Hindawi
IET Information Security
Volume 2024, Article ID 2786399, 20 pages
https://doi.org/10.1049/2024/2786399

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

Academic Editor: Jin Wook Byun

Lattice-based encryption schemes are significant cryptographic primitives to defend information security against quantum menace, and the decryption failure rate is related to both theoretical and realistic security. We quantitatively analyze how the floating-point arithmetic and neglecting small probabilities impact the precision, and propose a new effective and efficient test of the failure probability. Therein explicit criteria are given to select the floating-point datatype and to decide which small probabilities should be abandoned. Furthermore, the outcome is theoretically ensured to meet a given precision. Moreover, by combining the heuristic estimate and the precise simulation, this test is more efficient than previously neglecting small probabilities in a practical way.

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 $a \bmod q$ 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 $\mathbf{s} \in \mathbb{Z}_q^r$ given (sufficiently many) pairs $(\mathbf{a}, b)$, where $\mathbf{a} \leftarrow \$\mathbb{Z}_q^r$, $e \leftarrow \$D$, and $b = \mathbf{a}^T\mathbf{s} + e$. 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 $\mathbf{e}_1^T\mathbf{s}_2 - \mathbf{s}_1^T\mathbf{e}_2 + e_3 \bmod q$ is not as small as required. Explicitly, the decryption failure rate (DFR; the condition in Equation (1) [24] is also interpreted as $|\mathbf{e}_1^T\mathbf{s}_2 - \mathbf{s}_1^T\mathbf{e}_2 + e_3 \bmod q| > t$, for example, in D'Anvers et al. [23, 25, 26]. Whether t is counted in does not exert substantial influence on computing δ_{fail}), denoted by δ_{fail}, is the following probability

Key generation	Encryption
1: $A \leftarrow \$\, Z_q^{r\times r}$	1: $s_2 \leftarrow \$\, D_{s_2}^r$
2: $s_1 \leftarrow \$\, D_{s_1}^r$	2: $e_2 \leftarrow \$\, D_{e_2}^r$
3: $e_1 \leftarrow \$\, D_{e_1}^r$	3: $e_3 \leftarrow \$\, D_{e_3}$
4: $b = As_1 + e_1$	4: $c_1 = A^T s_2 + e_2$
5: pk = (A, b)	5: $c_2 = b^T s_2 + e_3 +$ Encode (m)
6: sk = s_1	6: ck = (c_1, c_2)

Decryption
1: Decode $(c_2 - c_1^T s_1)$

pk = public key; sk = secret key; ck = ciphertext; m = plaintext.

Figure 1: The Lindner–Peikert encryption scheme [19].

$$
\Pr\left[|\mathbf{e}_1^T \mathbf{s}_2 - \mathbf{s}_1^T \mathbf{e}_2 + e_3 \bmod q| \notin [-t, t): \begin{array}{r} \mathbf{s}_1 \leftarrow \$D_{s_1}^r, \\ \mathbf{s}_2 \leftarrow \$D_{s_2}^r, \\ \mathbf{e}_1 \leftarrow \$D_{e_1}^r, \\ \mathbf{e}_2 \leftarrow \$D_{e_2}^r, \\ e_3 \leftarrow \$D_{e_3} \end{array} \right], \tag{1}
$$

where the critical positive value t depends on the specific scheme, for example, involving the modulus q and the number of bits in the plaintext m.

Decryption failure is closely related to the security of latticed-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 "failure boosting" attack [26, 30, 31] and risk a loss of security level. Therefore, it is meaningful and interesting to efficiently compute the DFR δ_{fail} with confident accuracy.

The key to obtaining δ_{fail} is to characterize the distribution of $\mathbf{e}_1^T \mathbf{s}_2 - \mathbf{s}_1^T \mathbf{e}_2 + e_3 \bmod q$ 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 $r_n 2^n + r_{n-1} 2^{n-1} + \cdots + r_0$, and denote

$$
\begin{cases}
P_i & \text{the distribution of } \left[\sum_{j=0}^{i} r_{n-j} 2^{i-j}\right] \cdot (e_1 s_2 - s_1 e_2), \quad 0 \le i \le n; \\
P_i^{\text{dbl}} & \text{the distribution of } \left[\sum_{j=0}^{i-1} r_{n-j} 2^{i-j}\right] \cdot (e_1 s_2 - s_1 e_2), \quad 1 \le i \le n; \\
P_{\text{fin}} & \text{the distribution of } [r] \cdot (e_1 s_2 - s_1 e_2) + e_3;
\end{cases} \tag{2}
$$

where $s_1 \leftarrow \$D_{s_1}, s_2 \leftarrow \$D_{s_2}, e_1 \leftarrow \$D_{e_1}, e_2 \leftarrow \$D_{e_2}, e_3 \leftarrow \$D_{e_3}$, and $[m] \cdot (e_1 s_2 - s_1 e_2)$ denotes the sum of m independent random variable with the same distribution as $e_1 s_2 - s_1 e_2$. This is algorithmically feasible by recursive computation [15, 24, 25]

$$
\begin{cases}
P_i^{\text{dbl}} = P_{i-1} \circledast P_{i-1}, 1 \le i \le n; \\
P_i = P_i^{\text{dbl}} \circledast r_{n-i} \cdot P_0, 1 \le i \le n; \\
P_{\text{fin}} = P_n \circledast D_{e_3}.
\end{cases} \tag{3}
$$

where $X \circledast Y$ means the convolution of distributions X and Y. In addition, to optimize time and space cost in computation, the above "double-and-add" method [15, 24, 25] uses floating-point arithmetic and deliberately neglect tiny probabilities (e.g., those less than some assigned bound β).

In respect of the above estimation, there remain two unanswered questions below. In order to obtain δ_{fail} with a required precision,

(i) which floating-point datatype should be chosen?
(ii) which tiny probabilities should be neglected? Specifically, how large should we set β to be?

1.1. Our Results. This correspondence addresses these two questions above. First, we quantitatively analyze the impact of floating-point arithmetic and the trimming threshold β on the precision to approximate δ_{fail}. Second, derived from the analysis, it is specified how to select floating-point datatype based on cipher parameters. Third, a new test of DFR is proposed. This test has the following three properties:

(1) Instead of operating the heuristic estimation and the precise simulation independently, it combines both together and makes use of their internal relation with δ_{fail}. Specifically, the obtained heuristic approximation of decryption failure helps to select the trimming threshold β for the "double-and-add" method.
(2) Its returned estimate is ensured to approximate δ_{fail} with any assigned high precision as long as the machine precision allows. Particularly, whether its output is accurate enough can be theoretically verified.
(3) It selects the trimming bound β in a balanced and inexpensive way and thereby accelerates the "double-and-add" method, costing less time than previous computing [15, 24, 25]. Particularly, for Frodo640 [24] the new test, even including the heuristic estimate in it, takes time only 5.92% of that the previous method costs.

Fourth, we also analyze how the test of decryption failure is influenced by algebraic lattices and the rounding compression and whether the proposed test is feasible for lattice-based ID-based encryption (IBE) and attribute-based encryption (ABE).

1.2. Related Work. So far, it has not been analyzed how the floating-point arithmetic impacts the precision to approximate δ_{fail}, and a clear and explicit criterion to select floating-point datatype for DFR test has not been given though the DFR, for example, 2^{-136} for SABER [23], is possibly much 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_{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 $\log \delta_{\text{fail}}$ with three significant digits [15, 25], and FrodoKEM always removes probability less than 10^{-200} and gives $\log \delta_{\text{fail}}$ 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 ε_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 α_M be the minimal positive normalized floating-point number in machine. Both ε_M and α_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

$$g \cdot (1 - \varepsilon_M)^m \leq f \leq g \cdot (1 + \varepsilon_M)^m. \tag{4}$$

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*

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

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*

$$\sum_{a \in R} D(a) \leq 1. \tag{6}$$

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,

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

Let D_1 and D_2 be pseudo-laws. The *convolution* of D_1 and D_2, denoted by $D_1 \circledast D_2$ is

$$D_1 \circledast D_2(c) = \sum_{\substack{a, b \in \mathbb{Z}_q \\ a + b \equiv c \bmod q}} D_1(a) \cdot D_2(b), \tag{8}$$

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

$$D_1 \odot D_2(c) = \sum_{\substack{a, b \in \mathbb{Z}_q \\ a \cdot b \equiv c \bmod q}} D_1(a) \cdot D_2(b). \tag{9}$$

Notice that $C_{-1} \odot D$ is a mirror reflection of D over the Y-axis.

The convolution $D_1 \circledast D_2$ (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

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

in the quotient ring $\mathbb{R}[x]/(x^q - 1)$ of polynomials, where $(x^q - 1)$ denotes the principal ideal generated by $x^q - 1$. 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*

Input: the modulus q; the distributions D_{e_1}, D_{e_2}, D_{s_1}, D_{s_2} of coordinates of $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{s}_1$, $\mathbf{s}_2$, respectively; the distribution D_{e_3} of e_3; the dimension r; the critical value t of decryption failure; the trimming threshold β.

Output: an approximation of δ_{fail}.

1: 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 $e_2 \leftarrow D_{e_2}$, for example, $D = (D_{e_1} \odot D_{s_2}) \circledast (C_{-1} \odot D_{s_1} \odot D_{e_2})$.
2: $D_0 = \text{Trim}_\beta(D)$.
3: Get the binary representation $r = \sum_{i=0}^{n} r_i 2^i$, where $r_i \in \{0, 1\}$ and $r_n = 1$.
4: **for** $i = 1$ to n **do**
5: $\quad D_i^{\text{dbl}} = \text{Trim}_\beta(D_{i-1} \circledast D_{i-1})$.
6: $\quad$ **if** $r_{n-i} = 1$ **then**
7: $\quad\quad D_i = \text{Trim}_\beta(D_i^{\text{dbl}} \circledast D_0)$.
8: $\quad$ **else**
9: $\quad\quad D_i = D_i^{\text{dbl}}$.
10: $\quad$ **end if**
11: **end for**
12: $D_{\text{fin}} = \text{Trim}_\beta(D_n \circledast D_{e_3})$.
13: **return** $\delta_{\text{alg}} = \sum_{a \in \mathbb{Z}_q, a \notin [-t, t)} D_{\text{fin}}(a)$.

ALGORITHM 1: Estimate DFR with floating-point arithmetic.

$$\Delta(D_1, D_2) = \sum_{a \in \mathbb{Z}_q} |D_1(a) - D_2(a)|. \tag{11}$$

Lemmas 2 and 3 straightforwardly follow from Definition 2.

Lemma 2. *Let D_1, D_2, D_3 be pseudo-laws over $\mathbb{Z}_q$. Then*

$$\Delta(D_1, D_3) \le \Delta(D_1, D_2) + \Delta(D_2, D_3). \tag{12}$$

Lemma 3. *Let D_1, D_2, E_1, E_2 be pseudo-laws over $\mathbb{Z}_q$. Then*

$$\Delta(D_1 \circledast D_2, E_1 \circledast E_2) \le \Delta(D_1, E_1) + \Delta(D_2, E_2). \tag{13}$$

Lemma 3 upper-bounds propagation of statistical distances in the convolution of pseudo-laws, and a brief proof is included in Appendix A.

Definition 3 describes the procedure imposing zero probability over a given subspace S, and particularly the operation which removes positive probability not larger than a threshold β.

Definition 3. (Trim). *Let $S \subset \mathbb{Z}_q$. A trim of a pseudo-law D by S, denoted by* $\text{Trim}_S(D)$, *is a pseudo-law defined by*

$$\text{Trim}_S(D)(a) = \begin{cases} D(a), & a \in \mathbb{Z}_q \backslash S, \\ 0, & a \in S. \end{cases} \tag{14}$$

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

Input: the modulus q; the distributions $D_{e_1}^{\text{I}}$, $D_{e_2}^{\text{I}}$, $D_{s_1}^{\text{I}}$, $D_{s_2}^{\text{I}}$ of coordinates of $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{s}_1$, $\mathbf{s}_2$, respectively; the distribution $D_{e_3}^{\text{I}}$ of e_3; the dimension r; the critical value t of decryption failure; a number β for trimming.

Output: an approximation of δ_{fail}.

1: Compute the distribution D^{I} of $e_1 \cdot s_2 - s_1 \cdot e_2$, where $s_1 \leftarrow D_{s_1}^{\text{I}}$, $s_2 \leftarrow D_{s_2}^{\text{I}}$, $e_1 \leftarrow D_{e_1}^{\text{I}}$, and $e_2 \leftarrow D_{e_2}^{\text{I}}$.
2: $D_0^{\text{I}} = \text{Trim}_{S_0}(D^{\text{I}})$, where
$\quad S_0 = \{a \in \mathbb{Z}_q : D(a) \le \beta\}$
and D is given in Line 2 of Algorithm 1.
3: Get the binary representation $r = \sum_{i=0}^{n} r_i 2^i$, where $r_i \in \{0, 1\}$ and $r_n = 1$.
4: **for** $i = 1$ to n **do**
5: $\quad D_i^{\text{I-dbl}} = \text{Trim}_{S_i^{\text{dbl}}}(D_{i-1}^{\text{I}} \circledast D_{i-1}^{\text{I}})$, where
$\quad S_i^{\text{dbl}} = \{a \in \mathbb{Z}_q : (D_{i-1} \circledast D_{i-1})(a) \le \beta\}$
$\quad$ and $D_{i-1} \circledast D_{i-1}$ is given in Line 5 of Algorithm 1.
6: $\quad$ **if** $r_{n-i} = 1$ **then**
7: $\quad\quad D_i^{\text{I}} = \text{Trim}_{S_i^{\text{add}}}(D_i^{\text{I-dbl}} \circledast D_0^{\text{I}})$, where
$\quad\quad S_i^{\text{add}} = \{a \in \mathbb{Z}_q : (D_i \circledast D_0)(a) \le \beta\}$
$\quad\quad$ and $D_i \circledast D_0$ is given in Line 7 of Algorithm 1.
8: $\quad$ **else**
9: $\quad\quad D_i^{\text{I}} = D_i^{\text{I-dbl}}$.
10: $\quad$ **end if**
11: **end for**
12: $D_{\text{fin}}^{\text{I}} = \text{Trim}_{S_{n+1}}(D_n^{\text{I}} \circledast D_{e_3}^{\text{I}})$, where
$\quad S_{n+1} = \{a \in \mathbb{Z}_q : (D_n \circledast D_{e_3})(a) \le \beta\}$
and $D_n \circledast D_{e_3}$ is given in Line 12 of Algorithm 1.
13: **return** $\delta_{\text{alg}}^{\text{I}} = \sum_{a \in \mathbb{Z}_q, a \notin [-t, t)} D_{\text{fin}}^{\text{I}}(a)$.

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 β and deterministically computes the DFR δ_{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 $\mathcal{O}(q^2 \log r)$ since a convolution or a product cost at most $\mathcal{O}(q^2)$. As in Remark 1, this complexity can be further reduced using Fourier transformation.

Theorem 1. *Algorithm 1 runs in deterministic polynomial time $\mathcal{O}(q^2 \log r)$.*

Now we analyze effectiveness of Algorithm 1, that is, how closely it approximates the DFR δ_{fail}.

Theorem 2. *If $\beta \ge \sqrt{\alpha_M}$ and each nonzero value of distributions D_{s_1}, D_{s_2}, D_{e_1}, D_{e_2}, and D_{e_3} is not less than $\sqrt{\alpha_M}$, then*

$$\begin{cases} \delta_{\text{alg}} \ge (1 - \epsilon_M)^{1+4r(q+1)} \delta_{\text{fail}} - 2qr\beta; \\ \delta_{\text{alg}} \le \delta_{\text{fail}} (1 + \epsilon_M)^{1+4r(q+1)}. \end{cases} \tag{15}$$

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 "I" on pseudo-laws in Algorithm 2.

(ii) Instead of a fixed trimming threshold β, 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 δ_{alg} approximates δ_{fail}, 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 $D^{\mathrm{I}}_{\text{fin}}$ in Algorithm 2 is close to P_{fin}. Part IV integrates the above to complete the proof.

Part I (Lemma 4): Using a power of $(1+\varepsilon_M)$ 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_{fin}, $D_n \circledast D_{e_3}$, D, $D_{i-1} \circledast D_{i-1}$, and $D^{\text{dbl}}_i \circledast D_0$ $(1 \le i \le n)$ in Algorithm 1, and let E^{I} denote its ideally precise counterpart among $D^{\mathrm{I}}_{\text{fin}}$, $D^{\mathrm{I}}_n \circledast D^{\mathrm{I}}_{e_3}$, D^{I}, $D^{\mathrm{I}}_{i-1} \circledast D^{\mathrm{I}}_{i-1}$, and $D^{\mathrm{I\text{-}dbl}}_i \circledast D^{\mathrm{I}}_0$ in Algorithm 2. Given the condition in Theorem 2, it holds that*

$$E \sim E^{\mathrm{I}}(1 \pm \varepsilon_M)^{1+4r(q+1)}. \tag{16}$$

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 $(1+\varepsilon_M)^q$ (resp. $(1-\varepsilon_M)^q$). Thus, comparing Algorithms 1 and 2, we add (possible) errors by floating-point representation of D_{s_1}, D_{s_2}, D_{e_1}, D_{e_2}, and D_{e_3}, and count the number of convolutions, and then find that the following integers

$$\begin{cases} m_0 = 3q+4; & \\ m^{\text{dbl}}_i = 2m_{i-1} + q, & 1 \le i \le n; \\ m_i = 2m_{i-1} + q + r_{n-i}(q+m_0), & 1 \le i \le n; \\ m_{\text{fin}} = m_n + q + 1, & \end{cases} \tag{17}$$

satisfy

$$\begin{cases} D \sim D^{\mathrm{I}}(1 \pm \varepsilon_M)^{m_0}; & \\ (D_{i-1} \circledast D_{i-1}) \sim (D^{\mathrm{I}}_{i-1} \circledast D^{\mathrm{I}}_{i-1})(1 \pm \varepsilon_M)^{m^{\text{dbl}}_i}, & 1 \le i \le n; \\ \left(D^{\text{dbl}}_i \circledast D_0\right) \sim \left(D^{\mathrm{I\text{-}dbl}}_i \circledast D^{\mathrm{I}}_0\right)(1 \pm \varepsilon_M)^{m_i}, & r_{n-i} = 1, 1 \le i \le n; \\ \left(D_n \circledast D_{e_3}\right) \sim \left(D^{\mathrm{I}}_n \circledast D^{\mathrm{I}}_{e_3}\right)(1 \pm \varepsilon_M)^{m_{\text{fin}}}; & \\ D_{\text{fin}} \sim D^{\mathrm{I}}_{\text{fin}}(1 \pm \varepsilon_M)^{m_{\text{fin}}}. & \end{cases} \tag{18}$$

Explicitly, these integers are

$$\begin{cases} m_0 = 3q+4; & \\ m^{\text{dbl}}_i = -q + 4(q+1) \cdot \sum_{j=0}^{i-1} r_{n-j} 2^{i-j}, & 1 \le i \le n; \\ m_i = -q + 4(q+1) \cdot \sum_{j=0}^{i} r_{n-j} 2^{i-j}, & 1 \le i \le n; \\ m_{\text{fin}} = 1 + 4r(q+1). & \end{cases} \tag{19}$$

Finally, if $0 \le a < b$ and $f \sim g(1 \pm \varepsilon_M)^a$ then $f \sim g(1 \pm \varepsilon_M)^b$. Therefore, the proof of Lemma 4 is completed since $m_{\text{fin}} > \max\{m_0, m_i, m^{\text{dbl}}_i : 1 \le i \le n\}$. □

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 $\circledast$ in Line 1 of Algorithm 1, and the addend 4 is attributed to representing D_{s_1}, D_{s_2}, D_{e_1}, and D_{e_2} 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 ring $\mathbb{Z}_q$ and hence one $\odot$ there contributes a relative error much tamer than $(1 \pm \varepsilon_M)^q$. Taking Frodo640 [24] for example, the relative error of $\odot$ is bounded by $(1 \pm \varepsilon_M)^{625}$, far tamer than $(1 \pm \varepsilon_M)^{2^{15}}$. 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 $D^{\mathrm{I}}_n \circledast D^{\mathrm{I}}_{e_3}$, D^{I}, $D^{\mathrm{I}}_{i-1} \circledast D^{\mathrm{I}}_{i-1}$, and $D^{\mathrm{I\text{-}dbl}}_i \circledast D^{\mathrm{I}}_0$ $(1 \le i \le n)$ in Algorithm 2, and let* $\mathrm{Trim}(E)$ *denote its corresponding trim among $D^{\mathrm{I}}_{\text{fin}}$, D^{I}_0, $D^{\mathrm{I\text{-}dbl}}_i$, and D^{I}_i $(1 \le i \le n)$ in Algorithm 2. Then*

$$\Delta(E, \mathrm{Trim}(E)) \le q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. \tag{20}$$

Proof. Notice that for $S \subset \mathbb{Z}_q$ and any pseudo-law F over $\mathbb{Z}_q$,

$$\begin{aligned}\Delta(F, \mathrm{Trim}_S(F)) &= \sum_{a\in S} F(a) \\ &\leq |S| \cdot \max_{a\in S} F(a) \leq q \cdot \max_{a\in S} F(a),\end{aligned} \tag{21}$$

where $|S|$ denotes the cardinality of S. Considering the error caused by trimming in Line 2 of Algorithm 2, we have

$$\begin{aligned}\Delta(D_0^{\mathrm{I}}, D^{\mathrm{I}}) &= \Delta\left(\mathrm{Trim}_{S_0}(D^{\mathrm{I}}), D^{\mathrm{I}}\right) \leq q \cdot \max_{a\in S_0} D^{\mathrm{I}}(a) && \text{(using Equation (21))} \\ &\leq q \cdot (1-\varepsilon_M)^{-1-4r(q+1)} \max_{a\in S_0} D(a) && \text{(using Lemma 4)} \\ &\leq q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. && \text{(using Algorithm 2)}\end{aligned} \tag{22}$$

In a similar way, $\Delta(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, D_i^{\mathrm{I\text{-}dbl}})$, $\Delta\left(D_i^{\mathrm{I\text{-}dbl}} \circledast D_0^{\mathrm{I}}, D_i^{\mathrm{I}}\right)$, and $\Delta\left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}, D_{\mathrm{fin}}^{\mathrm{I}}\right)$ are also upper-bounded by $q\beta(1-\varepsilon_M)^{-1-4r(q+1)}$ and the detailed proof is included in Appendix B. □

Part III (Lemma 6): Using Part II, we upper-bound the statistical distance between $D_{\mathrm{fin}}^{\mathrm{I}}$ and P_{fin}.

Lemma 6. *Let P_{fin} be defined in Equation (2) and let $D_{\mathrm{fin}}^{\mathrm{I}}$ be as in Line 12 of Algorithm 2. Then it holds that*

$$\Delta\left(P_{\mathrm{fin}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) \leq 2qr\beta(1-\varepsilon_M)^{-1-4r(q+1)}. \tag{23}$$

Proof. Use the notations in Equation (2). Now we get

$$\begin{aligned}&\Delta\left(P_{\mathrm{fin}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) \\ &\leq \Delta\left(P_{\mathrm{fin}}, D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}\right) + \Delta\left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) && \text{(by Lemma 2)} \\ &\leq \Delta\left(P_n \circledast D_{e_3}^{\mathrm{I}}, D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}\right) + \Delta\left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) && \text{(by Equation (3))} \\ &\leq \Delta(P_n, D_n^{\mathrm{I}}) + \Delta\left(D_{e_3}^{\mathrm{I}}, D_{e_3}^{\mathrm{I}}\right) + \Delta\left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) && \text{(by Lemma 3)} \\ &\leq \Delta(P_n, D_n^{\mathrm{I}}) + q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. && \text{(by Lemma 5)}\end{aligned} \tag{24}$$

Similarly, we also have

$$\begin{cases}\Delta(P_i, D_i^{\mathrm{I}}) \leq \Delta\left(P_i^{\mathrm{dbl}}, D_i^{\mathrm{I\text{-}dbl}}\right) + r_{n-i} \cdot 2q\beta(1-\varepsilon_M)^{-1-4r(q+1)}; \\ \Delta\left(P_i^{\mathrm{dbl}}, D_i^{\mathrm{I\text{-}dbl}}\right) \leq 2 \cdot \Delta(P_{i-1}, D_{i-1}^{\mathrm{I}}) + q\beta(1-\varepsilon_M)^{-1-4r(q+1)}.\end{cases} \tag{25}$$

The detailed proof of Equation (25) is included in Appendix C.

Using Equations (3), (24), and (25), we derive that

$$\begin{aligned}&\Delta\left(P_{\mathrm{fin}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) \\ &\leq q\beta(1-\varepsilon_M)^{-1-4r(q+1)} + q\beta(1-\varepsilon_M)^{-1-4r(q+1)} \cdot 2^n + \textstyle\sum_{i=1}^n 2^{n-i} \cdot q\beta(1-\varepsilon_M)^{-1-4r(q+1)}(1+2r_{n-i}) \\ &= q\beta(1-\varepsilon_M)^{-1-4r(q+1)}\left(1+2^n+\textstyle\sum_{i=1}^n 2^{n-i}(1+2r_{n-i})\right) \\ &= 2qr\beta(1-\varepsilon_M)^{-1-4r(q+1)}.\end{aligned} \tag{26}$$

□

Part IV: Using Part I and III, we characterize how the output δ_{alg} returned by Algorithm 1 approximates the DFR δ_{fail}.

The lemma below for comparing pseudo-laws is straightforward.

Lemma 7. *If pseudo-laws D_1, D_2, E_1, E_2 satisfy $D_1 \leq E_1$ and $D_2 \leq E_2$, then*

$$D_1 \circledast D_2 \leq E_1 \circledast E_2. \tag{27}$$

Proof of Theorem 2. Since $\mathrm{Trim}_S(E) \le E$ for any pseudo-law E, the pseudo-laws in Algorithm 2 satisfy

$$\begin{cases} D_0^{\mathrm{I}} \le D^{\mathrm{I}} = P_0; \\ D_i^{\mathrm{I\text{-}dbl}} \le D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, & 1 \le i \le n; \\ D_i^{\mathrm{I}} \le D_i^{\mathrm{I\text{-}dbl}} \circledast (r_{n-i} \cdot D_0^{\mathrm{I}}), & 1 \le i \le n; \\ D_{\mathrm{fin}}^{\mathrm{I}} \le D_n^{\mathrm{I}} \circledast D_{e_3}. \end{cases} \tag{28}$$

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

$$D_{\mathrm{fin}}^{\mathrm{I}} \le P_{\mathrm{fin}}. \tag{29}$$

Since $\mathbf{e}_1^T \mathbf{s}_2 - \mathbf{s}_1^T \mathbf{e}_2 \bmod q$ is the sum of r independent random variables with the same distribution P_0, it holds that

$$\delta_{\mathrm{fail}} = \textstyle\sum_{a \in \mathbb{Z}_q, a \notin [-t,t)} P_{\mathrm{fin}}(a). \tag{30}$$

Recall that

$$\delta_{\mathrm{alg}}^{\mathrm{I}} = \textstyle\sum_{a \in \mathbb{Z}_q, a \notin [-t,t)} D_{\mathrm{fin}}^{\mathrm{I}}(a). \tag{31}$$

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

$$0 \le \delta_{\mathrm{fail}} - \delta_{\mathrm{alg}}^{\mathrm{I}} \le \Delta\left(P_{\mathrm{fin}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) \le 2qr\beta(1-\varepsilon_M)^{-1-4r(q+1)}. \tag{32}$$

In addition, it follows from Lemma 4 that

$$\delta_{\mathrm{alg}}^{\mathrm{I}} (1-\varepsilon_M)^{1+4r(q+1)} \le \delta_{\mathrm{alg}} \le \delta_{\mathrm{alg}}^{\mathrm{I}} (1+\varepsilon_M)^{1+4r(q+1)}. \tag{33}$$

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

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_{s_1}, D_{s_2}, D_{e_1}, D_{e_2}, and D_{e_3} 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 $\log_2 \delta_{\mathrm{fail}}$ as the final result we expect, and aim to control the absolute/relative error of $\log_2 \delta_{\mathrm{alg}}$.

Corollary 1. *Let $\varepsilon_{\mathrm{abs}} > 0$ and $\varepsilon_{\mathrm{rel}} > 0$. The statements below hold. (i) If $\varepsilon_M \le 1 - 2^{-\varepsilon_{\mathrm{abs}}/(1+4r(q+1))}$ and*

$$\sqrt{\alpha_M} \le \beta \le \frac{\delta_{\mathrm{fail}}}{2qr} \cdot \left((1-\varepsilon_M)^{1+4r(q+1)} - 2^{-\varepsilon_{\mathrm{abs}}}\right), \tag{34}$$

then

$$\left|\log_2 \delta_{\mathrm{alg}} - \log_2 \delta_{\mathrm{fail}}\right| \le \varepsilon_{\mathrm{abs}}. \tag{35}$$

(ii) If $\varepsilon_M \le 1 - \delta_{\mathrm{fail}}^{\varepsilon_{\mathrm{rel}}/(1+4r(q+1))}$ and

$$\sqrt{\alpha_M} \le \beta \le \frac{\delta_{\mathrm{fail}}}{2qr} \cdot \left((1-\varepsilon_M)^{1+4r(q+1)} - \delta_{\mathrm{fail}}^{\varepsilon_{\mathrm{rel}}}\right), \tag{36}$$

then

$$\left|\log_2 \delta_{\mathrm{alg}} / \log_2 \delta_{\mathrm{fail}} - 1\right| \le \varepsilon_{\mathrm{rel}}. \tag{37}$$

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

Corollary 2. *Let $\varepsilon_{\mathrm{abs}} > 0$ and $\varepsilon_{\mathrm{rel}} > 0$. If*

$$\varepsilon_M \le 1 - 2^{-\varepsilon_{\mathrm{abs}}/(1+4r(q+1))} \tag{38}$$

and

$$\sqrt{\alpha_M} \le \beta \le \frac{\delta_{\mathrm{alg}} \cdot \left((1-\varepsilon_M)^{1+4r(q+1)} - 2^{-\varepsilon_{\mathrm{abs}}}\right)}{2qr(1+\varepsilon_M)^{1+4r(q+1)}}, \tag{39}$$

then Equation (35) holds. If

$$\varepsilon_M \le 1 - (1-\varepsilon_M)^{-\varepsilon_{\mathrm{rel}}} \left(\delta_{\mathrm{alg}} + 2qr\beta\right)^{\varepsilon_{\mathrm{rel}}/(1+4r(q+1))} \tag{40}$$

and

$$\sqrt{\alpha_M} \le \beta \le \frac{\delta_{\mathrm{alg}}}{2qr} \cdot \left(\frac{1-\varepsilon_M}{1+\varepsilon_M}\right)^{1+4r(q+1)} \cdot \left(1 - \frac{\left(\delta_{\mathrm{alg}} + 2qr\beta\right)^{\varepsilon_{\mathrm{rel}}}}{(1-\varepsilon_M)^{(1+4r(q+1))(1+\varepsilon_{\mathrm{rel}})}}\right), \tag{41}$$

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 δ_{alg} in Schwabe [15], Alkim et al. [24], and D'Anvers et al. [25].

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

Experiment 1. *We use Corollary 2 to verify whether δ_{alg}'s in CRYSTALS-Kyber [15], SABER [25], and FrodoKEM [24] satisfy Equation (35) with $\varepsilon_{\mathrm{abs}} = 5\times10^{-3}$ and Equation (37) with $\varepsilon_{\mathrm{rel}} = 5\times10^{-6}$. CRYSTALS-Kyber and SABER use double-precision floating-point arithmetic while FrodoKEM uses float128 [15, 24, 25]. Here $\alpha_M = 2^{-16,382}$ and $\varepsilon_M = 2^{-64}$ [35] for FrodoKEM though the datatype numpy.float128 varies depending on machines and operating systems [36].*

As shown in Table 1 (as shown in Subsection 3.5, the DFR of CRYSTALS-Kyber is interpreted other than Equation (1), and hence the corresponding upper-bounds in Equations (38)–(41) are adapted. The tedious details are omitted here), for all these cipher, Equations (38)–(41) hold. Therefore, it is ensured by Corollary 2 that their failure probabilities [15, 24, 25] have met the required precision, and this is labeled as "Y" in the last column of Table 1. Because $\sqrt{2^{-1,022}} > 10^{-200} \approx 2^{-664}$, Corollary 2 does not convince the precision of δ_{fail} for FrodoKEM if Algorithm 1 utilizes double-precision floating-point arithmetic and sets $\beta = 10^{-200}$.

However, Corollaries 1 and 2 do not directly inform us how to determine β in practice because neither δ_{fail} nor δ_{alg} is known before a test. To ensure desired absolute (and relative) error for $\log_2\delta_{\mathrm{fail}}$, later we use Theorem 2 and Corollary 1 to select floating-point datatype and the trimming threshold β.

3.2. Select Floating-Point Datatype. A floating-point datatype is determined by its precision and range, respectively related to ε_M and α_M. Corollary 1 is helpful for selecting floating-point datatype in Algorithm 1.

If

$$\varepsilon_M \le 1 - 2^{-\varepsilon_{\mathrm{abs}}/(1+4r(q+1))}\left(\mathrm{resp.}\,\varepsilon_M \le 1 - \delta_{\mathrm{fail}}^{\varepsilon_{\mathrm{rel}}/(1+4r(q+1))}\right), \tag{42}$$

and

$$\alpha_M \le \left(\frac{\delta_{\mathrm{fail}}}{2qr}\right)^2 \cdot \left((1-\varepsilon_M)^{1+4r(q+1)} - 2^{-\varepsilon_{\mathrm{abs}}}\right)^2$$
$$\left(\mathrm{resp.}\ \alpha_M \le \left(\frac{\delta_{\mathrm{fail}}}{2qr}\right)^2 \cdot \left((1-\varepsilon_M)^{1+4r(q+1)} - \delta_{\mathrm{fail}}^{\varepsilon_{\mathrm{rel}}}\right)^2\right), \tag{43}$$

then Algorithm 1 with proper trimming returns δ_{alg} satisfying Equation (35) (resp. Equation (37)).

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

	q	r	$\varepsilon_{\mathrm{abs}} = 5\times10^{-3}$	$\varepsilon_{\mathrm{rel}} = 5\times10^{-6}$
Frodo640	2^{15}	640	$\varepsilon_M \le 2^{-34.50}$	$\varepsilon_M \le 2^{-44.47}$
Frodo976	2^{16}	976	$\varepsilon_M \le 2^{-36.11}$	$\varepsilon_M \le 2^{-46.07}$
Frodo1344	2^{16}	1, 344	$\varepsilon_M \le 2^{-36.57}$	$\varepsilon_M \le 2^{-46.53}$

For most nowadays lattice-based encryption schemes and KEMs, their parameters satisfy $q \le 2^{16}$ and $r \le 2^{14}$, and their DFRs are usually located in the range $2^{-256} < \delta_{\mathrm{fail}} < 2^{-80}$. Hence, the required absolute (resp. relative) error $\varepsilon_{\mathrm{abs}} = 5\times10^{-3}$ (resp. $\varepsilon_{\mathrm{rel}} = 5\times10^{-6}$) suffices. Under such conditions, $\varepsilon_M \le 2^{-43.82}$ and $\alpha_M \le 2^{-623.94}$ 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 remind that (i) the above datatype selection is based on the practical range of δ_{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 $\varepsilon_{\mathrm{abs}} = 5\times10^{-3}$, $\varepsilon_{\mathrm{rel}} = 5\times10^{-6}$, and list the parameters of FrodoKEM [24] and their corresponding ε_M estimated in Equation (42) (here conservatively using $\log_2\delta_{\mathrm{fail}} \le -1$). 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 δ_{fail}. Anyhow, it suffices to use double precision (64 bit) floating-point instead of float128 in the python numpy package to find the DFR δ_{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 β in a progressive way. Specifically, the heuristic estimate δ_{clt} through a continuous normal distribution helps to decide β for a tentative test, denoted by β_{abstnt} for the absolute error $\varepsilon_{\mathrm{abs}}$ (resp. by β_{reltnt} for the relative error $\varepsilon_{\mathrm{rel}}$), 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 $\mathbf{e}_1, \mathbf{e}_2, \mathbf{s}_1, \mathbf{s}_2$, respectively; the distribution D_{e_3} of e_3; the dimension r; the critical value t of decryption failure and $\varepsilon_{\text{abs}} > 0$ (resp. $\varepsilon_{\text{rel}} > 0$).

Output: estimate the DFR δ_{fail}.

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

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 δ_{clt}.

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

5: Set

$$\beta_{\text{abstnt}} = \frac{\delta_{\text{clt}}}{2qr} \cdot \left((1 - \varepsilon_M)^{1+4r(q+1)} - 2^{-\varepsilon_{\text{abs}}} \right)$$

$$\left(\text{resp. } \beta_{\text{reltnt}} = \frac{\delta_{\text{clt}}}{2qr} \cdot \left((1 - \varepsilon_M)^{1+4r(q+1)} - \delta_{\text{clt}}{}^{\varepsilon_{\text{rel}}} \right)\right).$$

6: Select the floating-point datatype such that $\alpha_M \leq \beta^2_{\text{abstnt}}$ (resp. β^2_{reltnt}).

7: Run Algorithm 1 with $\beta = \beta_{\text{abstnt}}$ (resp. $\beta = \beta_{\text{reltnt}}$). Denote its returned value by δ_{abstnt} (resp. δ_{reltnt}). {Skip Line 1 of Algorithm 1 as D is already available.}

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

9: Set

$$\beta_{\text{abscnf}} = \frac{\delta_{\text{abstnt}} \cdot \left((1-\varepsilon_M)^{1+4r(q+1)} - 2^{-\varepsilon_{\text{abs}}} \right)}{2qr(1+\varepsilon_M)^{1+4r(q+1)}}$$

$$\left(\text{resp. } \beta_{\text{relcnf}} = \frac{\delta_{\text{reltnt}}}{2qr} \cdot \left(\frac{1-\varepsilon_M}{1+\varepsilon_M} \right)^{1+4r(q+1)} \cdot \left(1 - \frac{(\delta_{\text{reltnt}} + 2qr\beta_{\text{reltnt}})^{\varepsilon_{\text{rel}}}}{(1-\varepsilon_M)^{(1+4r(q+1))(1+\varepsilon_{\text{rel}})}} \right) \right).$$

10: **if** $\beta_{\text{abscnf}} < \beta_{\text{abstnt}}$ (resp. $\beta_{\text{relcnf}} < \beta_{\text{reltnt}}$) **then**

11: Select the floating-point datatype such that $\alpha_M \leq \beta^2_{\text{abscnf}}$ (resp. β^2_{relcnf}).

12: Run Algorithm 1 with $\beta = \beta_{\text{abscnf}}$ (resp. $\beta = \beta_{\text{relcnf}}$). Denote its returned value by δ_{abscnf} (resp. δ_{relcnf}). {Skip Line 1 of Algorithm 1 as D is already available.}

13: **else**

14: Set $\delta_{\text{abscnf}} = \delta_{\text{abstnt}}$ (resp. $\delta_{\text{relcnf}} = \delta_{\text{reltnt}}$).

15: **end if**

16: **return** δ_{abscnf} (resp. δ_{relcnf}).

ALGORITHM 3: A Hybrid test of DFR.

better approximation δ_{abstnt} (resp. δ_{reltnt}) obtained by the tentative test determines β for a confirmatory test, denoted by β_{abscnf} for the absolute error ε_{abs} (resp. by β_{relcnf} for the relative error ε_{rel}). The final output δ_{abscnf} (resp. δ_{relcnf}) of the confirmatory test is ensured to satisfy the required precision.

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

$$\begin{cases} |\log_2 \delta_{\text{abscnf}} - \log_2 \delta_{\text{fail}}| \leq \varepsilon_{\text{abs}}; \\ |\log_2 \delta_{\text{relcnf}} / \log_2 \delta_{\text{fail}} - 1| \leq \varepsilon_{\text{rel}}. \end{cases} \tag{44}$$

Proof. Above all, the conditions Equation (42) in Corollary 1 follows from $\delta_{\text{fail}} \leq 1/2$, 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 δ_{fail} as below

$$\begin{cases} \delta_{\text{fail}} \leq (\delta_{\text{reltnt}} + 2qr\beta_{\text{reltnt}})(1 - \varepsilon_M)^{-1-4r(q+1)}; \\ \delta_{\text{fail}} \geq \delta \cdot (1 + \varepsilon_M)^{-1-4r(q+1)}, \delta \in \{\delta_{\text{abstnt}}, \delta_{\text{reltnt}}\}. \end{cases} \tag{45}$$

Hence, the trimming threshold β_{abscnf} (resp. β_{relcnf}) of Algorithm 3 is upper-bounded by the right hand of Equation (34) (resp. Equation (36)).

If $\beta_{\text{abscnf}} < \beta_{\text{abstnt}}$ (resp. $\beta_{\text{relcnf}} < \beta_{\text{reltnt}}$), then Algorithm 3 operates Line 12 and it follows from Corollary 1 that $\log_2 \delta_{\text{abscnf}}$ (resp. $\log_2 \delta_{\text{relcnf}}$) approximates $\log_2 \delta_{\text{fail}}$ with absolute (resp. relative) error not greater than ε_{abs} (resp. ε_{rel}).

If $\beta_{\text{abscnf}} \geq \beta_{\text{abstnt}}$ (resp. $\beta_{\text{relcnf}} \geq \beta_{\text{reltnt}}$), then β_{abstnt} (resp. β_{reltnt}) is also upper-bounded by the right hand of Equation (34) (resp. Equation (36)), implying that δ_{abstnt} (resp. δ_{reltnt}) has already met the desired precision. □

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

$$\delta_{\text{clt}} = \frac{1}{2} \sum_{x \in \mathbb{Z}_q} D_{e_3}(x) \cdot \left(\operatorname{erfc}\left(\frac{t - r \cdot m_D - x}{\sqrt{2r \cdot \sigma_D^2}} \right) + \operatorname{erfc}\left(\frac{t + 1 + r \cdot m_D + x}{\sqrt{2r \cdot \sigma_D^2}} \right) \right), \tag{46}$$

Input: the distributions D and D_{e_3}; the dimension r; the critical value t of decryption failure.
Output: a heuristic estimate of the DFR δ_{fail}.
1: Compute the mean m_D and the variance σ_D^2 of D, that is,
$m_D = \sum_{x\in\mathbb{Z}_q} x\cdot D(x)$ and $\sigma_D^2 = -m_D^2 + \sum_{x\in\mathbb{Z}_q} x^2\cdot D(x)$.
2: **return** $\delta_{\text{clt}} = \Pr_{x\leftarrow\$N(r\cdot m_D,\, r\cdot\sigma_D^2),\, y\leftarrow\$D_{e_3}}[x+y\notin[-t,t)]$,
where $N(r\cdot m_D, r\cdot\sigma_D^2)$ denotes the normal distribution with mean $r\cdot m_D$ and variance $r\cdot\sigma_D^2$.

Algorithm 4: A heuristic test of DFR.

Table 3: Trimming thresholds and DFR estimates.

Cipher	β	δ_{alg}	β_{abstnt}	β_{reltnt}	$\delta_{\text{abscnf}}(\delta_{\text{relcnf}})$
Kyber512	2^{-300}	2^{-139}	$2^{-185.49}$	$2^{-188.18}$	$2^{-138.94}$
Kyber768	2^{-300}	2^{-164}	$2^{-211.30}$	$2^{-213.76}$	$2^{-165.01}$
Kyber1024	2^{-300}	2^{-174}	$2^{-220.20}$	$2^{-222.60}$	$2^{-174.96}$
LightSaber	2^{-300}	2^{-120}	$2^{-162.98}$	$2^{-165.90}$	$2^{-120.35}$
Saber	2^{-300}	2^{-136}	$2^{-178.83}$	$2^{-181.60}$	$2^{-136.16}$
FireSaber	2^{-300}	2^{-165}	$2^{-208.50}$	$2^{-211.00}$	$2^{-165.26}$
Frodo640	10^{-200}	$2^{-138.7}$	$2^{-188.37}$	$2^{-191.06}$	$2^{-138.76}$
Frodo976	10^{-200}	$2^{-199.6}$	$2^{-254.40}$	$2^{-256.59}$	$2^{-199.60}$
Frodo1344	10^{-200}	$2^{-252.5}$	$2^{-310.07}$	$2^{-311.93}$	$2^{-252.60}$

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

$$\delta_{\text{clt}} \approx \Pr_{x\leftarrow\$N\left(r\cdot m_D + m_{D_{e_3}},\, r\cdot\sigma_D^2+\sigma_{D_{e_3}}^2\right)}[|x|\geq t], \tag{47}$$

where $m_{D_{e_3}}$ and $\sigma_{D_{e_3}}^2$ denote the mean and the variance of D_{e_3}, respectively. Generally speaking, Algorithm 4 costs more time than Equation (47) and yet gives a tighter approximation if D_{e_3} is far from a normal distribution. For example, SABER [23] has a uniform distribution D_{e_3} and its DFR $2^{-136.16}$, and Algorithm 4 yields an approximation $2^{-139.07}$ while Equation (47) derives a rough estimate $2^{-73.85}$. Therefore, Algorithm 4 is preferred to Equation (47) if the tentative test is expected to approximate the DFR with a high precision.

Remark 5. The confirmatory test in Algorithm 3 is not indispensable for specific applications. On the one hand, the inequalities Equation (15) are conservative and δ_{alg} is likely to be much closer to δ_{fail}. On the other hand, via the central limit theorem, the heuristic test possibly returns a value very near δ_{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.

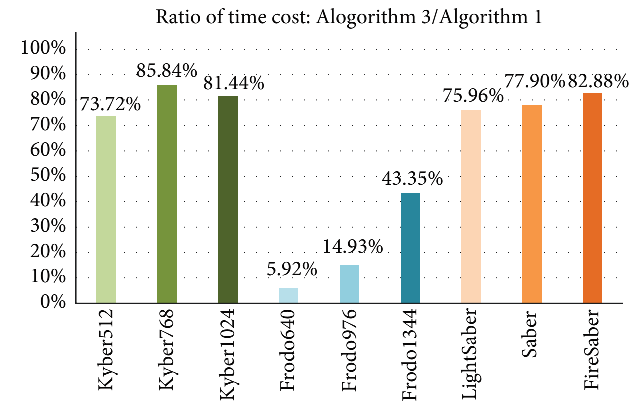

Figure 2: Ratio of time cost of Algorithm 3 to that of Algorithm 1.

3.4. An Experiment of DFR Test. Through the following experiment we compare Algorithm 3 with the previous DFR testing method in respect of their effectiveness and efficiency.

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 $\log_2 \delta_{\text{fail}}$ in Algorithm 3 is set $\varepsilon_{\text{abs}} = 5\times 10^{-3}$ (resp. $\varepsilon_{\text{rel}} = 5\times 10^{-6}$). For 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) i5-8350U 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 one hand, the data of Experiment 3 show that Algorithm 3, grounded on its theoretical proof (Theorem 3), ensures high accuracy to approximate δ_{fail} though its convolutions neglect more tiny probabilities than previous practical 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 therefore 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

Key generation	Encryption
1: $a \leftarrow \$ \mathscr{R}_q$	1: $s_2 \leftarrow \$ D^r_{s_2}\mathrm{B}$
2: $s_1 \leftarrow \$ D^r_{s_1}\mathrm{B}$	2: $e_2 \leftarrow \$ D^r_{e_2}\mathrm{B}$
3: $e_1 \leftarrow \$ D^r_{e_1}\mathrm{B}$	3: $e_3 \leftarrow \$ D^r_{e_3}\mathrm{B}$
4: $b = as_1 + e_1$	4: $c_1 = as_2 + e_2$
5: pk = (a, b)	5: $c_2 = bs_2 + e_3 + \text{Encode}\,(m)$
6: sk = s_1	6: ck = (c_1, c_2)

Decryption
1: Decode $(c_2 - c_1 s_1)$

pk = public key; sk = secret key; ck = ciphertext; m = plaintext.

FIGURE 3: The Lindner–Peikert encryption scheme using lattices over rings [20].

absolute error $\varepsilon_{\text{abs}} = 0.005$ over time running Algorithm 1 with assigned β in [15, 24, 25].

3.5. Use the Test for Practical Encryption Schemes. In the above, we only discussed the DFR determined by the distribution of $\mathbf{e}_1^T\mathbf{s}_2 - \mathbf{s}_1^T\mathbf{e}_2 + e_3 \bmod q$, setting other forms aside. When the plaintext is longer and enciphered in more than one elements of $\mathbb{Z}_q$, where c_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, $\mathscr{R}$ an order of K, and $\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{r-1}$ a basis of $\mathscr{R}$. Denote the quotient ring $\mathscr{R}/q\mathscr{R}$ by $\mathscr{R}_q$.

In Figure 3, $\boldsymbol{u} \leftarrow \$D^r\mathbf{B}$ means each coefficient of $\boldsymbol{u}$ with respect to the basis $\{\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{r-1}\}$ is sampled from D. Denote $\boldsymbol{e}_m = \sum_{k=0}^{r-1} e_m^{(k)}\boldsymbol{b}_k$ for $m \in \{1, 2, 3\}$ and $\boldsymbol{s}_m = \sum_{k=0}^{r-1} s_m^{(k)}\boldsymbol{b}_k$ for $m \in \{1, 2\}$.

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

$$\|\boldsymbol{e}_1\boldsymbol{s}_2 - \boldsymbol{s}_1\boldsymbol{e}_2 + \boldsymbol{e}_3 \bmod q\|_\infty \notin [-t, t), \tag{48}$$

where $\|\boldsymbol{e}_1\boldsymbol{s}_2 - \boldsymbol{s}_1\boldsymbol{e}_2 + \boldsymbol{e}_3 \bmod q\|_\infty$ denotes the greatest absolute value of the coefficients of $\boldsymbol{e}_1\boldsymbol{s}_2 - \boldsymbol{s}_1\boldsymbol{e}_2 + \boldsymbol{e}_3 \bmod q$ with respect to the basis $\{\boldsymbol{b}_0, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{r-1}\}$. Let $\boldsymbol{b}_i \cdot \boldsymbol{b}_j = \sum_{k=0}^{r-1} c_k^{(i,j)}\boldsymbol{b}_k$, $0 \leq i, j \leq r-1$. Then the coefficient of $\boldsymbol{b}_k$ in $\boldsymbol{e}_1\boldsymbol{s}_2 - \boldsymbol{s}_1\boldsymbol{e}_2 + \boldsymbol{e}_3 \bmod q$ is

$$\sum_{0 \leq i,j \leq r-1} c_k^{(i,j)}\left(e_1^{(i)}s_2^{(j)} - e_2^{(i)}s_1^{(j)}\right) + e_3^{(k)} \bmod q. \tag{49}$$

Let $D = \left(D_{e_1} \odot D_{s_2}\right) \circledast \left(C_{-1} \odot D_{s_1} \odot D_{e_2}\right)$ as in Line 2 of Algorithm 3. Then the distribution of Equation (49) is computed by

$$\left(\circledast_{0 \leq i,j < r}\left(C_{c_k^{(i,j)}} \odot D\right)\right) \circledast D_{e_3}. \tag{50}$$

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 $\mathscr{R} = \mathbb{Z}[x]/(x^r + 1)$ is the most popular in structured lattices [38], including NewHope [39], CRYSTALS-Kyber [21], and SABER [23]. As the conventional basis is $\boldsymbol{b}_k = x^k$, $0 \leq k \leq r-1$, we have

$$c_k^{(i,j)} = \begin{cases} 1, & 0 \leq i + j = k < r; \\ -1, & r \leq i + j = r + k; \\ 0, & \text{otherwise}. \end{cases} \tag{51}$$

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

$$D^{[k+1]} \circledast (C_{-1} \odot D)^{[r-1-k]} \circledast D_{e_3}, \tag{52}$$

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 $D_1 \circledast D_2$ 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 $D^{[r]} \circledast D_{e_3}$ and it is therefore feasible to compute δ_{fail} 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 $\boldsymbol{e}_1\boldsymbol{s}_2 - \boldsymbol{s}_1\boldsymbol{e}_2 + \boldsymbol{e}_3$ 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].

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 $\lfloor xp/q \rceil$, the integer nearest to xp/q, and this operation naturally extends on vectors in $\mathbb{Z}_q^r$ and algebraic numbers in $\mathscr{R}_q$. 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 $\mathbf{b}$ (or $\boldsymbol{b}$) and ciphertexts $\mathbf{c}_1, c_2$ (or $\boldsymbol{c}_1, \boldsymbol{c}_2$) in Figure 1 (or Figure 3). According to [21, Theorem 1], the DFR is computed the same as above except for that in Line 2 of Algorithm 3 and in Line 1 of Algorithm 1

$$D = \left(\left(D_{e_1} \circledast D_{p_1}\right) \odot D_{s_2}\right) \circledast \left(C_{-1} \odot \left(D_{e_1} \circledast D_{p_2}\right) \odot D_{s_1}\right). \tag{53}$$

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 adaption 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, CRYSTALS-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 $\odot$'s, two $\circledast$'s and relative errors in floating-point representation of D_{p_2} and D_{p_3}, it yields $m_0 = 4q + 2$.

3.5.3. 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 $\mathbf{s}_1$, and the corresponding public key is its syndrome $\mathbf{b} = \mathbf{A}^T\mathbf{s}_1 \in \mathbb{Z}_q^r$. The encryption algorithm chooses a pseudorandom LWE vector $\mathbf{c}_1 = \mathbf{A}\mathbf{s}_2 + \mathbf{e}_2 \bmod q$, and uses the syndrome $\mathbf{b}$ to generate one more LWE instance as a "pad" to hide the message, i.e., $\mathbf{c}_2 = \mathbf{b}^T\mathbf{s}_2 + e_3 + \mathrm{Encode}(m)$. The decryption algorithm proceeds similarly as in Regev [18] and Lindner and Peikert [19].

Then the key to obtaining δ_{fail} is to compute

$$\left(C_{-1} \odot \left(D_{s_1} \odot D_{e_2}\right)^{[w]}\right) \circledast D_{e_3}, \tag{54}$$

which characterizes the distribution of $-\mathbf{s}_1^T\mathbf{e}_2 + e_3 \bmod q$. Therefore, the results above in this paper also work for the dual Regev cryptosystem with slightly adaption.

Key generation	Encryption
1: $\mathrm{A} \leftarrow \$\ \mathbb{Z}_q^{w\times r}$ // $w > r$	1: $\mathrm{s}_2 \leftarrow \$\ D_{s_2}^r$
2: $\mathrm{s}_1 \leftarrow \$\ D_{s_1}^w$	2: $\mathrm{e}_2 \leftarrow \$\ D_{e_2}^w$
3: $\mathrm{b} = \mathrm{A}^T \cdot \mathrm{s}_1$	3: $e_3 \leftarrow \$\ D_{e_3}$
4: $\mathrm{pk} = (\mathrm{A}, \mathrm{b})$	4: $\mathrm{c}_1 = \mathrm{As}_2 + \mathrm{e}_2$
5: $\mathrm{sk} = \mathrm{s}_1$	5: $c_2 = \mathrm{b}^T \cdot \mathrm{s}_2 + e_3 + \mathrm{Encode}\ (m)$
Decryption	6: $\mathrm{ck} = (\mathrm{c}_1, c_2)$
1: Decode $(c_2 - \mathrm{s}_1^T \mathrm{c}_1)$	

pk = public key; sk = secret key; ck = ciphertext; m = plaintext.

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, δ_{fail} of the threshold ABE [45] can be determined by computing $\circledast_i (D_i \odot D_i')^{[w]}$, where D_i and D_i' 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 δ_{alg} of the "double-and-add" method with cipher parameters, the floating-point machine error ε_M and the trimming threshold β, 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 CRYSTALS-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 β 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 δ_{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:

$$
\begin{aligned}
&\Delta(D_1 \circledast D_2, E_1 \circledast E_2)\\
&= \sum_{k\in\mathbb{Z}_q} |D_1 \circledast D_2(k) - E_1 \circledast E_2(k)|\\
&= \sum_{k\in\mathbb{Z}_q} \left| \sum_{\substack{a,b\in\mathbb{Z}_q,\\ a+b\equiv k \bmod q}} D_1(a)\cdot D_2(b) - \sum_{\substack{c,d\in\mathbb{Z}_q,\\ c+d\equiv k \bmod q}} E_1(c)\cdot E_2(d) \right|\\
&= \sum_{k\in\mathbb{Z}_q} \left| \sum_{\substack{a,b\in\mathbb{Z}_q,\\ a+b\equiv k \bmod q}} (D_1(a)\cdot D_2(b) - E_1(a)\cdot E_2(b)) \right|\\
&= \sum_{k\in\mathbb{Z}_q} \left| \sum_{\substack{a,b\in\mathbb{Z}_q,\\ a+b\equiv k \bmod q}} (D_1(a) - E_1(a))\cdot D_2(b) + E_1(a)\cdot (D_2(b) - E_2(b)) \right|\\
&\le \sum_{k\in\mathbb{Z}_q} \sum_{\substack{a,b\in\mathbb{Z}_q,\\ a+b\equiv k \bmod q}} \left(|D_1(a) - E_1(a)|\cdot D_2(b) + E_1(a)\cdot |D_2(b) - E_2(b)| \right)\\
&= \sum_{k\in\mathbb{Z}_q} \sum_{\substack{a,b\in\mathbb{Z}_q,\\ a+b\equiv k \bmod q}} |D_1(a) - E_1(a)|\cdot D_2(b) + \sum_{l\in\mathbb{Z}_q} \sum_{\substack{c,d\in\mathbb{Z}_q,\\ c+d\equiv l \bmod q}} E_1(c)\cdot |D_2(d) - E_2(d)|\\
&= \sum_{a,b\in\mathbb{Z}_q} |D_1(a) - E_1(a)|\cdot D_2(b) + \sum_{c,d\in\mathbb{Z}_q} E_1(c)\cdot |D_2(d) - E_2(d)|\\
&= \sum_{a\in\mathbb{Z}_q} |D_1(a) - E_1(a)|\cdot \left(\sum_{b\in\mathbb{Z}_q} D_2(b)\right) + \left(\sum_{a\in\mathbb{Z}_q} E_1(a)\right)\cdot \sum_{b\in\mathbb{Z}_q} |D_2(b) - E_2(b)|\\
&\le \Delta(D_1, E_1) + \Delta(D_2, E_2).
\end{aligned}
\tag{A.1}
$$

□

B. Part of the proof of Lemma 5

The trimming error from Line 5 is estimated as

$$
\begin{aligned}
&\Delta\left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, D_i^{\text{I-dbl}}\right) && \\
&= \Delta\left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, \mathrm{Trim}_{S_i^{\mathrm{dbl}}}\left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}\right)\right) && \\
&\leq q \cdot \max_{a \in S_i^{\mathrm{dbl}}} \left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}\right)(a) && \text{using Equation (21)} \\
&\leq q \cdot (1-\varepsilon_M)^{q-4r(q+1)} \max_{a \in S_i^{\mathrm{dbl}}} \left(D_{i-1} \circledast D_{i-1}\right)(a) && \text{using Lemma 4} \\
&\leq q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. && \text{using Algorithm 2}
\end{aligned}
\tag{B.1}
$$

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

$$
\begin{aligned}
&\Delta\left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}, D_i^{\mathrm{I}}\right) && \\
&= \Delta\left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}, \mathrm{Trim}_{S_i^{\mathrm{add}}}\left(D_i^{\text{I-dbl}} \circledast r_{n-i} \cdot D_0^{\mathrm{I}}\right)\right) && \\
&\leq q \cdot \max_{a \in S_i^{\mathrm{add}}} \left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}\right)(a) && \text{using Equation (21)} \\
&\leq q \cdot (1-\varepsilon_M)^{-1-4r(q+1)} \max_{a \in S_i^{\mathrm{add}}} \left(D_i^{\mathrm{dbl}} \circledast D_0\right)(a) && \text{using Lemma 4} \\
&\leq q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. && \text{using Algorithm 2}
\end{aligned}
\tag{B.2}
$$

The trimming error from Line 12 is estimated as

$$
\begin{aligned}
&\Delta\left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}, D_{\mathrm{fin}}^{\mathrm{I}}\right) && \\
&= \Delta\left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}, \mathrm{Trim}_{S_{n+1}}\left(D_n^{\mathrm{I}} \circledast D_{e_3}\right)\right) && \\
&\leq q \cdot \max_{a \in S_{n+1}} \left(D_n^{\mathrm{I}} \circledast D_{e_3}^{\mathrm{I}}\right)(a) && \text{using Equation (21)} \\
&\leq q \cdot (1-\varepsilon_M)^{-1-4r(q+1)} \max_{a \in S_{n+1}} \left(D_n \circledast D_{e_3}\right)(a) && \text{using Lemma 4} \\
&\leq q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. && \text{using Algorithm 2}
\end{aligned}
\tag{B.3}
$$

C. Part of the proof of Lemma 6

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

$$
\begin{aligned}
&\Delta\left(P_i, D_i^{\mathrm{I}}\right) && \\
&\leq \Delta\left(P_i, D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}\right) + \Delta\left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}, D_i^{\mathrm{I}}\right) && \text{by Lemma 2} \\
&\leq \Delta\left(P_i^{\mathrm{dbl}} \circledast P_0, D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}\right) + \Delta\left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}, D_i^{\mathrm{I}}\right) && \text{by Equation (3)} \\
&\leq \Delta\left(P_i^{\mathrm{dbl}}, D_i^{\text{I-dbl}}\right) + \Delta\left(P_0, D_0^{\mathrm{I}}\right) + \Delta\left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}, D_i^{\mathrm{I}}\right) && \text{by Lemma 3} \\
&\leq \Delta\left(P_i^{\mathrm{dbl}}, D_i^{\text{I-dbl}}\right) + \Delta\left(D^{\mathrm{I}}, D_0^{\mathrm{I}}\right) + \Delta\left(D_i^{\text{I-dbl}} \circledast D_0^{\mathrm{I}}, D_i^{\mathrm{I}}\right) && \text{by } P_0 = D^{\mathrm{I}} \\
&\leq \Delta\left(P_i^{\mathrm{dbl}}, D_i^{\text{I-dbl}}\right) + 2q\beta(1-\varepsilon_M)^{-1-4r(q+1)} && \text{by Lemma 5}
\end{aligned}
\tag{C.1}
$$

and

$$
\begin{aligned}
&\Delta\left(P_i^{\mathrm{dbl}}, D_i^{\mathrm{I\text{-}dbl}}\right) \\
&\leq \Delta\left(P_i^{\mathrm{dbl}}, D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}\right) + \Delta\left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, D_i^{\mathrm{I\text{-}dbl}}\right) && \text{by Lemma 2} \\
&\leq \Delta\left(P_{i-1} \circledast P_{i-1}, D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}\right) + \Delta\left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, D_i^{\mathrm{I\text{-}dbl}}\right) && \text{by Equation (3)} \\
&\leq 2 \cdot \Delta\left(P_{i-1}, D_{i-1}^{\mathrm{I}}\right) + \Delta\left(D_{i-1}^{\mathrm{I}} \circledast D_{i-1}^{\mathrm{I}}, D_i^{\mathrm{I\text{-}dbl}}\right) && \text{by Lemma 3} \\
&\leq 2 \cdot \Delta\left(P_{i-1}, D_{i-1}^{\mathrm{I}}\right) + q\beta(1-\varepsilon_M)^{-1-4r(q+1)}. && \text{by Lemma 5}
\end{aligned} \tag{C.2}
$$

D. Proof of Corollary 1

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

$$2^{-\varepsilon_{\mathrm{abs}}}\delta_{\mathrm{fail}} \leq \delta_{\mathrm{fail}}(1-\varepsilon_M)^{1+4r(q+1)} - 2qr\beta. \tag{D.1}$$

Taking logarithm $\log_2$ on both sides implies

$$
\begin{aligned}
-\varepsilon_{\mathrm{abs}} &\leq \log_2\left(\delta_{\mathrm{fail}}(1-\varepsilon_M)^{1+4r(q+1)} - 2qr\beta\right) - \log_2 \delta_{\mathrm{fail}} \\
&\leq \log_2 \delta_{\mathrm{alg}} - \log_2 \delta_{\mathrm{fail}}. && \text{using Equation (15)}
\end{aligned} \tag{D.2}
$$

Furthermore, $\varepsilon_M \leq 1 - 2^{-\varepsilon_{\mathrm{abs}}/(1+4r(q+1))}$ 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

$$\varepsilon_M \leq 1 - 2^{-\varepsilon_{\mathrm{abs}}/(1+4r(q+1))} \leq 2^{\varepsilon_{\mathrm{abs}}/(1+4r(q+1))} - 1, \tag{D.3}$$

implying that

$$
\begin{aligned}
\varepsilon_{\mathrm{abs}} &\geq (1+4r(q+1))\log_2(1+\varepsilon_M) \\
&= \log_2\left(\delta_{\mathrm{fail}}(1+\varepsilon_M)^{1+4r(q+1)}\right) - \log_2 \delta_{\mathrm{fail}} \\
&\geq \log_2 \delta_{\mathrm{alg}} - \log_2 \delta_{\mathrm{fail}}. && \text{using Equation (15)}
\end{aligned} \tag{D.4}
$$

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] (β in the second column of Table 3), and the fourth and the fifth column show data of Algorithm 3 for absolute error $\varepsilon_{\mathrm{abs}} = 0.005$ and for relative error $\varepsilon_{\mathrm{rel}} = 5 \times 10^{-6}$, respectively.

Table 4: Algorithms 1 and 3 on Kyber512 [15].

	Algorithm 1	Algorithm 1 [15]	Algorithm 3 $\varepsilon_{\text{abs}} = 5 \times 10^{-3}$	Algorithm 3 $\varepsilon_{\text{rel}} = 5 \times 10^{-6}$
Compute D_0	Time = 0.19 ms			
Heuristic test	\	\	$\delta_{\text{clt}} = 2^{-147.61}$ Time = 0.81 ms	
Tentative test	$\beta = 0$ $\delta_{\text{alg}} = 2^{-138.94}$ Time = 39.18 ms	$\beta = 2^{-300}$ $\delta_{\text{alg}} = 2^{-138.94}$ Time = 14.27 ms	$\beta_{\text{abstnt}} = 2^{-185.49}$ $\delta_{\text{abstnt}} = 2^{-138.94}$ Time = 9.66 ms	$\beta_{\text{reltnt}} = 2^{-188.18}$ $\delta_{\text{reltnt}} = 2^{-138.94}$ Time = 10.91 ms
Confirmatory test	\	\	$\beta_{\text{abscnf}} = 2^{-176.82}$ $\delta_{\text{abscnf}} = 2^{-138.94}$ Time = 0.00 ms	$\beta_{\text{relcnf}} = 2^{-179.59}$ $\delta_{\text{relcnf}} = 2^{-138.94}$ Time = 0.00 ms
Total time	Time = 39.37 ms	Time = 14.46 ms	Time = 10.66 ms	Time = 11.91 ms

Table 5: Algorithms 1 and 3 on Kyber768 [15].

	Algorithm 1	Algorithm 1 [15]	Algorithm 3 $\varepsilon_{\text{abs}} = 5 \times 10^{-3}$	Algorithm 3 $\varepsilon_{\text{rel}} = 5 \times 10^{-6}$
Compute D_0	Time = 0.14 ms			
Heuristic test	\	\	$\delta_{\text{clt}} = 2^{-172.83}$ Time = 0.78 ms	
Tentative test	$\beta = 0$ $\delta_{\text{alg}} = 2^{-165.01}$ Time = 37.89 ms	$\beta = 2^{-300}$ $\delta_{\text{alg}} = 2^{-165.01}$ Time = 13.14 ms	$\beta_{\text{abstnt}} = 2^{-211.30}$ $\delta_{\text{abstnt}} = 2^{-165.01}$ Time = 10.48 ms	$\beta_{\text{reltnt}} = 2^{-213.76}$ $\delta_{\text{reltnt}} = 2^{-165.01}$ Time = 10.53 ms
Confirmatory test	\	\	$\beta_{\text{abscnf}} = 2^{-203.47}$ $\delta_{\text{abscnf}} = 2^{-165.01}$ Time = 0.00 ms	$\beta_{\text{relcnf}} = 2^{-206.00}$ $\delta_{\text{relcnf}} = 2^{-165.01}$ Time = 0.00 ms
Total time	Time = 38.03 ms	Time = 13.28 ms	Time = 11.40 ms	Time = 11.45 ms

Table 6: Algorithms 1 and 3 on Kyber1024 [15].

	Algorithm 1	Algorithm 1 [15]	Algorithm 3 $\varepsilon_{\text{abs}} = 5 \times 10^{-3}$	Algorithm 3 $\varepsilon_{\text{rel}} = 5 \times 10^{-6}$
Compute D_0	Time = 0.14 ms			
Heuristic test	\	\	$\delta_{\text{clt}} = 2^{-181.32}$ Time = 0.49 ms	
Tentative test	$\beta = 0$ $\delta_{\text{alg}} = 2^{-174.96}$ Time = 35.84 ms	$\beta = 2^{-300}$ $\delta_{\text{alg}} = 2^{-174.96}$ Time = 13.44 ms	$\beta_{\text{abstnt}} = 2^{-220.20}$ $\delta_{\text{abstnt}} = 2^{-174.96}$ Time = 10.43 ms	$\beta_{\text{reltnt}} = 2^{-222.60}$ $\delta_{\text{reltnt}} = 2^{-174.96}$ Time = 10.60 ms
Confirmatory test	\	\	$\beta_{\text{abscnf}} = 2^{-213.84}$ $\delta_{\text{abscnf}} = 2^{-174.96}$ Time = 0.00 ms	$\beta_{\text{relcnf}} = 2^{-216.29}$ $\delta_{\text{relcnf}} = 2^{-174.96}$ Time = 0.00 ms
Total time	Time = 35.98 ms	Time = 13.58 ms	Time = 11.06 ms	Time = 11.23 ms

Table 7: Algorithms 1 and 3 on Frodo640 [24].

	Algorithm 1	Algorithm 1 [24]	Algorithm 3 $\varepsilon_{abs} = 5 \times 10^{-3}$	Algorithm 3 $\varepsilon_{rel} = 5 \times 10^{-6}$
Compute D_0	Time = 1.22 ms			
Heuristic test	\	\	$\delta_{clt} = 2^{-148.87}$ Time = 0.24 ms	
Tentative test	$\beta = 0$ $\delta_{alg} = 2^{-138.76}$ Time = 4, 058.44 ms	$\beta = 10^{-200}$ $\delta_{alg} = 2^{-138.76}$ Time = 1, 057.97 ms	$\beta_{abstnt} = 2^{-188.37}$ $\delta_{abstnt} = 2^{-138.76}$ Time = 61.26 ms	$\beta_{reltnt} = 2^{-191.06}$ $\delta_{reltnt} = 2^{-138.76}$ Time = 64.56 ms
Confirmatory test	\	\	$\beta_{abscnf} = 2^{-178.26}$ $\delta_{abscnf} = 2^{-138.76}$ Time = 0.00 ms	$\beta_{relcnf} = 2^{-181.04}$ $\delta_{relcnf} = 2^{-138.76}$ Time = 0.00 ms
Total time	Time = 4, 059.66 ms	Time = 1, 059.19 ms	Time = 62.72 ms	Time = 66.02 ms

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

	Algorithm 1	Algorithm 1 [24]	Algorithm 3 $\varepsilon_{abs} = 5 \times 10^{-3}$	Algorithm 3 $\varepsilon_{rel} = 5 \times 10^{-6}$
Compute D_0	Time = 1.10 ms			
Heuristic test	\	\	$\delta_{clt} = 2^{-213.29}$ Time = 0.18 ms	
Tentative test	$\beta = 0$ $\delta_{alg} = 2^{-199.60}$ Time = 1, 803.06 ms	$\beta = 10^{-200}$ $\delta_{alg} = 2^{-199.60}$ Time = 431.80 ms	$\beta_{abstnt} = 2^{-254.40}$ $\delta_{abstnt} = 2^{-199.60}$ Time = 63.34 ms	$\beta_{reltnt} = 2^{-256.59}$ $\delta_{reltnt} = 2^{-199.60}$ Time = 64.01 ms
Confirmatory test	\	\	$\beta_{abscnf} = 2^{-240.71}$ $\delta_{abscnf} = 2^{-199.60}$ Time = 0.00 ms	$\beta_{relcnf} = 2^{-242.99}$ $\delta_{relcnf} = 2^{-199.60}$ Time = 0.00 ms
Total time	Time = 1, 804.16 ms	Time = 432.90 ms	Time = 64.62 ms	Time = 65.29 ms

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

	Algorithm 1	Algorithm 1 [24]	Algorithm 3 $\varepsilon_{abs} = 5 \times 10^{-3}$	Algorithm 3 $\varepsilon_{rel} = 5 \times 10^{-6}$
Compute D_0	Time = 0.37 ms			
Heuristic test	\	\	$\delta_{clt} = 2^{-268.49}$ Time = 0.06 ms	
Tentative test	$\beta = 0$ $\delta_{alg} = 2^{-252.60}$ Time = 148.71 ms	$\beta = 10^{-200}$ $\delta_{alg} = 2^{-252.60}$ Time = 56.08 ms	$\beta_{abstnt} = 2^{-310.07}$ $\delta_{abstnt} = 2^{-252.60}$ Time = 24.04 ms	$\beta_{reltnt} = 2^{-311.93}$ $\delta_{reltnt} = 2^{-252.60}$ Time = 24.10 ms
Confirmatory test	\	\	$\beta_{abscnf} = 2^{-294.17}$ $\delta_{abscnf} = 2^{-252.60}$ Time = 0.00 ms	$\beta_{relcnf} = 2^{-296.12}$ $\delta_{relcnf} = 2^{-252.60}$ Time = 0.00 ms
Total time	Time = 149.08 ms	Time = 56.45 ms	Time = 24.47 ms	Time = 24.53 ms

Table 10: Algorithms 1 and 3 on LightSaber [25].

	Algorithm 1	Algorithm 1 [25]	Algorithm 3 $\varepsilon_{\text{abs}} = 5 \times 10^{-3}$	 $\varepsilon_{\text{rel}} = 5 \times 10^{-6}$
Compute D_0	Time = 0.37 ms			
Heuristic test	\	\	$\delta_{\text{clt}} = 2^{-123.80}$ Time = 1.52 ms	
Tentative test	$\beta = 0$ $\delta_{\text{alg}} = 2^{-120.35}$ Time = 127.90 ms	$\beta = 2^{-300}$ $\delta_{\text{alg}} = 2^{-120.35}$ Time = 32.07 ms	$\beta_{\text{abstnt}} = 2^{-162.98}$ $\delta_{\text{abstnt}} = 2^{-120.35}$ Time = 22.75 ms	$\beta_{\text{reltnt}} = 2^{-165.90}$ $\delta_{\text{reltnt}} = 2^{-120.35}$ Time = 23.87 ms
Confirmatory test	\	\	$\beta_{\text{abscnf}} = 2^{-159.54}$ $\delta_{\text{abscnf}} = 2^{-120.35}$ Time = 0.00 ms	$\beta_{\text{relcnf}} = 2^{-162.50}$ $\delta_{\text{relcnf}} = 2^{-120.35}$ Time = 0.00 ms
Total time	Time = 128.27 ms	Time = 32.44 ms	Time = 24.64 ms	Time = 25.76 ms

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

	Algorithm 1	Algorithm 1 [25]	Algorithm 3 $\varepsilon_{\text{abs}} = 5 \times 10^{-3}$	 $\varepsilon_{\text{rel}} = 5 \times 10^{-6}$
Compute D_0	Time = 0.12 ms			
Heuristic test	\	\	$\delta_{\text{clt}} = 2^{-139.07}$ Time = 0.73 ms	
Tentative test	$\beta = 0$ $\delta_{\text{alg}} = 2^{-136.16}$ Time = 137.41 ms	$\beta = 2^{-300}$ $\delta_{\text{alg}} = 2^{-136.16}$ Time = 32.10 ms	$\beta_{\text{abstnt}} = 2^{-178.83}$ $\delta_{\text{abstnt}} = 2^{-136.16}$ Time = 24.25 ms	$\beta_{\text{reltnt}} = 2^{-181.60}$ $\delta_{\text{reltnt}} = 2^{-136.16}$ Time = 23.49 ms
Confirmatory test	\	\	$\beta_{\text{abscnf}} = 2^{-175.93}$ $\delta_{\text{abscnf}} = 2^{-136.16}$ Time = 0.00 ms	$\beta_{\text{relcnf}} = 2^{-178.73}$ $\delta_{\text{relcnf}} = 2^{-136.16}$ Time = 0.00 ms
Total time	Time = 137.53 ms	Time = 32.22 ms	Time = 25.10 ms	Time = 24.34 ms

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

	Algorithm 1	Algorithm 1 [25]	Algorithm 3 $\varepsilon_{\text{abs}} = 5 \times 10^{-3}$	 $\varepsilon_{\text{rel}} = 5 \times 10^{-6}$
Compute D_0	Time = 0.12 ms			
Heuristic test	\	\	$\delta_{\text{clt}} = 2^{-168.32}$ Time = 0.18 ms	
Tentative test	$\beta = 0$ $\delta_{\text{alg}} = 2^{-165.26}$ Time = 132.54 ms	$\beta = 2^{-300}$ $\delta_{\text{alg}} = 2^{-165.26}$ Time = 31.24 ms	$\beta_{\text{abstnt}} = 2^{-208.50}$ $\delta_{\text{abstnt}} = 2^{-165.26}$ Time = 25.69 ms	$\beta_{\text{reltnt}} = 2^{-211.00}$ $\delta_{\text{reltnt}} = 2^{-165.26}$ Time = 24.71 ms
Confirmatory test	\	\	$\beta_{\text{abscnf}} = 2^{-205.45}$ $\delta_{\text{abscnf}} = 2^{-165.26}$ Time = 0.00 ms	$\beta_{\text{relcnf}} = 2^{-207.97}$ $\delta_{\text{relcnf}} = 2^{-165.26}$ Time = 0.00 ms
Total time	Time = 132.66 ms	Time = 31.36 ms	Time = 25.99 ms	Time = 25.01 ms

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

[1] P. W. Shor, "Polynomial-time algorithms for discrete logarithms and factoring on a quantum computer," in *Algorithmic Number Theory*, L. M. Adleman and M. D. Huang, Eds., pp. 289–289, Springer, Berlin, Heidelberg, 1994.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[4] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[5] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology—EUROCRYPT '89*, J. J. Quisquater and J. Vandewalle, Eds., pp. 688-689, Springer, Berlin, Heidelberg, 1990.

[6] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology—CRYPTO '85 Proceedings*, H. C. Williams, Ed., pp. 417–426, Springer, Berlin, Heidelberg, 1986.

[7] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[8] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[9] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[10] National Institute of Standards and Technology, "Announcing request for nominations for public-key post-quantum cryptographic algorithms," 2016, https://csrc.nist.gov/News/2016/Public-Key-Post-Quantum-Cryptographic-Algorithms.

[11] National Institute of Standards and Technology, "Post-quantum cryptography PQC, selected algorithms 2022," September 2022, https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[12] National Institute of Standards and Technology, "Post-quantum cryptography PQC," September 2022, https://csrc.nist.gov/projects/post-quantum-cryptography.

[13] C. Peikert, "A decade of lattice cryptography," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016.

[14] National Institute of Standards and Technology, "NIST, post-quantum cryptography PQC round 3 submissions," 2022, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[15] P. Schwabe, "Crystals cryptographic suite for algebraic lattices," April 2022, https://pq-crystals.org/kyber/index.shtml.

[16] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems of Control and Information Theory*, vol. 15, no. 2, pp. 157–166, 1986.

[17] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: a ring-based public key cryptosystem," in *Algorithmic Number Theory*, J. P. Buhler, Ed., pp. 267–288, Springer, Berlin, Heidelberg, 1998.

[18] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, pp. 6–40, 2009.

[19] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Topics in Cryptology—CT-RSA 2011*, A. Kiayias, Ed., pp. 319–339, Springer, Berlin, Heidelberg, 2011.

[20] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *Journal of the ACM*, vol. 60, no. 6, pp. 6–35, 2013.

[21] J. Bos, L. Ducas, E. Kiltz et al., "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 353–367, IEEE, 2018.

[22] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *Advances in Cryptology—EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds., pp. 719–737, Springer, Berlin, Heidelberg, 2012.

[23] J. P. D'Anvers, A. Karmakar, S. Sinha Roy, and F. Vercauteren, "Saber: module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM," in *Progress in Cryptology—AFRICACRYPT 2018*, A. Joux, A. Nitaj, and T. Rachidi, Eds., pp. 282–305, Springer International Publishing, Cham, 2018.

[24] E. Alkim, J. W. Bos, L. Ducas et al., "FrodoKEM, practical quantum-secure key encapsulation from generic lattices," April 2022, https://frodokem.org.

[25] J. P. D'Anvers, A. Karmakar, S. S. Roy et al., "Saber MLWR-based KEM," April 2022, https://www.esat.kuleuven.be/cosic/pqcrypto/saber/.

[26] J. P. D'Anvers, Q. Guo, T. Johansson, A. Nilsson, F. Vercauteren, and I. Verbauwhede, "Decryption failure attacks on IND-CCA secure lattice-based schemes," in *Public-Key Cryptography—PKC 2019*, D. Lin and K. Sako, Eds., pp. 565–598, Springer International Publishing, Cham, 2019.

[27] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki-Okamoto transformation," in *Theory of Cryptography*, Y. Kalai and L. Reyzin, Eds., pp. 341–371, Springer International Publishing, Cham, 2017.

[28] H. Jiang, Z. Zhang, L. Chen, H. Wang, and Z. Ma, "IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited," in *Advances in Cryptology—CRYPTO*

2018, H. Shacham and A. Boldyreva, Eds., pp. 96–125, Springer International Publishing, Cham, 2018.

[29] J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, and G. Seiler, "Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. CCS '21*, pp. 2722–2737, Association for Computing Machinery, New York, NY, USA, 2021.

[30] N. Bindel and J. M. Schanck, "Decryption failure is more likely after success," in *Post-Quantum Cryptography*, J. Ding and J. P. Tillich, Eds., pp. 206–225, Springer International Publishing, Cham, 2020.

[31] J. P. D'Anvers, M. Rossi, and F. Virdia, "(One) failure is not an option: bootstrapping the search for failures in lattice-based encryption schemes," in *Advances in Cryptology—EUROCRYPT 2020*, A. Canteaut and Y. Ishai, Eds., pp. 3–33, Springer International Publishing, Cham, 2020.

[32] Y. Saad, "Floating point arithmetic—error analysis, lecture-notes of *computational aspects of matrix theory*," 2018, https://www-users.cselabs.umn.edu/Fall-2018/csci5304/FILES/LecN4.pdf.

[33] IEEE Computer Society, "IEEE standard for floating-point arithmetic," in *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, IEEE, 2019.

[34] D. Hofheinz and D. Unruh, "On the notion of statistical security in simulatability definitions," in *Information Security*, J. Zhou, J. Lopez, R. H. Deng, and F. Bao, Eds., pp. 118–133, Springer, Berlin, Heidelberg, 2005.

[35] N. Baulch, "Array types and conversions between types," 2004, https://www.codeproject.com/articles/6612/interpreting-intel-80-bit-long-double-byte-arrays.

[36] NumPy Developers, "Array types and conversions between types," October 2022, https://numpy.org/doc/stable/user/basics.types.html.

[37] C. Peikert and Z. Pepin, "Algebraically structured LWE, revisited," in *Theory of Cryptography*, D. Hofheinz and A. Rosen, Eds., pp. 1–23, Springer International Publishing, Cham, 2019.

[38] V. Lyubashevsky, C. Peikert, and O. Regev, "A toolkit for ring-LWE cryptography," in *Advances in Cryptology—EUROCRYPT 2013*, T. Johansson and P. Q. Nguyen, Eds., pp. 35–54, Springer, Berlin, Heidelberg, 2013.

[39] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—a new hope," in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 327–343, USENIX Association, Austin, TX, https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim, 2016.

[40] J. P. D'Anvers, F. Vercauteren, and I. Verbauwhede, "The impact of error dependencies on Ring/Mod-LWE/LWR based schemes," in *Post-Quantum Cryptography*, J. Ding and R. Steinwandt, Eds., pp. 103–115, Springer International Publishing, Cham, 2019.

[41] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[42] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.

[43] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. STOC '08*, pp. 197–206, Association for Computing Machinery, New York, NY, USA, 2008.

[44] D. Micciancio and C. Peikert, "Trapdoors for lattices: simpler, tighter, faster, smaller," in *Advances in Cryptology—EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds., pp. 700–718, Springer, Berlin, Heidelberg, 2012.

[45] J. Zhang, Z. Zhang, and A. Ge, "Ciphertext policy attribute-based encryption from lattices," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. ASIACCS '12*, pp. 16-17, Association for Computing Machinery, New York, NY, USA, 2012.

[46] D. Boneh, C. Gentry, S. Gorbunov et al., "Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits," in *Advances in Cryptology—EUROCRYPT 2014*, P. Q. Nguyen and E. Oswald, Eds., pp. 533–556, Springer, Berlin, Heidelberg, 2014.

[47] S. Gorbunov and D. Vinayagamurthy, "Riding on asymmetry: efficient abe for branching programs," in *Advances in Cryptology—ASIACRYPT 2015*, T. Iwata and J. H. Cheon, Eds., pp. 550–574, Springer, Berlin, Heidelberg, 2015.