Supplementary Material

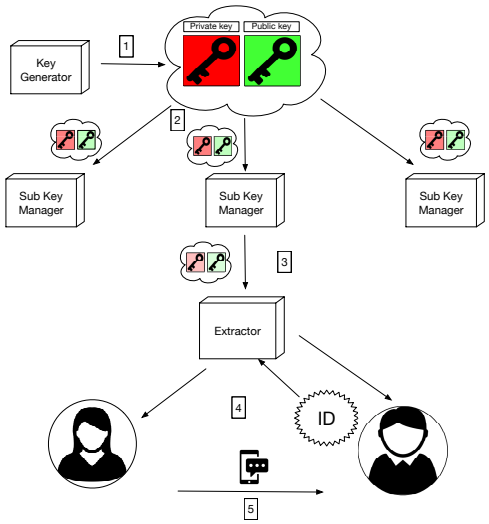

Figure 1. A 2-level HIBE scheme.

1. 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 1 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 1 indicates the notational practices used to identify each level of the hierarchy.

TABLE 1. 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

2. Cramer's Rule

Cramer's rule [1] 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 $\langle \mathbf{s}_{\ell+1}, \mathbf{s}_i \rangle$.

3. 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.

3.1. 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 $4N \cdot \kappa$.

3.2. 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 $(\ell+2) \cdot (\ell+2) \cdot N \cdot \kappa$.

3.3. 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 $\ell \cdot N \cdot \kappa$.

3.4. 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 $(\ell+1) \cdot N \cdot \kappa + 256$.

3.5. Comparison to FALCON

After adopting the NTRUSolve and lattice Gaussian sampling procedures from FALCON [2], 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 2. 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, [3]. (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 [2], 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.

4. 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 [4], 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 [4]. Any other differences are negligible and due to precision variations in the attack costing script.

4.1. 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:

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

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 most 2^{-48}. However, we also require the sampled vectors to

TABLE 2. PERFORMANCE RESULTS (OP/S) FOR THE FALCON [2] (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 3. 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 4. 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 5. 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

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 (1), 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 [5].

4.2. 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 [4], based on the method of [6]. 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 [4], 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 6. 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 7. 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

4.3. 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 3. The work required to find the shortest vector using this block size with the BKZ2.0 algorithm is also given.

4.4. 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 4 gives the minimum block size β required (as per below (2)) for a successful attack using BKZ2.0 and the classical and quantum cost of these attacks which depend on N and q.

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

4.5. 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 8. 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 (3) over β. Table 5 gives the minimum block size β required for a successful attack and the classical and quantum cost of these attacks.

$$\sigma_\ell \cdot \sqrt{2(\ell+1)} \cdot \sqrt{2N} \leq GH(\beta)^{(2N)/(\beta-1)} \cdot \det(\Lambda_0)^{1/2N}. \quad (3)$$

4.6. 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 6 and 7, depending on (N, q). The code to populate Tables 6 and 7 is that used in [6]. By considering the cost of all attacks covered in this Section, the security levels in Table 8 could be derived.

4.7. Setting up Parameters

The parameter sets are given in Table 8. These are the parameters recommended in the original specification [4]. We have extended the security estimates from [4] 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 [4], 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.

function ffLDLB($\mathbf{G}, \mathbf{D}'$)
  if $n = 1$ **then**
    $(T.\text{value})_0 \leftarrow (\mu_{\mathbf{G}_{0,0},R}, \sigma^2_{\mathbf{G}_{0,0},R}, 0, 0)$.
    Output $\mu_{\text{leaf},R} = \mu_{(T.\text{value})_0,R}$, $\sigma_{\text{leaf},R} = \sigma_{(T.\text{value})_0,R}$.
    return
  end if
  for $j \in \{0, \ldots, n-1\}$ **do**
    $(\mathbf{D}_{0,0})_j \leftarrow (\mu_{(\mathbf{G}_{0,0})_j,R}, \sigma^2_{(\mathbf{G}_{0,0})_j,R}, 0, 0)$.
    $(\mathbf{L}_{1,0})_j \leftarrow \text{DivB}((\mathbf{G}_{1,0})_j, (\mathbf{D}_{0,0})_j)$.
    if $d = 2$ **then**
      if $n = N$ **then**
        $(\mathbf{D}_{1,1})_j \leftarrow \text{DivB}(q^2, (\mathbf{D}_{0,0})_j)$.
      else
        $x \leftarrow \text{MultB}(\mathbf{D}'_{2j}, \mathbf{D}'_{2j+1})$.
        $(\mathbf{D}_{1,1})_j \leftarrow \text{DivB}(x, (\mathbf{D}_{0,0})_j)$.
      end if
    else if $d = 3$ **then**
      $x \leftarrow \text{DivB}(\text{AbsSqrB}((\mathbf{G}_{1,0})_j), (\mathbf{D}_{0,0})_j)$.
      $(\mathbf{D}_{1,1})_j \leftarrow \text{SubB}((\mathbf{G}_{1,1})_j, x)$.
      $x \leftarrow \text{MultB}((\mathbf{D}_{0,0})_j, (\mathbf{D}_{1,1})_j)$.
      $(\mathbf{D}_{2,2})_j \leftarrow \text{DivB}(q^2, x)$.
      $(\mathbf{L}_{2,0})_j \leftarrow \text{DivB}((\mathbf{G}_{2,0})_j, (\mathbf{D}_{0,0})_j)$.
      $x \leftarrow \text{MultB}((\mathbf{G}_{2,0})_j, (\mathbf{L}_{1,0})^*_j)$.
      $y \leftarrow \text{SubB}((\mathbf{G}_{2,1})_j, x)$.
      $(\mathbf{L}_{2,1})_j \leftarrow \text{DivB}(y, (\mathbf{D}_{1,1})_j)$.
    end if
  end for
  $T.\text{value} \leftarrow \mathbf{L}$.
  for $i \in \{0, \ldots, d-1\}$ **do**
    $\mathbf{d}_0, \mathbf{d}_1 \leftarrow \text{splitfftB}(\mathbf{D}_{i,i}, n)$.
    $\mathbf{G}' = \begin{pmatrix} \mathbf{d}_0 & \mathbf{d}_1 \\ \mathbf{d}_1^* & \mathbf{d}_0 \end{pmatrix}$.
    $T.\text{child}_i \leftarrow \text{ffLDLB}(\mathbf{G}', \mathbf{D}_{i,i})$.
  end for
  return T.
end function

Figure 2. The ffLDLB algorithm based on statistical model.

5. Algorithms in the Statistical Model

We present the supplementary algorithms (Figure 2–7) of our statistical model, including ffLDLB, ffSamplingB, splitfftB, mergefftB, FFTB, and FFTInvB.

References

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

[2] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon:

Input: $\mathbf{t} = (\mathbf{t}_0, \ldots, \mathbf{t}_\ell)$ in FFT format, tree T.
Output: $\mathbf{z} = (\mathbf{z}_0, \ldots, \mathbf{z}_\ell)$ in FFT format.

function ffSamplingB($\mathbf{t}, T$)
  if $n = 1$ **then**
    $z_0 \leftarrow (\mu_{(\mathbf{t}_0)_0,R}, 0, 0, 0)$.
    $z_1 \leftarrow (\mu_{(\mathbf{t}_1)_0,R}, 0, 0, 0)$.
    Output $\Delta'_{t_0} = \sigma_{(\mathbf{t}_0)_0,R} / (\sigma_\ell / \sqrt{\mu_{(T.\text{value})_0,R}})$.
    Output $\Delta'_{t_1} = \sigma_{(\mathbf{t}_1)_0,R} / (\sigma_\ell / \sqrt{\mu_{(T.\text{value})_0,R}})$.
    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.
      $\mathbf{t}'_j \leftarrow \mathbf{t}_j$.
      for $i \leftarrow j+1, \ldots, m-1$ **do**
        for $k \leftarrow 0, \ldots, n-1$ **do**
          $x \leftarrow \text{SubB}((\mathbf{t}_i)_k, (\mathbf{z}_i)_k)$.
          $y \leftarrow \text{MultB}(x, (T.\text{value}_{i,j})_k)$.
          $(\mathbf{t}'_j)_k \leftarrow \text{AddB}((\mathbf{t}'_j)_k, y)$.
        end for
      end for
      $\mathbf{f}_0, \mathbf{f}_1 \leftarrow \text{splitfftB}(\mathbf{t}'_j, n)$.
      $\mathbf{z}'_{j,0}, \mathbf{z}'_{j,1} \leftarrow \text{ffSamplingB}((\mathbf{f}_0, \mathbf{f}_1), T_j)$.
      $\mathbf{z}_j \leftarrow \text{mergefftB}(\mathbf{z}'_{j,0}, \mathbf{z}'_{j,1}, n)$.
    end for
    return $\mathbf{z}$.
  end if
end function

Figure 3. The ffSamplingB algorithm based on statistical model.

Input: $\mathbf{a}, n$.
Output: $\mathbf{f}_0, \mathbf{f}_1$.

function splitfftB($\mathbf{a}, n$)
  $(\mathbf{f}_0)_0 \leftarrow (\mu_{\mathbf{a}_0,R}, \sigma^2_{\mathbf{a}_0,R}, 0, 0)$.
  $(\mathbf{f}_1)_0 \leftarrow (\mu_{\mathbf{a}_0,I}, \sigma^2_{\mathbf{a}_0,I}, 0, 0)$.
  for $k \leftarrow 0, \ldots, n/2-1$ **do**
    $(\mathbf{f}_0)_k \leftarrow \text{MultB}(1/2, \text{AddB}(\mathbf{a}_{2k}, \mathbf{a}_{2k+1}))$.
    $x \leftarrow \text{SubB}(\mathbf{a}_{2k}, \mathbf{a}_{2k+1})$.
    $y \leftarrow \text{MultB}(x, \omega^{-\text{bitrev}(n/2+k)})$.
    $(\mathbf{f}_1)_k \leftarrow \text{MultB}(1/2, y)$.
  end for
  return $\mathbf{f}_0, \mathbf{f}_1$.
end function

Figure 4. The splitfftB algorithm based on statistical model.

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.

[3] 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.

Input: $\mathbf{f}_0, \mathbf{f}_1, n$.
Output: $\mathbf{a}$.

function mergefftB($\mathbf{f}_0, \mathbf{f}_1, n$)
  $\mathbf{a}_0 \leftarrow (\mu_{(\mathbf{f}_0)_0,R}, \sigma^2_{(\mathbf{f}_0)_0,R}, \mu_{(\mathbf{f}_1)_0,R}, \sigma^2_{(\mathbf{f}_1)_0,R})$.
  for $k \leftarrow 0, \ldots, n/2-1$ **do**
    $u \leftarrow \text{MultB}((\mathbf{f}_1)_k, \omega^{\text{bitrev}(n/2+k)})$.
    $\mathbf{a}_{2k} \leftarrow \text{AddB}((\mathbf{f}_0)_k, u)$.
    $\mathbf{a}_{2k+1} \leftarrow \text{SubB}((\mathbf{f}_0)_k, u)$.
  end for
  return $\mathbf{a}$.
end function

Figure 5. The mergefftB algorithm based on statistical model.

Input: $\mathbf{a}$.

function FFTB($\mathbf{a}$)
  $m = 1$.
  $t = n$.
  while $m < n$ **do**
    $t \leftarrow t/2$.
    for $i = 0$ **to** $m-1$ **do**
      $j_1 = 2it$.
      $j_2 = j_1 + t - 1$.
      for $j = j_1$ **to** j_2 **do**
        $u \leftarrow \mathbf{a}_j$.
        $v \leftarrow \text{MultB}(\mathbf{a}_{j+t}, \omega^{\text{bitrev}(m+i)})$.
        $\mathbf{a}_j \leftarrow \text{AddB}(u, v)$.
        $\mathbf{a}_{j+t} \leftarrow \text{SubB}(u, v)$.
      end for
    end for
    $m \leftarrow 2m$.
  end while
end function

Figure 6. The FFTB algorithm based on statistical model.

[4] "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.

[5] 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.

[6] 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.

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 7. The FFTInvB algorithm based on statistical model.