Quantum-safe HIBE: does it cost a LATTE?

Raymond K. Zhao*[†], Sarah McCarthy[‡][§], Ron Steinfeld[†], Amin Sakzad[†] and Máire O'Neill[§]

* *CSIRO's Data61, Marsfield, Australia*

[†] *Monash University, Clayton, Australia*

[‡] *University of Waterloo, Waterloo, Canada*

[§] *Queen's University Belfast, Belfast, United Kingdom*

Abstract—**The UK government is considering advanced primitives such as identity-based encryption (IBE) for adoption as they transition their public-safety communications network from TETRA to an LTE-based service. However, the current LTE standard relies on elliptic-curve-based IBE, which will be vulnerable to quantum computing attacks, expected within the next 20–30 years. Lattices can provide quantum-safe alternatives for IBE. These schemes have shown promising results in terms of practicality. To date, several IBE schemes over lattices have been proposed, but there has been little in the way of practical evaluation. This paper provides the first complete optimised practical implementation and benchmarking of LATTE, a promising Hierarchical IBE scheme proposed by the United Kingdom (UK) National Cyber Security Centre (NCSC) in 2017 and endorsed by European Telecommunications Standards Institute (ETSI). We propose optimisations for the KeyGen, Delegate, Extract and Gaussian sampling components of LATTE, to increas attack costs, reduce decryption key lengths by 2x–3x, ciphertext sizes by up to 33%, and improve speed. In addition, we conduct a precision analysis, bounding the Rényi divergence of the Gaussian sampling procedures from the ideal distribution in corroboration of our claimed security levels. Our resulting implementation of the Delegate function takes 0.4 seconds at 80-bit security level on a desktop machine at 4.2GHz, significantly faster than the order of minutes estimated in the ETSI technical report. Furthermore, our optimised LATTE Encrypt/Decrypt implementation reaches speeds up to 9.7x faster than the ETSI implementation.**

Index Terms—**lattice-based cryptography, hierarchical identity-based encryption, advanced primitives, software design, post-quantum**

1. Introduction

The UK Government anticipates the migration of its mission-critical communications network from Airwave TETRA to LTE-based Emergency Services Network (ESN) [1] will be complete by 2026 [2]. However, the current standard [3] relies on Elliptic Curve (ECC)-based Identity Based Encryption (IBE) scheme MIKEY-SAKKE for securing messages. The first such device authorised for ESN is the Panasonic Toughbook Tablet which runs on Intel i5 and transmits data via EM7511 Band 14 mobile broadband. An IBE scheme removes the need for a certificate repository by deriving a user's public key from their already established public identity. This provides a low latency setup with instantaneous communication capabilities, hence is ideal for this use-case. However, ECC will be rendered insecure under quantum computing attacks, as acknowledged by current post-quantum cryptography standardization efforts by the National Institute of Standards and Technology (NIST) [4].

One of the advantages of lattice-based cryptography (LBC), a contender for quantum-secure cryptographic solutions, is the ability to build advanced primitives such as IBE. Furthermore, a hierarchy can be built into an IBE scheme to provide a more distributed workload and allow for finer-grained control over private key distribution. Hierarchical identity-based encryption (HIBE) schemes extend the concept of using a personal identity as a public key to a multi-levelled scenario, such as one would find within a functioning company. HIBE has further applications such as forward-secure encryption [5] and public key broadcast encryption [6]. Besides ESN, there are other real-world application scenarios for HIBE, for example, in Messaging apps for forward-security of ratchet protocols like Signal (see [7], [8]). It is also well known that the key generation algorithm of HIBE can be used as a Hierarchical ID-Based Signature (HIBS), and HIBS also has potential real-world applications in forward secure signatures used in blockchain [9]. Other real-world deployments of IBEs for encrypted file transfer and email are offered by companies such as Voltage Security and TrendMicro. Also, HP utilised it in their time data release service Time Vault. However, IBE is set to grow in the post-quantum world, where key sizes become larger and the number of connected devices demanding instantaneous data transfer grows. However, (H)IBE is still new territory within the post-quantum field. Additionally, with the growth of the Internet of Things (IoT), which brings with it complex interconnected systems of constrained devices, there is a greater requirement for lightweight, advanced primitives unlike ever before. The long-term security considerations indicate that these should be made quantum-secure today. The aim of this paper is to assess the practicality and optimise the implementation/integration of a quantum-safe (H)IBE scheme.

The DLP-IBE scheme [10] was in 2017 combined with the Bonsai tree HIBE scheme introduced in [11] to create LATTE by Campbell and Groves [12]. This research was carried out by the National Cyber Security Centre (NCSC), with a view to utilising the scheme in UK public safety communications. They are currently working with the European Telecommunications Standards Institute (ETSI) in a move towards standardising the scheme [13]. However, the proposed specification [13] only provides the Encrypt and Decrypt performance results, and it is unclear if LATTE KeyGen, Delegate, and Extract are practical at all. There remains substantial analysis to be performed to determine if and how this scheme will work in the real world. This is the gap our research endeavours to bridge.

This paper provides the first performance benchmarking of a quantum-safe HIBE scheme, LATTE, written in C.[1] We also identify bottlenecks, propose optimisations, and provide further statistical and security analysis for LATTE and consider its suitability for such applications. In more detail, the contributions of this paper are:

Precision Analysis of LATTE: We develop a statistical model for floating-point arithmetic errors in our efficient LATTE implementation, verified by experimental analysis. This allows us to quantify the security impact on LATTE of the arithmetic precision. In particular, we bound the Rényi divergence (RD) from ideal, as recommended in [14], of the Gaussian lattice sampler (with its underlying fast ffSampling algorithm), and deduce that 53 bits of precision retain our claimed security levels for LATTE-1 and 2 with up to 2^{42} key Extract/Delegate queries. For LATTE-3 and 4, our analysis shows that about 90 bits of precision should be sufficient. We also apply our statistical model to the FALCON signature selected by NIST for PQC standardisation [15], and demonstrate a ≈ 3 bit improved precision estimate for it using a refined analysis compared to that in [15].

Optimised LATTE (Sub-)Algorithms: We first reduce the module dimension of the extracted user keys by one compared to [13] by extending a similar approach used in the DLP IBE [10]. This leads to faster performance and reduces user private key sizes by 2x–3x and ciphertext length by up to 33%. In addition, we also show a faster ffLDL algorithm for (Mod)NTRU basis in Sec. 5.2. We then adapt the NTRUSolve function from FALCON [15] in order to efficiently solve the NTRU equation in our optimised LATTE KeyGen algorithm. The NTRUSolve is both faster and more compact [16] than the resultant method [10] used in [13]. In addition, we adapt the technique from MODFALCON [17] and the length reduction technique by using Cramer's rule [13] in order to efficiently solve the NTRU equation for higher lattice dimensions in our optimised LATTE Delegate algorithm. We further adapt the FFT sampling procedures from FALCON [15], which is faster than the Klein-GPV sampler [18] used in [13]. In addition, the proposed LATTE specification [13] did not discuss the integer discrete Gaussian sampling techniques suitable for the needed standard deviations. We integrate efficient sampling techniques, including FACCT [19] and the variant [20] of COSAC [21] in our optimised LATTE implementation.

[1]https://gitlab.com/raykzhao/latte

New Parameter Sets for LATTE: We provide slightly revised parameter sets for LATTE, fixing a bug in the computation of a lattice smoothing parameter in the ETSI technical report [13], and also modify a Gaussian sampling standard deviation parameter to accommodate the more efficient FACCT [19] sampler for the Key Generation algorithm. Security estimates for these revised parameters are also presented as we discover that our redesign reduces the decryption failure rate and increases the cost of recovering the user key.

First Full and Practical Implementation of LATTE: Applying our optimisation techniques, we give the first complete practical performance results for a lattice-based HIBE scheme, including the KeyGen, Delegate, and Extract algorithms, whereas implementation results were unclear in [13].The proposed specification [13] estimated that the Delegate would have run-time in the order of minutes on a desktop machine. In contrast, we show that our efficient implementation can perform the Delegate function in 0.4s (resp. 1.3s) for 80-bit (resp. 160-bit) security level on a desktop machine. In addition, for the same ring dimension, our optimised LATTE implementation is up to 11.1x faster than the DLP IBE implementation result from [13] for the corresponding algorithms, and our LATTE Extract run-time overhead is less than 3.9x over the FALCON Sign algorithm run-time with the same lattice dimension.

The structure of the paper is as follows. Sec. 2 gives the background to HIBE and the lattice-based concepts used in HIBE schemes. Sec. 3 describes our improved LATTE HIBE scheme. Sec. 4 provides the precision and security analyses. Sec. 5 discusses our implementation techniques in making the scheme practical for real-world applications. Performance results for the scheme are given in Sec. 6.

2. Preliminaries

A lattice can be expressed as a collection of integer linear combinations of a set of basis vectors. Popular underlying hard lattice problems believed to be secure against quantum computing attacks include the Shortest Vector Problem (SVP) and Learning With Errors (LWE) alongside its ring variant (over ideal lattices), Ring-LWE. These are all concerned with finding short vectors in the lattice, which can be attempted to be solved by lattice reduction algorithms such as LLL [22], and BKZ [23], [24]. Another common lattice problem is the NTRU assumption [25]; that is, given a polynomial $\mathbf{h}$, one must find non-trivial short $\mathbf{f}, \mathbf{g}$ such that $\mathbf{h} = \mathbf{g} \cdot \mathbf{f}^{-1}$.

In this paper, vectors or, interchangeably through the canonical embedding, polynomials will be denoted by bold small letters like $\mathbf{f}$, matrices $\mathbf{M}$, polynomial ring of integers mod q as $\mathcal{R}_q := \mathbb{Z}_q[x]/\langle x^N + 1\rangle$ (for an integer N), and lattices as Λ. The field of integers mod q is denoted as $\mathbb{Z}_q$. Discrete Gaussian distributions with centre t and standard deviation σ are denoted as $\mathcal{D}_{\sigma,t}$, and we omit the centre if it is zero, i.e. $\mathcal{D}_\sigma$ if $t = 0$. A distribution is B-bounded for some $B \in \mathbb{R}^+$, if its support is in the interval $[-B, B]$. The smoothing parameter of $\mathbb{Z}$ is denoted as $\eta_\varepsilon(\mathbb{Z}) = (1/\pi)\sqrt{\ln(2+2/\varepsilon)/2}$. The Euclidean norm of a vector/polynomial $\mathbf{f}$ is denoted $\|\mathbf{f}\|$. The transpose $\mathbf{f}^*$ of polynomial $\mathbf{f} = f_0 + f_1 x + \cdots + f_{N-1}x^{N-1}$ is defined as $\mathbf{f}^* = f_0 - f_{N-1}x - \cdots - f_1 x^{N-1}$. We denote $\mathbf{M}^*$ as the transpose of matrix $\mathbf{M}$ where $\mathbf{M}^*_{i,j} = (\mathbf{M}_{j,i})^*$. The Hermitian product of vectors $\mathbf{a}, \mathbf{b}$ is denoted as $\langle \mathbf{a}, \mathbf{b} \rangle$. The concatenation of several vectors $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N$ will be written as $(\mathbf{f}_1|\mathbf{f}_2|\cdots|\mathbf{f}_N)$. In HIBE schemes, user identities at level ℓ are denoted by ID_ℓ. A hash function from an arbitrary length input to a vector of integers of length N is written as $H : \{0,1\}^* \to \mathbb{Z}_q^N$. An arrow $\leftarrow\!\!\$$ is used to show the uniform random sampling of an element from a set, e.g. $\mathbf{f} \leftarrow\!\!\$\ \mathbb{Z}_q^N$. The operator $\oplus$ means XOR. A Gram-Schmidt orthogonalised basis of $\mathbf{B}$ is denoted as $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_N\}$. The notation $\mathcal{A}(\mathbf{f})$ refers to the anti-circulant matrix associated with polynomial $\mathbf{f}$. The notation $\lfloor k \rceil$ indicates the real number k is to be rounded to the nearest integer. The rounding $\lfloor \mathbf{f} \rceil$ of a polynomial $\mathbf{f}$ is taken to be coefficient-wise rounding. The Fast Fourier Transform (FFT) and Number Theoretic Transform (NTT) of polynomial $\mathbf{f}$ are the evaluations $\mathbf{f}(\zeta^i)$ for $i \in \{0, \ldots, N-1\}$, where ζ is the $2N$-th complex root of unity in the FFT, and ζ is the $2N$-th root of unity mod q in the NTT. Let $\odot$ be the point-wise multiplication.

Definition 2.1 (Rényi Divergence [26])**.** *For two discrete distributions P and Q such that $\operatorname{Supp}(P) \subseteq \operatorname{Supp}(Q)$, the Rényi divergence (RD) of order $a \in (1, +\infty)$ is defined as:*

$$R_a(P||Q) = \left(\sum_{x \in \operatorname{Supp}(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

For $a = +\infty$, we have: $R_\infty(P||Q) = \max_{x \in \operatorname{Supp}(P)} \frac{P(x)}{Q(x)}$.

Lemma 2.1. *Let $a \in [1, +\infty]$. Let P and Q denote distributions with $\operatorname{Supp}(P) \subseteq \operatorname{Supp}(Q)$. Then the following properties hold:*

Data Processing Inequality: *$R_a(P^f \| Q^f) \leq R_a(P\|Q)$ for any function f, where P^f (resp. Q^f) denotes the distribution of $f(y)$ induced by sampling $y \leftarrow\!\!\$\ P$ (resp. $y \leftarrow\!\!\$\ Q$).*

Probability Preservation: *Let $A \subseteq \operatorname{Supp}(Q)$ be an arbitrary event. If $a \in (1, +\infty)$, then $Q(A) \geq P(A)^{\frac{a}{a-1}}/R_a(P\|Q)$. Further, we have $Q(A) \geq P(A)/R_\infty(P\|Q)$.*

We use the notation $\lesssim, \sim$ as in [27], to "absorb" all higher-order terms of negligible elements, e.g. if $\delta = o(1)$, then $\delta + \delta^2 \sim \delta$. The following remark bounds $R_\infty(D_2; D_1)$.

Remark 1. *Let $\tau \in \mathbb{Z}$ be the tailcut bound as above, and let $Q = 2^k$ for some $k \in \mathbb{Z}$. If $\tau \geq \sqrt{2\ln(2Q)}$, then:*

$$R_\infty(D_2; D_1) \leq 1/(1 - Q^{-1}) \lesssim 1 + 1/Q. \quad (1)$$

This can be verified by using classical tailcut bounds [28, Lemma 4.4].

In our analysis, we will apply the following proposition, adapted from Proposition 4 of [29].

Proposition 2.2. *[29] Let P and Q denote two distributions of a $N-$tuple of random variables $(x_i)_{i<N}$. For $0 \leq i < N$, assume P_i (resp Q_i) is the marginal distribution of x_i, and let $P_{i|<i}(\cdot|x < i)$ denote the conditional distribution of x_i given that $(x_0, \ldots, x_{i-1}) =: x_{<i}$. Let $a > 1$. Suppose that for all $0 \leq i < N$, there exists $B_i \geq 1$ such that for all i-tuples $x_{<i}$ in the support of Q restricted to its first i variables, $R_a(Q_i|x_{<i}, P_i|x_{<i}) \leq B_i$. Then $R_a(Q, P) \leq \prod_{i<N} B_i$.*

Theorem 2.3 (Tail-cut Bound, Adapted from [26], Thm. 2.11)**.** *Let $\mathcal{D}'_\sigma$ be the B-bounded distribution of $\mathcal{D}_\sigma$ by cutting its tail. For M independent samples, we have $R_\infty((\mathcal{D}'_\sigma)^M || (\mathcal{D}_\sigma)^M) \leq \exp(1)$ if $B \geq \sigma \cdot \sqrt{2\ln(2M)}$.*

2.1. Hierarchical Identity-based Encryption

Hierarchical identity-based encryption (HIBE) schemes were introduced by Horwitz and Lynn [30] and can be considered a generalisation of an IBE scheme to multiple levels. A HIBE scheme consists of five components: Keygen, Delegate, Extract, Encrypt, and Decrypt. Gentry and Silverberg proposed the first secure HIBE scheme in the random oracle model (ROM) in 2002 [31], which was an extension of the Boneh-Franklin IBE scheme [32], a Weil-pairing based scheme, the security of which relies on the bilinear Diffie-Hellman problem. This was shown to be secure against adaptive identity and chosen ciphertext attacks, by use of the Fujisaki-Okamoto (FO) transformation [33], although the security degrades exponentially with the number of levels.

Boneh-Boyen built on this in 2004 to create a scheme that was secure without random oracles [34]. However, as both the ciphertext and private keys grew linearly with the number of levels of the hierarchy, in 2005 a scheme [35] was proposed which fixed the ciphertext size to three group elements and curtailed private key growth to within level ℓ group elements. In 2018, an isogeny-based version of the Decisional Bilinear Diffie-Hellman-based scheme was proposed [36]. Despite isogenies possessing quantum-safe properties, this variant only serves to strengthen the existing classical security, by proving it secure under the assumption of *either* the classical version or the isogeny-based version of the problem and therefore is not necessarily quantum-safe. To the best of the authors' knowledge, the only quantum-safe HIBE schemes so far proposed are based on lattices. We now introduce the schemes upon which LATTE is built.

2.2. The Ingredients of LATTE

DLP IBE Scheme: In 2014, Ducas, Lyubashevsky and Prest proposed the first efficient lattice-based identity-based encryption (IBE) scheme [10]. They based their construction on the IBE scheme by Gentry, Peikert and Vaikuntanathan [18], using a variant of NTRU lattices. The underlying security problems are the NTRU problem for key generation and R-LWE for encryption. The ciphertexts, therefore, have more practical sizes than previous constructions, for example, 30kb (kilobits) for 192-bit classical security. The use of structured lattices also allowed for implementation optimisations such as the Number Theoretic Transform (NTT), as demonstrated by [37], whose software performance of the DLP-IBE outperformed that of the elliptic-curve-based Boneh-Franklin IBE scheme.

Bonsai Trees HIBE: Cash [11] proposed the use of Bonsai trees to create a hierarchical structure for IBE. They model the hierarchical network of users as a tree, whereby arborists, or sub-key-managers, have control over the subtrees and have the authority to delegate user private keys. Delegation requires the knowledge of a trapdoor basis of the lattice at that level. During the process whereby keys are delegated down the tree, the lattice is extended, and therefore its dimension and hence the key and ciphertext sizes increase. The public key size is of $\mathcal{O}(d^3kn^2)$ and ciphertext size is of $\mathcal{O}(d^3kn)$ at depth d, for security parameter n and hash output length k. The root authority has control of the whole tree by knowing the short trapdoor basis for the master root lattice. The security of this HIBE scheme is based on LWE over standard lattices.

3. Improved LATTE HIBE Scheme

LATTE was proposed in 2017 [12] and can be considered as a combination of the DLP-IBE scheme [10] and Bonsai Tree HIBE scheme [11] to create a lattice-based hierarchical IBE (HIBE) scheme. It can be shown to be ID-IND-CCA-secure, the proof for which is given in [13], based on the NTRU and R-LWE hardness assumptions.

3.1. Proposed Design Optimisations

For the optimised LATTE scheme presented in this Section and used in our software design and implementation, features of the FALCON [15] and the MODFALCON [17] signature schemes were utilised. This is the first time these features have been considered in LATTE, and so the rationale for this is expanded on in Sec. 5. For now, it suffices to acknowledge that the sub-algorithms NTRUSolve and ffSampling are taken from FALCON [15]. The currently presented LATTE in this Section also improves on the efficiency of the original proposal [12] by reducing the module dimension of the extracted secret keys by 1, by extending a similar approach used in the DLP IBE [10]. More concretely, we eliminate public key polynomial $\mathbf{b}$ by modifying the equation satisfied by the decryption key at level ℓ from the original rank $\ell+2$ module relation over $\mathcal{R}_q$:

$$\mathbf{t}_0 + \mathbf{t}_1 \cdot \mathbf{h} + \mathbf{t}_2 \cdot \mathbf{A}_1 + \cdots + \mathbf{t}_\ell \cdot \mathbf{A}_{\ell-1} + \mathbf{t}_{\ell+1} \cdot \mathbf{A}_\ell = \mathbf{b}, \quad (2)$$

where $\mathbf{A}_i = H(\mathsf{ID}_1|\ldots|\mathsf{ID}_i)$ for $1 \le i \le \ell$, to the following rank $\ell+1$ relation over $\mathcal{R}_q$:

$$\mathbf{t}_0 + \mathbf{t}_1 \cdot \mathbf{h} + \mathbf{t}_2 \cdot \mathbf{A}_1 + \cdots + \mathbf{t}_\ell \cdot \mathbf{A}_{\ell-1} = \mathbf{A}'_\ell, \quad (3)$$

where $\mathbf{A}'_\ell = H_E(\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell) := H(\texttt{"E"}|\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell)$. Furthermore, we remove the need for the Extract algorithm to be stateful. This is achieved by deriving randomness deterministically from the ID (see Sec. 4.4 for discussion).

3.2. Scheme Description

The full pseudocode for LATTE KeyGen, Delegate, Extract, Encryption, and Decryption are presented in Figure 3–6 in Appendix A, respectively. Let $H : \{0,1\}^* \to \mathbb{Z}_q^N$ and $\mu, Z \in \{0,1\}^{256}$, Table 6 in Appendix A further summarises the inputs and outputs of the LATTE algorithms. The KeyGen algorithm, given in Figure 3 in Appendix A, generates an NTRU-type basis. This is performed by sampling the short basis polynomials $\mathbf{f}, \mathbf{g}$ from a Gaussian distribution. Operations are over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^N + 1\rangle$, a variant of the NTRU ring. For the purposes of optimisation in the implementation, variables are stored in NTT representation where appropriate. The Gram-Schmidt norm of the associated basis is computed to ensure smallness allowing for short private keys to be delegated to the next level. If not, the polynomials are re-sampled. The rest of the basis, polynomials $\mathbf{F}, \mathbf{G}$, are computed so that they satisfy the NTRU equation, $\mathbf{fG} - \mathbf{gF} = q \mod x^N + 1$. This sub-algorithm is referred to as NTRUSolve, and its implementation will be discussed in Sec. 5. The solution to this is not unique, but any solution suffices provided it is short enough. This is taken care of by reduction of the coefficients. The public key consists of polynomial $\mathbf{h} = \mathbf{g} \cdot \mathbf{f}^{-1}$. The master public basis $\mathbf{B}_0$ and private basis $\mathbf{S}_0$ at level 0 are implicit in the polynomial master keys, as follows:

$$\mathbf{B}_0 = \begin{bmatrix} -\mathcal{A}(\mathbf{h}) & \mathbf{I}_N \\ q\mathbf{I}_N & \mathbf{0}_N \end{bmatrix}, \qquad \mathbf{S}_0 = \begin{bmatrix} \mathbf{g} & -\mathbf{f} \\ \mathbf{G} & -\mathbf{F} \end{bmatrix}.$$

The Delegate process, given in Figure 1, creates a public/secret key pair for the next level in the tree, allowing it to become a sub-key management service (sub-KMS). Suppose the KMS wishes to delegate a key from level $\ell-1$ to level ℓ. Then it can extend the public basis of the user at level ℓ, denoted by B_ℓ by placing $\mathbf{A}_\ell = H(\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell)$, where H is a hash function, to the beginning of the first column and filling the extra row with $\mathbf{I}_N$ and $\mathbf{0}_N$, as shown below. The dimension of the new matrix becomes $(\ell+2)N \times (\ell+2)N$. The corresponding private basis, S_ℓ, can then be generated. The i^{th} row $(\mathbf{s}_{i,0}, \mathbf{s}_{i,1}, \ldots, \mathbf{s}_{i,\ell+1})$ of the private basis is a short solution to the equation:

$$\mathbf{s}_{i,0} + \mathbf{s}_{i,1} \cdot \mathbf{h} + \mathbf{s}_{i,2} \cdot \mathbf{A}_1 + \cdots + \mathbf{s}_{i,\ell+1} \cdot \mathbf{A}_\ell = \mathbf{0} \mod q.$$

This can be found by sampling short vectors (using the Klein-GPV sampler [18] or its variant from FALCON [15]) from the $(\ell-1)$-level lattice using its secret basis, with centre vector $(-\mathbf{s}_{i,\ell+1} \cdot \mathbf{A}_\ell, \mathbf{0}, \ldots, \mathbf{0})$, where $\mathbf{s}_{i,\ell+1}$ is sampled from a discrete Gaussian distribution $\mathcal{D}_{\sigma_\ell}$ over $\mathcal{R}$. A check is made to ensure the GS-norm of the sampled lattice vector is within the bound $\sigma_\ell \cdot \sqrt{(\ell+2)N}$ to ensure the delegated basis will be of sufficient quality.

The remainder of the Delegate algorithm, in which the bottom row $(\mathbf{s}_{\ell+1,0}, \mathbf{s}_{\ell+1,1}, \ldots, \mathbf{s}_{\ell+1,\ell+1})$ is generated, is a higher-dimensional analogue of LATTE KeyGen. The resulting matrix has a determinant of size q. The final row is then reduced similarly to the KeyGen component to ensure the basis is of the required quality for further delegation. Cramer's rule is utilised here to find the reduction coefficients. Generalising to level ℓ, the public basis $\mathbf{B}_\ell$ and the private basis $\mathbf{S}_\ell$, respectively become:

$$\mathbf{B}_\ell = \begin{bmatrix} -\mathcal{A}(\mathbf{A}_\ell) & \mathbf{0}_N & \dots & \mathbf{I}_N \\ \vdots & \vdots & \ddots & \vdots \\ -\mathcal{A}(\mathbf{h}) & \mathbf{I}_N & \dots & \mathbf{0}_N \\ q\mathbf{I}_N & \mathbf{0}_N & \dots & \mathbf{0}_N \end{bmatrix},$$

and $\mathbf{S}_\ell = [\mathbf{s}_{i,j}]$, $0 \le i, j \le \ell + 1$.

In the LATTE Extract algorithm (Figure 4 in Appendix A), the user private key is a short solution $(\mathbf{t}_0, \mathbf{t}_1, \dots, \mathbf{t}_\ell)$ to:

$$\mathbf{t}_0 + \mathbf{t}_1 \cdot \mathbf{h} + \mathbf{t}_2 \cdot \mathbf{A}_1 + \dots + \mathbf{t}_\ell \cdot \mathbf{A}_{\ell-1} = \mathbf{A}_\ell \mod q, \quad (4)$$

where $\mathbf{A}_i = H(\mathsf{ID}_1|\dots|\mathsf{ID}_i)$ for $1 \le i \le \ell$. This is found using the Klein-GPV style sampler over the short basis from the previous level. An extended version of traditional R-LWE encryption/decryption [38] is used for ciphering messages as given in Figure 5 and 6 in Appendix A, respectively. A random *seed* is sampled and used together with a Key Derivation Function (KDF) to one-time-pad the message. The *seed* is encoded[2] and then encrypted using Ring-LWE and sent. The ciphertext consists of the encrypted message Z and deterministically sampled ephemeral public keys $\mathbf{C}_1, \dots, \mathbf{C}_\ell, \mathbf{C}_h$. This is a variant of the FO transform [33] to protect against invalid ciphertexts. The Decrypt process takes the user's private key to decrypt the *seed* and reconstruct the message. Using definitions of $\mathbf{C}_h, \mathbf{C}_i$, $1 \le i \le \ell$ and (4), we have

$$\mathbf{V} = \mathbf{e}_\ell + \mathbf{m} - \mathbf{t}_1 \cdot \mathbf{e}_h - \mathbf{t}_2 \cdot \mathbf{e}_1 - \dots - \mathbf{t}_\ell \cdot \mathbf{e}_{\ell-1} + \mathbf{t}_0 \cdot \mathbf{e}.$$

By construction, the error and private key terms are small enough so that $\mathbf{m}$ is decoded successfully to recover the *seed*. From the *seed*, the message is straightforwardly recovered from Z, which is sent as part of the ciphertext.

```
Input: $N, q, \sigma_\ell, \mathbf{S}_{\ell-1}, H : \{0,1\}^* \to \mathcal{R}_q, \mathsf{ID}_\ell$.
Output: $\mathbf{S}_\ell \in \mathcal{R}_q^{(\ell+2)\times(\ell+2)}$.
 1: function Delegate
 2:     $\mathbf{A}_\ell \leftarrow H(\mathsf{ID}_1|\dots|\mathsf{ID}_\ell)$ in NTT domain.
 3:     $T_{\ell-1} \leftarrow$ ffLDL(FFT($\mathbf{S}_{\ell-1} \cdot \mathbf{S}^*_{\ell-1}$)).
 4:     For each leaf of $T_{\ell-1}$, leaf.value $\leftarrow \sigma_\ell/\sqrt{\text{leaf.value}}$.
 5:     $seed \leftarrow_\$ \{0,1\}^{256}$.
 6:     for $i \in \{0,\dots,\ell\}$ do
 7:         $\mathbf{s}_{i,\ell+1} \leftarrow \mathcal{D}^N_{\sigma_\ell}$.
 8:         $\mathbf{t} \leftarrow (-\mathbf{s}_{i,\ell+1} \cdot \mathbf{A}_\ell, \mathbf{0}, \dots, \mathbf{0}) \cdot \mathbf{S}^{-1}_{\ell-1}$.
 9:         $\mathbf{z} \leftarrow \text{FFT}^{-1}(\text{ffSampling}(\mathbf{t}, T_{\ell-1}, seed))$.
10:         $(\mathbf{s}_{i,0},\dots,\mathbf{s}_{i,\ell}) \leftarrow \lfloor \bar{\mathbf{z}} \rceil$, where $\bar{\mathbf{z}} \leftarrow (\mathbf{t} - \mathbf{z})\mathbf{S}_{\ell-1}$.
11:         if $\|(\mathbf{s}_{i,0},\dots,\mathbf{s}_{i,\ell+1})\| > \sqrt{(\ell+2)N} \cdot \sigma_\ell$ then
12:             Resample.
13:         end if
14:     end for
15:     Set $\mathbf{M} = (\mathbf{s}_{i,j})$, for $0 \le i \le \ell$, $1 \le j \le \ell+1$.
16:     if $\mathbf{M}$ is not invertible then
17:         goto Step 4.
18:     end if
19:     $\mathbf{u} \leftarrow \text{adj}(\mathbf{M}) \cdot (\mathbf{s}_{0,0}, \mathbf{s}_{1,0}, \dots, \mathbf{s}_{\ell,0})^\mathsf{T}$.
20:     $(\mathbf{F}_\ell, \mathbf{G}_\ell) \leftarrow \text{NTRUSolve}_{N,q}(\det(\mathbf{M}), \mathbf{u}_0)$, where $\mathbf{u}_0$ is the first coordinate of $\mathbf{u}$.
21:     if NTRUSolve is aborted then
22:         goto Step 4.
23:     end if
24:     $(\mathbf{s}_{\ell+1,0},\dots,\mathbf{s}_{\ell+1,\ell+1}) \leftarrow (\mathbf{G}_\ell, \mathbf{F}_\ell, \mathbf{0}, \dots, \mathbf{0})$.
25:     Set $\mathbf{C} = (\mathbf{c}_{i,j})$, where $\mathbf{c}_{i,j} = \mathbf{s}_{j,0} \cdot \mathbf{s}^*_{i,0} + \dots + \mathbf{s}_{j,\ell+1} \cdot \mathbf{s}^*_{i,\ell+1}$, $0 \le i, j \le \ell$.
26:     Let $\mathbf{k} = (\mathbf{k}_i)_{0 \le i \le \ell}$ be the solution to $\mathbf{C} \cdot \mathbf{k} = \mathbf{d}$. By Cramer's rule, $\mathbf{k}_i = \frac{\det(\mathbf{C}_i(\mathbf{d}))}{\det(\mathbf{C})}$, where $\mathbf{C}_i(\mathbf{d})$ is the matrix $\mathbf{C}$ with its $i^{th}$ column replaced by $\mathbf{d}_i = \mathbf{s}_{\ell+1,0} \cdot \mathbf{s}^*_{i,0} + \dots + \mathbf{s}_{\ell+1,\ell+1} \cdot \mathbf{s}^*_{i,\ell+1}$.
27:     for $i \in \{0,\dots,\ell\}$ do
28:         $(\mathbf{s}_{\ell+1,0},\dots,\mathbf{s}_{\ell+1,\ell+1}) = (\mathbf{s}_{\ell+1,0},\dots,\mathbf{s}_{\ell+1,\ell+1}) - \lfloor \mathbf{k}_i \rceil \cdot (\mathbf{s}_{i,0},\dots,\mathbf{s}_{i,\ell+1})$.
29:     end for
30:     return $\mathbf{S}_\ell = (\mathbf{s}_{i,j})$, for $0 \le i, j \le \ell+1$.
31: end function
```

Figure 1. The LATTE Delegate algorithm (from level $\ell - 1$ to ℓ).

4. Security Analysis

A recurring concern around LBC is the precision requirements of the implementation, in particular, of the discrete Gaussian sampler. As noted in [39], the precision used is often excessive, leading to slow and impractical implementations. Traditional measures of statistical distance have recently been substituted for RD or Kullback-Leibler (KL) divergence to reduce memory and computational resources whilst maintaining security. In this Section, we make use of the RD argument initially proposed in [14] to answer the question of how low we can allow the precision of our implementation to be, without allowing an adversary to detect any distinction between the actual distribution and the ideal distribution of a true Gaussian sample *over the lattice*, hence maintaining our claimed security levels. In particular, we analyze the security impact on LATTE of finite precision errors in the floating-point arithmetic and in the $\mathbb{Z}$-samplers used in our implementation of the Extract and Delegate algorithm based on the ffSampling lattice Gaussian algorithm. For this, we follow the following steps:

Step 1 - RD Security Reduction: We give a security reduction (Sec. 4.1) based RD analysis to relate the security of finite precision LATTE to the security of its ideal (infinite precision) implementation, and bounds on the errors in the centre and standard deviation parameters of $\mathbb{Z}$ Gaussian samples used in lattice Gaussian ffSampling algorithm.

[2]The Encode/Decode are the same as described in [13].

Step 2 - RD Between $\mathbb{Z}$-Gaussians with Errors in Parameters: To support the above security reduction, we give a tight Lemma (in Sec. 4.2) giving a bound on RD between the output distribution of $\mathbb{Z}$ Gaussian samplers with errors in the centre and standard deviation parameters, extending the sharp RD results of [14].

Step 3 - Statistical Model for ffSampling Precision Errors: For use with the above security reduction, we introduce and empirically verify (in Sec. 4.3) a heuristic statistical model to compute upper bounds on the finite precision errors in the lattice Gaussian ffSampling algorithm. We give empirical evidence for the validity of our model, use it to compute estimated error bounds for LATTE parameter sets, and apply these with the above security reductions to evaluate the security impact of precision errors on LATTE.

4.1. RD Security Reduction

The security reduction to establish the (full chosen identity) indistinguishability against chosen-ciphertext attack security (ID-IND-CCA) of the IDEAL (infinite precision) LATTE HIBE encryption scheme is summarised in Annex C of the LATTE specification [13] and proceeds in two steps. Here, we show how to obtain a security reduction that takes into account and quantifies the security impact of the REAL (finite precision) implementation of LATTE. To do so, we introduce an additional middle step (step 2 below) in the security reduction steps for LATTE, and so we end up with the following three security reduction steps:

Step 1 - FO Transform: This generic reduction transforms any ID-IND-CCA attack (chosen ID indistinguishability against chosen ciphertext attacks) against REAL (finite precision) LATTE to an ID-OW-CPA (chosen ID onewayness against chosen plaintext attacks) of REAL (finite precision) LATTE$'$, assuming the random oracle model for the LATTE KDF hash function. Here, LATTE$'$ denotes the ID-OW-CPA encryption scheme underlying LATTE: the encryption/decryption algorithms of LATTE can be obtained by applying the tag-based Fujisaki-Okamoto KEM-DEM transform of [40] to the LATTE$'$ scheme. As pointed out in Annex C of [13], this reduction step follows directly from an (ID-based variant) of the composition of Theorem 3.1 and Theorem D.1 in [40].

Step 2 - RD - REAL to IDEAL: This presented reduction (in Lemma 4.1 below) transforms any ID-OW-CPA attack against REAL (finite precision) LATTE$'$ to an ID-OW-CPA attack against IDEAL (infinite precision) LATTE$'$, by using RD analysis techniques [14], [26]. For the latter RD reduction to apply, we exploit the fact that the onewayness notion ID-OW-CPA is a *search* problem rather than a *decision* problem.

Step 3 - ID-OW-CPA to NTRU/RLWE: This reduction transforms any attack against IDEAL LATTE$'$ ID-OW-CPA security into attacks against the NTRU or RLWE problems, assuming the random oracle model for the ID hash function H. As pointed out in Annex C of [13], this reduction is a variant of the Bonsai tree reduction presented in Theorem 5.2 of [41], with a minor modification for our improved LATTE construction (see Sec. 4.4 for more details).

The following result fills the missing Step 2 above (where we apply Lemma 4.1 with the ID-OW-CPA attack game and the event E being the winning of this game by the adversary), and quantifies the security impact of finite precision in the discrete integer Gaussian samplers and the floating point arithmetic used in the FFT lattice Gaussian sampler used within the LATTE$'$ Delegate algorithm (Figure 1 in Sec. 3.2) and Extract algorithm (Figure 4 in Appendix A). The latter security impact is expressed as a function of upper bounds $\delta^U_{\sigma^i}$ and $\Delta^U_{t^i}$ on the relative (resp. absolute) finite precision errors in the integer discrete Gaussian standard deviation parameters $\sigma^{(i)}$ (resp. Gaussian centre parameters $t^{(i)}$) used inside the Delegate and Extract algorithms, and an upper bound Δ^U_z on the absolute error in the final output value of the FFT sampling algorithm. We also allow for a negligible probability p_U (over the randomness of the key generation and discrete Gaussian samplers) that the above error upper bounds fail to hold. The next subsection explains our statistical model and results for estimating the latter error upper bounds and the probability p_U for the chosen implementation finite precision.

Consider an attack game REAL against LATTE$'$ with depth parameter d where the attack algorithm $\mathcal{A}$ is run on input a LATTE$'$ master public key h (where $(S_0, h) \leftarrow \text{KeyGen}(N, q, \sigma_0)$), makes at most Q_D total number of queries to the Delegate algorithm (Figure 1 in Sec. 3.2) and Q_E queries to the Extract algorithm (Figure 4 in Appendix A) implemented with:

- A finite precision $p_{\mathcal{D}}$ 1-dimensional Discrete Gaussian $\mathbb{Z}$-sampling algorithm in Lines 4–5 of Figure 7 in Appendix A outputting samples from a distribution $\bar{\mathcal{D}}_{\sigma,t}$ within RD of order a at most B from the ideal Discrete Gaussian distribution $\mathcal{D}_{\sigma,t}$ i.e. $R_a(\bar{\mathcal{D}}_{\sigma,t}, \mathcal{D}_{\sigma,t}) \leq B$.
- A finite precision p_{fp} floating-point arithmetic for Figure 1 in Sec. 3.2, Figure 4 in Appendix A, and Lines 11–14 of Figure 7 in Appendix A.

Let IDEAL denote the attack game against the ideal implementation of LATTE' where both $p_{\mathcal{D}}$ and p_{fp} are infinite precision. Let $(t^{(i)}, \sigma^{(i)})$ denote the center and std dev. parameter (resp.) for the i'th query to the 1-dim. $\mathbb{Z}$ Gaussian sampler (i.e. at Line 2 of Figure 3 in Appendix A, Line 4 or 5 of Figure 7 in Appendix A, or Line 7 of Figure 1 in Sec. 3.2) in the game IDEAL, and let $\bar{\mathbf{z}}^{(j)}$ denote the value of $\bar{\mathbf{z}}$ in the output of the j'th query to FFT^{-1}(ffSampling) in the game (i.e. at Line 9 of Figure 1 in Sec. 3.2 or Line 7 of Figure 4 in Appendix A). Suppose that, except for an event B_U, the absolute errors $\Delta_{t^{(i)}}$ in centers $t^{(i)}$ relative to $\sigma^{(i)}$ (i.e. $\Delta_{t^{(i)}}/\sigma^{(i)}$) are upper bounded by $\Delta'^U_{t^{(i)}}$ and relative errors $\delta_{\sigma^{(i)}}$ in standard deviations $\sigma^{(i)}$ are upper bounded by $\delta^U_{\sigma^{(i)}}$ for all $1 \leq i \leq M_{\mathbb{Z}}$, and the infinity-norm absolute errors $\Delta_{\bar{\mathbf{z}}^{(j)}}$ in $\bar{\mathbf{z}}^{(j)}$ is upper bounded by $\Delta^U_{\bar{\mathbf{z}}} < 1/2$ for all $1 \leq j \leq M_f$. The above errors are computed with respect to the same game with finite precision floating-point arithmetic. Here, $M_{\mathbb{Z}} \leq K \cdot (Q_E + (d+1) \cdot Q_D) + 2$ denotes the total number of queries to the 1-dim. $\mathbb{Z}$ Gaussian sampler in the

TABLE 1. LATTE SECURITY IMPACT OF FINITE PRECISION BASED ON EMPIRICAL ERROR ESTIMATION RESULTS FROM OUR STATISTICAL MODEL.

			$\ell = 1$					$\ell = 2$				
Set	$\mathbf{p}_{fp}$	$\mathbf{p}_{\mathcal{D}}$	$\ln C_K$	$\Delta^U_{\bar{\mathbf{z}}}$	$Q^C_{\max}$	Q^C_U	$Q^B_{\max}$	$\ln C_K$	$\Delta^U_{\bar{\mathbf{z}}}$	$Q^C_{\max}$	Q^C_U	$Q^B_{\max}$
LATTE-1	53	48	2^{-46}	2^{-23}	2^{46}	2^{39}	2^{74}	-	-	-	-	-
LATTE-2	53	48	2^{-42}	2^{-21}	2^{42}	2^{33}	2^{72}	-	-	-	-	-
LATTE-3	113	96	2^{-156}	2^{-71}	2^{156}	2^{149}	2^{171}	2^{-95}	2^{-35}	2^{95}	2^{88}	2^{75}
LATTE-4	113	96	2^{-149}	2^{-68}	2^{149}	2^{142}	2^{169}	2^{-85}	2^{-30}	2^{85}	2^{77}	2^{66}

game, K denotes the number of $\mathbb{Z}$ sampler calls of each call of Figure 7 in Appendix A, and $M_f \le Q_E + (d+1) \cdot Q_D$ denotes the number of calls of Figure 7 in the game.

Lemma 4.1. *Let $M_{\mathbb{Z}}$ be as above. Let also $\tau, \epsilon > 0$, $Q_M := \exp(\tau^2/2)/(2M_{\mathbb{Z}})$ and $\sigma^{(i)} \ge \eta_\varepsilon(\mathbb{Z})$ for $1 \le i \le M_{\mathbb{Z}}$. Let p_U denote the probability of event B_U in game* IDEAL. *Let E denote any event defined over the view of $\mathcal{A}$, $B_T := B^{M_{\mathbb{Z}}}$, $C_T := \prod_{i<M_{\mathbb{Z}}} C^{(i)}$, where $C^{(i)}$ is given by the right hand side of* (6) *in Lemma 4.2 with $\delta_\sigma := \delta^U_{\sigma^{(i)}}$, $\Delta'_t := \Delta'^U_{t^{(i)}}$ for $1 \le i \le M_{\mathbb{Z}}$. Then,*

$$\Pr[E_{\text{IDEAL}}] \ge \frac{1}{C_T} \cdot \left(\frac{\Pr[E_{\text{REAL}}]^{a/(a-1)}}{B_T} - \eta \right)^{a/(a-1)} - p_U,$$

where $\eta := C_T(p_U + 1/Q_M)^{(a-1)/a}$.

4.2. RD between $\mathbb{Z}$-Gaussians with errors

This step builds upon unpublished work by Prest [42]. We will consider the following Gaussians:

- D_1 is an ideal Gaussian of standard deviation σ and center t.
- D_2 is a Gaussian of standard deviation σ and center t, restricted to the interval $I = [t - \tau \cdot \sigma, t + \tau \cdot \sigma]$.
- D_3 is a Gaussian of standard deviation $\bar{\sigma}$ and center $\bar{t}$, restricted to the interval I.

Now, we present Lemma 4.2 showing for adequate values of $\tau, |\bar{t} - t|/\sigma, |\frac{\bar{\sigma}}{\sigma} - 1|$, D_1 and D_3 are close in the RD sense. The proof appears in Appendix C.

Lemma 4.2. *Consider D_1, D_2, D_3, τ as defined above. Suppose that there exist $\delta_\sigma, \varepsilon, \delta, Q > 0$ such that:*

1) $\max(\delta_\sigma, \varepsilon) \le \delta = o(1)$*;*
2) $|\bar{t} - t|/\sigma \le \Delta'_t$ *(bounded absolute error);*
3) $|\frac{\bar{\sigma}}{\sigma} - 1| = \delta_\sigma$ *(bounded relative error);*
4) $\sigma \ge \eta_\varepsilon(\mathbb{Z})$*;*
5) $Q \le \exp(\sigma^2\tau^2/2)(2\pi\sqrt{\sigma\tau}(1-\varepsilon)\sigma)$*;*
6) $\mathsf{num}(a, b, c) := |a^2 + 2a\frac{\sqrt{2\pi b}}{1-b} + (2c + c^2)(1 + \frac{2\pi b}{1-b})|/2(1 - c^2)$*;*
7) $\mathsf{ub} := 2/Q + \mathsf{num}(\Delta'_t, \varepsilon, \delta_\sigma) + \frac{1}{1-\delta_\sigma} \cdot (\tau\Delta'_t + \tau^2\delta_\sigma)$.

Then the RDs of D_3 and D_2 (resp. D_1) is:

$$R_a(D_3; D_2) \lesssim 1 + \frac{a \cdot \mathsf{ub}^2}{2}. \quad (5)$$

$$R_a(D_3; D_1) \lesssim 1 + \frac{1}{Q} + \frac{a \cdot \mathsf{ub}^2}{2}. \quad (6)$$

TABLE 2. EMPIRICAL RESULTS OF ACTUAL ARITHMETIC ERRORS.

		$\ell = 1$		$\ell = 2$	
Set	$\mathbf{p}_{fp}$	$\ln C_K$	$\Delta^U_{\bar{\mathbf{z}}}$	$\ln C_K$	$\Delta^U_{\bar{\mathbf{z}}}$
LATTE-1	53	2^{-46}	2^{-22}	-	-
LATTE-2	53	2^{-40}	2^{-20}	-	-
LATTE-3	113	2^{-155}	2^{-70}	2^{-94}	2^{-34}
LATTE-4	113	2^{-148}	2^{-67}	2^{-84}	2^{-29}

4.3. Statistical Model for ffSampling Precision

In this Section, we present a statistical model to estimate bounds for floating point arithmetic errors in LATTE ffSampling algorithm using our chosen implementation floating point precision for the LATTE parameter sets, and we use those bounds to analyse the security impact of those errors on our LATTE implementation by applying Lemma 4.1.

Our statistical model makes the heuristic but natural assumption that the floating point error introduced in each arithmetic operation in the ffSampling algorithm can be modelled as an independent zero-centered continuous Gaussian random variable, and the model estimates the maximum standard deviations $\delta_\sigma, \Delta'_t, \Delta_{\bar{\mathbf{z}}}$ of the errors $\delta_{\sigma^{(i)}}$, $\Delta'_{t^{(i)}}$, $\Delta_{\bar{\mathbf{z}}^{(j)}}$ over all $\mathbb{Z}$-sampler query indices $1 \le i \le M_{\mathbb{Z}}$ and ffSampler query indices $1 \le j \le M_f$ in the IDEAL game of Lemma 4.1 by propagating the standard deviations of the independent errors through the ffSampling algorithm arithmetic steps, assuming uniformly random input matrices $\mathbf{A}_i \in \mathcal{R}_q$ at the input to the Extract and Delegate algorithms. We explain at the end of this Section how we apply the standard deviations in the $\mathbb{Z}$ sampler queries to derive the security impact of floating point errors on LATTE. We remark that the use of the random oracles H and G to hash the attacker's choice of identities queried to Extract or Delegate algorithms to derive the ffSampling input ring elements $\mathbf{A}_i$ uniformly at random in $\mathcal{R}_q$ and seed for Extract uniformly random supports our statistical (rather than adversarial) model of floating point errors since the attacker cannot control the randomness of H and G and the Delegate, Extract and Key Generation algorithms. A similar heuristic statistical model is commonly used in the context of evaluating the propagation of LWE errors via a circuit computed homomorphically with Fully Homomorphic Encryption schemes [43].

We now present the details of our statistical model for estimating the standard deviations of errors, i.e. δ_σ, Δ'_t, and $\Delta_{\bar{\mathbf{z}}}$. For a complex number $a = \mu_R + i\mu_I$, with $\mu_R, \mu_I \in \mathbb{R}$, let us denote the absolute error of the real part μ_R as σ_R and the absolute error of the imaginary part

μ_I as σ_I, respectively. We assume the real and imaginary parts of *every* complex number in our statistical model are independent Gaussian variables, e.g. for complex number $a = \mu_R + i\mu_I$, $\mathrm{Re}(a)$ follows the normal Gaussian distribution $\mathcal{N}(\mu_R, \sigma_R^2)$ and $\mathrm{Im}(a)$ follows the normal distribution $\mathcal{N}(\mu_I, \sigma_I^2)$, respectively, no matter whether $\mathrm{Re}(a)$, $\mathrm{Im}(a)$ are linear combinations of one or more independent normal variables. Therefore, we use the tuple $(\mu_R, \sigma_R^2, \mu_I, \sigma_I^2)$ to represent a complex number with errors.

Definition 4.1 (AddB, SubB, and MultB). *For independent* $a = (\mu_{a,R}, \sigma_{a,R}^2, \mu_{a,I}, \sigma_{a,I}^2)$ *and* $b = (\mu_{b,R}, \sigma_{b,R}^2, \mu_{b,I}, \sigma_{b,I}^2)$*, let us define* $\mathrm{AddB}(a,b)$ *as* $(\mu_{a,R} + \mu_{b,R}, \sigma_{a,R}^2 + \sigma_{b,R}^2, \mu_{a,I} + \mu_{b,I}, \sigma_{a,I}^2 + \sigma_{b,I}^2)$, $\mathrm{SubB}(a,b)$ *as* $(\mu_{a,R} - \mu_{b,R}, \sigma_{a,R}^2 + \sigma_{b,R}^2, \mu_{a,I} - \mu_{b,I}, \sigma_{a,I}^2 + \sigma_{b,I}^2)$*, and* $\mathrm{MultB}(a,b)$ *as:* $(\mu_{a,R}\mu_{b,R} - \mu_{a,I}\mu_{b,I}, \mu_{a,R}^2\sigma_{b,R}^2 + \mu_{b,R}^2\sigma_{a,R}^2 + \sigma_{a,R}^2\sigma_{b,R}^2 + \mu_{a,I}^2\sigma_{b,I}^2 + \mu_{b,I}^2\sigma_{a,I}^2 + \sigma_{a,I}^2\sigma_{b,I}^2, \mu_{a,R}\mu_{b,I} + \mu_{a,I}\mu_{b,R}, \mu_{a,R}^2\sigma_{b,I}^2 + \mu_{b,I}^2\sigma_{a,R}^2 + \sigma_{a,R}^2\sigma_{b,I}^2 + \mu_{a,I}^2\sigma_{b,R}^2 + \mu_{b,R}^2\sigma_{a,I}^2 + \sigma_{a,I}^2\sigma_{b,R}^2)$.

Definition 4.2 (DivB [44]). *Let* $a = (\mu_{a,R}, \sigma_{a,R}^2, \mu_{a,I}, \sigma_{a,I}^2)$ *and* $b = (\mu_{b,R}, \sigma_{b,R}^2, 0, 0)$*. Assuming* $\mathrm{Re}(a)$, $\mathrm{Im}(a)$*, and* b *are independent normal variables such that* $\sqrt{\sigma_{a,R}^2/\mu_{a,R}^2 + \sigma_{b,R}^2/\mu_{b,R}^2} < 1$ *and* $\sqrt{\sigma_{a,I}^2/\mu_{a,I}^2 + \sigma_{b,R}^2/\mu_{b,R}^2} < 1$*, we define* $\mathrm{DivB}(a,b)$ *as:* $\left(\frac{\mu_{a,R}}{\mu_{b,R}}, \frac{\mu_{a,R}^2}{\mu_{b,R}^2}\left(\frac{\sigma_{a,R}^2}{\mu_{a,R}^2} + \frac{\sigma_{b,R}^2}{\mu_{b,R}^2}\right), \frac{\mu_{a,I}}{\mu_{b,R}}, \frac{\mu_{a,I}^2}{\mu_{b,R}^2}\left(\frac{\sigma_{a,I}^2}{\mu_{a,I}^2} + \frac{\sigma_{b,R}^2}{\mu_{b,R}^2}\right)\right)$.

Definition 4.3 (AbsSqrB). *For* $a = (\mu_{a,R}, \sigma_{a,R}^2, \mu_{a,I}, \sigma_{a,I}^2)$*, assuming* $\mathrm{Re}(a)$ *and* $\mathrm{Im}(a)$ *are independent normal variables, let us define* $\mathrm{AbsSqrB}(a)$ *as* $\mathrm{AddB}((\mathrm{Re}(a))^2, (\mathrm{Im}(a))^2)$.

We can use the above absolute arithmetic error bound approximations to rewrite our optimised ffLDL in Figure 2 in Sec. 5.2 and ffSampling in Figure 7 in Appendix A, in order to estimate δ_σ and Δ_t', respectively. For δ_σ, we first use the ffLDLB algorithm in Appendix J to estimate the absolute errors of the leaf values (real numbers) in ffLDL tree T for a given Gram matrix $\mathbf{G}$, i.e. the standard deviation $\sigma_{\mathrm{leaf},R}$. Since σ for the 1-D integer Gaussian sampler is computed by $\sigma_\ell/\sqrt{\mu_{\mathrm{leaf},R}}$ during tree normalisation, assuming the relative error of the floating-point arithmetic is u, we have the following arithmetic error bound:

$$\delta_\sigma \le \max_{\text{all leaves}} \left[\frac{(1+u)\frac{(1+u)\sigma_\ell}{(1-u)\sqrt{\mu_{\mathrm{leaf},R} - \sigma_{\mathrm{leaf},R}}}}{\frac{\sigma_\ell}{\sqrt{\mu_{\mathrm{leaf},R}}}} - 1\right]$$
$$= \frac{(1+u)^2}{1-u}\sqrt{\max_{\text{all leaves}} \frac{\mu_{\mathrm{leaf},R}}{\mu_{\mathrm{leaf},R} - \sigma_{\mathrm{leaf},R}}} - 1.$$

Similarly, we can use ffSamplingB algorithm in Appendix J to output Δ_t' for a given vector $\mathbf{t}$ and ffLDL tree T. In addition, we can compute the rounding errors $\Delta_{\bar{\mathbf{z}}}$ i.e. Line 10 in Figure 1 in Sec. 3.2 and Line 8 in Figure 4 in Appendix A, by combining the FFT/FFT^{-1} errors of the input and the errors $\sigma_{\mathbf{z},R}$, $\sigma_{\mathbf{z},I}$ of $\mathbf{z}$ computed by ffSamplingB. We also use a similar statistical modelling approach to estimate the errors of FFT/FFT^{-1} for a given vector $\mathbf{a}$.

Let $C_K := \prod_{i<K} C^{(i)}$, where $K, C^{(i)}$ are defined in Lemma 4.1. Here we show the empirical results of $\ln C_K$ and the error $\Delta_{\bar{\mathbf{z}}}$ estimated by our statistical model. For the target floating-point precisions used by our implementation of the LATTE scheme (see Sec. 6 for the rationale behind the chosen precision), we compute the errors for 100 random $(\mathbf{S}, \mathbf{t})$ pairs, where $\mathbf{S}$ is the basis and $\mathbf{t}$ is the input of the ffSampling in Figure 7 in Appendix A. The $\ln C_K$ and $\Delta_{\bar{\mathbf{z}}}^U$ among these 100 iterations are shown in Table 1. To provide empirical evidence for supporting the accuracy of our statistical model, for the same 100 pairs of $(\mathbf{S}, \mathbf{t})$, we also give the actual arithmetic errors between the values computed by using a very high precision (1024 bits) and the values computed by using the target precisions. The actual arithmetic errors computed by this approach are shown in Table 2. By comparing the results in Table 1 and 2, the difference between the actual arithmetic errors and the estimated errors from our statistical model is at most 2 bits in this empirical experiment. We will leave modelling the distributions of $(\mathbf{S}, \mathbf{t})$ to make our statistical model fully deterministic as future works.

Security Impact of Finite Precision Errors: In order to use the results in Table 1 with Lemma 4.1 to derive the security impact of floating point errors, we first derive corresponding upper bounds $\delta_{\sigma^{(i)}}^U := \tau_U \cdot \delta_{\sigma^{(i)}}$, $\Delta'^U_{t^{(i)}} := \tau_U \cdot \Delta'_{t^{(i)}}$, and $\Delta_{\bar{\mathbf{z}}} := \tau_U \cdot \Delta_{\bar{\mathbf{z}}}^U$ on the absolute value of the errors, where τ_U is chosen so that each individual Gaussian error's absolute value exceeds its bound with probability $\le 2^{-\lambda}$, which by the standard Gaussian tail bound is satisfied by setting $2\exp(-\tau^2/2) \le 2^{-\lambda}$. Therefore by a union bound, all bounds hold except with a negligible probability $p_U \le (2M_{\mathbb{Z}} + M_f)2^{-\lambda}$, with λ denoting the target security level. Applying Lemma 4.1 with $a := 2\lambda$ we conclude using $a/(a-1) \approx 1$ and p_U is negligible, that $\Pr[E]_{\mathrm{IDEAL}} \approx \frac{1}{B_T C_T}\Pr[E]_{\mathrm{REAL}}$ so that finite precision causes a bit security loss $L \approx \log_2(B_T) + \log_2(C_T)$ bits. We use the above floating point arithmetic upper bounds to compute an estimate for the maximum number of the delegate/extract queries $Q_{\max}^C$ (resp. $Q_{\max}^B$) that ensures $\log_2(C_T) \le 1$ (resp. $\log_2(B_T) \le 1$) so that if $\max(Q_D, Q_E) \le \min(Q_{\max}^C, Q_{\max}^B)$, then $L \le 2$ bits of security are lost overall for our finite arithmetic precision p_{fp} LATTE implementation versus the infinite precision implementation. To compute $B_T \le B^{M_{\mathbb{Z}}}$, we use the RD bound B on the COSAC $\mathbb{Z}$ sampler RD from the ideal $\mathbb{Z}$ sampler distribution derived in [21] corresponding to the COSAC sampler precision $p_{\mathcal{D}}$ used in our COSAC implementation (see Sec. 5.4 for the discussions). The finite precision security impact results are summarised in Table 1. Note that in Table 1, $Q_{\max}^C$ is computed using the max. error values $\delta_{\sigma^{(i)}}$ and $\Delta'^U_{t^{(i)}}$ estimated by our statistical model over 100 runs with random $(\mathbf{S}, \mathbf{t})$ input pairs, whereas Q_U^C is a more conservative estimate using tail bounds $\delta_{\sigma^{(i)}}^U := \tau_U \cdot \delta_{\sigma^{(i)}}$ and $\Delta'^U_{t^{(i)}} := \tau_U \cdot \Delta'_{t^{(i)}}$ on the statistical model error estimates. We conjecture the former estimates are more accurate, although our existing security proof in Lemma 4.1 only implies the latter estimates.

The results show that for LATTE-1 and LATTE-2, 2^{42} Extract and/or Delegate queries can be supported with at most 2 bits of security loss with our 53-bit double-precision floating-point precision implementation. This should suffice for most practical applications. For LATTE-3 and LATTE-4, the main bottleneck in precision is the $\Delta_{\bar{z}}$ bound, but the results indicate that even reducing the precision by about 25 bits from our chosen 113-bit arithmetic precision to ≈ 90 bits precision would suffice for security.

Similarly, we apply our statistical model on the Python implementation of FALCON[3] and compute the numerical results of $Q^C_{\max}$ based on the empirical results of $\ln C_K$. We use the `Decimal` fixed-point number type in Python with 1000 decimal digits and perform the experiment by using the same seeds as in the Known Answer Tests (KATs). We get $Q^C_{\max} = 2^{60}$ for FALCON-512 and $Q^C_{\max} = 2^{56}$ for FALCON-1024, respectively. We also compute $Q^C_{\max}$ by using the upper bounds of δ_σ, Δ'_t for the whole ffLDL tree from our statistical model (similar to the approach in the FALCON specification [15], which uses the empirical error upper bounds of δ_σ, Δ_t for the whole ffLDL tree)[4], and get $Q^C_{\max} = 2^{57}$ for FALCON-512 and $Q^C_{\max} = 2^{53}$ for FALCON-1024, respectively. The results indicate that our error analysis by using the error upper bounds for each leaf instead of the whole ffLDL tree has a 3-bit improvement over the approach in [15]. Although our results are lower than the claimed number of queries $Q_s = 2^{64}$ in the FALCON specification [15], the empirical error upper bound $\delta_\sigma + \Delta_t \le 2^{-40}$ given in the specification does not satisfy the required upper bound $\delta_\sigma + \Delta_t \le 2^{-46}$ from the authors' analysis for 2^{64} queries, and the specification did not discuss the impact of such larger errors on Q_s in detail. In addition, polynomials are converted between the original domain and the FFT domain in every polynomial arithmetic operation when computing the Gram matrix $\mathbf{G}$ in the FALCON Python implementation, instead of only doing the conversions at the beginning/end of the $\mathbf{G}$ computation. These redundant FFTs will also increase the errors of $\mathbf{G}$ and thus increase the errors in the ffLDL tree.

4.4. ID-OW-CPA security of Improved LATTE

Recall from Sec. 3 that our improved LATTE scheme achieves improved efficiency and shorter decryption keys output by the Extract algorithm relative to the original LATTE scheme. This change to Extract necessitates a different strategy for simulating the Extract oracle at level ℓ in the ID-OW-CPA security proof (step 3 in the overview of Sec. 4.1), compared with the strategy outlined in [13] based on Theorem 5.2 in [41]. In particular, the Extract oracle simulation at level ℓ must simulate the decryption key $\mathbf{t}$ at that level without knowing the delegation secret key $\mathbf{S}_{\ell-1}$ at level $\ell - 1$. In the original LATTE using (2), this can be done by programming $A_\ell = H(\mathsf{ID}_1|\dots|\mathsf{ID}_\ell)$ to be a matrix with an embedded NTRU trapdoor and using the basis extension method used in the Delegate oracle and its simulation at level ℓ. But with our improved LATTE Extraction using (3), we cannot use a trapdoor for $\mathbf{A}'_\ell$ to simulate multiple such decryption key vectors $\mathbf{t}$; indeed, if this were possible then subtracting two such distinct short vectors would reveal a short vector $\mathbf{s}$ in the (secret) level $\ell - 1$ delegation module lattice $\mathbf{s} : \mathbf{s}_0 + \mathbf{s}_1\mathbf{h} + \dots + \mathbf{t}_\ell \mathbf{A}_{\ell-1} = \mathbf{0}$.

Instead, our Extract simulator generates a *single* such short vector $\mathbf{t}$ using the GPV signature simulation strategy [18], i.e. programming $\mathbf{A}'_\ell = H_E(\mathsf{ID}_1|\dots|\mathsf{ID}_\ell) := \mathbf{t}_0 + \mathbf{t}_1\mathbf{h} + \dots + \mathbf{t}_\ell\mathbf{A}_{\ell-1}$ for short discrete Gaussian $\mathbf{t}_i$'s sampled by the Extract simulator. To avoid a contradiction with the different programming strategy for $\mathbf{A}_\ell$, our modified LATTE uses a different hash function H_E modelled as a random oracle (obtained from the random oracle H by using the prefix `"E"`) for computing $\mathbf{A}'_\ell$ used in Extract so that H and H_E can be programmed independently. Also, since our programming strategy for $\mathbf{A}'_\ell = H_E(\mathsf{ID}_1|\dots|\mathsf{ID}_\ell)$ only works for a *single* decryption key $\mathbf{t}$, we must make Extract deterministic so that it returns the same secret key $\mathbf{t}$ again if queried again at the same $\mathsf{ID}_1|\dots|\mathsf{ID}_\ell$; this is the purpose of the hash function G used to derive the randomness seed for Extract deterministically from $\mathsf{ID}_1|\dots|\mathsf{ID}_\ell$.

5. Implementation Techniques

We now discuss the implementation techniques used in our optimised LATTE scheme. First, we summarise the difference of parameters compared to [13] and discuss the security impact in Sec. 5.1. Then, we present our faster novel ffLDL variant for (Mod)NTRU basis in Sec. 5.2. Then, we discuss the techniques adapted from FALCON [15] and MODFALCON [17] in Sec. 5.3. Finally, we discuss the integer discrete Gaussian sampling techniques in Sec. 5.4, including the adaption of FACCT [19] and COSAC [20], [21] samplers in our LATTE implementation.

5.1. Summary of Differences to ETSI Report

There are three main differences: (1) We find that the discrete Gaussian statistical parameter $\varepsilon = 2^{-22.5}/(\ell+1)N$ used by σ_ℓ in [13] was miscalculated. The KL-divergence between the sampled distribution and the ideal discrete Gaussian distribution is bounded by approximately $8((\ell + 1)N)^2\varepsilon^2$. Choosing $\varepsilon = 2^{-25.5}/(\ell+1)N$ ensures the divergence is at most 2^{-48}, as specified by the proposed LATTE specification [13]. If the sampled distribution has a KL-divergence of 2^{-48} from the ideal distribution, then using the sampler at most 2^{47} times will only reduce the security of the scheme by up to one-bit [45]. However, in [13], the $\varepsilon = 2^{-22.5}/(\ell+1)N$ would only ensure the KL-divergence is at most 2^{-42}. (2) To accommodate the use of the FACCT sampler in KeyGen, as described in Sec. 5, we modify the value of σ_0, as displayed in Table 3. This also has an effect on the subsequent σ_ℓ, and therefore the difficulty of the

[3]https://github.com/tprest/falcon.py

[4]The computed statistical model results are close to the empirical errors, see Table 1 and 2.

TABLE 3. LATTE σ_ℓ AND DECRYPTION FAIL. PROB.

Set	σ_ℓ			**Fail. Prob.**	
	$\ell = 0$	$\ell = 1$	$\ell = 2$	$\ell = 1$	$\ell = 2$
LATTE-1	106.2	5513.3	-	2^{-191}	-
LATTE-2	106.2	7900.2	-	2^{-380}	-
LATTE-3	6777.6	351968.4	22559988.0	$2^{-\text{inf}}$	2^{-126}
LATTE-4	9583.7	713167.	64997288.2	$2^{-\text{inf}}$	2^{-246}

underlying lattice problems and success of each attack. (3) As our redesign of LATTE discards the polynomial $\mathbf{b}$ in the master public key and reduces the module dimension of the user private key, as described in Section 3.1, we update the attack costings accordingly in Appendix I. First, it reduces the decryption failure rate, as there is one less error term. The best user key recovery attack reduces to CVP in the master lattice, so the attack is on the same lattice, but it demands a marginally shorter vector to be successful.

5.2. Improved ffLDL Algorithm for NTRU Basis

We observe the following theorem, which can be adapted to accelerate the computation of the ffLDL algorithm from FALCON [15], for the Fast Fourier $\mathbf{LDL}^*$ decomposition of the (Mod)NTRU basis $\mathbf{S}_\ell$ in LATTE:

Theorem 5.1. *Let $\mathbf{S}_\ell$ be a (Mod)NTRU basis. In ffLDL tree of the matrix $\mathbf{G} = \mathbf{S}_\ell \mathbf{S}_\ell^* \in (\mathbb{C}[x]/\langle x^N + 1\rangle)^{d\times d}$ in FFT domain, we get:*

1) *$\mathbf{D}_{i,i} \in \mathbb{R}^n$ for some $n = 2^k \le N$ in every node of the tree, $0 \le i \le d-1$.*
2) *$\prod_{i=0}^{d-1} (\mathbf{D}_{i,i})_j = q^2$ in the root of tree, $0 \le j \le N-1$.*
3) *$(\mathbf{D}_{0,0})_j (\mathbf{D}_{1,1})_j = \mathbf{D}'_{2j}\mathbf{D}'_{2j+1}$ for some $n = 2^k \le N/2$ in every non-root node of the tree, where $\mathbf{D}' \in \{\mathbf{D}_{i,i}\}_{i=0}^{d-1}$ is from its parent, $0 \le j \le n-1$.*
4) *$(\mathbf{D}_{i,i})_j \in \mathbb{R}^+$ for some $n = 2^k \le N$ in every node of the tree, $0 \le i \le d-1$ and $0 \le j \le n-1$.*

Proof: 1) From the original ffLDL algorithm in FALCON [15], we have $(\mathbf{D}_{0,0})_j = (\mathbf{G}_{0,0})_j$ and $(\mathbf{D}_{1,1})_j = (\mathbf{G}_{1,1})_j - |(\mathbf{L}_{1,0})_j|^2 (\mathbf{G}_{0,0})_j$, $0 \le j \le n-1$, for some input matrix $\mathbf{G}$ in the FFT domain in every node of the tree. In addition, we have $(\mathbf{D}_{i,i})_j = (\mathbf{G}_{i,i})_j - \sum_{k<i}(|(\mathbf{L}_{i,k})_j|^2(\mathbf{D}_{k,k})_j)$ at the root when $d > 2$. Therefore, we have $\mathbf{D}_{i,i} \in \mathbb{R}^n$ assuming that $\mathbf{G}_{i,i} \in \mathbb{R}^n$ for all $i \in \{0, \dots, d-1\}$. To show that latter assumption is true, we observe that at the root we have the input $\mathbf{G} = \mathbf{S}_\ell\mathbf{S}_\ell^*$ in the FFT domain, $\mathbf{G}_{i,i} \in \mathbb{R}^N$ for $0 \le i \le d-1$. Thus, $\mathbf{D}_{i,i} \in \mathbb{R}^N$ for $0 \le i \le d-1$ at the root. Assuming $\mathbf{D}_{i,i} \in \mathbb{R}^n$ for $0 \le i \le d-1$ at an non-leaf node, for its i-th child, we have the ffLDL input $\mathbf{G}'_{0,0} = \mathbf{G}'_{1,1} = \mathbf{d}_0$, where $(\mathbf{d}_0)_j = \frac{1}{2}[(\mathbf{D}_{i,i})_{2j} + (\mathbf{D}_{i,i})_{2j+1}] \in \mathbb{R}$ for $0 \le j \le n/2-1$. Thus, $\mathbf{D}'_{0,0}, \mathbf{D}'_{1,1} \in \mathbb{R}^{n/2}$ in this child and we can deduce the conclusion by induction.

2) Since by definition of the $\mathbf{LDL}^*$ decomposition [46], $\mathbf{L}$ is a lower triangular matrix with 1 on its diagonal and $\mathbf{D}$ is a diagonal matrix, we have $\det(\mathbf{D}) = \prod_{i=0}^{d-1} \mathbf{D}_{i,i} = \det(\mathbf{G})$. Because $\mathbf{G} = \mathbf{S}_\ell\mathbf{S}_\ell^*$ at the root and the determinant of a (Mod)NTRU basis $\mathbf{S}_\ell$ is q, we have $\prod_{i=0}^{d-1}(\mathbf{D}_{i,i})_j = q^2$ in the FFT domain at the root for $0 \le j \le N-1$.

```
Input: Gram matrix $\mathbf{G} \in (\mathbb{C}[x]/\langle x^n + 1\rangle)^{d\times d}$ in the
    FFT domain. $d \in \{2,3\}$. $\mathbf{D}' \in (\mathbb{R}^+)^{2n}$.
Output: Tree $T$.
function ffLDL($\mathbf{G}, \mathbf{D}'$)
    if $n = 1$ then
        $T$.value $\leftarrow \mathbf{G}_{0,0}$.
    else
        $\mathbf{L} \leftarrow \mathbf{I}_d, \mathbf{D} \leftarrow \mathbf{0}_d$.
        for $j = 0$ to $n-1$ do
            $(\mathbf{D}_{0,0})_j \leftarrow (\mathbf{G}_{0,0})_j$.
            $(\mathbf{L}_{1,0})_j \leftarrow \frac{(\mathbf{G}_{1,0})_j}{(\mathbf{D}_{0,0})_j}$.
            if $d = 2$ then
                if $n = N$ then
                    $(\mathbf{D}_{1,1})_j \leftarrow \frac{q^2}{(\mathbf{D}_{0,0})_j}$.
                else
                    $(\mathbf{D}_{1,1})_j \leftarrow \frac{\mathbf{D}'_{2j}\mathbf{D}'_{2j+1}}{(\mathbf{D}_{0,0})_j}$.
                end if
            else if $d = 3$ then
                $(\mathbf{D}_{1,1})_j \leftarrow (\mathbf{G}_{1,1})_j - \frac{|(\mathbf{G}_{1,0})_j|^2}{(\mathbf{D}_{0,0})_j}$.
                $(\mathbf{D}_{2,2})_j \leftarrow \frac{q^2}{(\mathbf{D}_{0,0})_j(\mathbf{D}_{1,1})_j}$.
                $(\mathbf{L}_{2,0})_j \leftarrow \frac{(\mathbf{G}_{2,0})_j}{(\mathbf{D}_{0,0})_j}$.
                $(\mathbf{L}_{2,1})_j \leftarrow \frac{(\mathbf{G}_{2,1})_j - (\mathbf{G}_{2,0})_j(\mathbf{L}_{1,0})_j^*}{(\mathbf{D}_{1,1})_j}$.
            end if
        end for
        $T$.value $\leftarrow \mathbf{L}$.
        for $i = 0$ to $d-1$ do
            $\mathbf{d}_0, \mathbf{d}_1 \leftarrow$ splitfft$(\mathbf{D}_{i,i})$.
            $\mathbf{G}' = \begin{pmatrix}\mathbf{d}_0 & \mathbf{d}_1\\ \mathbf{d}_1^* & \mathbf{d}_0\end{pmatrix}$.
            $T$.child$_i \leftarrow$ ffLDL$(\mathbf{G}', \mathbf{D}_{i,i})$.
        end for
    end if
    return $T$.
end function
```

Figure 2. Optimised ffLDL algorithm for (Mod)NTRU basis in LATTE.

3) For the i-th child of an non-leaf node, we have the ffLDL input $\mathbf{G}' = \begin{pmatrix}\mathbf{d}_0 & \mathbf{d}_1\\ \mathbf{d}_1^* & \mathbf{d}_0\end{pmatrix}$ for $\mathbf{d}_0, \mathbf{d}_1 \leftarrow \text{splitfft}(\mathbf{D}_{i,i})$, $0 \le i \le d-1$ (see [15] for the definition of splitfft). By the definition of the $\mathbf{LDL}^*$ decomposition, for this child, we have $\mathbf{D}'_{0,0}\mathbf{D}'_{1,1} = \det(\mathbf{G}') = \mathbf{d}_0^2 - \mathbf{d}_1\mathbf{d}_1^*$. Thus, in the FFT domain, we have: $(\mathbf{D}'_{0,0})_j(\mathbf{D}'_{1,1})_j = (\mathbf{d}_0)_j^2 - |(\mathbf{d}_1)_j|^2 = (1/2\cdot[(\mathbf{D}_{i,i})_{2j} + (\mathbf{D}_{i,i})_{2j+1}])^2 - |1/2\cdot[(\mathbf{D}_{i,i})_{2j} - (\mathbf{D}_{i,i})_{2j+1}]\omega^{-\text{bitrev}(n/2+j)}|^2$, for $0 \le j \le n/2-1$. Since $(\mathbf{D}_{i,i})_{2j}, (\mathbf{D}_{i,i})_{2j+1} \in \mathbb{R}$ and $|\omega| = 1$, we get $(\mathbf{D}'_{0,0})_j(\mathbf{D}'_{1,1})_j = (\mathbf{D}_{i,i})_{2j}(\mathbf{D}_{i,i})_{2j+1}$.

4) The ffLDL algorithm computes the $\mathbf{LDL}^*$ decomposition in the FFT domain. Let $\mathbf{S}_\ell = \mathbf{L}\cdot\tilde{\mathbf{S}}_\ell$ be the GSO decomposition of $\mathbf{S}_\ell \in \mathcal{R}^{d\times d}$ where rows of $\tilde{\mathbf{S}}_\ell$ are pairwise

TABLE 4. SUMMARY OF PERFORMANCE RESULTS (OP/S) AT 4.2GHZ.

Set	Sec.	N	$\log_2 q$	KeyGen	$\ell = 1$				$\ell = 2$		
					Ext	Enc	Dec	Del	Ext	Enc	Dec
Orig. LATTE-1 [13]	128	1024	24	-	-	2911	2987	-	-	-	-
Our LATTE-1				9.4	1361.8	23061.4	18041.3	-	-	-	-
Orig. LATTE-2 [13]	256	2048	25	-	-	1335	1351	-	-	-	-
Our LATTE-2				3.3	636.9	10690.7	8456.4	-	-	-	-
Orig. LATTE-3 [13]	80	1024	36	-	-	1892	1774	-	-	1455	1474
Our LATTE-3				5.7	36.3	14331.1	12134.7	2.4	20.0	11429.8	9713.4
Orig. LATTE-4 [13]	160	2048	38	-	-	709	668	-	-	568	541
Our LATTE-4				1.7	17.1	6846.6	5785.6	0.8	9.4	5450.2	4642.1
DLP-0 [13]	80	512	22	14.7	873.2	8731.8	6202.9	-	-	-	-
DLP-3 [13]	192	1024	22	4.9	454.1	2639.8	1621.6	-	-	-	-
[47]	40	512	50	717.9	711.6	3589.7	3152.0	-	-	-	-
	80	1024	51	336.9	401.8	1615.4	1442.3	-	-	-	-
	195	2048	62	164.9	197.2	662.0	477.9	-	-	-	-
[48]	96	1024	30	225.1	133.6	453.9	377.2	-	-	-	-
	126	1280	30	49.2	122.2	403.4	339.0	-	-	-	-

orthogonal. For the input $\mathbf{G} = \mathbf{S}_\ell \mathbf{S}_\ell^*$ at the root, we have $\mathbf{G} = \mathbf{LDL}^*$ where $\mathbf{D} = \tilde{\mathbf{S}}_\ell \tilde{\mathbf{S}}_\ell^*$ [46]. Thus, in the FFT domain, $\mathbf{D}_{i,i} \in (\mathbb{R}^+)^N$ at the root. Assuming $\mathbf{D}_{i,i} \in (\mathbb{R}^+)^n$ for some $i \in \{0, \ldots, d-1\}$ at an non-leaf node, for the i-th child of this node, we have $(\mathbf{D}'_{0,0})_j (\mathbf{D}'_{1,1})_j = (\mathbf{D}_{i,i})_{2j} (\mathbf{D}_{i,i})_{2j+1} \in \mathbb{R}^+$ for $0 \le j \le n/2 - 1$. Because $(\mathbf{D}'_{0,0})_j = (\mathbf{d}_0)_j = \frac{1}{2}[(\mathbf{D}_{i,i})_{2j} + (\mathbf{D}_{i,i})_{2j+1}] \in \mathbb{R}^+$ due to the ffLDL input $\mathbf{G}' = \begin{pmatrix} \mathbf{d}_0 & \mathbf{d}_1 \\ \mathbf{d}_1^* & \mathbf{d}_0 \end{pmatrix}$ where $\mathbf{d}_0, \mathbf{d}_1 \leftarrow$ splitfft$(\mathbf{D}_{i,i})$, we get $\mathbf{D}'_{0,0}, \mathbf{D}'_{1,1} \in (\mathbb{R}^+)^{n/2}$. Thus, we deduce the conclusion by induction. □

We can utilise Theorem 5.1 when computing $\mathbf{D}$ in the ffLDL algorithm, see Figure 2, for the (Mod)NTRU basis $\mathbf{S}_\ell$ in LATTE with $d \in \{2, 3\}$: $\mathbf{D}_{d-1,d-1}$ at the root can be computed by $(\mathbf{D}_{d-1,d-1})_j = q^2 / \prod_{i=0}^{d-2} (\mathbf{D}_{i,i})_j$ for $0 \le j \le N-1$. For all the non-root nodes, we can directly compute $\mathbf{D}_{0,0}, \mathbf{D}_{1,1}$ by using $(\mathbf{D}_{0,0})_j = (\mathbf{G}_{0,0})_j$ and $(\mathbf{D}_{1,1})_j = \mathbf{D}'_{2j} \mathbf{D}'_{2j+1} / (\mathbf{D}_{0,0})_j$, $0 \le j \le n-1$, for some $\mathbf{D}' \in \mathbb{R}^{2n}$, $\mathbf{G}_{0,0} = \mathbf{d}'_0 \in \mathbb{R}^n$ from its parent. Since for all $0 \le i \le d-1$, we have $\mathbf{D}_{i,i} \in \mathbb{R}^n$ in every node of the tree, $\mathbf{D}$ can be computed solely by using the real number arithmetic, i.e. without complex number arithmetic. Because every complex number arithmetic computation contains multiple underlying floating-point arithmetic operations, by replacing complex number arithmetic with real number arithmetic when computing $\mathbf{D}$, we reduce the total amount of floating-point arithmetic operations. Therefore, this optimisation technique will accelerate the run-time speed of the ffLDL algorithm.

5.3. Techniques from FALCON and MODFALCON

Our LATTE utilises techniques from the signature scheme FALCON [15]. The two schemes are closely related; they are instantiated over the same type of lattice and share key generation and sampling procedures. FALCON makes use of the "tower of rings" structure to find a solution to the NTRU equation $\mathbf{fG} - \mathbf{gF} = q \mod x^N + 1$, for a given $\mathbf{f}$ and $\mathbf{g}$ in the NTRUSolve sub-algorithm of KeyGen, and in the lattice Gaussian sampling (ffSampling) component of LATTE Delegate and Extract. The tower of rings approach utilises the fact that computations over polynomials $\mathbf{f}, \mathbf{g} \in \mathbb{C}[x]/\langle x^{N/2} + 1 \rangle$ are equivalent to computations over $\mathbf{f}(x^2), \mathbf{g}(x^2) \in \mathbb{C}[x]/\langle x^N + 1 \rangle$. When $N = 2^k$, for some $k \in \mathbb{Z}$, this can be applied repeatedly so that computations are performed over polynomials of degree 1. This is advantageous in terms of both memory usage, and speed [16].

Furthermore, in LATTE Delegate, to complete the delegated basis $\mathbf{S}_\ell$ for lattice dimension larger than $2N$, we adapt techniques from MODFALCON [17]. Let $\mathbf{S}_\ell = \begin{pmatrix} \mathbf{v}^\mathsf{T} & \mathbf{M} \\ \mathbf{G}_\ell & \mathbf{F}'_\ell \end{pmatrix}$ be the delegated basis, where $\mathbf{G}_\ell = \mathbf{s}_{\ell+1,0}$, $\mathbf{F}'_\ell = (\mathbf{s}_{\ell+1,1}, \ldots, \mathbf{s}_{\ell+1,\ell+1})$, $\mathbf{v} = (\mathbf{s}_{0,0}, \mathbf{s}_{1,0}, \ldots, \mathbf{s}_{\ell,0})$, and $\mathbf{M} = (\mathbf{s}_{i,j})$ for $0 \le i \le \ell$ and $1 \le j \le \ell + 1$. By Schur complement, if $\mathbf{M}$ is invertible, we have: $\det(\mathbf{S}_\ell) = \det(\mathbf{G}_\ell - \mathbf{F}'_\ell \mathbf{M}^{-1} \mathbf{v}^\mathsf{T}) \det(\mathbf{M}) = (\mathbf{G}_\ell - \mathbf{F}'_\ell \mathbf{M}^{-1} \mathbf{v}^\mathsf{T}) \det(\mathbf{M}) = \mathbf{G}_\ell \det(\mathbf{M}) - \mathbf{F}'_\ell \mathrm{adj}(\mathbf{M}) \mathbf{v}^\mathsf{T}$. Since one can choose $(\mathbf{G}_\ell, \mathbf{F}'_\ell)$ such that $\det(\mathbf{S}_\ell) = q$ when filling the bottom row of $\mathbf{S}_\ell$, we assume $\mathbf{F}'_\ell$ have the form $(\mathbf{F}_\ell, \mathbf{0}, \ldots, \mathbf{0})$. We have $\det(\mathbf{S}_\ell) = \det(\mathbf{M}) \cdot \mathbf{G}_\ell - \mathbf{F}_\ell \cdot \mathbf{u}_0$ where $\mathbf{u}_0$ is the first coordinate of $\mathbf{u} = \mathrm{adj}(\mathbf{M}) \cdot \mathbf{v}^\mathsf{T}$. In order to fill the bottom row $(\mathbf{s}_{\ell+1,0}, \ldots, \mathbf{s}_{\ell+1,\ell+1})$ of $\mathbf{S}_\ell$, if $\mathbf{M}$ is invertible, we can use the same NTRUSolve algorithm as in LATTE KeyGen to find $\mathbf{F}_\ell, \mathbf{G}_\ell$ such that $\det(\mathbf{M}) \cdot \mathbf{G}_\ell - \mathbf{F}_\ell \cdot \mathbf{u}_0 = q$, and resample when $\det(\mathbf{M}) = 0$.

However, since the NTRUSolve algorithm [16] performs the length reduction based on the size of the coefficients in the input, the coefficient size of $\mathbf{F}_\ell, \mathbf{G}_\ell$ will be approximately the same as $\det(\mathbf{M}), \mathbf{u}_0$. Since $\mathbf{M}$ is an $(\ell+1) \times (\ell+1)$ sub-matrix of $\mathbf{S}_\ell$ with coordinate sizes being in the order of q among each element, the size of coefficients of $\det(\mathbf{M})$, $\mathbf{u}_0$, $\mathbf{F}_\ell$, and $\mathbf{G}_\ell$ is in the order of $q^{\ell+1}$. To make the infinity norm of $\mathbf{S}_\ell$ less than q, we employ length reduction using Cramer's rule.

5.4. Discrete Gaussian Sampling over Integers

In LATTE KeyGen, $\mathbf{f}, \mathbf{g}$ may need to be resampled multiple times due to the norm check and possible failure to find solutions to the NTRU equation. In order to sample $2N$ coordinates efficiently from $\mathcal{D}_{\sigma_0}$, we employ the FACCT sampler [19], which is fast and compact even for larger σ_0 used in LATTE-3 and 4. However, since the FACCT sampler can only sample with $\sigma = k\sqrt{1/(2\ln 2)}$ where k is a positive integer, we slightly increase $\sigma_0 \approx 1.17\sqrt{q/(2N)}$ in LATTE parameters by setting $k = \lceil 1.17\sqrt{q/(2N)}/\sqrt{1/(2\ln 2)}\rceil$.

Let $\mathbf{S}_\ell = \mathbf{L} \cdot \tilde{\mathbf{S}}_\ell$ be the GSO decomposition of the delegated basis $\mathbf{S}_\ell \in \mathcal{R}^{(\ell+2)\times(\ell+2)}$, where $\mathbf{L}$ is a unit lower triangular and rows $\tilde{\mathbf{s}}_i$ of $\tilde{\mathbf{S}}_\ell$ are pairwise orthogonal. We find that the Euclidean norm of the last GSO vector $\tilde{\mathbf{s}}_{\ell+1}$ is very small compared to $\tilde{\mathbf{s}}_0, \ldots, \tilde{\mathbf{s}}_\ell$. This is because rows $\mathbf{s}_0, \ldots, \mathbf{s}_\ell$ of $\mathbf{S}_\ell$ are sampled with a large σ_ℓ but $\det(\mathbf{S}_\ell \cdot \mathbf{S}_\ell^*) = \prod_{i=0}^{\ell+1}\langle \tilde{\mathbf{s}}_i, \tilde{\mathbf{s}}_i\rangle$ is constant and always equal to q^2 [17]. The experiment results in Fig.3 of [49] also verified that $\|\tilde{\mathbf{s}}_{\ell+1}\|$ decreases significantly by increasing $\|\mathbf{s}_0\|$ for $\mathbf{S}_\ell \in \mathcal{R}^{3\times 3}$. In this case, the ratio between the maximal and minimal standard deviation σ' used by the integer discrete Gaussian sampling subroutine in ffSampling is very large and the isochronous sampler [50] used by FALCON [15] will be inefficient for our scheme, since the rejection rate of [50] is proportional to $\max(\sigma')/\min(\sigma')$. In order to sample with σ' in a broad range, we employ a variant [20] of the COSAC sampler [21] instead, which is scalable to large σ' without sacrificing efficiency.

The precision analysis in Sec. 4.3 requires the bound B on RD between a single sample from COSAC and an ideal Gaussian $\mathbb{Z}$ sample. In [21] it is shown that $B \leq 1 + 4\sigma^2 e_x^2 \lambda$, where e_x denotes the absolute error of the underlying Box-Muller continuous Gaussian sampler used by the COSAC sampler and σ denotes the upper bound of the integer Gaussian standard deviation σ.

When $\ell = 1$, we have $\sigma \leq \sigma_{\min} \cdot \frac{\max_i \|(\tilde{\mathbf{S}}_0)_i\|}{\min_i \|(\tilde{\mathbf{S}}_0)_i\|}$, $0 \leq i \leq 2N-1$, for σ of the integer Gaussian in ffSampling [50]. We have $\sigma_{\min} = \eta_\epsilon(\mathbb{Z})$ and $\max_i \|(\tilde{\mathbf{S}}_0)_i\| \leq \sigma_0\sqrt{2N}$. By symplecticity of $\mathbf{S}_0$ [50], we have $\min_i \|(\tilde{\mathbf{S}}_0)_i\| \geq q/(\sigma_0\sqrt{2N})$. Therefore, we get $\sigma \leq \eta_\epsilon(\mathbb{Z}) \cdot (\sigma_0\sqrt{2N})^2/q$. In order to analyse the upper bound of σ when $\ell = 2$, first we introduce the following Lemmas with proofs in Appendix D.

Lemma 5.2. *Every non-root, non-leaf node in a ffLDL tree satisfies* $\min_{k=0}^{2n-1} \mathbf{D}'_k \leq (\mathbf{D}_{i,i})_j \leq \max_{k=0}^{2n-1} \mathbf{D}'_k$*, for some* $\mathbf{D}' \in (\mathbb{R}^+)^{2n}$ *from its parent,* $0 \leq j \leq n-1$*,* $i \in \{0, 1\}$*.*

From Lemma 5.2, if the ancestor of a non-root, non-leaf node is the m-th child of the root, $0 \leq m \leq d-1$, then $(\mathbf{D}_{i,i})_j$ of this node has the minimal value $\min_{k=0}^{N-1}(\mathbf{D}'_{m,m})_k$ and the maximal value $\max_{k=0}^{N-1}(\mathbf{D}'_{m,m})_k$, $i \in \{0,1\}$, $0 \leq j \leq n-1$, for $\mathbf{D}'_{m,m}$ from the root, respectively. The leaf value of an ffLDL tree is $\sigma = \sigma_\ell/\sqrt{(\mathbf{G}_{0,0})_0}$, where $(\mathbf{G}_{0,0})_0 = \frac{1}{2}(\mathbf{D}'_0 + \mathbf{D}'_1)$ for some $\mathbf{D}'$ from its parent. Following a similar approach in the proof of Lemma 5.2, we have: $\min\{\mathbf{D}'_0, \mathbf{D}'_1\} \leq (\mathbf{G}_{0,0})_0 \leq \max\{\mathbf{D}'_0, \mathbf{D}'_1\}$. Therefore, similar to a non-root, non-leaf node, if the ancestor of a leaf node is the m-th child of the root, then the leaf value σ has the minimal value $\sigma_\ell/\sqrt{\max_{k=0}^{N-1}(\mathbf{D}'_{m,m})_k}$ and the maximal value $\sigma_\ell/\sqrt{\min_{k=0}^{N-1}(\mathbf{D}'_{m,m})_k}$.

In order to analyse the minimal and maximal values of $\mathbf{D}'_{m,m}$ from the root, we introduce the following Lemma:

Lemma 5.3. *For FFT domain Gram matrix* $\mathbf{G} = \mathbf{S}_{\ell-1}\mathbf{S}_{\ell-1}^* \in (\mathbb{C}[x]/\langle x^N + 1\rangle)^{(\ell+1)\times(\ell+1)}$*, we have* $|(\mathbf{G}_{i,i})_j| \leq \sigma_{\ell-1}^2 N^2(\ell+1)^2$, $0 \leq i \leq \ell-1$, $0 \leq j \leq N-1$.

For the root of an ffLDL tree when $\ell = 2$, we have $(\mathbf{D}_{0,0})_j = (\mathbf{G}_{0,0})_j \leq 9\sigma_1^2 N^2$ by Lemma 5.3. For $(\mathbf{D}_{1,1})_j = (\mathbf{G}_{1,1})_j - \frac{|(\mathbf{G}_{1,0})_j|^2}{(\mathbf{D}_{0,0})_j}$, since $(\mathbf{D}_{0,0})_j \in \mathbb{R}^+$ from Theorem 5.1, we have $(\mathbf{D}_{1,1})_j \leq (\mathbf{G}_{1,1})_j \leq 9\sigma_1^2 N^2$. By Theorem 5.1, we have $(\mathbf{D}_{2,2})_j = \frac{q^2}{(\mathbf{D}_{0,0})_j(\mathbf{D}_{1,1})_j} \geq \frac{q^2}{81\sigma_1^4 N^4}$, by taking the upper bound $9\sigma_1^2 N^2$ of $(\mathbf{D}_{0,0})_j$, $(\mathbf{D}_{1,1})_j$. Thus, for the leaf values σ, we have $\sigma \leq \sigma_2/\sqrt{q^2/(81\sigma_1^4 N^4)} = \sigma_2 \cdot 9\sigma_1^2 N^2/q$.

We use double precision, i.e. 53-bit floating-point arithmetic precision in the COSAC sampler for LATTE-1 and 2, which provides $e_x \leq 2^{-48}$ [21]. Since the run-time speed of the underlying Box-Muller continuous Gaussian sampler is critical for the speed of the COSAC sampler [21], for the COSAC implementation in LATTE-3 and 4, we use binary128, i.e. 113-bit floating-point arithmetic precision and reduce the absolute precision of uniform sampling in the underlying Box-Muller continuous Gaussian sampler to 96 bits. This will make e_x less than approximately 2^{-96}.

To accelerate the LATTE Encrypt and Decrypt speed, we sample the ephemeral keys $\mathbf{e}, \mathbf{e}_1, \ldots, \mathbf{e}_\ell, \mathbf{e}_h$ from a binomial distribution with center 0 and small standard deviation $\sigma_e = 2.0$ instead of $\mathcal{D}_{\sigma_e}$ used by [13]. Sampling from a binomial distribution is much faster than sampling from $\mathcal{D}_{\sigma_e}$, and the impact on security is negligible in the encryption [51].

6. Performance Results

The first published specification of LATTE [13] only provided the Encrypt and Decrypt performance results, as displayed in "Orig. LATTE" rows in Table 4, scaled and converted into op/s at 4.2GHz. Here, we give the first full performance results for our optimised variant of LATTE, including KeyGen, Extract, and Delegate.

We adapt Plantard's multiplication modular reduction algorithm [52] with word size $w = 32$ bits for LATTE-1 and 2, and $w = 64$ bits for LATTE-3 and 4, respectively. Since Plantard's algorithm requires multiplication in $2w$ bits, we use the 128-bit integer variable type `__uint128` from gcc to implement the modular reduction in LATTE-3 and 4. We employ the gmp [53] library for multi-precision integer arithmetic. For precisions of floating-point and complex number arithmetic, we use 53 bits, i.e. double precision for LATTE-1 and 2, and 113 bits i.e. binary128 for LATTE-3 and 4. We use the `__float128` and `__complex128` variable types from gcc to implement the 113-bit floating-point and complex number arithmetic for LATTE-3 and 4, respectively. Although the error analysis in Sec. 4.3 indicates that the arithmetic precisions for LATTE-3 and 4 can be further reduced, however, the generic multi-precision floating-point library such as MPFR [54] is not optimised for less than 1,000-bit precision in terms of the run-time speed [55].

TABLE 5. SUMMARY OF KEY AND CIPHERTEXT SIZES (BYTES).

Set	Sec.	Master Public Key	Master Private Key	User Private Key		Ciphertext		Delegated Public Key	Delegated Private Key
				$\ell = 1$	$\ell = 2$	$\ell = 1$	$\ell = 2$		
Orig. LATTE-1 [13]	128	6144	12288	9216	-	9248	-	-	-
Our LATTE-1		3072	12288	3072	-	6176	-	-	-
Orig. LATTE-2 [13]	256	12800	25600	19200	-	19232	-	-	-
Our LATTE-2		6400	25600	6400	-	12832	-	-	-
Orig. LATTE-3 [13]	80	9216	18432	13824	18432	13856	18464	9216	41472
Our LATTE-3		4608	18432	4608	9216	9248	13856	9216	41472
Orig. LATTE-4 [13]	160	19456	38912	29184	38912	29216	38944	19456	87552
Our LATTE-4		9728	38912	9728	19456	19488	29216	19456	87552
[47]	40	169600	6400	166400	-	169600	-	-	-
	80	352512	13056	345984	-	352512	-	-	-
	195	1031680	31744	1015808	-	1031680	-	-	-
[48]	96	126720	7680	122880	-	126720	-	-	-
	126	772800	48000	153600	-	154560	-	-	-

We will leave using hand-optimised floating-point arithmetic routines with lower precision as future works.

We use AES-256 CTR mode with hardware AES-NI instructions as the pseudorandom generator, and use SHAKE-256 [56] as the KDF in LATTE Encrypt and Decrypt. The performance results have been obtained from a desktop machine with an Intel i7-7700K CPU at 4.2GHz, with both hyper-threading and TurboBoost disabled. We use gcc 11.2.0 compiler with compiling options `-O3 -march=native` enabled. Results are given as "Our LATTE" rows in Table 4.

As expected, the KeyGen, Extract, and Delegate processes are the most time-consuming components of the scheme, and this increases as security and therefore lattice dimension increase. The trend down the hierarchical levels is that the Extract, Encrypt, and Decrypt all become more time-consuming as the hierarchical level increases. For Extract in LATTE-3 and 4, this corresponds to about 45% decrease in op/s from level 1 to level 2. On the other hand, for the Encrypt and Decrypt, our implementation is 6.0x–9.7x faster compared to the previous performance results from [13]. The speedup might be due to: (1) We change the distribution of the ephemeral keys from discrete Gaussian distribution to the binomial distribution. (2) We only perform NTT for the ephemeral keys and $\mathbf{m}$ during the Encrypt and Decrypt, since other inputs are already in the NTT domain. (3) Since we reduce the dimension of extracted user keys by 1, there is also 1 less ephemeral key in Encrypt/Decrypt. Since the run-time speed of Encrypt/Decrypt in our LATTE implementation is in the order of microseconds, these algorithms should also be feasible on lightweight devices. In addition, our optimised LATTE Delegate only takes about 0.4–1.3 seconds on a desktop machine at 4.2GHz, which is practical and much faster than the estimated run-time (in the order of minutes) for the Delegate in [13].

The key/ciphertext sizes are summarised in Table 5. Since we reduce the dimension of extracted user keys by 1 in our improved LATTE scheme, we compare the key and ciphertext sizes of "our LATTE" scheme with the original LATTE [13] labelled as "Orig. LATTE". From Table 5, our improved LATTE scheme reduces the key/ciphertext sizes by 25%–67% among all LATTE parameter sets.

Our current implementation is not constant-time since the gmp multiprecision integer arithmetic library [53] and the gcc run-time library for the binary128 floating-point and complex number arithmetic are unlikely to be constant-time [57]. We will leave the constant-time implementation of our optimised LATTE scheme as future works.

6.1. Comparison to Other Lattice-based IBEs

Comparison to DLP IBE: Performance results of the DLP IBE scheme from [13] (converted to op/s at 4.2GHz) are given in Table 4. Since the decryption in the DLP IBE did not include ciphertext validation, for a fair comparison with LATTE, we use the sum of DLP encryption and decryption run-time to compute the op/s of decryption in Table 4. We focus on the comparison between LATTE-1 and DLP-3, since the sizes of parameters N and q are similar. The KeyGen speed of our LATTE-1 implementation is 1.9x faster than DLP-3, and the speed of our LATTE-1 Extract implementation is about 3x faster than DLP-3 extraction. This is mainly because we adapt the faster NTRUSolve and lattice Gaussian sampling procedure from FALCON [15]. In addition, the Encrypt/Decrypt speed is 8.7x–11.1x faster in our implementation.

Comparison to IBEs on Standard Models: Performance results and key/ciphertext sizes of the IBEs based on standard models [47], [48] are given in Table 4 and 5, respectively. Similar to the DLP IBE, the decryption run-time in [47], [48] did not include ciphertext validation, so we use the sum of encryption and decryption run-time as the decryption performance results in Table 4. Additionally, since the reported KeyGen run-time of [48] is the sum of KeyGen and the pre-processing of the discrete Gaussian sampler, for a fair comparison, we also use the sum of the KeyGen and pre-processing run-time as the KeyGen performance results of [47] in Table 4 and compare with the performance results of our LATTE variant computing the ffLDL tree during KeyGen in Table 7 in Appendix E. For the run-time comparison, the Extract/Encrypt/Decrypt in our LATTE-1 are 4.5x/14.3x/12.5x faster than the 80-bit secure IBE [47], 14.8x/57.2x/53.2x faster than the 126-bit secure IBE on module lattice [48], and our LATTE-2 are 4.2x/16.1x/17.7x faster than the 195-bit secure IBE [47], respectively. On the other hand, the KeyGen in 80/126/195-bit secure IBEs are 36.2x/5.3x/50.0x faster than LATTE-1/1/2 KeyGen, respectively, due to the fast gadget trapdoor generation algorithms [47], [48]. Note that the implementation in [47] uses 2 threads for parallelisation. For size comparison, the sizes of master public key/master private key/user private key/ciphertext in our LATTE-1 are 114.8x/1.1x/112.6x/57.1x smaller than the 80-bit secure IBE [47], 251.6x/3.9x/50.0x/25.0x smaller than the 126-bit secure IBE on module lattice [48], and the sizes in our LATTE-2 are 161.2x/1.2x/158.7x/80.4x smaller than the 195-bit secure IBE [47], respectively.

References

[1] U. H. Office, "Emergency services network: overview," https://www.gov.uk/government/publications/the-emergency-services-mobile-communications-programme/emergency-services-network, 2022.

[2] U. Comm, "Esn sees 'good progress' but challenges remain in the uk, director says," https://urgentcomm.com/2022/03/19/esn-sees-good-progress-but-challenges-remain-in-the-uk-director-says/, 2022.

[3] *LTE;Security of the mission critical service (3GPP TS 33.180 version 14.8.0 Release 14)*, ETSI-3GPP TS 33.180 version 14.8.0, 2020.

[4] NIST, "Post-quantum crypto project," https://csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization, 2016.

[5] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *EUROCRYPT*, ser. LNCS, vol. 2656. Springer, 2003, pp. 255–271.

[6] Y. Dodis and N. Fazio, "Public key broadcast encryption for stateless receivers," in *Digital Rights Management Workshop*, ser. LNCS, vol. 2696. Springer, 2002, pp. 61–80.

[7] J. Jaeger and I. Stepanovs, "Optimal channel security against fine-grained state compromise: The safety of messaging," in *CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds. Cham: Springer International Publishing, 2018, pp. 33–62.

[8] B. Poettering and P. Rösler, "Towards bidirectional ratcheted key exchange," in *CRYPTO 2018*. Berlin, Heidelberg: Springer-Verlag, 2018, p. 3–32.

[9] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," in *29th USENIX Security Symposium (USENIX Security 20)*, Aug 2020, pp. 2093–2110.

[10] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over NTRU lattices," in *ASIACRYPT (2)*, ser. LNCS, vol. 8874. Springer, 2014, pp. 22–41.

[11] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," in *EUROCRYPT*, ser. LNCS, vol. 6110. Springer, 2010, pp. 523–552.

[12] P. Campbell and M. Groves, "Practical post-quantum hierarchical identity-based encryption," 2017, available at https://www.qub.ac.uk/csit/FileStore/Filetoupload,785752,en.pdf.

[13] "Quantum-Safe Identity-based Encryption," https://www.etsi.org/deliver/etsi_tr/103600_103699/103618/01.01.01_60/tr_103618v010101p.pdf, The European Telecommunications Standards Institute, Sophia-Antipolis, France, Technical Report, 2019.

[14] T. Prest, "Sharper bounds in lattice-based cryptography using the rényi divergence," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 347–374.

[15] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," National Institute of Standards and Technology, Tech. Rep., 2020, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[16] T. Pornin and T. Prest, "More efficient algorithms for the NTRU key generation using the field norm," in *Public Key Cryptography (2)*, ser. LNCS, vol. 11443. Springer, 2019, pp. 504–533.

[17] C. Chuengsatiansup, T. Prest, D. Stehlé, A. Wallet, and K. Xagawa, "ModFalcon: Compact signatures based on Module-NTRU lattices," in *AsiaCCS*. ACM, 2020, pp. 853–866.

[18] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *STOC*. ACM, 2008, pp. 197–206.

[19] R. K. Zhao, R. Steinfeld, and A. Sakzad, "FACCT: fast, compact, and constant-time discrete Gaussian sampler over integers," *IEEE Trans. Computers*, vol. 69, no. 1, pp. 126–137, 2020.

[20] S. Sun, Y. Zhou, Y. Ji, R. Zhang, and Y. Tao, "Generic, efficient and isochronous gaussian sampling over the integers," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 199, 2021.

[21] R. K. Zhao, R. Steinfeld, and A. Sakzad, "COSAC: compact and scalable arbitrary-centered discrete Gaussian sampling over integers," in *PQCrypto*, ser. LNCS, vol. 12100. Springer, 2020, pp. 284–303.

[22] A. Lenstra, H. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, pp. 515–534, 1982.

[23] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *ASIACRYPT*, ser. LNCS, vol. 7073. Springer, 2011, pp. 1–20.

[24] C. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, pp. 181–199, 1994.

[25] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *ANTS*, ser. LNCS, vol. 1423. Springer, 1998, pp. 267–288.

[26] S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. Steinfeld, "Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance," *J. Cryptology*, vol. 31, no. 2, pp. 610–640, 2018.

[27] D. Micciancio and M. Walter, "Gaussian sampling over the integers: Efficient, generic, constant-time," in *CRYPTO 2017*, ser. LNCS, vol. 10402. Springer, 2017, pp. 455–485.

[28] V. Lyubashevsky, "Lattice signatures without trapdoors," in *EUROCRYPT 2012*, ser. LNCS, D. Pointcheval and T. Johansson, Eds., vol. 7237. Springer, 2012, pp. 738–755.

[29] J. Howe, T. Prest, T. Ricosset, and M. Rossi, "Isochronous gaussian sampling: From inception to implementation." in *PQCrypto*, 2020, pp. 53–71.

[30] J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in *EUROCRYPT*, ser. LNCS, vol. 2332. Springer, 2002, pp. 466–481.

[31] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," in *ASIACRYPT*, ser. LNCS, vol. 2501. Springer, 2002, pp. 548–566.

[32] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," in *CRYPTO*, ser. LNCS, vol. 2139. Springer, 2001, pp. 213–229.

[33] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *CRYPTO*, ser. LNCS, vol. 1666. Springer, 1999, pp. 537–554.

[34] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *EUROCRYPT*, ser. LNCS, vol. 3027. Springer, 2004, pp. 223–238.

[35] D. Boneh, X. Boyen, and E. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *EUROCRYPT*, ser. LNCS, vol. 3494. Springer, 2005, pp. 440–456.

[36] T. Koshiba and K. Takashima, "New assumptions on isogenous pairing groups with applications to attribute-based encryption," in *ICISC*, ser. LNCS, vol. 11396. Springer, 2018, pp. 3–19.

[37] S. McCarthy, N. Smyth, and E. O'Sullivan, "A practical implementation of identity-based encryption over NTRU lattices," in *IMACC*, ser. LNCS, vol. 10655. Springer, 2017, pp. 227–246.

[38] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *EUROCRYPT*, ser. LNCS, vol. 6110. Springer, 2010, pp. 1–23.

[39] M.-J. O. Saarinen, "Gaussian sampling precision in lattice cryptography," *IACR ePrint*, vol. 953, p. 2015, 2015.

[40] M. Abe, R. Gennaro, and K. Kurosawa, "Tag-kem/dem: A new framework for hybrid encryption," *IACR Cryptol. ePrint Arch.*, p. 27, 2005.

[41] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," *Journal of cryptology*, vol. 25, no. 4, pp. 601–639, 2012.

[42] T. Prest, "Renyi divergence analysis," Unpublished, 2021, private communication.

[43] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption library," August 2016, https://tfhe.github.io/tfhe/.

[44] E. Díaz-Francés and F. J. Rubio, "On the existence of a normal approximation to the distribution of the ratio of two independent normal random variables," *Statistical Papers*, vol. 54, no. 2, pp. 309–323, 2013.

[45] T. Pöppelmann, L. Ducas, and T. Güneysu, "Enhanced lattice-based signatures on reconfigurable hardware," in *CHES*, ser. LNCS, vol. 8731. Springer, 2014, pp. 353–370.

[46] L. Ducas and T. Prest, "Fast fourier orthogonalization," in *ISSAC*. ACM, 2016, pp. 191–198.

[47] P. Bert, P. Fouque, A. Roux-Langlois, and M. Sabt, "Practical implementation of ring-sis/lwe based signature and IBE," in *PQCrypto*, ser. LNCS, vol. 10786. Springer, 2018, pp. 271–291.

[48] P. Bert, G. Eberhart, L. Prabel, A. Roux-Langlois, and M. Sabt, "Implementation of lattice trapdoors on modules and applications," in *PQCrypto*, ser. LNCS, vol. 12841. Springer, 2021, pp. 195–214.

[49] J. H. Cheon, D. Kim, T. Kim, and Y. Son, "A new trapdoor over Module-NTRU lattice and its application to ID-based encryption," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1468, 2019.

[50] J. Howe, T. Prest, T. Ricosset, and M. Rossi, "Isochronous Gaussian sampling: From inception to implementation," in *PQCrypto*, ser. LNCS, vol. 12100. Springer, 2020, pp. 53–71.

[51] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange - A new hope," in *USENIX Security Symposium*. USENIX Association, 2016, pp. 327–343.

[52] T. Plantard, "Efficient word size modular arithmetic," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1506–1518, 2021.

[53] T. Granlund and G. D. Team, *GNU MP 6.0 Multiple Precision Arithmetic Library*. London, GBR: Samurai Media Limited, 2015.

[54] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A multiple-precision binary floating-point library with correct rounding," *ACM Trans. Math. Softw.*, vol. 33, no. 2, p. 13, 2007.

[55] C. Q. Lauter, "Easing development of precision-sensitive applications with a beyond-quad-precision library," in *ACSSC*. IEEE, 2015, pp. 742–746.

[56] NIST, "SHA-3 standard: Permutation-based hash and extendable-output functions," https://doi.org/10.6028/NIST.FIPS.202, 2015.

TABLE 6. LATTE ALGORITHM SUMMARY.

Alg.	**Inputs**	**Outputs**
KeyGen	$N, q \in \mathbb{Z}, \sigma_0 \in \mathbb{R}$	$\mathbf{h} \in \mathcal{R}_q$, $\mathbf{B}_0, \mathbf{S}_0 \in \mathcal{R}_q^{2\times 2}$
Delegate	$\mathbf{S}_{\ell-1} \in \mathcal{R}_q^{(\ell+1)\times(\ell+1)}$, $\sigma_\ell \in \mathbb{R}$, ID_ℓ, H	$\mathbf{S}_\ell \in \mathcal{R}_q^{(\ell+2)\times(\ell+2)}$
Extract	$\mathbf{S}_{\ell-1} \in \mathcal{R}_q^{(\ell+1)\times(\ell+1)}$, $\sigma_\ell \in \mathbb{R}$, ID_ℓ, H	$\mathbf{t}_0, \ldots, \mathbf{t}_\ell \in \mathcal{R}_q$
Encrypt	$\sigma_e \in \mathbb{R}$, $\mathbf{h} \in \mathcal{R}_q$, KDF, ID_ℓ, μ, H	$Z \in \{0,1\}^{256}$, $\mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h \in \mathcal{R}_q$
Decrypt	$Z \in \{0,1\}^{256}$, $\mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h \in \mathcal{R}_q$, $\mathbf{t}_0, \ldots, \mathbf{t}_\ell \in \mathcal{R}_q$	$\mu' \in \{0,1\}^{256}$

[57] T. Oder, J. Speith, K. Höltgen, and T. Güneysu, "Towards practical microcontroller implementation of the signature scheme falcon," in *PQCrypto*, ser. LNCS, vol. 11505. Springer, 2019, pp. 65–80.

[58] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on gaussian measures," in *FOCS 2004*. IEEE Computer Society, 2004, pp. 372–381.

[59] A. Langlois, D. Stehlé, and R. Steinfeld, "GGHLite: More efficient multilinear maps from ideal lattices," in *EUROCRYPT 2014*, ser. LNCS, P. Q. Nguyen and E. Oswald, Eds., vol. 8441. Springer, 2014, pp. 239–256.

[60] N. Brisebarre, M. Joldes, J. Muller, A. Nanes, and J. Picot, "Error analysis of some operations involved in the cooley-tukey fast fourier transform," *ACM Trans. Math. Softw.*, vol. 46, no. 2, pp. 11:1–11:27, 2020.

[61] Cramer and Gabriel, *Introduction a l'analyse des lignes courbes algebriques par Gabriel Cramer*. chez les freres Cramer & Cl. Philibert, 1750.

Appendix A. LATTE HIBE Scheme

We summarise the inputs/ouputs of the LATTE algorithms in Table 6 and present the full pseudocode of LATTE KeyGen, Extract, Encrypt, and Decrypt in Figure 3–6. Additionally, the ffSampling algorithm from FALCON [15] is presented in Figure 7. Readers may refer to the FALCON specification [15] for other subroutines (NTRUSolve, splitfft, mergefft, etc.) used by these algorithms.

Appendix B. Proof of Lemma 4.1

Proof: Consider the ffSampling Figure 7 in Appendix A. We employ a sequence of games $\mathbf{G_0}, \ldots, \mathbf{G_5}$, and track the probability of the events E and B_U over those games using an RD approach. Let E_i and $B_{U,i}$ denote the events E and B_U in game i for $i = 0, \ldots, 5$. The games REAL and IDEAL are defined in the Lemma statement. The sequence of games is as follows:

- $\mathbf{G_0}$: Game REAL.
- $\mathbf{G_1}$: $\mathbf{G_0}$, but we change the 1-dimensional $\mathbb{Z}$-sampler from the finite precision sampler distribution $\bar{\mathcal{D}}$ to infinite precision sampler distribution $\mathcal{D}$.
- $\mathbf{G_2}$: $\mathbf{G_1}$, but we abort the game if B_U happens, meaning either there exists i such that the errors $\Delta'_{t^{(i)}}$

```
Input: $N, q, \sigma_0$.
Output: $\mathbf{S}_0 \in \mathcal{R}_q^{2\times 2}, \mathbf{h} \in \mathcal{R}_q$.
 1: function KeyGen
 2:     $\mathbf{f}, \mathbf{g} \leftarrow \mathcal{D}_{\sigma_0}^N$.
 3:     $\nu \leftarrow \max\left(\|\mathbf{g}, -\mathbf{f}\|, \left\|\left(\frac{q\cdot\mathbf{f}^*}{\mathbf{f}\cdot\mathbf{f}^*+\mathbf{g}\cdot\mathbf{g}^*}, \frac{q\cdot\mathbf{g}^*}{\mathbf{f}\cdot\mathbf{f}^*+\mathbf{g}\cdot\mathbf{g}^*}\right)\right\|\right)$.
 4:     if $\nu > \sigma_0 \cdot \sqrt{2N}$ then
 5:         goto Step 2.
 6:     end if
 7:     $\mathbf{F}, \mathbf{G} \leftarrow$ NTRUSolve$_{N,q}(\mathbf{f}, \mathbf{g})$.
 8:     if NTRUSolve is aborted then
 9:         goto Step 2.
10:     end if
11:     if $\mathbf{f}$ is not invertible on $\mathcal{R}_q$ then
12:         goto Step 2.
13:     end if
14:     $\mathbf{h} \leftarrow \mathbf{g} \cdot \mathbf{f}^{-1} \mod q$ in NTT domain.
15:     return $\mathbf{S}_0 = \begin{pmatrix} \mathbf{g} & -\mathbf{f} \\ \mathbf{G} & -\mathbf{F} \end{pmatrix}, \mathbf{h}$.
16: end function
```

Figure 3. The LATTE KeyGen algorithm.

```
Input: $N, q, \sigma_\ell, \mathbf{S}_{\ell-1}, H : \{0,1\}^* \to \mathbb{Z}_q^N, G : \{0,1\}^* \to \{0,1\}^{256}, \mathsf{ID}_\ell$.
Output: $\mathbf{t}_1, \ldots, \mathbf{t}_\ell \in \mathcal{R}_q$.
 1: function Extract
 2:     $\mathbf{A}'_\ell \leftarrow H(\texttt{"E"}|\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell)$ in NTT domain.
 3:     $seed \leftarrow G(\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell)$.
 4:     $T_{\ell-1} \leftarrow$ ffLDL(FFT($\mathbf{S}_{\ell-1} \cdot \mathbf{S}_{\ell-1}^*$)).
 5:     For each leaf of $T_{\ell-1}$, leaf.value $\leftarrow \sigma_\ell/\sqrt{\text{leaf.value}}$.
 6:     $\mathbf{t} \leftarrow (\mathbf{A}'_\ell, \mathbf{0}, \ldots, \mathbf{0}) \cdot \mathbf{S}_{\ell-1}^{-1}$.
 7:     $\mathbf{z} \leftarrow$ FFT$^{-1}$(ffSampling($\mathbf{t}, T_{\ell-1}, seed$)).
 8:     $(\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_\ell) \leftarrow \lfloor \bar{\mathbf{z}} \rceil$, where $\bar{\mathbf{z}} \leftarrow (\mathbf{t} - \mathbf{z}) \cdot \mathbf{S}_{\ell-1}$.
 9:     return $\mathbf{t}_1, \ldots, \mathbf{t}_\ell \in \mathcal{R}_q$ in NTT domain.
10: end function
```

Figure 4. LATTE Extract algorithm (from level $\ell - 1$ to user at level ℓ).

(relative to $\sigma^{(i)}$) in centers $t^{(i)}$ exceed $\Delta'^{U}_{t^{(i)}}$ or relative errors $\delta_{\sigma^{(i)}}$ in standard deviations $\sigma^{(i)}$ exceed $\delta^U_{\sigma^{(i)}}$ or there exists j such that the infinity-norm absolute errors $\Delta_{\bar{\mathbf{z}}^{(j)}}$ in $\bar{\mathbf{z}}^{(j)}$ exceed $\Delta^U_{\bar{\mathbf{z}}}$.

- $\mathbf{G_3}$: $\mathbf{G_2}$, but we restrict the 1-dimensional $\mathbb{Z}$ samplers $\mathcal{D}$ to the corresponding τ-bounded distribution $\mathcal{D}^\tau$.
- $\mathbf{G_4}$: $\mathbf{G_3}$, but changing arithmetic from finite precision to infinite precision, and removing the τ-tailcut on the 1-dimensional $\mathbb{Z}$ samplers to return to the ideal Gaussian distribution $\mathcal{D}$. This game is identical to $\mathbf{G}_{\text{IDEAL}}$, except for the abort condition introduced in the previous game.
- $\mathbf{G_5}$: $\mathbf{G_4}$, but remove the abort introduced in $\mathbf{G_2}$. This game is identical to $\mathbf{G}_{\text{IDEAL}}$.

```
Input: $N, q, \sigma_e, \mathbf{h}$, KDF, $\mathsf{ID}_\ell, \mu \in \{0,1\}^{256}$.
Output: $Z \in \{0,1\}^{256}, \mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h \in \mathcal{R}_q$.
 1: function Encrypt
 2:     $seed \leftarrow_\$ \{0,1\}^{256}$.
 3:     $Z \leftarrow \mu \oplus$ KDF($seed$).
 4:     Sample $\mathbf{e}, \mathbf{e}_1, \ldots, \mathbf{e}_\ell, \mathbf{e}_h$ from a binomial distribution with center 0 and standard deviation $\sigma_e$ using the seed KDF($seed|Z$).
 5:     for $i \in \{1, \ldots, \ell - 1\}$ do
 6:         $\mathbf{C}_i \leftarrow \mathbf{A}_i \cdot \mathbf{e} + \mathbf{e}_i$, where $\mathbf{A}_i = H(\mathsf{ID}_1|\ldots|\mathsf{ID}_i)$ in NTT domain.
 7:     end for
 8:     $\mathbf{m} \leftarrow$ Encode($seed$).
 9:     $\mathbf{C}_\ell \leftarrow \mathbf{A}'_\ell \cdot \mathbf{e} + \mathbf{e}_\ell + \mathbf{m}$, where $\mathbf{A}'_\ell = H(\texttt{"E"}|\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell)$ in NTT domain.
10:     $\mathbf{C}_h \leftarrow \mathbf{h} \cdot \mathbf{e} + \mathbf{e}_h$.
11:     return $Z \in \{0,1\}^{256}, \mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h \in \mathcal{R}_q$ in NTT domain.
12: end function
```

Figure 5. The LATTE Encrypt algorithm (at level ℓ).

```
Input: $N, q, \sigma_e, \mathbf{h}$, KDF, $\mathsf{ID}_\ell, Z, (\mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h), \mathbf{t}$.
Output: $\mu'$.
 1: function Decrypt
 2:     $\mathbf{V} \leftarrow \mathbf{C}_\ell - \mathbf{C}_h \cdot \mathbf{t}_1 - \mathbf{C}_1 \cdot \mathbf{t}_2 - \cdots - \mathbf{C}_{\ell-1} \cdot \mathbf{t}_\ell$.
 3:     $seed' \leftarrow$ Decode($\mathbf{V}$).
 4:     Sample $\mathbf{e}', \mathbf{e}'_1, \ldots, \mathbf{e}'_\ell, \mathbf{e}'_h$ from a binomial distribution with center 0 and standard deviation $\sigma_e$ using the seed KDF($seed'|Z$).
 5:     for $i \in \{1, \ldots, \ell - 1\}$ do
 6:         $\mathbf{C}'_i \leftarrow \mathbf{A}_i \cdot \mathbf{e}' + \mathbf{e}'_i$, where $\mathbf{A}_i = H(\mathsf{ID}_1|\ldots|\mathsf{ID}_i)$ in NTT domain.
 7:     end for
 8:     $\mathbf{m}' \leftarrow$ Encode($seed'$).
 9:     $\mathbf{C}'_\ell \leftarrow \mathbf{A}'_\ell \cdot \mathbf{e}' + \mathbf{e}'_\ell + \mathbf{m}'$, where $\mathbf{A}'_\ell = H(\texttt{"E"}|\mathsf{ID}_1|\ldots|\mathsf{ID}_\ell)$ in NTT domain.
10:     $\mathbf{C}'_h \leftarrow \mathbf{h} \cdot \mathbf{e}' + \mathbf{e}'_h$.
11:     Check $(\mathbf{C}'_1, \ldots, \mathbf{C}'_\ell, \mathbf{C}'_h)$ agrees with $(\mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h)$, else return $\perp$.
12:     return $\mu' = Z \oplus$ KDF($seed'$).
13: end function
```

Figure 6. The LATTE Decrypt algorithm (at level ℓ).

$\mathbf{G_0} \to \mathbf{G_1}$: Changing the 1-D $\mathbb{Z}$-sampler. Let $(\bar{\sigma}^{(i)}, \bar{t}^{(i)}) = (\sigma^{(i)}(1 + \delta_{\sigma^{(i)}}, t^{(i)} + \Delta_{t^{(i)}})$ denote the i'th query to the 1-D sampler in the execution of these games, and denote by $\zeta^{(i)}$ the output integer returned by the sampler for the i'th query. We apply Proposition 4, with x_0 denoting the remaining source of randomness in the game (i.e. the random coins of $\mathcal{A}$ and the hash function H), and we let

```
Input: $\mathbf{t} = (\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_\ell)$ in FFT format, tree $T$,
    $seed \in \{0,1\}^{256}$.
Output: $\mathbf{z} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$ in FFT format.
function ffSampling($\mathbf{t}, T$)
    if $n = 1$ then
        $\sigma' \leftarrow T$.value.
        Sample $z_0 \leftarrow \mathcal{D}_{\sigma', t_0}$ using $seed$.
        Sample $z_1 \leftarrow \mathcal{D}_{\sigma', t_1}$ using $seed$.
        return $\mathbf{z} = (z_0, z_1)$.
    else
        $m \leftarrow$ number of children of $T$.
        for $j \leftarrow m-1, \ldots, 0$ do
            $T_j \leftarrow j$-th child of $T$.
            $\mathbf{t}'_j \leftarrow \mathbf{t}_j + \sum_{i=j+1}^{m-1} (\mathbf{t}_i - \mathbf{z}_i) \cdot T.\text{value}_{i,j}$.
            $\mathbf{t}'_j \leftarrow$ splitfft($\mathbf{t}'_j$).
            $\mathbf{z}'_j \leftarrow$ ffSampling($\mathbf{t}'_j, T_j$).
            $\mathbf{z}_j \leftarrow$ mergefft($\mathbf{z}'_j$).
        end for
        return $\mathbf{z} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_m)$.
    end if
end function
```

Figure 7. The ffSampling algorithm [15].

$x_i := \zeta^{(i)}$ for $i = 1, \ldots, M_{\mathbb{Z}}$. Consider $(x_i|x_{j<i})$, the conditional distribution of x_i, conditioned on all previous x_j, for $j < i$, and the RD between this distribution in $\mathbf{G_0}$ and $\mathbf{G_1}$. Observe that conditioned on the same value of $x_{j<i}$, the values of the following query $\bar{t}^{(i)}$ and std. deviation $\bar{\sigma}^{(i)}$ are identical in both $\mathbf{G_0}$ and $\mathbf{G_1}$ since they both use the same finite precision arithmetic. We have:

$$R_a((x_i|x_{j<i})_{G_0}, (x_i|x_{j<i})_{G_1}) = R_a(\bar{\mathcal{D}}_{\bar{\sigma}^{(i)}, \bar{t}^{(i)}}, \mathcal{D}_{\bar{\sigma}^{(i)}, \bar{t}^{(i)}}) \le B,$$

then Proposition 2.2 implies:

$$R_a\Big((x_0, \ldots, x_K)_{G_0}, (x_0, \ldots, x_K)_{G_1}\Big) \le B^{M_{\mathbb{Z}}} := B_T.$$

By the data processing and probability preservation properties of RD, $\Pr[E_1] \ge \Pr[E_0]^{a/(a-1)}/B_T$.

$\mathbf{G_1} \to \mathbf{G_2}$: Adding a τ tailcut to the $\mathbb{Z}$ Gaussian samplers. By a standard tail bound [28, Lemma 4.4], the statistical distance between this game and the previous one is $\le M_{\mathbb{Z}} \cdot 2\exp(-\tau^2/2) \le 1/Q_M$. Hence, we have $\Pr[E_2] \ge \Pr[E_1] - 1/Q_M$.

$\mathbf{G_2} \to \mathbf{G_3}$: Aborting the game if the errors exceed the bounds. Recall that $B_{U,2}$ denotes the event B_U in $\mathbf{G_2}$ that the errors exceed the bounds in the Lemma statement. If the event $B_{U,2}$ does not occur, games $\mathbf{G_2}$ and $\mathbf{G_3}$ proceed identically. Hence, we have $\Pr[E_3] \ge \Pr[E_2] - \Pr[B_{U,2}]$ and $\Pr[B_{U,2}] = \Pr[B_{U,3}]$.

$\mathbf{G_3} \to \mathbf{G_4}$: Changing finite precision arithmetic to infinite precision and removing the τ-tailcut on the Gaussians. We again apply Proposition 2.2, except that this time $x_i := \zeta^{(i)}$ for $1 \le i \le M_{\mathbb{Z}}$ except if the event B_U occurs at the i'th query to the $\mathbb{Z}$ sampler (determined by $x_0, \ldots, x_{i-1}$), in which case $x_i := \perp$, and all subsequent $x_j := \perp$ for $j > i$. As in the previous game, we consider the conditional distribution $(x_i|x_{j<i})$, of x_i conditioned on all previous x_j for $j < i$, and the RD between this conditional distribution in $\mathbf{G_3}$ and $\mathbf{G_4}$. When the event B_U occurs at the at (or before) the i'th query to the $\mathbb{Z}$ sampler, the conditional distribution $(x_i|x_{j<i})$ is identical in both games (as both conditional distributions return $\perp$ with probability 1) and have RD 0. Whereas, if the event B_U does not occur at (or before) the i'th query to the $\mathbb{Z}$ conditioned on the same fixed value of $x_{j<i}$ in the support of the j'th 1-D $\mathbb{Z}$-samplers, we have $\Delta'_{t^{(i)}} \le \Delta'^{U}_{t^{(i)}}$. Also, the query std deviation values $\sigma^{(i)}$ in G_4 and $\bar{\sigma}^{(i)}$ in G_3 have a relative error $\delta_{\sigma^{(i)}} \le \delta^{U}_{\sigma^{(i)}}$ by definition of event B_U. We therefore have:

$$R_a((x_i|x_{j<i})_{G_3}, (x_i|x_{j<i})_{G_4}) \tag{7}$$
$$\le R_a(\mathcal{D}^{\tau}_{\sigma^{(i)}\cdot(1\pm\delta_{\sigma^{(i)}}), t^{(i)}+\Delta'_{t^{(i)}}\cdot\sigma^{(i)}}, \mathcal{D}_{\sigma^{(i)}, t^{(i)}}) \le C^{(i)}, \tag{8}$$

where in the last inequality, we used Lemma 4.2. Then Proposition 2.2 above implies:

$$R_a((x_0, \ldots, x_{M_{\mathbb{Z}}})_{G_2}, (x_0, \ldots, x_{M_{\mathbb{Z}}})_{G_3}) \le \prod_{i<M_{\mathbb{Z}}} C^{(i)} := C_T.$$

Due to the abort condition, we have that conditioned on the same fixed value of x_i's that do not cause an abort, the values $\bar{\mathbf{z}}^{(j)}$ in $\mathbf{G_3}$ and $\mathbf{G_4}$ differ by an absolute error at most $\Delta_{\bar{\mathbf{z}}} < 1/2$, and therefore, observing that in $\mathbf{G_4}$ the $\bar{\mathbf{z}}^{(j)}$ has integer coordinates (due to the infinite precision), the rounded $\bar{\mathbf{z}}^{(j)}$values in $\mathbf{G_4}$ are identical to those in $\mathbf{G_3}$ conditioned on the same x_i's. Since the adversary's view in the game depends on x_i's only via the rounded $\bar{\mathbf{z}}^{(j)}$, we conclude by the Rényi probability preservation property that $\Pr[E_4] \ge \Pr[E_3]^{a/(a-1)}/C_T$, and $\Pr[B_{U,4}] \ge \Pr[B_{U,3}]^{a/(a-1)}/C_T$.

$\mathbf{G_4} \to \mathbf{G_5}$: In this game, we remove the abort introduced in $\mathbf{G_2}$. Since the games $\mathbf{G_4}$ and $\mathbf{G_5}$ proceed identically until an abort occurs, we have $\Pr[B_{U,5}] = \Pr[B_{U,4}]$ and $\Pr[E_5] \ge \Pr[E_4] - \Pr[B_{U,4}]$. Furthermore, by the Lemma hypothesis, we have $\Pr[B_{U,5}] := p_U$. Putting together the above bounds, we obtain that the probability of E_5 (i.e. event E in IDEAL) is lower bounded by

$$\begin{aligned}\Pr[E_5] &\ge \Pr[E_3]^{a/(a-1)}/C_T - p_U \\ &\ge \frac{(\Pr[E_0]^{a/(a-1)}/B_T - (\Pr[B_{U,3}] + 1/Q_M))^{a/(a-1)}}{C_T} - p_U\end{aligned}$$

and using $p_U = \Pr[B_{U,5}] = \Pr[B_{U,4}] \ge \Pr[B_{U,3}]^{a/(a-1)}/C_T$, we get bound on $\Pr[E_4] := \Pr[E_{\text{IDEAL}}]$. □

Appendix C. Proof of Lemma 4.2

Proof: We have that $\frac{D_3(z)}{D_2(z)} = \frac{\rho_{t,\sigma}(I)}{\rho_{\bar{t},\bar{\sigma}}(I)} \cdot \exp(u(z))$, where, following assumption (3) in the statement of Lemma and the notations of the proof of [14, Lemma 7], we get (9).

$$u(z) = \frac{(z-t)^2}{2\sigma^2} - \frac{(z-\bar{t})^2}{2\bar{\sigma}^2} \tag{9}$$

$$= -\frac{(t-\bar{t})^2 + 2(t-\bar{t})(z-t) - (2\delta_\sigma + \delta_\sigma^2)(z-t)^2}{2(1-\delta_\sigma)^2\sigma^2}. \tag{10}$$

We first bound $\frac{\rho_{t,\sigma}(I)}{\rho_{\bar{t},\bar{\sigma}}(I)}$. Let $\mathsf{erfc}(\alpha) := \frac{1}{\sqrt{2\pi}}\int_\alpha^\infty \exp(-y^2/2)dy$ be the complementary error function; then we get:

$$\begin{aligned}\rho_{t,\sigma}(I) &= \rho_{t,\sigma}(\mathbb{Z}) - 2\cdot \mathsf{erfc}(\tau\cdot\sigma)\\ &= \rho_{t,\sigma}(\mathbb{Z})\cdot\left(1 - \frac{2\cdot\mathsf{erfc}(\tau\cdot\sigma)}{\rho_{t,\sigma}(\mathbb{Z})}\right) \geq \rho_{t,\sigma}(\mathbb{Z})\cdot(1-1/Q),\end{aligned}$$

where the last inequality uses $\rho_{t,\sigma}(\mathbb{Z}) \sim \sqrt{2\pi}(1-\varepsilon)\sigma$, $\mathsf{erfc}(\tau\sigma) \leq \frac{\exp(-(\tau\sigma)^2/2)}{\sqrt{2\pi}\tau\sigma}$, and $Q \leq 2\pi\sqrt{\sigma\tau}(1-\varepsilon)\sigma\exp(\sigma^2\tau^2/2)$ set in condition 5). Deriving a similar inequality for $\rho_{\bar{t},\bar{\sigma}}(I)$, we have:

$$1 - 2/Q \lesssim \frac{\rho_{t,\sigma}(I)}{\rho_{\bar{t},\bar{\sigma}}(I)} \Bigg/ \frac{\rho_{t,\sigma}(\mathbb{Z})}{\rho_{\bar{t},\bar{\sigma}}(\mathbb{Z})} \lesssim 1 + 2/Q. \tag{11}$$

Let $\mathsf{n} := ((\Delta_t')^2 + 2\Delta_t'(z-t)/\sigma - (2\delta_\sigma+\delta_\sigma^2)((z-t)/\sigma)^2)$. By applying [14, Lemma 7], followed by (10), followed by [58, Lemma 4.4] and finally using $\max(\delta_\sigma, \epsilon) \leq \delta$, it follows that:

$$\ln\left(\frac{\rho_{t,\sigma}(\mathbb{Z})}{\rho_{\bar{t},\bar{\sigma}}(\mathbb{Z})}\right) \leq |\mathbb{E}_{z\leftarrow D_1}[u]| \leq \left|\mathbb{E}_{z\leftarrow D_1}\left[\frac{\mathsf{n}}{2(1-\delta_\sigma)^2}\right]\right| \tag{12}$$

$$\leq \frac{1}{2(1-\delta_\sigma)^2}\left|\mathbb{E}_{z\leftarrow D_1}[\mathsf{n}]\right| \tag{13}$$

$$\leq \mathsf{num}(\Delta_t', \varepsilon, \delta_\sigma) \tag{14}$$

Similarly, we bound $\exp(u(z))$ over I by using (10):

$$\max_I |u| \lesssim \frac{1}{1-\delta_\sigma}\cdot(\tau\Delta_t' + \tau^2\delta_\sigma). \tag{15}$$

Combining (11), (14), and (15), we bound the relative error:

$$\begin{aligned}\left|\ln\left(\frac{D_3}{D_2}\right)\right| &\leq \ln(1+2/Q) + \mathsf{num}(\Delta_t', \varepsilon, \delta_\sigma)\\ &\quad + \frac{1}{1-\delta_\sigma}\cdot(\tau\Delta_t' + \tau^2\delta_\sigma)\\ &\leq 2/Q + \mathsf{num}(\Delta_t', \varepsilon, \delta_\sigma)\\ &\quad + \frac{1}{1-\delta_\sigma}\cdot(\tau\Delta_t' + \tau^2\delta_\sigma) = \mathsf{ub}.\end{aligned} \tag{16}$$

Combining (16) and [14, Lemma 3], the RD between D_3 and D_2 is derived as (5). Finally, we combine the first weak triangle inequality of [59, Lemma 4.1] with remark 1 to obtain the RD between D_3 and D_1 as in (6). □

Appendix D. Proofs of Lemma 5.2 and 5.3

Proof: From Theorem 5.1, for a non-root, non-leaf node, since $(\mathbf{D}_{0,0})_j = \frac{1}{2}(\mathbf{D}'_{2j} + \mathbf{D}'_{2j+1})$, $0 \leq j \leq n-1$, for some $\mathbf{D}' \in (\mathbb{R}^+)^{2n}$, $(\mathbf{D}_{0,0})_j$ gets the minimal value $\min_{k=0}^{2n-1}\mathbf{D}'_k$ when both $\mathbf{D}'_{2j}$ and $\mathbf{D}'_{2j+1}$ are equal to $\min_{k=0}^{2n-1}\mathbf{D}'_k$. Similarly, $(\mathbf{D}_{0,0})_j$ gets the maximal value $\max_{k=0}^{2n-1}\mathbf{D}'_k$ when both $\mathbf{D}'_{2j}$ and $\mathbf{D}'_{2j+1}$ are equal to $\max_{k=0}^{2n-1}\mathbf{D}'_k$. For $(\mathbf{D}_{1,1})_j = \mathbf{D}'_{2j}\mathbf{D}'_{2j+1}/(\mathbf{D}_{0,0})_j = \frac{\mathbf{D}'_{2j}\mathbf{D}'_{2j+1}}{1/2\cdot(\mathbf{D}'_{2j}+\mathbf{D}'_{2j+1})}$, it gets the minimal value $\min_{k=0}^{2n-1}\mathbf{D}'_k$ when both $\mathbf{D}'_{2j}$ and $\mathbf{D}'_{2j+1}$ are equal to $\min_{k=0}^{2n-1}\mathbf{D}'_k$ and $(\mathbf{D}_{1,1})_j$ gets the maximal value $\max_{k=0}^{2n-1}\mathbf{D}'_k$ when both $\mathbf{D}'_{2j}$ and $\mathbf{D}'_{2j+1}$ are equal to $\max_{k=0}^{2n-1}\mathbf{D}'_k$ for $\mathbf{D}' \in (\mathbb{R}^+)^{2n}$. □

Proof: We have $\mathbf{G}_{i,i} = \sum_{k=0}^{\ell}\mathrm{FFT}(\mathbf{S}_{\ell-1})_{i,k} \odot \mathrm{FFT}(\mathbf{S}^*_{\ell-1})_{k,i}$, and thus $|(\mathbf{G}_{i,i})_j| = \sum_{k=0}^{\ell}|(\mathrm{FFT}(\mathbf{S}_{\ell-1})_{i,k})_j|^2$. For N-point FFT result $\mathbf{z}$ of scalar $\mathbf{a}$, we have $|\mathbf{z}_i| \leq \|\mathbf{z}\| = \sqrt{N}\|\mathbf{a}\|$ for $0 \leq i \leq N-1$ [60]. Thus, we have $|(\mathbf{G}_{i,i})_j| \leq (\ell+1)\cdot N\cdot\|(\mathbf{S}_{\ell-1})_{i,k}\|^2 \leq \sigma_{\ell-1}^2 N^2(\ell+1)^2$, since $\|(\mathbf{S}_{\ell-1})_{i,k}\| \leq \sigma_{\ell-1}\cdot\sqrt{(\ell+1)N}$. □

Appendix E. Compute ffLDL Tree in KeyGen/Delegate

If the key extraction speed is critical for the application, similar to FALCON [15], we can move the ffLDL Tree computation from the LATTE Extract (Line 4 in Figure 4 in Appendix A) to the LATTE KeyGen/Delegate when generating a master/delegated private key $\mathbf{S}_\ell$, at the expense of significantly larger master/delegated private key size. The KeyGen/Delegate/Extract speed of this LATTE variant is shown in Table 7. The LATTE Extract in this variant is about 1.3x–1.7x faster than the run-time speed in Table 4, while the KeyGen/Delegate is at most 6% slower.

Here we also analyse the overhead in the master/delegated private key size of this variant due to the ffLDL tree T. Assuming a floating-point value has p bytes, the size of T consists of the following 3 parts: (1) For a $d\times d$ basis $\mathbf{S}_\ell$, the root of T stores $d(d-1)/2$ components of $\mathbf{L}$ from the $\mathbf{LDL}^*$ decomposition, with each component in $\mathbb{C}[x]/\langle x^N+1\rangle$. Thus, the root of T has $d(d-1)/2\cdot 2Np = Npd(d-1)$ bytes. (2) The root of T has d sub-trees. The i-th non-leaf level of a sub-tree has 2^i nodes, $0 \leq i \leq \log_2 N - 2$. Each node at i-th level of a sub-tree stores $\mathbf{L}_{1,0} \in \mathbb{C}[x]/\langle x^n+1\rangle$ from the $\mathbf{LDL}^*$ decomposition, where $n = N/2^{i+1}$. Therefore, the total size of i-th level of a sub-tree is $2^i\cdot 2(N/2^{i+1})p = Np$ bytes, and the total size of all non-leaf nodes in a sub-tree is $Np(\log_2 N - 1)$ bytes. (3) A sub-tree has $N/2$ leaf nodes. Each leaf node stores a p-byte floating-point value. Therefore, the total size of all leaf nodes in a sub-tree is $Np/2$ bytes. Thus, the total size of T is $Npd(d-1) + d(Np(\log_2 N - 1) + Np/2) = Npd(\log_2 N + d - 3/2)$ bytes. Columns "T Size" in Table 7 summarise the ffLDL

TABLE 7. Performance Results (op/s) and ffLDL Tree Size (Bytes) of LATTE Variant at 4.2GHz.

		$\ell = 1$			$\ell = 2$	
Set	**KeyGen**	**Ext**	**Del**	**T Size**	**Ext**	**T Size**
LATTE-1	9.3	1802.8	-	172032	-	-
LATTE-2	3.3	826.3	-	376832	-	-
LATTE-3	5.4	54.6	2.3	344064	33.6	565248
LATTE-4	1.6	26.0	0.8	753664	16.0	1228800

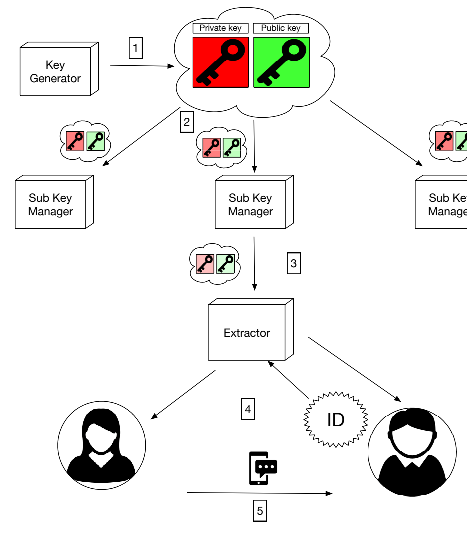

Figure 8. A 2-level HIBE scheme.

tree size for the parameters and floating-point precisions in our LATTE implementation.

Appendix F. HIBE Scheme Description

This Section presents the components of a HIBE scheme and how they interact. They are as follows:

1) **KeyGen:** The master key generator establishes the master public and private keys.
2) **Delegate:** Through a delegation function, the master key generator creates a public/private key pair for the sub-key manager. This gives it the ability to delegate further key pairs, and extract user private keys at that level.
3) **Delegate:** The sub-key manager delegates a further public/private key to the next level of the hierarchy.
4) **Extract:** The extractor uses their public/private key pair to extract and share user public/private keys, as in the single-level IBE scheme.
5) **Encrypt/Decrypt:** Encryption/decryption works as a regular encryption scheme, such as R-LWE encryption.

Figure 8 depicts a diagram of a 2-level HIBE scheme with all its algorithms.

A HIBE scheme is said to be IND-CCA-secure if it is indistinguishable under chosen-ciphertext attacks; that is, an adversary with the ability to decrypt any other ciphertext does not possess an advantage in decrypting the challenge ciphertext. ID-IND-CCA further implies the adversary has access to an extraction oracle that allows them to extract keys for other identities before committing to the challenge identity, yet gains no advantage. The challenge consists of the ciphertext *and* the identity under which it is encrypted.

TABLE 8. Explanation of Notational Practice of HIBE Functions.

Level	**Function**
Level 0	Master KeyGen $2N \times 2N$
Level 1	Extracting with $2N \times 2N \rightarrow$ Enc/Dec
	Delegating to $3N \times 3N$
Level 2	Extracting with $3N \times 3N \rightarrow$ Enc/Dec

Table 8 indicates the notational practices used to identify each level of the hierarchy.

Appendix G. Cramer's Rule

Cramer's rule [61] is used for solving systems of linear equations. Considering a system of N equations with N unknowns $\mathbf{x}$, represented as $\mathbf{Ax} = \mathbf{b}$. Cramer's rule states that the solution can be written as $\mathbf{x}_i = \frac{\det(\mathbf{A}_i)}{\det(\mathbf{A})}$, where $\mathbf{A}_i$ is the matrix formed by replacing the i-th column of $\mathbf{A}$ by the column vector $\mathbf{b}$.

The formulae for the reduction coefficients in the KeyGen and Delegate process come directly from Cramer's Rule applied to the system $\mathbf{Ax} = \mathbf{b}$, where, in the first level, $\mathbf{A}$ is the 2×2 matrix whose (i, j)-entry is the Hermitian product $\langle \mathbf{s}_i, \mathbf{s}_j \rangle$ of the i^{th} and j^{th} rows of the delegation matrix, and where $\mathbf{b}$ is the two-dimensional column vector whose i^{th} coefficient is $\langle \mathbf{s}_2, \mathbf{s}_i \rangle$. This result generalises to arbitrary levels; i.e., for any given number of levels $\ell \geq 1$, the reduction of the vector $\mathbf{s}_{\ell+1}$ is effected by replacing it with $\mathbf{s}_{\ell+1} - \lfloor \mathbf{k}_0 \rceil \mathbf{s}_0 - \ldots - \lfloor \mathbf{k}_\ell \rceil \mathbf{s}_\ell$, where the $\mathbf{k}_i$ are the coefficients of the solution $\mathbf{x}$ to the system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is the $(\ell + 1) \times (\ell + 1)$ matrix whose (i, j)-entry is the Hermitian product $\langle \mathbf{s}_i, \mathbf{s}_j \rangle$ of the i^{th} and j^{th} rows of the delegation matrix, and where $\mathbf{b}$ is the $(\ell + 1)$-dimensional column vector whose i-th coefficient is $\langle \mathbf{s}_{\ell+1}, \mathbf{s}_i \rangle$.

Appendix H. Key and Ciphertext Size Calculations

The LATTE keys and ciphertexts are mainly collections of polynomials in $\mathcal{R}$. The degree of each polynomial is N and the number of bits in each coefficient is $\kappa = \lceil \log_2 q \rceil$. The parameters N and q are dependent on the security level required. The key/ciphertext bit-size is equal to $N \cdot \kappa \cdot$ *number of polynomials*, plus any additional bit strings sent, in the case of the ciphertext. Furthermore, we usually consider the key and ciphertext sizes in bytes, and so when the total bit size is computed, it will be divided by 8 to give the size in bytes.

H.1. Master Keys

The master public key consists of a polynomial $\mathbf{h} \in \mathcal{R}_q$. Therefore the bit-size is $N \cdot \kappa$. The master private key $\mathbf{S}_0$ consists of $(\mathbf{f}, \mathbf{g}, \mathbf{F}, \mathbf{G})$. However, $\mathbf{F}$ and $\mathbf{G}$ can be recomputed on the fly from $\mathbf{f}$ and $\mathbf{g}$ using NTRUSolve. The solution is not unique but as long as it is a short solution, it will suffice. However, this is not efficient and so this research considers the entire $(\mathbf{f}, \mathbf{g}, \mathbf{F}, \mathbf{G})$ to be stored as the private key. Therefore, the master private key is of size $4N \cdot \kappa$.

H.2. Delegated Keys

The delegated public key can be straightforwardly generated using the master public key and the chain of user IDs along which the delegation process is happening. Although this can be efficiently generated on the fly, given the user ID chain, we will consider it being stored as the polynomials $\mathbf{h}, \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_\ell$, which translates as $(\ell + 1)$ polynomials in $\mathcal{R}$, and so the total bit-size of the delegated public key is $(\ell + 1) \cdot N \cdot \kappa$. The delegated private key generated from level $\ell - 1$ to level ℓ, to be passed onto users at level $\ell + 1$, is a $(\ell + 2) \times (\ell + 2)$ matrix of polynomials in $\mathcal{R}_q$. Its size is therefore $(\ell + 2) \cdot (\ell + 2) \cdot N \cdot \kappa$.

H.3. User Private Keys

The user's public key is entirely dependent on the identity, so we only examine the size of the user's private key. In LATTE for a user at level ℓ, this is a tuple of $(\ell + 1)$ polynomials in $\mathcal{R}_q$. However, we only need to store ℓ of these polynomials (disregarding t_0) and so the user private key is of bit size $\ell \cdot N \cdot \kappa$.

H.4. Ciphertexts

Let's consider the ciphertext at level ℓ. This consists of $\ell + 1$ polynomials $\mathbf{C}_1, \ldots, \mathbf{C}_\ell, \mathbf{C}_h \in \mathcal{R}_q$ along with a 256-bit string Z (which is essentially the encrypted message). Therefore, at level ℓ, the bit-size of the full ciphertext is $(\ell + 1) \cdot N \cdot \kappa + 256$.

H.5. Comparison to FALCON

After adopting the NTRUSolve and lattice Gaussian sampling procedures from FALCON [15], our optimised LATTE KeyGen becomes similar to the FALCON KeyGen, and our optimised LATTE Extract becomes similar to the FALCON Sign, in terms of the operations used by these algorithms. Therefore, here we compare the run-time speed of our optimised LATTE KeyGen/Extract against the FALCON KeyGen/Sign, respectively. The performance results of the FALCON is summarised in Table 9. We focus on the comparison between LATTE-1 and FALCON-1024 since the size of parameter N is the same. The KeyGen speed of our LATTE-1 implementation is about 7.1x slower than FALCON-1024, and the speed of our LATTE-1 Extract implementation is about 3.9x slower than FALCON-1024 Sign. This is mainly because (1) The size of q in LATTE is much larger than FALCON (24 bits for LATTE-1 compared to 14 bits for FALCON-1024), which will significantly increase the maximal integer size in NTRUSolve, as well as the run-time overhead in KeyGen, [16]. (2) FALCON computes the ffLDL Tree during the KeyGen, while the ffLDL Tree is computed during the Extract in our LATTE scheme. This difference will add overhead to the run-time speed of our LATTE Extract implementation. (3) From the FALCON specification [15], the AVX2 and FMA instructions were used in the source code during the benchmark. However, these instructions are not used in the source code of our LATTE implementation.

Appendix I. Concrete Parameter Sets Based on Best Known Attacks

The security of each component of LATTE depends on an associated lattice problem and so the computational security of each of these problems must be considered to derive parameters, with the most vulnerable component determining the overall security for a given parameter set. The global parameters for the scheme are dimension N and modulus q, but we will also need to consider level-specific parameters, namely the standard deviation used for sampling at each level, σ_ℓ. The six security constraints to be considered are: (1) Gaussian sampler security (2) Decryption failure (3) Master key recovery (breaking the NTRU problem/finding short vectors in the NTRU lattice) (4) Delegated key recovery (finding short vectors in the lattice) (5) User key recovery (solving closest vector problem) (6) Message recovery (breaking the R-LWE encryption scheme). These are discussed in detail in [13], so here we only state the mathematical conditions which must be satisfied and compute the security levels using our updated parameters and modifications to the scheme. We first summarise the differences between our security analysis and [13]. Any other differences are negligible and due to precision variations in the attack costing script.

I.1. Gaussian Sampler Security

The statistical security of the Gaussian sampler used for sampling short vectors from lattice cosets in extraction and delegation to level ℓ is determined by the standard deviation of the sampler σ_ℓ and its relation to the Gram-Schmidt norm of the input basis. As this property of the basis is determined from the master key generation and any previous delegations, i.e. $\|\tilde{\mathbf{B}}\| \leq \sqrt{(\ell + 2)N} \cdot \sigma_\ell$, we can draw the following condition based on the relationship of the standard deviations at each level:

$$\sigma_\ell \geq \eta_\varepsilon(\mathbb{Z}) \sqrt{(\ell + 1)N} \cdot \sigma_{\ell - 1}, \qquad (17)$$

taking ε as $2^{-25.5}/(\ell + 1)N$ in order to make the KL-divergence of the sampler from the discrete Gaussian is at

TABLE 9. PERFORMANCE RESULTS (OP/S) FOR THE FALCON [15] (SCALED TO 4.2GHZ).

Set	Sec.	n	$\log_2 q$	KeyGen	Sign	Verify
FALCON-512	128	512	14	211.4	10861.7	51008.1
FALCON-1024	256	1024	14	66.5	5319.4	24926.1

TABLE 10. LATTE ESTIMATED COST OF MASTER KEY RECOVERY.

Set	β	Classical Security	Quantum Security
LATTE-1	974	301	275
LATTE-2	1501	455	414
LATTE-3	973	301	274
LATTE-4	1501	455	414

TABLE 11. LATTE ESTIMATED COST OF DELEGATED KEY RECOVERY.

Set	ℓ	β	Classical Security	Quantum Security
LATTE-1	1	1020	314	287
LATTE-2	1	1051	323	295
LATTE-3	1	1021	315	287
	2	388	130	119
LATTE-4	1	1051	323	295
	2	907	281	257

TABLE 12. LATTE ESTIMATED COST OF USER KEY RECOVERY.

Set	ℓ	β	Classical Security	Quantum Security
LATTE-1	1	829	258	236
LATTE-2	1	1863	560	510
LATTE-3	1	830	259	236
	2	334	114	105
LATTE-4	1	1864	561	510
	2	799	250	228

most 2^{-48}. However, we also require the sampled vectors to be short for the purposes of keeping the underlying lattice problem hard. Therefore, we can set σ_ℓ to be equal to right hand side of (17), where $\sigma_0 \approx 1.17\sqrt{q/(2N)}$. The quantity σ_0 is chosen to be this as it minimises the Gram-Schmidt norm of the master basis (resulting in short user private keys in the single-level IBE), as deduced in [10].

I.2. Decryption Failure

To protect against attacks which exploit random decryption failures, we must bound the error term incurred in the R-LWE encryption scheme. The probability that the error term is too large is derived in [13], based on the method of [51]. Essentially, the decryption failure rate cannot exceed $2^{-\lambda}$, where λ is the security level in bits of the scheme. For each parameter set and level, we can compute the probability of decryption failure, noting that our design consists of one less ephemeral private key than in [13], reducing the standard deviation τ of the Gaussian distribution of the coefficients of the error term d to $\tau = \sqrt{\sigma_e^2 + (\ell+1)N\sigma_\ell^2\sigma_e^2}$, marginally reducing the failure rate.

TABLE 13. COST OF PRIMAL MESSAGE RECOVERY ATTACK.

Set	m	β	Classical Security	Quantum Security
LATTE-1	1018	423	140	128
LATTE-2	1962	967	299	273
LATTE-3	998	232	84	78
LATTE-4	2037	561	180	165

TABLE 14. COST OF DUAL MESSAGE RECOVERY ATTACK.

Set	m	β	Classical Security	Quantum Security
LATTE-1	1039	422	140	128
LATTE-2	1974	964	298	272
LATTE-3	1005	232	84	78
LATTE-4	2101	560	180	165

I.3. Master Key Recovery

The security of the master key recovery depends upon the difficulty of finding the short vector $(\mathbf{g}, \mathbf{f})$ in the lattice, given the public NTRU basis. The attack is successful if the projection of the short vector onto the vector space spanned by the final β Gram-Schmidt vectors is shorter than the length of the $(2N-\beta+1)^{th}$ Gram-Schmidt vector. This corresponds to minimising block size β, for:

$$\sigma_0\sqrt{\beta} \leq GH(\beta)^{(2\beta-2N)/(\beta-1)} \cdot \det(\Lambda_0)^{1/2N}.$$

The minimum solutions to this inequality for different parameter sets is given in Table 10. The work required to find the shortest vector using this block size with the BKZ2.0 algorithm is also given.

I.4. Delegated Key Recovery

For delegated key recovery, the attacker must find a short sequence of vectors in $\Lambda_{\ell-1}$. This can reduce to solving SVP in the master lattice Λ_0 to find a vector of length $\sigma_\ell \cdot \sqrt{2N}$. Table 11 gives the minimum block size β required (as per below (18)) for a successful attack using BKZ2.0 and the classical and quantum cost of these attacks which depend on N and q.

$$\sigma_\ell \cdot \sqrt{2N} \leq GH(\beta)^{(2N)/(\beta-1)} \cdot \det(\Lambda_0)^{1/2N}. \qquad (18)$$

I.5. User Key Recovery

User key recovery requires finding a short solution to $\mathbf{t}_0 + \mathbf{t}_1 \cdot \mathbf{h} + \mathbf{t}_2 \cdot \mathbf{A}_1 + \cdots + \mathbf{t}_\ell \cdot \mathbf{A}_{\ell-1} = \mathbf{A}_\ell$, which reduces to solving the CVP in the master lattice Λ_0, of the form

TABLE 15. LATTE PARAMETERS.

Set	Security	N	q
LATTE-1	128	1024	$2^{24} - 2^{14} + 1$
LATTE-2	256	2048	$2^{25} - 2^{12} + 1$
LATTE-3	80	1024	$2^{36} - 2^{20} + 1$
LATTE-4	160	2048	$2^{38} - 2^{26} + 1$

$t_0 + t_1 \cdot A_0 = A_\ell$. It is enough to find a short (t_0, t_1) with length $\leq \sigma_\ell \cdot \sqrt{2(\ell+1)} \cdot \sqrt{2N}$. To do this, it is required to minimise (19) over β. Table 12 gives the minimum block size β required for a successful attack and the classical and quantum cost of these attacks.

$$\sigma_\ell \cdot \sqrt{2(\ell+1)} \cdot \sqrt{2N} \leq GH(\beta)^{(2N)/(\beta-1)} \cdot \det(\Lambda_0)^{1/2N}. \qquad (19)$$

I.6. Message Recovery

There are two attacks to consider for this. Message recovery depends on solving an extended version of R-LWE, which reduces to an instance of the primal-CVP or dual-SVP. In the primal-CVP attack, the ephemeral private keys are recovered via a close vector problem. In the dual-SVP attack, an attempt is made to distinguish the ciphertext elements from uniformly random polynomials in $\mathcal{R}_q$. In fact, it is enough for the attacker to recover one of the ephemeral private keys, $\mathbf{e}$ and so message recovery cost is not affected by hierarchical level, or by our redesign.

The minimal block size β needed for a successful attack, and the cost of these attacks are given in Tables 13 and 14, depending on (N, q). The code to populate Tables 13 and 14 is that used in [51]. By considering the cost of all attacks covered in this Section, the security levels in Table 15 could be derived.

I.7. Setting up Parameters

The parameter sets are given in Table 15. These are the parameters recommended in the original specification [13]. We have extended the security estimates from [13] to give them on a per-level basis. The security decreases as we move down the hierarchy. However, it turns out that each parameter set's security is determined by the message recovery capabilities, which remain constant down the levels. Therefore our parameter security conclusions match that of [13], and furthermore are not affected by our optimisations, as the message recovery attack is independent of the modified parameter ℓ.

Parameter sets LATTE-1 and 2 are only applicable to a single level, essentially an IBE rather than HIBE, version of the scheme. LATTE-3 and 4 can be used for up to two levels. The reason we cannot use these parameters beyond these levels is that the decryption failure rate exceeds the target security level. In fact, the failure rate is so high it renders the scheme completely insecure and also not suitable for use.

```
Input: $\mathbf{G}, \mathbf{D}'$.
Output: Tree $T$.
1:  function ffLDLB($\mathbf{G}, \mathbf{D}'$)
2:      if $n = 1$ then
3:          $(T.\text{value})_0 \leftarrow (\mu_{\mathbf{G}_{0,0},R}, \sigma^2_{\mathbf{G}_{0,0},R}, 0, 0)$.
4:          Output $\mu_{\text{leaf},R} = \mu_{(T.\text{value})_0,R}$, $\sigma_{\text{leaf},R} = \sigma_{(T.\text{value})_0,R}$.
5:          return
6:      end if
7:      for $j \in \{0, \ldots, n-1\}$ do
8:          $(\mathbf{D}_{0,0})_j \leftarrow (\mu_{(\mathbf{G}_{0,0})_j,R}, \sigma^2_{(\mathbf{G}_{0,0})_j,R}, 0, 0)$.
9:          $(\mathbf{L}_{1,0})_j \leftarrow \text{DivB}((\mathbf{G}_{1,0})_j, (\mathbf{D}_{0,0})_j)$.
10:         if $d = 2$ then
11:             if $n = N$ then
12:                 $(\mathbf{D}_{1,1})_j \leftarrow \text{DivB}(q^2, (\mathbf{D}_{0,0})_j)$.
13:             else
14:                 $x \leftarrow \text{MultB}(\mathbf{D}'_{2j}, \mathbf{D}'_{2j+1})$.
15:                 $(\mathbf{D}_{1,1})_j \leftarrow \text{DivB}(x, (\mathbf{D}_{0,0})_j)$.
16:             end if
17:         else if $d = 3$ then
18:             $x \leftarrow \text{DivB}(\text{AbsSqrB}((\mathbf{G}_{1,0})_j), (\mathbf{D}_{0,0})_j)$.
19:             $(\mathbf{D}_{1,1})_j \leftarrow \text{SubB}((\mathbf{G}_{1,1})_j, x)$.
20:             $x \leftarrow \text{MultB}((\mathbf{D}_{0,0})_j, (\mathbf{D}_{1,1})_j)$.
21:             $(\mathbf{D}_{2,2})_j \leftarrow \text{DivB}(q^2, x)$.
22:             $(\mathbf{L}_{2,0})_j \leftarrow \text{DivB}((\mathbf{G}_{2,0})_j, (\mathbf{D}_{0,0})_j)$.
23:             $x \leftarrow \text{MultB}((\mathbf{G}_{2,0})_j, (\mathbf{L}_{1,0})^*_j)$.
24:             $y \leftarrow \text{SubB}((\mathbf{G}_{2,1})_j, x)$.
25:             $(\mathbf{L}_{2,1})_j \leftarrow \text{DivB}(y, (\mathbf{D}_{1,1})_j)$.
26:         end if
27:     end for
28:     $T.\text{value} \leftarrow \mathbf{L}$.
29:     for $i \in \{0, \ldots, d-1\}$ do
30:         $\mathbf{d}_0, \mathbf{d}_1 \leftarrow \text{splitfftB}(\mathbf{D}_{i,i}, n)$.
31:         $\mathbf{G}' = \begin{pmatrix} \mathbf{d}_0 & \mathbf{d}_1 \\ \mathbf{d}_1^* & \mathbf{d}_0 \end{pmatrix}$.
32:         $T.\text{child}_i \leftarrow \text{ffLDLB}(\mathbf{G}', \mathbf{D}_{i,i})$.
33:     end for
34:     return $T$.
35: end function
```

Figure 9. The ffLDLB algorithm based on statistical model.

Appendix J. Algorithms in the Statistical Model

We present the supplementary algorithms (Figure 9–14) of our statistical model, including ffLDLB, ffSamplingB, splitfftB, mergefftB, FFTB, and FFTInvB.

```
Input: $\mathbf{t} = (\mathbf{t}_0, \dots, \mathbf{t}_\ell)$ in FFT format, tree $T$.
Output: $\mathbf{z} = (\mathbf{z}_0, \dots, \mathbf{z}_\ell)$ in FFT format.
 1: function ffSamplingB($\mathbf{t}, T$)
 2:     if $n = 1$ then
 3:         $z_0 \leftarrow (\mu_{(\mathbf{t}_0)_0,R}, 0, 0, 0)$.
 4:         $z_1 \leftarrow (\mu_{(\mathbf{t}_1)_0,R}, 0, 0, 0)$.
 5:         Output $\Delta'_{t_0} = \sigma_{(\mathbf{t}_0)_0,R}/(\sigma_\ell/\sqrt{\mu_{(T.\text{value})_0,R}})$.
 6:         Output $\Delta'_{t_1} = \sigma_{(\mathbf{t}_1)_0,R}/(\sigma_\ell/\sqrt{\mu_{(T.\text{value})_0,R}})$.
 7:         return $\mathbf{z} = (z_0, z_1)$.
 8:     else
 9:         $m \leftarrow$ number of children of $T$.
10:         for $j \leftarrow m-1, \dots, 0$ do
11:             $T_j \leftarrow j$-th child of $T$.
12:             $\mathbf{t}'_j \leftarrow \mathbf{t}_j$.
13:             for $i \leftarrow j+1, \dots, m-1$ do
14:                 for $k \leftarrow 0, \dots, n-1$ do
15:                     $x \leftarrow \mathrm{SubB}((\mathbf{t}_i)_k, (\mathbf{z}_i)_k)$.
16:                     $y \leftarrow \mathrm{MultB}(x, (T.\text{value}_{i,j})_k)$.
17:                     $(\mathbf{t}'_j)_k \leftarrow \mathrm{AddB}((\mathbf{t}'_j)_k, y)$.
18:                 end for
19:             end for
20:             $\mathbf{f}_0, \mathbf{f}_1 \leftarrow$ splitfftB($\mathbf{t}'_j, n$).
21:             $\mathbf{z}'_{j,0}, \mathbf{z}'_{j,1} \leftarrow$ ffSamplingB($(\mathbf{f}_0, \mathbf{f}_1), T_j$).
22:             $\mathbf{z}_j \leftarrow$ mergefftB($\mathbf{z}'_{j,0}, \mathbf{z}'_{j,1}, n$).
23:         end for
24:         return $\mathbf{z}$.
25:     end if
26: end function
```

Figure 10. The ffSamplingB algorithm based on statistical model.

```
Input: $\mathbf{f}_0, \mathbf{f}_1, n$.
Output: $\mathbf{a}$.
 1: function mergefftB($\mathbf{f}_0, \mathbf{f}_1, n$)
 2:     $\mathbf{a}_0 \leftarrow (\mu_{(\mathbf{f}_0)_0,R}, \sigma^2_{(\mathbf{f}_0)_0,R}, \mu_{(\mathbf{f}_1)_0,R}, \sigma^2_{(\mathbf{f}_1)_0,R})$.
 3:     for $k \leftarrow 0, \dots, n/2-1$ do
 4:         $u \leftarrow \mathrm{MultB}((\mathbf{f}_1)_k, \omega^{\mathrm{bitrev}(n/2+k)})$.
 5:         $\mathbf{a}_{2k} \leftarrow \mathrm{AddB}((\mathbf{f}_0)_k, u)$.
 6:         $\mathbf{a}_{2k+1} \leftarrow \mathrm{SubB}((\mathbf{f}_0)_k, u)$.
 7:     end for
 8:     return $\mathbf{a}$.
 9: end function
```

Figure 12. The mergefftB algorithm based on statistical model.

```
Input: $\mathbf{a}, n$.
Output: $\mathbf{f}_0, \mathbf{f}_1$.
 1: function splitfftB($\mathbf{a}, n$)
 2:     $(\mathbf{f}_0)_0 \leftarrow (\mu_{\mathbf{a}_0,R}, \sigma^2_{\mathbf{a}_0,R}, 0, 0)$.
 3:     $(\mathbf{f}_1)_0 \leftarrow (\mu_{\mathbf{a}_0,I}, \sigma^2_{\mathbf{a}_0,I}, 0, 0)$.
 4:     for $k \leftarrow 0, \dots, n/2-1$ do
 5:         $(\mathbf{f}_0)_k \leftarrow \mathrm{MultB}(1/2, \mathrm{AddB}(\mathbf{a}_{2k}, \mathbf{a}_{2k+1}))$.
 6:         $x \leftarrow \mathrm{SubB}(\mathbf{a}_{2k}, \mathbf{a}_{2k+1})$.
 7:         $y \leftarrow \mathrm{MultB}(x, \omega^{-\mathrm{bitrev}(n/2+k)})$.
 8:         $(\mathbf{f}_1)_k \leftarrow \mathrm{MultB}(1/2, y)$.
 9:     end for
10:     return $\mathbf{f}_0, \mathbf{f}_1$.
11: end function
```

Figure 11. The splitfftB algorithm based on statistical model.

```
Input: $\mathbf{a}$.
 1: function FFTB($\mathbf{a}$)
 2:     $m = 1$.
 3:     $t = n$.
 4:     while $m < n$ do
 5:         $t \leftarrow t/2$.
 6:         for $i = 0$ to $m-1$ do
 7:             $j_1 = 2it$.
 8:             $j_2 = j_1 + t - 1$.
 9:             for $j = j_1$ to $j_2$ do
10:                 $u \leftarrow \mathbf{a}_j$.
11:                 $v \leftarrow \mathrm{MultB}(\mathbf{a}_{j+t}, \omega^{\mathrm{bitrev}(m+i)})$.
12:                 $\mathbf{a}_j \leftarrow \mathrm{AddB}(u, v)$.
13:                 $\mathbf{a}_{j+t} \leftarrow \mathrm{SubB}(u, v)$.
14:             end for
15:         end for
16:         $m \leftarrow 2m$.
17:     end while
18: end function
```

Figure 13. The FFTB algorithm based on statistical model.

```
Input: $\mathbf{a}$.
function FFTInvB($\mathbf{a}$)
    $t = 1$.
    $h = n$.
    $m = n$.
    while $m > 1$ do
        $j_1 = 0$.
        $h \leftarrow h/2$.
        for $i = 0$ to $h - 1$ do
            $j_2 = j_1 + t - 1$.
            for $j = j_1$ to $j_2$ do
                $u \leftarrow \mathrm{AddB}(\mathbf{a}_j, \mathbf{a}_{j+t})$.
                $x \leftarrow \mathrm{SubB}(\mathbf{a}_j, \mathbf{a}_{j+t})$.
                $\mathbf{a}_j \leftarrow u$.
                $\mathbf{a}_{j+t} \leftarrow \mathrm{MultB}(x, \omega^{-\mathrm{bitrev}(h+i)})$.
            end for
            $j_1 \leftarrow j_1 + 2t$.
        end for
        $t \leftarrow 2t$.
    end while
    for $i = 0$ to $n - 1$ do
        $\mathbf{a}_i \leftarrow \mathrm{MultB}(1/n, \mathbf{a}_i)$.
    end for
end function
```

Figure 14. The FFTInvB algorithm based on statistical model.