: Select the floating-point datatype such that $\alpha_M \leq \beta_{abscnt}^2$ (resp. $\beta_{abscnt}$).

7: Run Algorithm 1 with $\beta = \beta_{abscnt}$ (resp. $\beta = \beta_{relcnt}$). Denote its returned value by $\delta_{abscnt}$ (resp. $\delta_{relcnt}$). [Skip Line 1 of Algorithm 1 as $D$ is already available.]

8: **[A confirmatory test]** includes Lines 9–15.

9: Set

$$
\beta_{abscnt} = \frac{\delta_{abscnt} \cdot (1 - \epsilon_M)^{1+4r(q+1)} - 2^{-e_{as}}}{2^{pr}(1 + \epsilon_M)^{1+4r(q+1)}}
$$

10: if $\beta_{abscnt} < \beta_{abscnt}$ (resp. $\beta_{abscnt} < \beta_{relcnt}$) then

11: Select the floating-point datatype such that $\alpha_M \leq \beta_{abscnt}^2$ (resp. $\beta_{abscnt}$).

12: Run Algorithm 1 with $\beta = \beta_{abscnt}$ (resp. $\beta = \beta_{relcnt}$). Denote its returned value by $\delta_{abscnt}$ (resp. $\delta_{relcnt}$). [Skip Line 1 of Algorithm 1 as $D$ is already available.]

13: else

14: Set $\delta_{abscnt} = \delta_{abscnt}$ (resp. $\delta_{relcnt} = \delta_{relcnt}$).

15: end if

16: **return** $\delta_{abscnt}$ (resp. $\delta_{relcnt}$).

**Algorithm 3: A Hybrid test of DFR.**

better approximation $\delta_{abscnt}$ (resp. $\delta_{relcnt}$) obtained by the tentative test determines $\beta$ for a confirmatory test, denoted by $\beta_{abscnt}$ for the absolute error $\epsilon_{abs}$ (resp. by $\beta_{relcnt}$ for the relative error $\epsilon_{rel}$). The final output $\delta_{abscnt}$ (resp. $\delta_{relcnt}$) of the confirmatory test is ensured to satisfy the required precision.

**Theorem 3.** If $\delta_{fail} \leq 1/2$, then

$$
\begin{align*}
|\log_2 \delta_{abscnt} - \log_2 \delta_{fail}| & \leq \epsilon_{abs}; \\
|\log_2 \delta_{relcnt} / \log_2 \delta_{fail} - 1| & \leq \epsilon_{rel}.
\end{align*}
$$

**Proof.** Above all, the conditions Equation (42) in Corollary 1 follows from $\delta_{fail} \leq 1/2$, and the range of floating-point numbers in Equations (34) and (36) are ensured in Lines 6 and 11 of Algorithm 3.

Moreover, by Theorem 2, the tentative test ensures to bound $\delta_{fail}$ as below

$$
\delta_{clt} = \frac{1}{2} \sum_{x \in \mathbb{Z}_q} D_{e_3}(x) \cdot \left( \text{erfc} \left( \frac{t - r \cdot m_D - x}{\sqrt{2r \cdot \sigma_D^2}} \right) + \text{erfc} \left( \frac{t + r \cdot m_D + x}{\sqrt{2r \cdot \sigma_D^2}} \right) \right).
$$

Hence, the trimming threshold $\beta_{abscnt}$ (resp. $\beta_{relcnt}$) of Algorithm 3 is upper-bounded by the right hand of Equation (34) (resp. Equation (36)).

If $\beta_{abscnt} < \beta_{abscnt}$ (resp. $\beta_{relcnt} < \beta_{relcnt}$), then Algorithm 3 operates Line 12 and it follows from Corollary 1 that

$$
\begin{align*}
\delta_{abscnt} & \leq \left( \delta_{relcnt} + 2qr \beta_{relcnt} \right) (1 - \epsilon_M)^{-1-4r(q+1)}; \\
\delta_{abscnt} & \geq \delta \cdot (1 + \epsilon_M)^{-1-4r(q+1)}, \delta \in \{\delta_{abscnt}, \delta_{relcnt}\}.
\end{align*}
$$

**Remark 4.** Line 2 of Algorithm 4 can be computed as below

$$
\begin{align*}
\delta_{fail} & \leq \left( \delta_{relcnt} + 2qr \beta_{relcnt} \right) (1 - \epsilon_M)^{-1-4r(q+1)}; \\
\delta_{fail} & \geq \delta \cdot (1 + \epsilon_M)^{-1-4r(q+1)}, \delta \in \{\delta_{abscnt}, \delta_{relcnt}\}.
\end{align*}
$$
De scenarios where strict proof of the DFR is not compulsory. Therefore, the con
below shows that the tentative test is suf
while Equation (47) derives a rough estimate 2
computes
For example, distinct from Algorithm 4, FrodoKEM [24]
ever, it is not unique to implement the heuristic test.

Remark 5. The confirmatory test in Algorithm 3 is not indispen-
sable for specific applications. On the one hand, the
inequalities Equation (15) are conservative and \( \delta_{\text{alg}} \) is likely
to be much closer to \( \delta_{\text{fail}} \). On the other hand, via the central
limit theorem, the heuristic test possibly returns a value very
near \( \delta_{\text{fail}} \). Hence, it is probable that the tentative test already
obtains the DFR with a desirable precision. Experiment 3 below shows
that the tentative test is sufficient for CRYSTALS-Kyber [15], SABER [25],
and FrodoKEM [24]. Therefore, the confirmatory test of Algorithm 3 is optional in
scenarios where strict proof of the DFR is not compulsory.

### Table 3: Trimming thresholds and DFR estimates.

<table>
<thead>
<tr>
<th>Cipher</th>
<th>( \beta )</th>
<th>( \delta_{\text{alg}} )</th>
<th>( \delta_{\text{abst}} )</th>
<th>( \beta_{\text{abst}} )</th>
<th>( \delta_{\text{abst}}(\delta_{\text{abst}}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kyber512</td>
<td>2(^{-300})</td>
<td>2(^{-139}) 49</td>
<td>2(^{-185.49})</td>
<td>2(^{-188.18})</td>
<td>2(^{-138.94})</td>
</tr>
<tr>
<td>Kyber768</td>
<td>2(^{-300})</td>
<td>2(^{-164}) 73</td>
<td>2(^{-211.30})</td>
<td>2(^{-213.76})</td>
<td>2(^{-165.81})</td>
</tr>
<tr>
<td>Kyber1024</td>
<td>2(^{-300})</td>
<td>2(^{-174}) 73</td>
<td>2(^{-220.20})</td>
<td>2(^{-222.60})</td>
<td>2(^{-174.96})</td>
</tr>
<tr>
<td>LightSaber</td>
<td>2(^{-300})</td>
<td>2(^{-120}) 49</td>
<td>2(^{-162.98})</td>
<td>2(^{-165.90})</td>
<td>2(^{-120.35})</td>
</tr>
<tr>
<td>Saber</td>
<td>2(^{-300})</td>
<td>2(^{-136}) 73</td>
<td>2(^{-178.83})</td>
<td>2(^{-181.60})</td>
<td>2(^{-136.16})</td>
</tr>
<tr>
<td>FireSaber</td>
<td>2(^{-300})</td>
<td>2(^{-165}) 73</td>
<td>2(^{-208.50})</td>
<td>2(^{-211.00})</td>
<td>2(^{-165.26})</td>
</tr>
<tr>
<td>Frodo640</td>
<td>10(^{-200})</td>
<td>2(^{-138.7})</td>
<td>2(^{-188.37})</td>
<td>2(^{-191.06})</td>
<td>2(^{-138.76})</td>
</tr>
<tr>
<td>Frodo976</td>
<td>10(^{-200})</td>
<td>2(^{-199.6})</td>
<td>2(^{-254.40})</td>
<td>2(^{-256.59})</td>
<td>2(^{-199.60})</td>
</tr>
<tr>
<td>Frodo1344</td>
<td>10(^{-200})</td>
<td>2(^{-252.5})</td>
<td>2(^{-310.07})</td>
<td>2(^{-311.93})</td>
<td>2(^{-252.60})</td>
</tr>
</tbody>
</table>

where erfc denotes the complementary error function. How-
ever, it is not unique to implement the heuristic test.
For example, distinct from Algorithm 4, FrodoKEM [24] computes

\[
\delta_{\text{clt}} \approx \Pr_{x \sim \mathcal{N}(\mu_{D_{\text{exit}}}, \sigma_{D_{\text{exit}}})} \left[ |x| \geq t \right],
\]  

(47)

where \( \mu_{D_{\text{exit}}} \) and \( \sigma_{D_{\text{exit}}}^2 \) denote the mean and the variance of \( D_{\text{exit}} \), respectively. Generally speaking, Algorithm 4 costs more

### 3.4. An Experiment of DFR Test. Through the following experiment we compare Algorithm 3 with the previous

**Experiment 3.** For parameter sets of CRYSTALS-Kyber [15],
SABER [25], and FrodoKEM [24], we run Algorithm 1 without
trimming (\( \beta = 0 \)), Algorithm 1 with previous practical trimming
[15, 24, 25] and also Algorithm 3. The absolute (resp. relative) error
for log2 \( \delta_{\text{fail}} \) in Algorithm 3 is set \( \epsilon_{\text{abs}} = 5 \times 10^{-3} \) (resp. \( \epsilon_{\text{rel}} = 5 \times 10^{-6} \). A fair comparison, all tests employ the convolution
speedup from FrodoKEM [24] (as in Remark 1). Pseudo-laws are
memorized and processed in double precision (64 bit) floating-point
numbers. The computation is programmed in Python, compiled by
Visual Studio Community 16.11.17, and operated on Intel(R)
Core(TM) i5-8500U CPU 1.70 GHz with memory 8 GB. The results
(including all trimming thresholds, DFR estimates, and
time costs) are detailed in Tables 4–12 of Appendix E.

On the other hand, the data of Experiment 3 show that
Algorithm 3, grounded on its theoretical proof (Theorem 3),
ensures high accuracy to approximate \( \delta_{\text{fail}} \) though its con-
volutions neglect more tiny probabilities than previous prac-
tical methods. In Table 3, the second column lists the
trimming thresholds in previous tests, and the fourth and
fifth columns list the trimming thresholds used in Algorithm
3; the third column lists the previously given DFRs in the
submissions to NIST [14], and the last column lists the DFRs
outputted by Algorithm 3.

On the other hand, Algorithm 3 outperforms previous
practical DFR tests in efficiency for all parameter sets of
CRYSTALS-Kyber, SABER, and FrodoKEM. The experiment
data show that

(i) All the parameter sets dissatisfy the condition in Line
10 of Algorithm 3 and the confirmatory test is there-
fore almost free.

(ii) As in Figure 2, among all nine parameter sets, \( s \) achieves
its minimum 5.92\% for Frodo640 and its maximum
85.84\% for Kyber768, where \( s \) denotes the ratio of time running Algorithm 3 for the

![FIGURE 2: Ratio of time cost of Algorithm 3 to that of Algorithm 1.](image-url)
3.5. Use the Test for Practical Encryption Schemes. In the above, we only discussed the DFR determined by the distribution of \( e_1^t s_2 - s_1 e_2 + e_3 \mod q \), setting other forms aside. When the plaintext is longer and enciphered in more than one elements of \( \mathbb{Z}_q \), where \( e_2 \) in Figure 1 is parallelized as a matrix over \( \mathbb{Z}_q \), incorporating the union bound into Algorithm 3 will estimate the DFR of the encryption scheme. Anyhow, a practical lattice-base encryption scheme probably integrates other techniques and computes its DFR in other ways. In the rest of this subsection, we analyze the influence of algebraic lattices and the rounding compression on decryption failure, and also consider using the test for lattice-based IBE/ABE schemes.

3.5.1. The Impact of Using Structured Lattices. Lyubashevsky et al. [20] proposed the LWE over rings and also an algebraic version of the Lindner–Peikert cryptosystem (Figure 3). Despite variants of structured lattices in cryptography [37], here we consider the following algebraic lattice utilized in most practical schemes.

Let \( K \) be a number field of degree \( r \), \( \mathcal{R} \) an order of \( K \), and \( b_0, b_1, \ldots, b_{r-1} \) a basis of \( \mathcal{R} \). Denote the quotient ring \( \mathcal{R}/q\mathcal{R} \) by \( \mathcal{R}_q \).

In Figure 3, \( u \leftarrow SD^B \mathcal{R} \) means each coefficient of \( u \) with respect to the basis \( \{ b_0, b_1, \ldots, b_{r-1} \} \) is sampled from \( D \). Denote \( e_m = \sum_{k=0}^{r-1} \lambda_k^j b_k \) for \( m \in \{ 1, 2, 3 \} \) and \( s_m = \sum_{k=0}^{r-1} \gamma_k^j b_k \) for \( m \in \{ 1, 2 \} \).

Similar to Equation (1), decryption fails in this encryption scheme if

\[
\| e_1 s_2 - s_1 e_2 + e_3 \|_\infty \notin [-t, t),
\]

where \( \| e_1 s_2 - s_1 e_2 + e_3 \|_\infty \) denotes the greatest absolute value of the coefficients of \( e_1 s_2 - s_1 e_2 + e_3 \mod q \) with respect to the basis \( \{ b_0, b_1, \ldots, b_{r-1} \} \). Let \( b_i \cdot b_i = \sum_{k=0}^{r-1} \lambda_i^j b_k \) for \( 0 \leq i, j \leq r - 1 \). Then the coefficient of \( b_k \) in \( e_1 s_2 - s_1 e_2 + e_3 \mod q \) is

\[
\sum_{0 \leq j < r} \lambda_i^j (e_i^j s_2^j - s_1^j e_2^j + e_3^j) \mod q.
\]

Let \( D = (D_0 \odot D_s) \odot (C_{-1} \odot D_0 \odot D_0) \) as in Line 2 of Algorithm 3. Then the distribution of Equation (49) is computed by

\[
\left( \odot_{0 \leq t < \epsilon}(C_{\epsilon} D) \odot D_{c, t} \right).
\]

Therefore, we conclude that testing DFR depends on the algebraic rings and their chosen basis, and the “double-and-add” method is not universally effective.

Fortunately, rings in the present practical encryption schemes cause not much trouble. The power-of-two cyclotomic ring \( \mathcal{R} = \mathbb{Z}[x]/(x^r + 1) \) is the most popular in structured lattices [38], including NewHope [39], CRYSTALS-Kyber [21], and SABER [23]. As the conventional basis is \( b_i = x^i, 0 \leq i \leq r - 1 \), we have

\[
\epsilon_k^{(ij)} = \begin{cases} 1, & 0 \leq i + j < k < r; \\ -1, & r \leq i + j = r + k; \\ 0, & \text{otherwise}. \end{cases}
\]

In this specific case, the distribution of Equation (49) is computed by

\[
D^{[k+1]} \odot (C_{-1} \odot D)^{[r-1-k]} \odot D_{c, r},
\]

where \( D^{[m]} \) denotes the \( m \)-fold convolution of \( D \).

We call a pseudo-law \( D \) to be symmetric if \( D(a) = D(-a) \) for any \( a \in \mathbb{Z}_q \). The following lemma is straightforwardly derived from definitions.

**Lemma 8.** Let \( D_1, D_2 \) be pseudo-laws. If \( D_1 \) is symmetric, then \( D_1 \odot D_2 \) is symmetric. If \( D_1 \) and \( D_2 \) are symmetric, then \( D_1 \odot D_2 \) is symmetric.

In practical schemes, most secrets and errors comply with symmetric laws, for example, the centered binomial distribution and the discrete approximate Gaussian in FrodoKEM [24]. By Lemma 8, due to symmetry of \( D \) in such schemes, Equation (52) is exactly \( D^{[i]} \odot D_{c, i} \) and it is therefore feasible to compute \( \epsilon_{\text{had}} \) by Algorithms 1 and 3.

Recall that Algorithm 3 proceeds decryption failure of one coordinate of the algebraic number. If the encryption scheme based on structured lattices employs no error correcting codes, then taking the \( r \) coordinates of \( e_1 s_2 - s_1 e_2 + e_3 \) as independent random variables is appropriate [40]; on the contrast, using the independence assumption in such cryptosystems with error correcting codes possibly results in overestimation of the DFR and a method has been proposed to calculate the DFR for those schemes [40].

<table>
<thead>
<tr>
<th>Key generation</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( a \leftarrow \mathcal{R}_q )</td>
<td>1: ( s_2 \leftarrow D^B_0 )</td>
</tr>
<tr>
<td>2: ( s_1 \leftarrow D^B_1 )</td>
<td>2: ( e_2 \leftarrow D^B_2 )</td>
</tr>
<tr>
<td>3: ( e_1 \leftarrow D^B_0 )</td>
<td>3: ( e_3 \leftarrow D^B_0 )</td>
</tr>
<tr>
<td>4: ( b = a s_3 + e_1 )</td>
<td>4: ( c_1 = a s_3 + e_2 )</td>
</tr>
<tr>
<td>5: ( \text{pk} = (a, b) )</td>
<td>5: ( s_3 = b s_2 + e_3 + \text{Encode} (m) )</td>
</tr>
<tr>
<td>6: ( \text{sk} = s_2 )</td>
<td>6: ( c_3 = (c_1, c_2) )</td>
</tr>
</tbody>
</table>

**Figure 3:** The Lindner–Peikert encryption scheme using lattices over rings [20].
In addition, the above test of DFR can naturally extend to cryptosystems based on module-LWE [21, 41, 42] or module-LWR [23].

3.5.2. The Impact of Compressing Public Key/Ciphertexts. Let \( p \) be a positive integer less than \( q \). The rounding function maps \( x \in \mathbb{Z}_q \) to \([xp/q], q\). This operation naturally extends on vectors in \( \mathbb{Z}_q^n \) and algebraic numbers in \( \mathcal{A}_q \). This technique is used to reduce bandwidth. Conventionally, the truncated information is also taken as errors [21, 23]. Let \( D_{p_1}, D_{p_2}, \) and \( D_{p_3} \), respectively, denote the distributions by compressing the public key \( b \) (or \( b \)) and ciphertext \( c_1, c_2 \) (or \( c_1, c_2 \)) in Figure 1 (or Figure 3). According to [21, Theorem 1], the DFR is computed in the same as above except for that in Line 2 of Algorithm 3 and in Line 1 of Algorithm 1

\[
D = ((D_{c_1} \odot D_{p_1}) \odot D_{c_2}) \odot (C_{-1} \odot (D_{c_3} \odot D_{p_2}) \odot D_{c_4}).
\]  

(53)

The distributions from rounding are not necessarily symmetric. Fortunately, by Lemma 8, the encryption schemes with symmetric secret distributions have symmetric \( D \). Therefore, Algorithms 1 and 3 are able to test their DFR, with slight adaptation as in Equation (53).

Furthermore, the same as in Remark 2, changes in computing \( D \) lead to distinct \( m_0 \)'s in the proof of Lemma 4. The involved results following from it should be adjusted and this is straightforward. For example, CRYSALS-Kyber [15] in the third round of NIST PQC program [14], different from its previous version, compresses only ciphertexts, that is, \( D_{p_1} = C_0 \). Note that the operation \( \odot \) with \( C_{-1} \) results in no loss of precision. Counting in two \( \oplus \)'s, two \( \otimes \)'s and relative errors in floating-point representation of \( D_{p_2} \) and \( D_{p_3} \), it yields \( m_0 = 4q + 2 \).

5.3.3. Test DFR in Lattice-Based IBE/ABE Schemes. In typical constructions of lattice-based IBE [43, 44] and ABE [45–47], instead of using the original Regev encryption scheme [18], its dual version is used as a primitive, in which the key generation and encryption procedures are essentially swapped. Specifically, in the dual system with unstructured lattices (Figure 4), the secret key is a short vector \( s_1 \), and the corresponding public key is its syndrome \( b = A^T s_1 \in \mathbb{Z}_q^r \). The encryption algorithm chooses a pseudorandom LWE vector \( c_1 = As_2 + e_2 \mod q \), and uses the syndrome \( b \) to generate one more LWE instance as a “pad” to hide the message, i.e., \( c_2 = b^T s_2 + e_3 + \text{Encode}(m) \). The decryption algorithm proceeds similarly as in Regev [18] and Lindner and Peikert [19].

Then the key to obtaining \( \delta_{\text{fail}} \) is to compute

\[
(C_{-1} \odot (D_{c_1} \odot D_{c_2})^{\|w|} \odot D_{c_4}),
\]  

which characterizes the distribution of \(-s_1^T e_2 + e_3 \mod q \).

Therefore, the results above in this paper also work for the dual Regev cryptosystem with slightly adaption.

### Table 1

<table>
<thead>
<tr>
<th>Key generation</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( A \in \mathbb{Z}_q^{m \times n} ) ( \text{// } w &gt; r )</td>
<td>1: ( s_2 \in D_{c_2}^w )</td>
</tr>
<tr>
<td>2: ( s_1 \in D_{c_2}^w )</td>
<td>2: ( e_2 \in D_{c_2}^w )</td>
</tr>
<tr>
<td>3: ( b = A^T s_1 )</td>
<td>3: ( e_3 \in D_{c_3} )</td>
</tr>
<tr>
<td>4: ( \text{pk} = (A, b) )</td>
<td>4: ( c_1 = A s_2 + e_2 )</td>
</tr>
<tr>
<td>5: ( \text{sk} = s_1 )</td>
<td>5: ( c_2 = b^T s_2 + e_3 + \text{Encode}(m) )</td>
</tr>
</tbody>
</table>

**Figure 4:** The dual Regev encryption scheme [43].

In respect of lattice-based ABE, the above method allows to efficiently and precisely estimate the DFR for primitive components, and deciding its DFR of the whole ABE scheme highly depends on specific access structures. For example, \( \delta_{\text{fail}} \) of the threshold ABE [45] can be determined by computing \( \odot(D_1 \odot D_f^{\|w|}) \), where \( D_1 \) and \( D_f \) are pseudo-laws, and Algorithms 1 and 3 with slight modification are effective for such computation.

### 4. Conclusion and Future Work

In this article, we bound the output \( \delta_{\text{alg}} \) of the “double-and-add” method with cipher parameters, the floating-point machine error \( \varepsilon_M \) and the trimming threshold \( \beta \), and we also propose an algorithm to determine the DFR of the LWE-based encryption schemes. The main outcomes are as below.

First, an explicit way is given to select the proper floating-point datatype enabling to output of the DFR with assigned accuracy. Particularly, according to theoretical analysis and experimental verification, the IEEE standardized double precision float-pointing, which is supported by a variety of computing devices and operating systems, suffices for common nowadays lattice-based encryption while single precision (32 bit) floating-point arithmetic does not guarantee a precise approximation.

Second, inequalities in Corollary 2 enables to quantitatively confirm whether the “double-and-add” algorithm returns an estimate satisfying the precision. Particularly, therefrom it immediately follows that \( \log_2 \delta_{\text{fail}} \)'s obtained in CRYSALS-Kyber [15], SABER [25], and ForoKEM [24] are theoretically proved to be precise in respect of a given absolute (resp. relative) error \( \varepsilon_{\text{abs}} = 5 \times 10^{-3} \) (resp. \( \varepsilon_{\text{rel}} = 5 \times 10^{-6} \)).

Third, the proposed new test of DFR includes an explicit criterion to select the trimming threshold \( \beta \) and is theoretically ensured to achieve an assigned precision. Moreover, realistic processing shows that this test accelerates previous “double-and-add” computation with practical trimming. For example, computing \( \delta_{\text{fail}} \) of Frodo640 in double-precision floating-point allows trimming probability less than \( 2^{-191.06} \).
instead of previous $10^{-200} \approx 2^{-664}$, and thereby the new test neglects more distribution data and hence runs faster.

Finally, we analyze the impact of algebraic lattices and the rounding compression, and also consider applying the results in lattice-based IBE/ABE. The "double-and-add" philosophy is effective if the cryptosystem samples symmetric secrets and errors and utilizes the power-of-two cyclotomic ring together with its natural power basis.

We hope that this work can serve as an inspiration to effectively and efficiently test (or search) parameters of lattice-based cryptosystems. For instance, it is interesting to apply the techniques and methods in this paper, adapted if necessary, to estimate the failure probability of LWE-based fully homomorphic encryption schemes.

**Appendix**

A. Proof of Lemma 3

**Proof of Lemma 3.** The proof is by straightforward computation:

\[
\Delta(D_1 \otimes D_2, E_1 \otimes E_2) = \sum_{k \in \mathbb{Z}_q} |D_1 \otimes D_2(k) - E_1 \otimes E_2(k)|
\]

\[
= \sum_{k \in \mathbb{Z}_q} \left| \sum_{a \in \mathbb{Z}_q} D_1(a) \cdot D_2(b) - \sum_{c \in \mathbb{Z}_q} E_1(c) \cdot E_2(d) \right|
\]

\[
= \sum_{k \in \mathbb{Z}_q} \left| \sum_{a, b \in \mathbb{Z}_q, \quad a + b \equiv k \pmod{q}} (D_1(a) \cdot D_2(b) - E_1(a) \cdot E_2(b)) \right|
\]

\[
\leq \sum_{k \in \mathbb{Z}_q} \left| \sum_{a, b \in \mathbb{Z}_q, \quad a + b \equiv k \pmod{q}} (\Delta(D_1, E_1) \cdot D_2(b) + E_1(a) \cdot \Delta(D_2, E_2)) \right|
\]

\[
= \sum_{a, b \in \mathbb{Z}_q} |D_1(a) - E_1(a)| \cdot |D_2(b) + \sum_{c \in \mathbb{Z}_q} E_1(c) \cdot |D_2(d) - E_2(d)|
\]

\[
= \sum_{a, b \in \mathbb{Z}_q} |D_1(a) - E_1(a)| \cdot \sum_{b \in \mathbb{Z}_q} D_2(b) + \sum_{c \in \mathbb{Z}_q} E_1(c) \cdot \sum_{d \in \mathbb{Z}_q} |D_2(d) - E_2(d)|
\]

\[
\leq \Delta(D_1, E_1) + \Delta(D_2, E_2).
\]
B. Part of the proof of Lemma 5

The trimming error from Line 5 is estimated as

\[
\Delta(D^l_{i-1} \otimes D^i_{r-1}, D^i_{l-1}) \\
= \Delta(D^l_{i-1} \otimes D^i_{r-1}, \text{Trim}_{s_{\text{ub}}}(D^l_{i-1} \otimes D^i_{r-1})) \\
\leq q \cdot \max_{a \in S^n_{\text{ub}}} (D^l_{i-1} \otimes D^i_{r-1})(a) \quad \text{using Equation (21)} \\
\leq q \cdot (1 - \epsilon_M)^{4 - \alpha(q+1)} \max_{a \in S^n_{\text{ub}}} (D^l_{i-1} \otimes D^i_{r-1})(a) \quad \text{using Lemma 4} \\
\leq q\beta(1 - \epsilon_M)^{-1 - 4\alpha(q+1)}. \quad \text{using Algorithm 2}
\]

The trimming errors from Line 7 (under the condition \(r_{n-1} = 1\)) is estimated as

\[
\Delta(D^l_{1-\text{dbl}} \otimes D^i_0, D^i_1) \\
= \Delta(D^l_{1-\text{dbl}} \otimes D^i_0, \text{Trim}_{s_{\text{ub}}}(D^l_{1-\text{dbl}} \otimes r_{n-1} \cdot D^i_0)) \\
\leq q \cdot \max_{a \in S^n_{\text{ub}}} (D^l_{1-\text{dbl}} \otimes D^i_0)(a) \quad \text{using Equation (21)} \\
\leq q \cdot (1 - \epsilon_M)^{1 - \alpha\epsilon(q+1)} \max_{a \in S^n_{\text{ub}}} (D^l_{1-\text{dbl}} \otimes D^i_0)(a) \quad \text{using Lemma 4} \\
\leq q\beta(1 - \epsilon_M)^{-1 - 4\alpha(q+1)}. \quad \text{using Algorithm 2}
\]

The trimming error from Line 12 is estimated as

\[
\Delta(D^i_n \otimes D^i_{e_1}, D^i_{\text{lin}}) \\
= \Delta(D^i_n \otimes D^i_{e_1}, \text{Trim}_{S_{n+1}}(D^i_n \otimes D^i_{e_1})) \\
\leq q \cdot \max_{a \in S_{n+1}} (D^i_n \otimes D^i_{e_1})(a) \quad \text{using Equation (21)} \\
\leq q \cdot (1 - \epsilon_M)^{-1 - \alpha\epsilon(q+1)} \max_{a \in S_{n+1}} (D^i_n \otimes D^i_{e_1})(a) \quad \text{using Lemma 4} \\
\leq q\beta(1 - \epsilon_M)^{-1 - 4\alpha(q+1)}. \quad \text{using Algorithm 2}
\]

C. Part of the proof of Lemma 6

Below is the proof of Equation (25). It holds that

\[
\Delta(P_i, D^i_l) \\
\leq \Delta(P_i, D^i_{l-\text{dbl}} \otimes D^i_0) + \Delta(D^i_{l-\text{dbl}} \otimes D^i_0, D^i_l) \quad \text{by Lemma 2} \\
\leq \Delta(P^l_{\text{dbl}} \otimes P_0, D^i_{l-\text{dbl}} \otimes D^0_0) + \Delta(D^i_{l-\text{dbl}} \otimes D^i_0, D^i_l) \quad \text{by Equation (3)} \\
\leq \Delta(P^l_{\text{dbl}}, D^i_{l-\text{dbl}}) + \Delta(P_0, D^i_0) + \Delta(D^i_{l-\text{dbl}} \otimes D^i_0, D^i_l) \quad \text{by Lemma 3} \\
\leq \Delta(P^l_{\text{dbl}}, D^i_{l-\text{dbl}}) + \Delta(D^i_{l}, D^i_0) + \Delta(D^i_{l-\text{dbl}} \otimes D^i_0, D^i_l) \quad \text{by Lemma 3} \\
\leq \Delta(P^l_{\text{dbl}}, D^i_{l-\text{dbl}}) + 2q\beta(1 - \epsilon_M)^{-1 - 4\alpha(q+1)} \quad \text{by Lemma 5}
\]
and

\[
\Delta(P_{i-1} \odot D_{i-1}^{-1} \odot D_{i-1}^{-1}) \leq \Delta(P_{i-1} \odot P_{i-1} \odot D_{i-1}^{-1} \odot D_{i-1}^{-1}) \quad \text{by Lemma 2}
\]

\[
\leq 2 \cdot \Delta(P_{i-1}, D_{i-1}^{-1}) + \Delta(D_{i-1}^{-1} \odot D_{i-1}^{-1}, D_{i-1}^{-1}) \quad \text{by Equation (3)}
\]

\[
\leq 2 \cdot \Delta(P_{i-1}, D_{i-1}^{-1}) + q \beta (1 - \varepsilon_M)^{-1-(r+1)} \quad \text{by Lemma 3}
\]

\[
\leq 2 \cdot \Delta(P_{i-1}, D_{i-1}^{-1}) + q \beta \quad \text{by Lemma 5}
\]

**D. Proof of Corollary 1**

**Proof of Corollary 1.** The inequality Equation (34) implies

\[
2^{-\varepsilon_{abs}} \delta_{fail} \leq \delta_{fail}(1 - \varepsilon_M)^{1+4r(q+1)} - 2q r \beta.
\]  

(D.1)

Taking logarithm \(\log_2\) on both sides implies

\[
-\varepsilon_{abs} \leq \log_2 \left( \delta_{fail}(1 - \varepsilon_M)^{1+4r(q+1)} - 2q r \beta \right) - \log_2 \delta_{fail}
\]

\[
\leq \log_2 \delta_{alg} - \log_2 \delta_{fail}.
\]  

(D.2)

Furthermore, \(\varepsilon_M \leq 1 - 2^{-\varepsilon_{abs}/(1+4r(q+1))}\) derives that the right hand of Equation (34) is non-negative, and is hence coherent with the fact that \(\beta \geq 0\) in Algorithm 1.

In addition, we have

\[
\varepsilon_M \leq 1 - 2^{-\varepsilon_{abs}/(1+4r(q+1))} \leq 2^{-\varepsilon_{abs}/(1+4r(q+1))} - 1,
\]  

(D.3)

implying that

\[
\varepsilon_{abs} \geq (1 + 4r(q + 1)) \log_2 (1 + \varepsilon_M)
\]

\[
= \log_2 \left( \delta_{fail}(1 + \varepsilon_M)^{1+4r(q+1)} \right) - \log_2 \delta_{fail}
\]

\[
\geq \log_2 \delta_{alg} - \log_2 \delta_{fail}.
\]  

(D.4)

Then Equation (35) holds.

The proof of Equation (37) is similar and omitted here. □

**E. Data of Experiment 3**

This section includes the data of Experiment 3. Specifically, each of Tables 4–12 shows the data for one of the parameter sets of CRYSTALS-Kyber Schwabe [15], FrodoKEM Alkim et al. [24], and SABER D’Anvers et al. [25]. In the tables below, the second row gives time cost of computing \(D_0\) (Line 2 of Algorithm 3), and the third, the fourth, and the fifth row give the data of the heuristic test, the tentative test, and the confirmatory test, respectively. The second column shows data for Algorithm 1 without trimming \((\beta = 0)\), the third column shows data for Algorithm 1 with trimming \([15, 24, 25]\) \((\beta \text{ in the second column of Table 3})\), and the fourth and the fifth column show data of Algorithm 3 for absolute error \(\varepsilon_{abs} = 0.005\) and for relative error \(\varepsilon_{rel} = 5 \times 10^{-6}\), respectively.
<table>
<thead>
<tr>
<th>Table 4: Algorithms 1 and 3 on Kyber512 [15].</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Compute $D_0$</td>
</tr>
<tr>
<td>Heuristic test</td>
</tr>
<tr>
<td>Tentative test</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Confirmatory test</td>
</tr>
<tr>
<td>Total time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5: Algorithms 1 and 3 on Kyber768 [15].</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Compute $D_0$</td>
</tr>
<tr>
<td>Heuristic test</td>
</tr>
<tr>
<td>Tentative test</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Confirmatory test</td>
</tr>
<tr>
<td>Total time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6: Algorithms 1 and 3 on Kyber1024 [15].</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Compute $D_0$</td>
</tr>
<tr>
<td>Heuristic test</td>
</tr>
<tr>
<td>Tentative test</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Confirmatory test</td>
</tr>
<tr>
<td>Total time</td>
</tr>
</tbody>
</table>
**Table 7: Algorithms 1 and 3 on Frodo1344 [24].**

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Algorithm 1</th>
<th>Algorithm 1 [24]</th>
<th>Algorithm 3</th>
<th>$\epsilon_{\text{abs}} = 5 \times 10^{-3}$</th>
<th>$\epsilon_{\text{rel}} = 5 \times 10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $D_0$</td>
<td></td>
<td>$\delta_{\text{clt}} = 2^{-148.87}$</td>
<td></td>
<td>Time = 1.22 ms</td>
<td>$\delta_{\text{clt}} = 2^{-148.87}$</td>
</tr>
<tr>
<td>Heuristic test</td>
<td></td>
<td>$\beta = 0$</td>
<td>$\beta = 10^{-200}$</td>
<td>Time = 0.24 ms</td>
<td>$\beta_{\text{clt}} = 2^{-191.06}$</td>
</tr>
<tr>
<td>Tentative test</td>
<td></td>
<td>$\delta_{\text{alg}} = 2^{-138.76}$</td>
<td>$\delta_{\text{alg}} = 2^{-138.76}$</td>
<td>Time = 4,058.44 ms</td>
<td>$\delta_{\text{alg}} = 2^{-138.76}$</td>
</tr>
<tr>
<td>Confirmatory test</td>
<td></td>
<td>$\beta_{\text{alcf}} = 2^{-178.76}$</td>
<td>$\beta_{\text{alcf}} = 2^{-178.76}$</td>
<td>Time = 1,057.97 ms</td>
<td>$\delta_{\text{alcf}} = 2^{-138.76}$</td>
</tr>
<tr>
<td>Total time</td>
<td></td>
<td>$\beta_{\text{relcf}} = 2^{-191.04}$</td>
<td>$\delta_{\text{relcf}} = 2^{-138.76}$</td>
<td>Time = 64.56 ms</td>
<td>$\delta_{\text{relcf}} = 2^{-138.76}$</td>
</tr>
</tbody>
</table>

**Table 8: Algorithms 1 and 3 on Frodo976 [24].**

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Algorithm 1</th>
<th>Algorithm 1 [24]</th>
<th>Algorithm 3</th>
<th>$\epsilon_{\text{abs}} = 5 \times 10^{-3}$</th>
<th>$\epsilon_{\text{rel}} = 5 \times 10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $D_0$</td>
<td></td>
<td>$\delta_{\text{clt}} = 2^{-213.29}$</td>
<td></td>
<td>Time = 1.10 ms</td>
<td>$\delta_{\text{clt}} = 2^{-213.29}$</td>
</tr>
<tr>
<td>Heuristic test</td>
<td></td>
<td>$\beta = 0$</td>
<td>$\beta = 10^{-200}$</td>
<td>Time = 0.18 ms</td>
<td>$\beta_{\text{clt}} = 2^{-256.59}$</td>
</tr>
<tr>
<td>Tentative test</td>
<td></td>
<td>$\delta_{\text{alg}} = 2^{-199.60}$</td>
<td>$\delta_{\text{alg}} = 2^{-199.60}$</td>
<td>Time = 63.34 ms</td>
<td>$\delta_{\text{alg}} = 2^{-199.60}$</td>
</tr>
<tr>
<td>Confirmatory test</td>
<td></td>
<td>$\beta_{\text{alcf}} = 2^{-240.71}$</td>
<td>$\beta_{\text{alcf}} = 2^{-240.71}$</td>
<td>Time = 0.00 ms</td>
<td>$\delta_{\text{alcf}} = 2^{-199.60}$</td>
</tr>
<tr>
<td>Total time</td>
<td></td>
<td>$\beta_{\text{relcf}} = 2^{-242.99}$</td>
<td>$\delta_{\text{relcf}} = 2^{-199.60}$</td>
<td>Time = 65.29 ms</td>
<td>$\delta_{\text{relcf}} = 2^{-199.60}$</td>
</tr>
</tbody>
</table>

**Table 9: Algorithms 1 and 3 on Frodo1344 [24].**

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Algorithm 1</th>
<th>Algorithm 1 [24]</th>
<th>Algorithm 3</th>
<th>$\epsilon_{\text{abs}} = 5 \times 10^{-3}$</th>
<th>$\epsilon_{\text{rel}} = 5 \times 10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $D_0$</td>
<td></td>
<td>$\delta_{\text{clt}} = 2^{-268.49}$</td>
<td></td>
<td>Time = 0.37 ms</td>
<td>$\delta_{\text{clt}} = 2^{-268.49}$</td>
</tr>
<tr>
<td>Heuristic test</td>
<td></td>
<td>$\beta = 0$</td>
<td>$\beta = 10^{-200}$</td>
<td>Time = 0.06 ms</td>
<td>$\beta_{\text{clt}} = 2^{-311.93}$</td>
</tr>
<tr>
<td>Tentative test</td>
<td></td>
<td>$\delta_{\text{alg}} = 2^{-252.60}$</td>
<td>$\delta_{\text{alg}} = 2^{-252.60}$</td>
<td>Time = 24.04 ms</td>
<td>$\delta_{\text{alg}} = 2^{-252.60}$</td>
</tr>
<tr>
<td>Confirmatory test</td>
<td></td>
<td>$\beta_{\text{alcf}} = 2^{-294.17}$</td>
<td>$\beta_{\text{alcf}} = 2^{-294.17}$</td>
<td>Time = 0.00 ms</td>
<td>$\delta_{\text{alcf}} = 2^{-252.60}$</td>
</tr>
<tr>
<td>Total time</td>
<td></td>
<td>$\beta_{\text{relcf}} = 2^{-296.12}$</td>
<td>$\delta_{\text{relcf}} = 2^{-252.60}$</td>
<td>Time = 24.10 ms</td>
<td>$\delta_{\text{relcf}} = 2^{-252.60}$</td>
</tr>
</tbody>
</table>
Table 10: Algorithms 1 and 3 on LightSaber [25].

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Algorithm 1</th>
<th>Algorithm 1 [25]</th>
<th>Algorithm 3</th>
<th>( \varepsilon_{abs} = 5 \times 10^{-3} )</th>
<th>( \varepsilon_{rel} = 5 \times 10^{-6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic test</td>
<td>( \beta = 0 )</td>
<td>( \beta = 2^{-300} )</td>
<td>( \beta_{abst} = 2^{-162.98} )</td>
<td>( \delta_{clt} = 2^{-123.30} )</td>
<td>( \beta_{rel} = 2^{-165.90} )</td>
</tr>
<tr>
<td>Time</td>
<td>0.37 ms</td>
<td>1.52 ms</td>
<td>22.75 ms</td>
<td>23.87 ms</td>
<td>25.00 ms</td>
</tr>
<tr>
<td>Tentative test</td>
<td>( \delta_{dg} = 2^{-120.35} )</td>
<td>( \delta_{dg} = 2^{-120.35} )</td>
<td>( \delta_{abst} = 2^{-120.35} )</td>
<td>( \delta_{rel} = 2^{-120.35} )</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>127.90 ms</td>
<td>32.07 ms</td>
<td>22.75 ms</td>
<td>23.87 ms</td>
<td>25.00 ms</td>
</tr>
<tr>
<td>Confirmatory test</td>
<td>( \beta_{abscnf} = 2^{-159.54} )</td>
<td>( \beta_{abscnf} = 2^{-120.35} )</td>
<td>( \delta_{abscnf} = 2^{-120.35} )</td>
<td>( \delta_{relcnd} = 2^{-120.35} )</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>0.00 ms</td>
<td>32.44 ms</td>
<td>24.64 ms</td>
<td>25.76 ms</td>
<td>25.00 ms</td>
</tr>
</tbody>
</table>

Table 11: Algorithms 1 and 3 on Saber [25].

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Algorithm 1</th>
<th>Algorithm 1 [25]</th>
<th>Algorithm 3</th>
<th>( \varepsilon_{abs} = 5 \times 10^{-3} )</th>
<th>( \varepsilon_{rel} = 5 \times 10^{-6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic test</td>
<td>( \beta = 0 )</td>
<td>( \beta = 2^{-300} )</td>
<td>( \beta_{abst} = 2^{-178.83} )</td>
<td>( \delta_{clt} = 2^{-139.07} )</td>
<td>( \beta_{rel} = 2^{-181.60} )</td>
</tr>
<tr>
<td>Time</td>
<td>0.12 ms</td>
<td>0.73 ms</td>
<td>24.25 ms</td>
<td>23.49 ms</td>
<td>24.34 ms</td>
</tr>
<tr>
<td>Tentative test</td>
<td>( \delta_{dg} = 2^{-136.16} )</td>
<td>( \delta_{dg} = 2^{-136.16} )</td>
<td>( \delta_{abst} = 2^{-136.16} )</td>
<td>( \delta_{rel} = 2^{-136.16} )</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>137.41 ms</td>
<td>32.10 ms</td>
<td>24.25 ms</td>
<td>23.49 ms</td>
<td>24.34 ms</td>
</tr>
<tr>
<td>Confirmatory test</td>
<td>( \beta_{abscnf} = 2^{-173.93} )</td>
<td>( \beta_{abscnf} = 2^{-120.35} )</td>
<td>( \delta_{abscnf} = 2^{-120.35} )</td>
<td>( \delta_{relcnd} = 2^{-120.35} )</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>0.00 ms</td>
<td>32.22 ms</td>
<td>25.10 ms</td>
<td>24.34 ms</td>
<td>24.34 ms</td>
</tr>
</tbody>
</table>

Table 12: Algorithms 1 and 3 on FireSaber [25].

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Algorithm 1</th>
<th>Algorithm 1 [25]</th>
<th>Algorithm 3</th>
<th>( \varepsilon_{abs} = 5 \times 10^{-3} )</th>
<th>( \varepsilon_{rel} = 5 \times 10^{-6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic test</td>
<td>( \beta = 0 )</td>
<td>( \beta = 2^{-300} )</td>
<td>( \beta_{abst} = 2^{-208.50} )</td>
<td>( \delta_{clt} = 2^{-168.32} )</td>
<td>( \beta_{rel} = 2^{-211.00} )</td>
</tr>
<tr>
<td>Time</td>
<td>0.12 ms</td>
<td>0.18 ms</td>
<td>25.69 ms</td>
<td>24.71 ms</td>
<td>25.01 ms</td>
</tr>
<tr>
<td>Tentative test</td>
<td>( \delta_{dg} = 2^{-165.26} )</td>
<td>( \delta_{dg} = 2^{-165.26} )</td>
<td>( \delta_{abst} = 2^{-165.26} )</td>
<td>( \delta_{rel} = 2^{-165.26} )</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>132.54 ms</td>
<td>31.24 ms</td>
<td>25.69 ms</td>
<td>24.71 ms</td>
<td>25.01 ms</td>
</tr>
<tr>
<td>Confirmatory test</td>
<td>( \beta_{abscnf} = 2^{-205.45} )</td>
<td>( \beta_{abscnf} = 2^{-165.26} )</td>
<td>( \delta_{abscnf} = 2^{-165.26} )</td>
<td>( \delta_{relcnd} = 2^{-165.26} )</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>0.00 ms</td>
<td>31.36 ms</td>
<td>25.99 ms</td>
<td>25.01 ms</td>
<td>25.01 ms</td>
</tr>
</tbody>
</table>
Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This paper include appendices: Appendix A is the proof of Lemma 3, Appendix B is part of the proof of Lemma 5, Appendix C is part of the proof of Lemma 6, Appendix D is the proof of Corollary 1, and Appendix E includes the data of Experiment 3.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank Dr. Leixiao Cheng for her enlightening discussion. This work is supported by the National Key R&D Program of China (Grant No. 2021YFB3100200), the Open Research Fund of State Key Laboratory of Cryptology (Grant No. MMKFKT202207), and the Shandong Provincial Natural Science Foundation (Grant No. ZR2022QF039).

References


