LOTUS*: Algorithm Specifications and Supporting Documentation

Le Trieu Phong[1], Takuya Hayashi[1,2], Yoshinori Aono[1], Shiho Moriai[1]

[1] National Institute of Information and Communications Technology (NICT), Tokyo, Japan
[2] Kobe University, Kobe, Japan
{phong, takuya.hayashi.2, aono, shiho.moriai}@nict.go.jp, t-hayashi@eedept.kobe-u.ac.jp

Version date: 30 November 2017

Abstract

In this document submitted to the NIST's Post-Quantum Cryptography Standardization Process, we present a lattice-based cryptosystem named LOTUS. Specifically, we describe a PKE scheme and a KEM scheme called LOTUS-PKE and LOTUS-KEM respectively. Both are proven to be IND-CCA2-secure *solely* under the standard LWE assumption, in the random oracle model. This security notion is a security requirement described in the NIST's Call For Proposals.

* **L**earning with err**O**rs based encryption with chosen ciphertex**T** sec**U**rity for po**S**t quantum era.

Table of Contents

1 Preliminaries

1.1 Notations

This section defines a few notations used throughout this document. The symbols $\|$ and $\oplus$ stand for the concatenation and XOR of bit strings. Let $\mathbb{Z}$ and $\mathbb{R}$ be the set of integer and real numbers respectively. Let $\mathbb{Z}_q \subset (-q/2, q/2]$ be the set of integers centered modulus q. Taking modulus q is via the formula

$$x \bmod q = x - \lfloor x/q \rceil q$$

for $x \in \mathbb{R}$ in which $\lfloor x/q \rceil$ is the rounding of x/q to the nearest integer in the interval $(x/q - 1/2, x/q + 1/2]$. Let $\mathbb{Z}_q^{a \times b}$ be the set of matrices of size $a \times b$ whose elements are from $\mathbb{Z}_q$.

We use $\mathbb{Z}_{(0,s)}$ to denote the discrete Gaussian distribution of mean 0 and standard deviation s, and $\mathbb{Z}_{(0,s)}^{a \times b}$ as the set of all matrices of size $a \times b$ whose elements are sampled from $\mathbb{Z}_{(0,s)}$.

1.2 Public key encryption

PKE. A public key encryption (PKE) scheme consists of key generation algorithm KeyGen_{pke}, encryption algorithm Enc_{pke}, and decryption algorithm Dec_{pke}. $\mathsf{KeyGen}_{pke}(\lambda)$ with security parameter λ outputs public key pk and secret key sk, or $(pk, sk) \leftarrow \mathsf{KeyGen}_{pke}(\lambda)$ for short. The algorithm $\mathsf{Enc}_{pke}(pk, M)$ encrypting a message M returns a ciphertext C, or $C \leftarrow \mathsf{Enc}_{pke}(pk, M)$ for short. Correctness holds if $\mathsf{Dec}_{pke}(sk, C) = M$.

IND-CCA2 security of PKE. IND-CCA2 is the standard security notion for public key encryption. To define the security of PKE, consider the following game with adversary $\mathcal{A}$. First, the key pair (pk, sk) is generated by running $\mathsf{KeyGen}_{pke}(\lambda)$ and pk is given to $\mathcal{A}$. In the so-called find stage, $\mathcal{A}$ can query any C of its choice to oracle $\mathsf{Dec}_{pke}(sk, \cdot)$.

Then $\mathcal{A}$ invokes a challenge oracle with two messages of equal length (M_0, M_1), who takes $b \in \{0, 1\}$ uniformly at random and computes $C^* \leftarrow \mathsf{Enc}_{pke}(pk, M_b)$. The oracle returns the challenge ciphertext C^*.

After that, in the guess stage, $\mathcal{A}$ can again access to the oracle $\mathsf{Dec}_{pke}(sk, \cdot)$, but is not allowed to query C^* to the decryption oracle. Finally, $\mathcal{A}$ returns b' as a guess of the hidden b.

The PKE is IND-CCA2-secure if the advantage

$$\mathbf{Adv}^{\mathrm{ind-cca2}}_{\mathrm{PKE},\mathcal{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible in λ for all poly-time adversaries $\mathcal{A}$.

1.3 Key encapsulation mechanisms

KEM. A key encapsulation mechanism (KEM) consists of key generation KeyGen_{kem}, encapsulation Enc_{kem}, and decapsulation Dec_{kem} algorithms. $\mathsf{KeyGen}_{kem}(\lambda)$ with security parameter λ outputs public key pk and secret key sk. The algorithm $\mathsf{Enc}_{kem}(pk)$ returns a pair (C, K). Correctness holds if $\mathsf{Dec}_{kem}(sk, C) = K$.

IND-CCA2 security of KEM. To define the security of KEM, consider the following game with adversary $\mathcal{A}$. First, $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and pk is given to $\mathcal{A}$. In the so-called find stage, $\mathcal{A}$ can query any C of its choice to oracle $\mathsf{Dec}(sk, \cdot)$.

Then $\mathcal{A}$ invokes a challenge oracle who computes $(C^*, K^*) \leftarrow \mathsf{Enc}(pk)$, then takes K_* randomly and of equal size of K^*, and chooses $b \in \{0, 1\}$ uniformly at random. The oracle returns challenge pair $(C^*, K(b))$ in which $K(0) = K^*$ and $K(1) = K_*$.

After that, in the guess stage, $\mathcal{A}$ can again access to the oracle $\mathsf{Dec}(sk, \cdot)$, but is not allowed to query C^* to the decapsulation oracle. Finally, $\mathcal{A}$ returns b' as a guess of the hidden b.

The KEM is IND-CCA2-secure if the advantage

$$\mathbf{Adv}^{\mathrm{ind-cca2}}_{\mathrm{KEM},\mathcal{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible in λ for all poly-time adversaries $\mathcal{A}$.

1.4 The Learning with Errors (LWE) assumption

The Learning with Errors (LWE) assumption [37] ensures the security of our proposed LOTUS-PKE and LOTUS-KEM schemes, and is recalled here.

The symbol $\xleftarrow{\$}$ is for randomly sampling from the uniform distribution, while $\xleftarrow{\mathrm{g}}$ for randomly sampling from the discrete Gaussian distribution. Related to the decision LWE assumption LWE(n, s, q), where n, s, q depend on the security parameter λ, consider matrix $A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, vectors $r \xleftarrow{\$} \mathbb{Z}_q^{m \times 1}$, $x \xleftarrow{\mathrm{g}} \mathbb{Z}_{(0,s)}^{n \times 1}$, $e \xleftarrow{\mathrm{g}} \mathbb{Z}_{(0,s)}^{m \times 1}$. Then vector $Ax + e$ is computed over $\mathbb{Z}_q$. Define the following advantage of a poly-time probabilistic algorithm $\mathcal{D}$:

$$\mathbf{Adv}^{\mathrm{LWE}(n,s,q)}_{\mathcal{D}}(\lambda) = \Big| \Pr[\mathcal{D}(A, Ax + e) \to 1] - \Pr[\mathcal{D}(A, r) \to 1] \Big|.$$

The decision LWE assumption asserts that $\mathbf{Adv}^{\mathrm{LWE}(n,s,q)}_{\mathcal{D}}(\lambda)$ is negligible as a function of λ. Also note that, originally, x is chosen randomly from $\mathbb{Z}_q^{n \times 1}$ in [37]. However, as showed in [8, 31], it is possible to take $x \xleftarrow{\mathrm{g}} \mathbb{Z}_{(0,s)}^{n \times 1}$ without weakening the assumption as we do here. In addition, the computational LWE problem is defined and analysed in Section A.1.3. Certainly, if the computational LWE problem is solved, then so is the decisional one. Therefore, a parameter set for the computational LWE problem implies a parameter set for the decisional LWE problem.

2 A Known IND-CPA-secure PKE

The following PKE scheme is from Lindner-Peikert [27], which is used as a component in the construction of LOTUS-PKE and LOTUS-KEM.

Key generation $\mathsf{KeyGen}^{cpa}_{pke}(pp, \lambda)$**:** Choose positive integers q, n, l, and take matrix $A \in \mathbb{Z}_q^{n\times n}$ uniformly at random. Fix deviation $s \in \mathbb{R}$ and take Gaussian noise matrices $R, S \in \mathbb{Z}^{n\times l}_{(0,s)}$ at random. The public key is $pk = (P, A, q, n, l, s)$ for $P = R - AS \in \mathbb{Z}_q^{n\times l}$, and the secret key is $sk = S$. Here, l is the message length in bits, while n is the key dimension. Return (pk, sk).

Encryption $\mathsf{Enc}^{cpa}_{pk}(M; \mathsf{randomness})$**:** To encrypt $M \in \{0,1\}^l$, use $\mathsf{randomness}$ to take Gaussian noise vectors $e_1, e_2 \in \mathbb{Z}^{1\times n}_{(0,s)}$, and $e_3 \in \mathbb{Z}^{1\times l}_{(0,s)}$, and return ciphertext $c = (c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$ where

$$c_1 = e_1A + e_2 \in \mathbb{Z}_q^{1\times n}, c_2 = e_1P + e_3 + M \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^{1\times l}.$$

Decryption $\mathsf{Dec}^{cpa}_{sk}(c)$**:** To decrypt $c = (c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$ by secret key S, compute $\overline{M} = c_1S + c_2 \in \mathbb{Z}_q^l$. Let $\overline{M} = (\overline{M}_1, \ldots, \overline{M}_l)$. If $\overline{M}_i \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor) \subset \mathbb{Z}_q$, let $M_i = 0$; otherwise $M_i = 1$. Return $M = M_0 \parallel \cdots \parallel M_l \in \{0,1\}^l$.

Theorem 1 (Lindner-Peikert [27]) *Given the public key, the above PKE scheme has pseudo-random ciphertexts under the LWE assumption.*

The above scheme of Lindner-Peikert is not IND-CCA2-secure. Indeed, given a ciphertext $c = (c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$, the decryption of the modified ciphertext $c' = (c_1, c_2') \in \mathbb{Z}_q^{1\times(n+l)}$ where $c_2' = c_2 + e'$ for vector $e' = (1, 0, \ldots, 0) \in \mathbb{Z}^{1\times l}$ is

$$\begin{aligned} c_1S + c_2' &= (e_1A + e_2)S + e_1P + (e_3 + e') + M \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ &= (e_1A + e_2)S + e_1(R - AS) + (e_3 + e') + M \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ &= \underbrace{e_2S + e_1R + (e_3 + e')}_{\text{small noise}} + M \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^l. \end{aligned}$$

Therefore, the message M inside the original ciphertext $c = (c_1, c_2)$ can be recovered with non-negligible probability by the decryption of $c' = (c_1, c_2')$.

3 Algorithm Specifications

3.1 Design rationale: security first, cost second

We choose to emphasize on security in our design of LOTUS, so that we make use of the standard Learning with Errors (LWE) assumption proposed by Regev [37] in 2005. There are

several reasons to believe that the LWE problem is hard, as listed in [38]: the best algorithm solving LWE runs in exponential time of the LWE dimension and even quantum algorithms do not seem to help; LWE is an extension of the Learning Parity with Noise (LPN) problem, which is also extensively studied; and LWE hardness is linked to the worst-case hardness of standard lattice problems [12]. In addition, the practical hardness of LWE is extensively studied in the literature [2–5, 27, 28], on which we revisit and provide enhancements in Appendix A.

Based on the LWE assumption, LOTUS-PKE has been already appeared in Aono et al. [5, Section 4] (by removing the proxy re-encryption part) and LOTUS-KEM is the trivial adaptation of LOTUS-PKE. Both make use of the public key encryption scheme in Lindner-Peikert [27] (recalled in Section 2) in combination with the Fujisaki-Okamoto transformation in [19]. Thanks to this design rationale, both LOTUS-PKE and LOTUS-KEM are expected to achieve IND-CCA2 security. See also Section 5 for details.

It may be possible to further enhance the efficiency of LOTUS. For example, the technique of reconciliation [36] may help reduce the ciphertext sizes a little. For the sake of simplicity in the design, in this version of LOTUS we do not employ the technique.

3.2 Knuth-Yao discrete Gaussian sampling

For generating discrete Gaussian noises, we employ the Knuth-Yao algorithm [25]. The motivation of Knuth-Yao algorithm is to sample from a finite discrete set X with using as small number of random bits as possible. The method consists of two parts: tree construction as a preprocess, and sampling. Suppose for each element $s \in X$, the generating probability $p(s)$ is given and these probabilities are normalized so that $\sum_{s \in X} p(s) = 1$.

The preprocessing step constructs the binary tree, called a discrete distribution generating (DGG) tree, on which randomly searching outputs s with probability $p(s)$. Suppose we have L-binary digits numbers $\overline{p(s)}$ that approximates $p(s)$: $|\overline{p(s)} - p(s)| \leq 2^{-L}$. For $\overline{p(s)}$ for $s \in X$, construct a binary matrix called a probability matrix, whose each row is corresponding to the binary representation of $\overline{p(s)}$, and each column is corresponding to the k-th digit of $\overline{p(s)}$. As the step 0 of the DGG tree construction method, consider the tree having only one node (root) without label. Then, at step k, for all non-labeled nodes at depth $k-1$, add two children and set the label corresponding to s such that the k-th digit of $p(s)$ is `1`. Since all $\overline{p(s)}$ are in L bits after the point, it finishes at the step L and all leaves are labeled.

In the sampling step, the algorithm starts at the root and moves to its left/right children with both probability 1/2, until it arrives the leaf, i.e., labeled node, and then it outputs the label of last node. By construction, the probability that the algorithm touches a depth k node is always 2^{-k}, whenever it finishes the search or not. Thus, for each element $s \in X$, the probability that it was outputted at depth k is 0 (resp. 2^{-k}) if the k-th bit of $p(s)$ is `0` (resp. `1`). This fact means the total probability of s is indeed $p(s)$. This is an outline of the standard Knuth-Yao algorithm (KY).

To sample from the discrete Gaussian $\mathbb{Z}_{(0,s)}$, we need to compute the probability that an integer $i \in \mathbb{Z}$ is generated. By definition, it can be easily computed within sufficiently high

accuracy:

$$q(x) = \frac{\exp(-\pi x^2/s^2)}{1 + 2\sum_{i=1}^{\infty} \exp(-\pi i^2/s^2)}.$$

To determine a finite set $X \subset \mathbb{Z}$ for a precision parameter L, we use Lemma 4.4.1 of [29]:

Lemma 1 *For any $x_0 > 0$,*

$$\Pr\left[|x| > \frac{sx_0}{\sqrt{2\pi}} \;\middle|\; x \xleftarrow{g} \mathbb{Z}_{(0,s)}\right] \leq 2\exp\left(\frac{-x_0^2}{2}\right).$$

For the smallest x_0 such that

$$2\exp\left(\frac{-x_0^2}{2}\right) \leq 2^{-L}$$

set $X = \{0, 1, \ldots, x_0\}$, then corresponding probabilities are $p(0) = \overline{q(0)}$ and $p(x) = \overline{2 \cdot q(x)}$ for $x \geq 1$. The desired distribution can be obtained by changing the sign of the output with probability 1/2. The statistical distance between the ideal discrete Gaussian and the above algorithm is

$$\begin{aligned}\sum_{x=-\infty}^{\infty} |\Pr[x \xleftarrow{g} \mathbb{Z}_{(0,s)}] - \Pr[x \leftarrow \mathsf{KY}]| &= \sum_{|x|>x_0} |\Pr[x \xleftarrow{g} \mathbb{Z}_{(0,s)}]| \\ &\quad + \sum_{x=-x_0}^{x_0} |\Pr[x \xleftarrow{g} \mathbb{Z}_{(0,s)}] - \Pr[x \leftarrow \mathsf{KY}]| \\ &< 3 \cdot 2^{-L} + (2x_0 + 1)2^{-L}.\end{aligned}$$

In our implementation, we set $L = 262$ to argue the 256-bit security of the scheme. Moreover, to speed up the algorithm, we employ following improvements:

- **Online tree construction [39]**: One of the bottlenecks of the Knuth-Yao sampling algorithm is the relatively large memory usage because of the DGG tree. Instead of pre-constructing the tree, one can construct the tree at the sampling step to reduce the memory usage.
  Let n_j be the number of unlabeled nodes and h_j be the number of leaf nodes at the depth j. Suppose the current node at the sampling step is the i-th unlabeled node, counted from the most left side, and let $d_j = n_j - i$ be the distance from the current node to the most right side unlabeled node. Then there are $2d_j + 2$ child nodes between the i-th node and the n-th node, and the distance from the left (resp. right) child node of the i-th node to the most right side unlabeled node is $d_{j+1} = 2d_j + 1 - h_{j+1}$ (resp. $d_{j+1} = 2d_j - h_{j+1}$). The child node is a leaf node if $d_{j+1} < 0$, then scan $j+1$-th column of the probability matrix from the top until an index k s.t. $d_{j+1} + \sum_{l=1}^{k} v_{l,j+1} = -1$, where $v_{i,j}$ is a (i,j)-th element of the probability matrix, is found, and output k as a label. Using this method, one can construct (part of) the DGG tree from the current node and the probability matrix, and the whole DGG tree is not required.

- **Optimization for hamming weight calculation [16, 39]**: On the above method, the most time-consuming parts are in the calculation of hamming weight and the process of scanning columns of the probability matrix. To reduce the cost, we store the probability matrix as column-major order to sequentially access to the memory for scanning columns; and, we store a lookup table of hamming weights for several columns. We set the probability to miss the lookup table as $< 2^{-30}$.
- **Lookup table for first few depths [16]**: Instead of coin flipping each time to choose a child node for the first few depths, we employ a lookup table to use several random bits at once. We set the probability to miss the lookup table as < 0.01 and the lookup table represents first 8 columns of the probability matrix. The lookup table only requires 256 Bytes but significantly improves the sampling performance.

Specifically, our implementation of the Knuth-Yao discrete Gaussian sampler is in the file `sampler.c`, and the sampling step is written in the code snippet below.

```
U16 sample_unit_discrete_gaussian(U8 *randomness, U32 *idx){
  const U8 msb = 0x80;
  U8 coin;
  U16 r;
  U32 p;
  int j, d;

  while(1){
    coin = csprng_sample_byte(randomness, idx);
    r = _LOTUS_KYDG_SAMPLER_LUT[coin];
    if(!(r & msb)) return extend_sign_with_random_bit(r, randomness, idx);

    d = r ^ msb;
    for(j = 0; j < _LOTUS_KYDG_SAMPLER_L1_WEIGHTDEPTH; ++j){
      coin = csprng_sample_bit(randomness, idx);
      d = (d << 1) + coin;
      d -= _LOTUS_KYDG_SAMPLER_L1_weight[j];
      if(d < 0){
        r = scan_bit_and_output(d, j);
        return extend_sign_with_random_bit(r, randomness, idx);
      }
    }

    for(; j < _LOTUS_KYDG_SAMPLER_L1_NCOL; ++j){
      coin = csprng_sample_bit(randomness, idx);
      d = (d << 1) + coin;
      p = _LOTUS_KYDG_SAMPLER_L1_pMat[j];
      d -= __builtin_popcount(p);
      if(d < 0){
        r = scan_bit_and_output(d, j);
        return extend_sign_with_random_bit(r, randomness, idx);
      }
    }
  }
}
```

Code 1.1. Code snippet for the Knuth-Yao discrete Gaussian sampler.

The function takes 512 Bytes randomness pool and its index from the pool, and outputs a sample in $\mathbb{Z}_q$ according to the discrete Gaussian distribution. At lines 9–11, sample using a lookup table `_LOTUS_KYDG_SAMPLER_LUT` with 8 random bits. It will be hit with probability > 0.99. If a hit is found, which is checked using the most significant bit of the value `r`, return the value with extending sign using the `extend_sign_with_random_bit()` function below.

```
1 U16 extend_sign_with_random_bit(const U16 r, U8 *randomness, U32 *idx){
2   U16 ret[2];
3   ret[0] = r;
4   ret[1] = (-r) & (_LOTUS_LWE_MOD - 1);
5   return ret[csprng_sample_bit(randomness, idx)];
6 }
```

Here `ret` is an array of signed value of `r`, where `ret[0]` stores positive and `ret[1]` stores negative. The output value is determined by a random bit sampled using `csprng_sample_bit()`.

If the above lookup misses, sample using the online tree construction approach described above, at lines 14–33. At lines 14–22, the distance `d` is calculated using the lookup table of hamming weight `_LOTUS_KYDG_SAMPLER_L1_weight`, and if $\texttt{d} < 0$, which means the current node is a leaf and the corresponding label is an output value, scan the probability matrix to determine the label using `scan_bit_and_output()` function below, then return the label value with extending sign.

```
1   U16 scan_bit_and_output(long long d, const int j){
2     int t;
3     U32 p = _LOTUS_KYDG_SAMPLER_L1_pMat[j];
4     for(t = 32 - 1; d < -1; --t) d += (p >> t) & 1;
5     while(!((p >> t) & 1)) --t;
6     return 32 - t - 1;
7   }
```

Here the array `_LOTUS_KYDG_SAMPLER_L1_pMat` stores the probability matrix in column-major order, and k-th bit of a value of the array is corresponding to the row of the matrix labeled as $32 - k - 1$. The distance `d` is updated at line 4 until $\texttt{d} = -1$. Then skip `0`'s at line 5 and return the label of the current node (the row index).

The size of `_LOTUS_KYDG_SAMPLER_L1_weight` is fixed so that the probability to exit from the loop of lines 14–22 is $< 2^{-30}$, and if it goes down to lines 24–33 of the Code 1.1, do almost the same things. The difference is, instead of using lookup table, just count the hamming weight straightforwardly using the `popcount()` function at line 28.

3.3 LOTUS-PKE: our proposed LWE-based PKE

The symbols and parameters in Table 1 are used in the specification of LOTUS.

Let $(\mathsf{SE}, \mathsf{SD})$ be a symmetric encryption scheme which is one-time IND-CPA-secure with message space $\{0,1\}^*$. Symmetrically encrypting a message $M \in \{0,1\}^*$ with a key K to obtain a ciphertext c_{sym} is written as $c_{sym} = \mathsf{SE}_K(M)$. Decrypting the ciphertext c_{sym} is written as $\mathsf{SD}_K(c_{sym})$.

Table 1. Symbols and parameters in LOTUS.

SE, SD	Symmetric encryption and decryption algorithms.
M	Message to be encrypted.
K	Symmetric key.
c_{sym}	Symmetric ciphertext.
G, H	Hash functions.
q	LWE modulus.
n	LWE dimension.
l	An integer in the set {128, 192, 256} (security levels).
σ	A uniformly random bit string of length l.
h	A hash value.
s	Noise derivation.
KeyLen	An integer in the set {128, 192, 256} (security levels).
$\{0,1\}^*$	Bit string of arbitrary length.

- **Key generation** $\mathsf{KeyGen}^{cca}_{pke}(\lambda)$**:**
  1. Fix two distinct hash functions G and H.
  2. Choose positive integers q, n, l.
  3. Take matrix $A \in \mathbb{Z}_q^{n\times n}$ uniformly at random.
  4. Fix deviation $s \in \mathbb{R}$. Take Gaussian noise matrices $R, S \in \mathbb{Z}^{n\times l}_{(0,s)}$ randomly.
  5. The public key is $pk = (P, A, q, n, l, s)$ for $P = R - AS \in \mathbb{Z}_q^{n\times l}$, and the secret key is $sk = (S, pk)$. Return (pk, sk).
- **Encryption** $\mathsf{Enc}^{cca}_{pke}(M \in \{0,1\}^*)$**:**
  1. Choose $\sigma \in \{0,1\}^l$ uniformly at random.
  2. Symmetrically encrypt message M by letting $c_{sym} = \mathsf{SE}_{G(\sigma)}(M)$.
  3. Let $h = H(\sigma \parallel c_{sym})$.
  4. Encrypt σ: use randomness h, take Gaussian noise vectors $e_1, e_2 \in \mathbb{Z}^{1\times n}_{(0,s)}$, and $e_3 \in \mathbb{Z}^{1\times l}_{(0,s)}$ randomly, and compute ciphertext $c = (c_1, c_2) = \mathsf{Enc}^{cpa}_{pk}(\sigma; h)$, namely compute
  $$c_1 = e_1 A + e_2 \in \mathbb{Z}_q^{1\times n}, c_2 = e_1 P + e_3 + \sigma \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^{1\times l}.$$
  5. Return $ct = (c_1, c_2, c_{sym})$.
- **Decryption** $\mathsf{Dec}^{cca}_{pke}(sk, ct)$**:** to decrypt $ct = (c_1, c_2, c_{sym})$ by secret key $sk = (S, pk)$, execute following steps.
  1. (Reconstruction) Compute $\overline{\sigma} = c_1 S + c_2 \in \mathbb{Z}_q^l$.
  2. Let $\overline{\sigma} = (\overline{\sigma}_1, \ldots, \overline{\sigma}_l)$. If $\overline{\sigma}_i \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor) \subset \mathbb{Z}_q$, let $\sigma'_i = 0$; otherwise $\sigma'_i = 1$.
  3. Let $\sigma' = \sigma'_1 \cdots \sigma'_l$ and compute $h' = H(\sigma' \parallel c_{sym})$.
  4. (Integrity check and output) Using σ', h', check
  $$(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk}(\sigma'; h'),$$
  namely compute $(c'_1, c'_2) = \mathsf{Enc}^{cpa}_{pk}(\sigma'; h')$, $M' = \mathsf{SD}_{G(\sigma')}(c_{sym})$ and
  (a) if $(c'_1, c'_2) \neq (c_1, c_2)$, the decryption fails;
  (b) otherwise return M' as the message.

3.4 LOTUS-KEM: our proposed LWE-based KEM

In this section we consider the one-time-pad as the underlying symmetric encryption, so that $\mathsf{SE}_u(v) = \mathsf{SD}_u(v) = u \oplus v$ for two bit strings u and v.

- **Key generation $\mathsf{KeyGen}^{cca}_{kem}(\lambda)$:**
  1. Fix two distinct hash functions G and H.
  2. Choose positive integers q, n, l, and KeyLen.
  3. Take matrix $A \in \mathbb{Z}_q^{n\times n}$ uniformly at random.
  4. Fix deviation $s \in \mathbb{R}$.
  5. Take Gaussian noise matrices $R, S \in \mathbb{Z}^{n\times l}_{(0,s)}$ randomly.
  6. The public key is $pk = (P, A, q, n, l, s, \mathsf{KeyLen})$ for $P = R - AS \in \mathbb{Z}_q^{n\times l}$, and the secret key is $sk = (S, pk)$. Return (pk, sk).
- **Encapsulation $\mathsf{Enc}^{cca}_{kem}(pk)$:**
  1. Choose $K \in \{0,1\}^{\mathsf{KeyLen}}$ uniformly at random.
  2. Choose $\sigma \in \{0,1\}^l$ uniformly at random.
  3. Let $c_{sym} = \mathsf{SE}_{G(\sigma)}(K) = G(\sigma) \oplus K \in \{0,1\}^{\mathsf{KeyLen}}$.
  4. Let $h = H(\sigma \parallel c_{sym})$.
  5. Encrypt σ: use randomness h, take Gaussian noise vectors $e_1, e_2 \in \mathbb{Z}^{1\times n}_{(0,s)}$, and $e_3 \in \mathbb{Z}^{1\times l}_{(0,s)}$, and obtain $(c_1, c_2) \leftarrow \mathsf{Enc}^{cpa}_{pk}(\sigma; h)$, namely compute
  $$c_1 = e_1 A + e_2 \in \mathbb{Z}_q^{1\times n}, c_2 = e_1 P + e_3 + \sigma \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^{1\times l}.$$
  6. Return $CT = (c_1, c_2, c_{sym})$ as the encapsulation and K as the symmetric key.
- **Decapsulation $\mathsf{Dec}^{cca}_{kem}(sk, CT)$:** to decrypt $CT = (c_1, c_2, c_{sym})$ by secret key $sk = (S, pk)$, execute following steps.
  1. (Reconstruction) Compute $\overline{\sigma} = c_1 S + c_2 \in \mathbb{Z}_q^l$.
  2. Let $\overline{\sigma} = (\overline{\sigma}_1, \ldots, \overline{\sigma}_l)$. If $\overline{\sigma}_i \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor) \subset \mathbb{Z}_q$, let $\sigma'_i = 0$; otherwise $\sigma'_i = 1$.
  3. Let $\sigma' = \sigma'_1 \cdots \sigma'_l$ and compute $h' = H(\sigma' \parallel c_{sym})$.
  4. (Integrity check and output) Using σ', h', check
  $$(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk}(\sigma'; h'),$$
  namely compute $(c'_1, c'_2) = \mathsf{Enc}^{cpa}_{pk}(\sigma'; h')$, $K = \mathsf{SD}_{G(\sigma')}(c_{sym}) = G(\sigma') \oplus c_{sym}$ and
  (a) if $(c'_1, c'_2) \neq (c_1, c_2)$, the decapsulation fails;
  (b) otherwise return K as the symmetric key.

4 Parameter Sets and Performance Analysis

4.1 Parameter sets

Table 2 contains our suggested parameter sets.

Table 2. Parameters for LOTUS and expected security strengths.

	LWE parameter (n, q, s)	**Other parameters**	**Security level**	**NIST's security category**
`lotus-params128`	$(n = 576, q = 8192, s = 3.0)$	$l = 128, \mathsf{KeyLen} = 128$	128-bit security	AES-128, SHA3-256
`lotus-params192`	$(n = 704, q = 8192, s = 3.0)$	$l = 192, \mathsf{KeyLen} = 192$	192-bit security	AES-192, SHA3-384
`lotus-params256`	$(n = 832, q = 8192, s = 3.0)$	$l = 256, \mathsf{KeyLen} = 256$	256-bit security	AES-256

Claim 1 (LWE bit security) *The LWE assumption with the parameter* `lotus-params256` *(resp.,* `lotus-params192`*,* `lotus-params128`*) in Table 2 has at least 256-bit security (resp., 192-, 128-bit security). The equivalence with NIST's security category is given in Table 2.*

The evidence of Claim 1 is given in depth in Appendix A.

Choice of building blocks in LOTUS. For our implementations, we choose symmetric ingredients for LOTUS as in Table 3. In particular, the hash functions G and H will be of the following form: $G(x) = \text{SHA-512}(x \parallel \texttt{0x01})$ and $H(x) = \text{SHA-512}(x \parallel \texttt{0x02})$ for all bit strings x.

Table 3. Symmetric ingredients in LOTUS.

Symmetric encryption (SE, SD)	AES-128-CTR for 128-bit security AES-192-CTR for 192-bit security AES-256-CTR for 256-bit security
Hash function $G(x)$	SHA-512($x \parallel \texttt{0x01}$)
Hash function $H(x)$	SHA-512($x \parallel \texttt{0x02}$)

4.2 Correctness of LOTUS

The parameters in Table 2 also ensure that probability of the decryption error due to unexpected large noises in encryption is less than 2^{-256}.

We will use the following lemmas [10, 11, 27]. Below $||\cdot||$ stands for either the Euclidean norm of a vector or the absolute value; $\langle \cdot, \cdot \rangle$ for inner product. Writing $||\mathbb{Z}^n_{(0,s)}||$ is a short hand for taking a vector from the distribution and computing its norm.

Lemma 2 *Let $c \geq 1$ and $C = c \cdot \exp(\frac{1-c^2}{2})$. Then for any real $s > 0$ and any integer $n \geq 1$, we have*

$$\Pr\left[||\mathbb{Z}^n_{(0,s)}|| \geq \frac{c \cdot s\sqrt{n}}{\sqrt{2\pi}}\right] \leq C^n.$$

Lemma 3 *For any real $s > 0$ and $T > 0$, and any $x \in \mathbb{R}^n$, we have*

$$\Pr\left[||\langle x, \mathbb{Z}^n_{(0,s)}\rangle|| \geq Ts||x||\right] < 2\exp(-\pi T^2).$$

Theorem 2 (Correctness of LOTUS) *The decryption error in both LOTUS-KEM and LOTUS-PKE are less than* 2^{-256} *for all parameter sets* `lotus-params256`*,* `lotus-params192`*, and* `lotus-params128`*.*

Proof (of Theorem 2). The decryption in LOTUS-PKE and LOTUS-KEM yields noises, as follows:

$$c_1 S + c_2 = e_1 R + e_2 S + e_3 + m \cdot \left\lfloor \frac{q}{2} \right\rfloor.$$

Each component in $\mathbb{Z}_q$ of the noise vector $e_1 R + e_2 S + e_3$ can be written as the inner product of two vectors of form

$$e = \Big(e_1, e_2, e_3\Big)$$
$$x = \Big(r, r', \mathbf{0}1\mathbf{0}_{1\times l}\Big)$$

where vectors $r, r' \in \mathbb{Z}_{(0,s)}^{1\times n}$ represent corresponding columns in matrices R, S and $\mathbf{0}1\mathbf{0}_{1\times l}$ stands for a vector of length l with all 0's except one 1.

Use $||\cdot||$ to denote the Euclidean norm, we have

$$e \in \mathbb{Z}_{(0,s)}^{1\times(2n+l)}, \text{ and } ||x|| \leq ||(r, r')|| + 1$$

where $(r, r') \in \mathbb{Z}_{(0,s)}^{1\times 2n}$. Applying Lemma 2 for vector of length $2n$, with high probability of

$$1 - C^{2n} = 1 - \left(c \cdot \exp\left(\frac{1-c^2}{2}\right)\right)^{2n} \geq 1 - 2^{-256} \tag{1}$$

we have

$$||x|| \leq \frac{c \cdot s\sqrt{2n}}{\sqrt{2\pi}} + 1.$$

We now use Lemma 3 with vectors x and e. Let ρ be the error per message symbol in decryption, we set $2\exp(-\pi T^2) = \rho$, so $T = \sqrt{\ln(2/\rho)}/\sqrt{\pi}$. For correctness, we need $T \cdot s \cdot ||x|| \leq q/4$, which holds true provided that

$$\frac{\sqrt{\ln(2/\rho)}}{\sqrt{\pi}} \cdot s \cdot \left(\frac{cs\sqrt{2n}}{\sqrt{2\pi}} + 1\right) \leq \frac{q}{4}. \tag{2}$$

Concrete bounds. When $n = 576$ as in `lotus-params128` we take $c = 1.42$ so that condition (1) is satisfied. In (2) we use $s = 3.0$, $\rho = 2^{-256}$, $\pi \approx 3.14$ so that $q \geq 5302.7$ suffices.

When $n = 704$ as in `lotus-params192` we take $c = 1.38$ so that condition (1) is satisfied. In (2) we use $s = 3.0$, $\rho = 2^{-256}$, $\pi \approx 3.14$ so that $q \geq 5690.5$ suffices.

When $n = 832$ as in `lotus-params256` we take $c = 1.35$ so that condition (1) is satisfied. In (2) we use $s = 3.0$, $\rho = 2^{-256}$, $\pi \approx 3.14$ so that $q \geq 6046$ suffices.

For all cases, as we choose $q = 8192$, the theorem follows. □

4.3 Performance analysis of LOTUS-PKE and LOTUS-KEM

The performance analyses of LOTUS-PKE and LOTUS-KEM are given in Tables 4, 5, 6, 7.

Sizes. The key and ciphertext sizes in LOTUS are computed as follows in bits:

- $size(pk) = size(P) + size(A)$ where $size(P) = nl\log_2(q)$ and $size(A) = n^2\log_2(q)$.
- $size(sk) = size(pk) + nl\log_2(7.54s)$.
- $size(CT) = size(c_1) + size(c_2) + size(c_{sym}) = (n+l) \times \log_2(q) + size(c_{sym})$.

The constant 7.54 in estimating the size of sk is to ensure that its elements in absolute values are less than $7.54s$ approximately with probability 2^{-256}.

Speed. We provide the three implementations for LOTUS-PKE and LOTUS-KEM: (1) reference implementation, (2) optimized implementation, and (3) AVX2-based implementation. Timings reported in Tables 6 and 7 are from the reference implementation on a computer with CPU Core i7-7700K (4.20 GHz). The timings are averaged over 2^{12} times of executions for key generation and 2^{17} times for encryption/encapsulation and decryption/decapsulation. The optimized and AVX2-based implementation (whose timings are not in the tables) are a little faster than the reference implementation.

Table 4. LOTUS-PKE sizes.

Parameter set	Public key size	Secret key size	Encapsulation size
`lotus-params128`	658.95 (KB)	700.42 (KB)	1.144 (KB) + $size(c_{sym})$
`lotus-params192`	1025.0 (KB)	1101.0 (KB)	1.456 (KB) + $size(c_{sym})$
`lotus-params256`	1471.0 (KB)	1590.8 (KB)	1.768 (KB) + $size(c_{sym})$

Table 5. LOTUS-KEM sizes.

Parameter set	Public key size	Secret key size	Encapsulation size
`lotus-params128`	658.95 (KB)	700.42 (KB)	1.144 (KB)
`lotus-params192`	1025.0 (KB)	1101.0 (KB)	1.456 (KB)
`lotus-params256`	1471.0 (KB)	1590.8 (KB)	1.768 (KB)

5 Expected Security Strength

We provide the evidence for the security strength of LOTUS-PKE and LOTUS-KEM.

5.1 Classical security strength

Theorem 3 *The LOTUS-PKE scheme is IND-CCA2-secure under the LWE assumption provided that G, H are random oracles.*

Table 6. LOTUS-PKE running times (for 1 KB messages, reference implementation).

Parameter set	$\mathsf{KeyGen}_{pke}^{cca}$	Enc_{pke}^{cca}	Dec_{pke}^{cca}
`lotus-params128`	6,385.842 usec	75.299 usec	91.091 usec
	26,820,400 clock	316,252 clock	382,582 clock
`lotus-params192`	11,109.302 usec	105.636 usec	139.862 usec
	46,658,849 clock	443,667 clock	587,417 clock
`lotus-params256`	17,197.583 usec	149.075 usec	210.126 usec
	72,229,496 clock	626,112 clock	882,523 clock

(usec denotes microsecond.)

Table 7. LOTUS-KEM running times (reference implementation).

Parameter set	$\mathsf{KeyGen}_{kem}^{cca}$	Enc_{kem}^{cca}	Dec_{kem}^{cca}
`lotus-params128`	6387.009 usec	75.146 usec	90.165 usec
	26,825,276 clock	315,611 clock	378,690 clock
`lotus-params192`	10,975.066 usec	110.201 usec	142.581 usec
	46,095,015 clock	462,842 clock	598,836 clock
`lotus-params256`	17,106.305 usec	139.266 usec	206.540 usec
	71,846,095 clock	584,915 clock	867,464 clock

(usec denotes microsecond.)

Proof. We utilize the Fujisaki-Okamoto transformation in [19]. Indeed, the encryption in the LOTUS-PKE scheme can be viewed as follows:

$$\mathsf{Enc}_{pk}^{cca}(m;\sigma) = \Big(\underbrace{\mathsf{Enc}_{pk}^{cpa}\Big(\sigma; H(\sigma \parallel c_{sym})\Big)}_{(c_1,c_2)}, \underbrace{\mathsf{SE}_{G(\sigma)}(m)}_{c_{sym}} \Big) \tag{3}$$

which is exactly the transformation in [19]. To be complete, the encryption scheme with $\mathsf{Enc}_{pk}^{cpa}(\cdot,\cdot)$ is one-way and $(n+l)\log_2 q$-spread (see [19, Definition 5.2]) due to Theorem 1. Applying [19, Theorem 6.1], the above theorem follows. □

Theorem 4 *LOTUS-KEM scheme is IND-CCA2-secure under the LWE assumption provided that G, H are random oracles.*

The proof is similar to the proof of [19, Theorem 6.1].

Claim 2 (Security strength of LOTUS) *Both LOTUS-PKE and LOTUS-KEM schemes achieve IND-CCA2 security with 256-bit security (resp., 192- and 128-bit security) with the parameter set* `lotus-params256` *(resp.,* `lotus-params192` *and* `lotus-params128`*). The equivalence with NIST's security category is also given in Table 2.*

The evidence of Claim 2 directly comes from Claim 1, together with Theorem 3 and Theorem 4.

5.2 Quantum security strength

Our proposed schemes LOTUS-PKE (Section 3.3) and LOTUS-KEM (Section 3.4) can be modified to have asymptotic security in the quantum random oracle model [45]. Specifically following [45], to have security in the quantum oracle model, the transformation given in (3) is turned to

$$\mathsf{Enc}^{cca}_{pk}(m;\sigma) = \Big(\underbrace{\mathsf{Enc}^{cpa}_{pk}\Big(\sigma; H(\sigma \parallel c_{sym})\Big)}_{(c_1,c_2)}, \underbrace{\mathsf{SE}_{G(\sigma)}(m)}_{c_{sym}}, H'(\sigma)\Big) \tag{4}$$

in which H' is an additional hash function viewed as a random oracle. Using (4), the encryption in both LOTUS-PKE and LOTUS-KEM will have an additional line computing $d = H'(\sigma)$, and the decryption will check whether $d = H'(\sigma)$ accordingly.

The IND-CCA2 security of the transformation given in (4) in the quantum oracle model are proved in [45, Theorem 4]. The additional assumption is that H' is a random oracle.

Below we discuss our decisions on building LOTUS-PKE and LOTUS-KEM regarding the quantum oracle model:

- We choose not to add $H'(\sigma)$ to the ciphertexts of LOTUS-PKE and LOTUS-KEM. The reason is that we think $H'(\sigma)$ only helps the security proof in the quantum oracle model, but plays little role in the actual security of LOTUS-PKE and LOTUS-KEM.
- We currently ignore the concrete reduction in [45, Theorem 4] because it is very loose. Instead, we tentatively choose to increase the LWE dimension n in the future if there exist attacks (either classical or quantum) on the LWE assumption or directly on LOTUS.

6 Known Attacks

The only known attack to the IND-CCA2 security of LOTUS is to solve the LWE problem with parameters (n, q, s) in Table 2.

7 Advantages and Limitations

The advantage of LOTUS-PKE and LOTUS-KEM is on the fact that their IND-CCA2 securities relying *solely* on the LWE assumption [37] whose maturity and strength are recalled in Section 3.1. The use of the LWE assumption causes a cost in key sizes, and we see this cost as a trade-off for strong security assurance.

A limitation is perhaps on the relatively big key sizes described in Tables 4 and 5. Another possible limitation is that the arguments (proofs) for IND-CCA2 security are in the random oracle model. Despite of these limitations, LOTUS may serve as a good candidate for *general use* of post-quantum PKE and KEM.

References

1. Data Age 2025: The Evolution of Data to Life-Critical. `http://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf`.
2. M. R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in helib and SEAL. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 103–129, 2017.
3. M. R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, and L. Perret. On the complexity of the BKW algorithm on LWE. *Des. Codes Cryptography*, 74(2):325–354, 2015.
4. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015.
5. Y. Aono, X. Boyen, L. T. Phong, and L. Wang. Key-private proxy re-encryption under LWE. In G. Paul and S. Vaudenay, editors, *INDOCRYPT*, volume 8250 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.
6. Y. Aono, T. Seito, and J. Shikata. A theoretical cost lower bound of lattice vector enumeration. Computer Security Symposium 2017 (CSS2017), 1E4-5, 2017.
7. Y. Aono, Y. Wang, T. Hayashi, and T. Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. *IACR Cryptology ePrint Archive*, 2016:146, 2016.
8. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
9. S. Bai, T. Laarhoven, and D. Stehle. Tuple lattice sieving. *LMS Journal of Computation and Mathematics*, 19(A):146–162, 2016.
10. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.
11. W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 13(1):217–231, 1995.
12. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584, 2013.
13. M. R. Bremner. *Lattice Basis Reduction: An Introduction to the LLL Algorithm and Its Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011.
14. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.
15. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 3–33, 2016.
16. R. Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede. Efficient software implementation of Ring-LWE encryption. Cryptology ePrint Archive, Report 2014/725, 2014. `http://eprint.iacr.org/2014/725`.
17. T. F. development team. fplll, a lattice reduction library. Available at `https://github.com/fplll/fplll`, 2016.
18. U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *J-MATH-COMPUT*, 44(170):463–471, Apr. 1985.
19. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptology*, 26(1):80–101, 2013.
20. M. Fukase and K. Kashiwabara. An accelerated algorithm for solving SVP based on statistical analysis. *JIP*, 23(1):67–80, 2015.
21. N. Gama and P. Q. Nguyen. *Predicting Lattice Reduction*, pages 31–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
22. N. Gama, P. Q. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2010.
23. R. Kannan. Improved algorithms for integer programming and related lattice problems. In D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, editors, *STOC*, pages 193–206. ACM, 1983.

24. S. V. Khrushchev. Men′shov's correction theorem and Gaussian processes (translated from Russian). *Trudy Mat. Inst. Steklov.*, 155:151–181, 1981.
25. D. E. Knuth and A. C. Yao. The complexity of non-uniform random number generation. *Algorithms and Complexity, Academic Press, New York*, pages 357–428, 1976.
26. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261:513–534, 1982.
27. R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.
28. M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In E. Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2013.
29. V. Lyubashevsky. Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537, 2011. `https://eprint.iacr.org/2011/537`. Full version of a paper appearing at Eurocrypt 2012.
30. V. Lyubashevsky and D. Micciancio. *On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem*, pages 577–594. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
31. D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.
32. D. Micciancio and M. Walter. Practical, predictable lattice basis reduction. In *Proceedings, Part I, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9665*, pages 820–849, New York, NY, USA, 2016. Springer-Verlag New York, Inc.
33. National Institute of Standards and Technology (NIST). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. Website: `http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf`.
34. P. Q. Nguyen and D. Stehlé. *LLL on the Average*, pages 238–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
35. P. Q. Nguyen and B. Valle. *The LLL Algorithm: Survey and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.
36. C. Peikert. Lattice cryptography for the internet. In M. Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.
37. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
38. O. Regev. The learning with errors problem (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, 2010*, pages 191–204, 2010.
39. S. S. Roy, F. Vercauteren, and I. Verbauwhede. High precision discrete gaussian sampling on FPGAs. In *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, 2013*, pages 383–401, 2013.
40. M. Schneider and N. Gama. SVP challenge. Available at `http://www.latticechallenge.org/svp-challenge/`.
41. C. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201 – 224, 1987.
42. C.-P. Schnorr. Lattice reduction by random sampling and birthday methods. In H. Alt and M. Habib, editors, *STACS 2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2003.
43. C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *Math. Programming*, pages 181–191, 1993.
44. C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In L. C. Guillou and J.-J. Quisquater, editors, *EUROCRYPT 1995*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1995.
45. E. E. Targhi and D. Unruh. Quantum security of the Fujisaki-Okamoto and OAEP transforms. Cryptology ePrint Archive, Report 2015/1210, 2015. `http://eprint.iacr.org/2015/1210`.
46. S. S. Venkatesh. *The theory of probability: Explorations and applications*. Cambridge: Cambridge University Press, 2013.

Appendix

A Hardness Estimation and Proposed Parameters for LWE in LOTUS

Our parameter sets `lotus-params128`, `lotus-params192`, and `lotus-params256` are based on the theory to bound the cost of lattice based attack for LWE presented recently by Aono et al. in [6]. This section is mainly based on [6], with additional contents dedicated to the LWE assumption.

The lattice based attacks against an LWE instance with parameter (n, q, s) consisting of three steps:

(i) construct a lattice basis B from the instance,
(ii) perform lattice reduction for B, and
(iii) solve the bounded distance decoding corresponding to LWE.

This attack model is completely the same as Liu-Nguyen [28], and typically called "the primal attack." The attacker's cost is given as follows and the minimum is over all lattice reduction algorithms and the success probability of the point search algorithm.

$$Cost(Problem) = \min \frac{Cost(LatticeReduction) + Cost(PointSearch)}{\text{Success probability of PointSearch}}. \tag{5}$$

Roughly speaking, if one uses a stronger algorithm, the cost of lattice reduction increases, as that of point search is decreases.

A.1 Technical backgrounds

A.1.1 Basic notations and special functions

Let $\mathbb{Z}$ and $\mathbb{Z}_m$ are the set of integers and the ring $\{0, \ldots, m-1\}$ respectively. $\mathbb{Q}$ and $\mathbb{R}$ are the set of rational numbers and real numbers respectively. For natural numbers $n \leq m$, $[n, m]$ is the set $\{n, \ldots, m\} \subset \mathbb{Z}$ and we denote $[m] := [1, m]$. Throughout this paper, m and k are usually used for the considered and projected dimension respectively.

The gamma and beta functions are respectively defined by

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt \text{ and } B(a, b) = \int_0^1 z^{a-1}(1-z)^{b-1} dz.$$

The basic relations $\Gamma(a+1) = a\Gamma(a)$ and $B(a, b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ hold.

For $m \in \mathbb{N}$, let $B_m(\boldsymbol{x}, c)$ be the m-dimensional ball whose center is $\boldsymbol{x} \in \mathbb{R}^m$ and radius $c > 0$. The center-origin ball is denoted by $B_m(c) := B_m(\mathbf{0}, c)$. The volume of m-dimensional ball with radius c is $V_m(c) = \frac{\pi^{m/2} c^m}{\Gamma(\frac{m}{2}+1)}$. In particular, we denote $V_m := V_m(1)$. Also, $S_m(r)$ is the surface of $B_m(r)$. The log function $\log(\cdot)$ is always the natural logarithm.

Incomplete gamma function: For a parameter a, the lower incomplete gamma function regularized lower incomplete gamma function are

$$\gamma(a,x) := \int_0^x t^{a-1}e^{-t}dt \quad \text{and} \quad P(a,x) = \frac{\gamma(a,x)}{\Gamma(a)}$$

respectively. $P(a,x)$ can be used for the closed formula of the high-dimensional Gaussian integral.

$$\int_{B_k(\rho)} \frac{1}{s^k} e^{-\pi\|\boldsymbol{x}\|^2/s^2} d\boldsymbol{x} = \frac{2\pi^{k/2}}{s^k\Gamma(\frac{k}{2})} \int_0^\rho e^{-\pi r^2/s^2} r^{k-1} dr = \frac{1}{\Gamma(\frac{k}{2})}\gamma\left(\frac{k}{2}, \frac{\pi\rho^2}{s^2}\right) = P\left(\frac{k}{2}, \frac{\pi\rho^2}{s^2}\right). \tag{6}$$

By the relation $P(a,x) \le \dfrac{1}{\Gamma(a)} \displaystyle\int_0^x t^{a-1}dt = \frac{x^a}{a\Gamma(a)}$, we can bound the inverse function of P from lower:

$$P^{-1}(a,x) \ge (a\Gamma(a)x)^{1/a}. \tag{7}$$

Beta distribution and incomplete beta function: For $a, b > 0$, the beta distribution $B(a,b)$ is defined by the probability density function

$$f(x;a,b) = \frac{x^{a-1}(1-x)^{b-1}}{B(a,b)}.$$

Its cumulative distribution function is given by the incomplete beta function:

$$I_x(a,b) := \frac{1}{B(a,b)} \int_0^x z^{a-1}(1-z)^{b-1}dz,$$

and its inverse function is defined by $x = I_y^{-1}(a,b) \Leftrightarrow y = I_x(a,b)$. Both functions are strictly increasing from $[0,1]$ to $[0,1]$.

A bound

$$I_x(a,b) \le \frac{1}{B(a,b)} \int_0^x z^{a-1}dz = \frac{x^a}{a \cdot B(a,b)}$$

holds and thus

$$I_x^{-1}(a,b) \ge (aB(a,b)x)^{1/a}. \tag{8}$$

Fact 1 *Suppose $(x_1, \ldots, x_m) \leftarrow S_m(1)$. Then, $x_1^2 + \cdots + x_k^2$ follows the beta distribution of parameters $(a,b) = \left(\frac{k}{2}, \frac{m-k}{2}\right)$. Thus,*

$$\Pr_{(x_1,\ldots,x_m)\leftarrow S_m(1)}\left[x_1^2 + \cdots + x_k^2 \le C\right] = I_C\left(\frac{k}{2}, \frac{m-k}{2}\right) := \frac{\int_0^C x^{\frac{k}{2}-1}(1-x)^{\frac{m-k}{2}-1}dx}{B(\frac{k}{2}, \frac{m-k}{2})}.$$

In particular, $(x_1, \ldots, x_{m-2})$ follows the uniform distribution in $B_{m-2}(1)$, which implies the following formula.

$$\Pr_{(x_1,\ldots,x_m)\leftarrow B_m(1)}\left[x_1^2 + \cdots + x_k^2 \le C\right] = I_C\left(\frac{k}{2}, \frac{m+2-k}{2}\right)$$

.

A.1.2 Basics on lattices

This section is an outline of lattices which are used to construct our theory. For more fundamental explanations and backgrounds, readers should refer the textbooks [13, 35].

For a set of linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Q}^m$, the lattice is defined by the set of all the integer linear combination:

$$L(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{ \sum_{i=1}^{n} a_i \boldsymbol{b}_i : a_i \in \mathbb{Z} \right\}.$$

The ordered set $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ is called *a basis* of lattice. For a basis, its Gram-Schmidt basis is defined by recursively $\boldsymbol{b}_1^* = \boldsymbol{b}_1$ and $\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{b}_j^*$ where $\mu_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle / \langle \boldsymbol{b}_j^*, \boldsymbol{b}_j^* \rangle$ for $i = 2, \ldots, n$. The new basis $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$ that are orthogonal to each other spans the same space to the original basis:

$$\mathrm{span}(L) := \left\{ \sum_{i=1}^{n} w_i \boldsymbol{b}_i : w_i \in \mathbb{R} \right\} = \left\{ \sum_{i=1}^{n} x_i \boldsymbol{b}_i^* : x_i \in \mathbb{R} \right\}.$$

Thus, any lattice point $\boldsymbol{v} = \sum_{i=1}^{n} a_i \boldsymbol{b}_i$ can be represented by using Gram-Schmidt basis: $\boldsymbol{v} = \sum_{i=1}^{n} x_i \boldsymbol{b}_i^*$. For this, the j-th projection is

$$\pi_j(\boldsymbol{v}) = \sum_{i=j}^{n} x_i \boldsymbol{b}_i^*.$$

The volume of lattice $\mathrm{vol}(L)$ is defined by

$$|\det(L)| = \prod_{i=1}^{n} \|\boldsymbol{b}_i^*\|.$$

Hard lattice problems: For a lattice L, we denote by $\lambda_1(L)$ the smallest nonzero norm of points in L, i.e., the length of the shortest vector. The problem for searching $\boldsymbol{v} \in L$ such that $\|\boldsymbol{v}\| = \lambda_1(L)$ is called the shortest vector problem (SVP). The approximate Hermite shortest vector problem (HSVP_α) [21] is the problem of finding vector $\boldsymbol{v}$ shorter than $\alpha \cdot \mathrm{vol}(L)^{1/n}$. The bounded distance decoding (BDD) problem [3] for a given lattice basis B, target point $\boldsymbol{t}$ and the distance d, is the problem of finding a lattice point $\boldsymbol{v}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leq d$.

Gaussian heuristic [44, Section. 3]: Consider a set $S \subset \mathrm{span}(L)$ with a finite volume: $\mathrm{vol}(S) < \infty$. The Gaussian heuristic assumption claims that the number of lattice points in S is approximately given by $\mathrm{vol}(S)/\mathrm{vol}(L)$. In particular, we can see $\lambda_1(L)$ is close to $\ell = V_n^{-1/n} \mathrm{vol}(L)^{1/n}$ so that $V_n(\ell) = \mathrm{vol}(L)$. We denote this length by $GH(L)$ and call it the Gaussian heuristic length of L.

[3] The standard version of BDD is defined by using the shortest vector length $\lambda_1(B)$; see, for example Lyubashevsky and Micciancio [30]. Throughout this paper, we use this notation by following the work of Liu and Nguyen [28].

Dual lattices: For a lattice L, its *dual lattice* $L^\times$ is defined by the set

$$L^\times = \{\boldsymbol{w} \in \mathrm{span}(L) : \forall \boldsymbol{v} \in L, \langle \boldsymbol{v}, \boldsymbol{w} \rangle \in \mathbb{Z}\}.$$

For the basis matrix $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m \times n}$, its dual basis matrix $D = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n)$ satisfies $B^T D = I$. Explicitly written, $D = B(B^T B)^{-1}$ in general and $D = (B^T)^{-1}$ for the square matrix B.

Fact 2 *Let $(\boldsymbol{d}_n, \ldots, \boldsymbol{d}_1)$ be the reversed order of dual basis of $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$. Its Gram-Schmidt orthogonalization $(\boldsymbol{d}_n^\star, \ldots, \boldsymbol{d}_1^\star)$ satisfies $\boldsymbol{d}_i^\star = \boldsymbol{b}_i^* / \|\boldsymbol{b}_i^*\|^2$.*

Thus, $\|\boldsymbol{d}_i^\star\| = 1/\|\boldsymbol{b}_i^*\|$ holds and it deduces $\mathrm{vol}(L^\times) = 1/\mathrm{vol}(L)$. The Gaussian heuristic over the dual lattice predicts that

$$\lambda_1(L^\times) \approx V_n^{-1/n} \mathrm{vol}(L^\times)^{1/n} = V_n^{-1/n} \mathrm{vol}(L)^{-1/n}.$$

Lattice basis reduction: Since the celebrated LLL algorithm [26], a series of lattice reduction algorithm has wide applications in many area. In general, such an algorithm selects a suitable unimodular matrix U for given lattice basis matrix B, update the basis by multiplication: $B \leftarrow B \cdot U$. In this document, we focus on these algorithms as the approximate SVP algorithm as [14, 41, 43, 44]. The strongest lattice reduction algorithm finds the shortest vector at the first vector $\boldsymbol{b}_1$ of basis. The Gaussian heuristic suggests the practical lower (resp. upper) bounds of $\|\boldsymbol{b}_1\|$ (resp. $\|\boldsymbol{b}_n^*\|$), that is, for a majority of random lattices, its reduced basis satisfies

$$\|\boldsymbol{b}_1\| > GH(L) = V_n^{-1/n} \cdot \mathrm{vol}(L)^{1/n} \text{ and } \|\boldsymbol{b}_n^*\| < GH(L^\times) = V_n^{1/n} \cdot \mathrm{vol}(L)^{1/n}.$$

We also claim the following assumption to prove Lemma 1 in Section A.3.2, which is a bit stronger than the original Gaussian heuristic.

Assumption 1 *For almost all lattices and its reduced basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$, its projections all satisfy*

$$\|\boldsymbol{b}_n^*\| < GH(L^\times) = V_n^{1/n} \cdot (\|\boldsymbol{b}_1\| \cdots \|\boldsymbol{b}_n^*\|)^{1/n}$$

for a reasonable range of n, such as $(3/4)m \leq n \leq m$.

Root-Hermite factor and geometric series assumption: From the experimental observations by Gama, Nguyen and Stehlé [21, 34] for lattice reduction algorithms that work on any lattice dimension n, there exists a constant δ_0 so that the output of lattice reduction algorithm over random lattices satisfies $\|\boldsymbol{b}_1\| \approx \delta_0^n \mathrm{vol}(L)^{1/n}$. This δ_0 is called *the root Hermite factor* of the algorithm. We call the basis is δ_0-reduced if $\|\boldsymbol{b}_1\| \leq \delta_0^n \mathrm{vol}(L)^{1/n}$ holds, thus, it is a solution of the $\mathrm{HSVP}_{\delta_0^n}$ problem.

Depending on varieties of algorithms, the graph of Gram-Schmidt lengths $\|\boldsymbol{b}_i^*\|$ has a variety if they all achieve the same root Hermite factor δ_0. However, they are typically concave curves close to a line. Schnorr's geometric series assumption (GSA) [42] claims that

$||\mathbf{b}_i^*||^2$ is approximated by $||\mathbf{b}_1||^2 r^{i-1}$ by a constant $r < 1$. Hence, each of Gram-Schmidt lengths of a δ_0-reduced basis can be approximated by

$$||\mathbf{b}_i^*|| = r^{\frac{2i-1-n}{4}} \mathrm{vol}(L)^{1/n} \text{ where } r = \delta_0^{\frac{-4n}{n-1}}. \tag{9}$$

We call the sequence

$$(||\mathbf{b}_1^*||, \ldots, ||\mathbf{b}_n^*||) = (r^{(1-n)/4}\mathrm{vol}(L)^{1/n}, \ldots, r^{(n-1)/4}\mathrm{vol}(L)^{1/n})$$

the δ_0-*GSA basis*, and sometimes we denote it by B_{δ_0}, which is an abused notation because it is not a lattice basis.

This assumption has been used to estimate the practical complexity of lattice problems in some published works [20,27]. However, for highly reduced lattice bases, the lengths $\|\boldsymbol{b}_i^*\|$ do not follow a line in the last indexes. Such phenomenon is justified by the Gaussian heuristic. Hence, it is not reasonable to estimate the *expected* complexity by using GSA. On the other hand, we will demonstrate it can be used to estimate a lower bound in Section A.3.

A.1.3 Lattice-based attack against LWE

Definition 1. *(Search learning with errors (LWE) problem with Gaussian error) Fixing the problem parameter $n, m, q \in \mathbb{N}$ and $\alpha \in \mathbb{R}$, the challenger generates random vectors $\boldsymbol{a}_i \in \mathbb{Z}_q^n$ $(i = 1, \ldots, m)$, a secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, and a random Gaussian noise vector $\boldsymbol{e} \in \mathbb{Z}^m$ whose coordinates are independently and randomly sampled from the discrete Gaussian distribution $\mathcal{D}_s$ whose probability density function at $i \in \mathbb{Z}$ is*

$$p(i) = \frac{e^{-\pi i^2/s^2}}{\sum_{j=-\infty}^{\infty} e^{-\pi j^2/s^2}}. \tag{10}$$

Then, the adversary gets pairs $(\boldsymbol{a}_i, b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ for $i = 1, \ldots, m$ and tries to recover the secret vector $\boldsymbol{s}$.

For the uniqueness of the solution, the number of vectors m is usually taken much larger than n. The primal attack for LWE which we will consider in the later section is converting the instance to a BDD instance with a lattice of rank $m' \leq m$ which is selected by the attacker to minimize the cost.

We introduce an outline of the primal attack. From the instance, suppose we pick some vectors and construct a matrix $A \in \mathbb{Z}_q^{n \times m'}$ by concatenating $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m'}$. Then, $A\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{b} \pmod q$ holds and there exists an integer vector $\boldsymbol{w} \in \mathbb{Z}_q^{m'}$, the equation can be expressed by $A\boldsymbol{s} + q\boldsymbol{w} + \boldsymbol{e} = \boldsymbol{b}$. Thus, the unknown vectors $\boldsymbol{s}$ and $\boldsymbol{w}$ satisfy

$$[A \ qI] \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{w} \end{bmatrix} + \boldsymbol{e} = \boldsymbol{b}$$

and if the error vector is short, there exists a point of lattice spanned by the columns of $[A \ qI] \in \mathbb{Z}^{m' \times (m'+n)}$ close to $\boldsymbol{b}$. Since the basis given by the matrix is degenerate, it needs

to reconstruct the independent vectors for which we know an efficient method. Therefore, the LWE problem is regarded as a variant of BDD problem whose distance is $\|\boldsymbol{e}\|$. Since the distribution of $\|\boldsymbol{e}\|$ is well-known, we can take an appropriate bound so that the error vector is shorter than it with high probability.

A.1.4 Enumeration algorithm and cost estimation

We give a brief overview of the lattice vector enumeration algorithms by Kannan [23], and Fincke-Phost [18], pruned enumeration by Schnorr [44], and rigid analysis under the Gaussian heuristic assumption by Gama-Nguyen-Regev [22].

For a given lattice basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$, suppose we have data of Gram-Schmidt vectors $\boldsymbol{b}_i^*$ and Gram-Schmidt coefficients $\mu_{i,j}$ with sufficient accuracy. The enumeration algorithm finds combination of coefficients a_i such that $\boldsymbol{v} = \sum_{i=1}^{n} a_i \boldsymbol{b}_i \in L$ is shorter than a threshold radius c. It is easy to convert the coefficients to the lattice vector.

The standard pruned enumeration algorithm takes as input the Gram-Schmidt vectors and coefficients, thresholding radius c, and pruning coefficients $0 < R_1 \leq \ldots \leq R_m = 1$. It searches the (possibly infinite) labelled tree of depth m. Each node at depth k is labelled by a vector $\boldsymbol{v} \in \mathbb{R}^m$ and the projective length $\|\pi_{m-k+1}(\boldsymbol{v})\|$, and it has children labelled by vectors $\boldsymbol{v} + a_{m-k}\boldsymbol{b}_{m-k}$ $(a_{m-k} \in \mathbb{Z})$. The root has the zero vector. Hence, every node at depth k has the label $\sum_{i=m-k+1}^{m} a_i \boldsymbol{b}_i$. The algorithm carries out the depth-first search and pruning the edge when the projective length exceeds $c \cdot R_k$ at depth k to limit the searching space. For the detailed algorithm and efficient implementation, see for example [22, 43].

We remark that it can be easily extended to the algorithm for the BDD problem [28]; the projective length corresponding to the label $\boldsymbol{v}$ is changed to $\|\pi_{m-k+1}(\boldsymbol{v} - \boldsymbol{t})\|$ where $\boldsymbol{t}$ is the target vector.

These enumeration algorithms output all lattice vectors satisfying $\|\boldsymbol{v}\| \leq c$ (or $\|\boldsymbol{v}-\boldsymbol{t}\| \leq c$) if there is no pruning, i.e., $R_i = 1$ for all i.

Cost estimation under the model by Gama-Nguyen-Regev [22]: They gave models and efficient techniques for analyzing success probability and complexity for the pruned enumeration. For given searching radius c and pruning coefficients, The space searched by the above tree-searching algorithm at depth k is

$$\begin{aligned} &\{\boldsymbol{v} \in \mathbb{R}^k : \|\pi_{k-i+1}(\boldsymbol{v})\| \leq c \cdot R_i \text{ for } i \in [k]\} \\ &= \left\{ \sum_{j=m-k+1}^{m} x_j \boldsymbol{b}_i^* : \sum_{j=m-i+1}^{m} x_j^2 \|\boldsymbol{b}_j^*\|^2 \leq (c \cdot R_i)^2 \text{ for } i \in [k] \right\}. \end{aligned} \tag{11}$$

Thus, by using the appropriate rotation that transforms the point $\sum_{j=m-k+1}^{m} x_j \boldsymbol{b}_i^*$ into $(x_{m-k+1}\|\boldsymbol{b}_{m-k+1}^*\|, \ldots, x_m\|\boldsymbol{b}_m^*\|)$, we can find that the searching space is the rotation of the object

$$\left\{ (x_1, \ldots, x_k) \in \mathbb{R}^k : \sum_{i=1}^{\ell} x_i^2 < (c \cdot R_\ell)^2 \text{ for } \forall \ell \in [k] \right\}.$$

For simplicity, one can define the set C_k as below

$$C_k = \left\{ (x_1, \ldots, x_k) \in \mathbb{R}^k : \sum_{i=1}^{\ell} x_i^2 < R_\ell^2 \text{ for } \forall \ell \in [k] \right\}. \tag{12}$$

By the Gaussian heuristic over the projective sublattice $\pi_{m-k}(B)$, the number of lattice points in the space (11) is

$$\frac{c^k \mathrm{vol}(C_k)}{\prod_{i=m-k+1}^{m} \|\boldsymbol{b}_i^*\|},$$

and is the approximation of the number of touched nodes at depth k. Therefore, the cost of enumeration, which is defined by the total number of nodes touched by the tree searching algorithm, is given as follows.

$$N = \frac{1}{2} \sum_{k=1}^{m} \frac{c^k \mathrm{vol}(C_k)}{\prod_{i=m-k+1}^{m} \|\mathbf{b}_i^*\|}. \tag{13}$$

Note that the factor $1/2$ comes from the symmetry in the shortest vector computation, and it is vanished when we consider the closest vector problem and its variants.

A.1.5 Complexity measurements

Several measurements have been proposed for security estimation whereas our estimation is based on the number of nodes searched in the enumeration tree [22, 28]. However, we need to be careful to compare other measurements such as computing time in seconds [4, 27] or the number of logical gates.

A typical efficient implementation of tree searching algorithm needs to compute the integer linear combination of Gram-Schmidt coefficients

$$\sum_{i=m-k+1}^{m} y_i \mu_{i,m-k+1} \tag{14}$$

by floating numbers with some accuracy to check whether the square of projective length $\|\pi_{m-k+1}(\boldsymbol{v})\|^2 = s^2 + (\text{computed term in above } k-1 \text{ terms })$ exceeds the bound. That is, the complexity to check a node at depth k includes $k+1$ multiplication, $(k-1)+1$ addition and 1 comparison operations over floating numbers. Denoting the cost depending on each measurement for a node at depth k by $Cost_{Measure}(k)$, the complexity is evaluated as

$$EC_{Model,single,Measure}(B; params) = \frac{1}{2} \sum_{k=1}^{m} \frac{c^k \mathrm{vol}(C_k) \cdot Cost_{Measure}(k)}{\prod_{i=m-k+1}^{m} \|\mathbf{b}_i^*\|}. \tag{15}$$

Here, B and *parmas* are respectively the lattice basis and given parameters for models such as success probability.

Since our target is the lower bounds, we provide how many costs are required at minimum in each unit to process one node in the searching tree. In other words, we propose how to bound $Cost_{Measure}(k)$ from lower for $Measure \in \{gates, time\}$, which means that in the measure of number of logical gates as required in the NIST's criteria [33], and in the single-thread CPU time.

Below we assume that floating point numbers with a suitable precision are used to approximate and compute the Gram-Schmidt coefficients and lengths.

Computing time in seconds: To the best of our knowledge, the currently known fastest implementation for lattice vector enumeration can process about $5.0 \cdot 10^7$ nodes per second in a single thread in a practical dimension about 100. (A bit less than $6.0 \cdot 10^7$ in [7] and about $3.3 \cdot 10^7$ in [32]).

On the other hand, since the latest CPU can carry out many floating point operations per a cycle, a highly optimized implementation for each CPU can be faster. For example, the latest Intel CPU has two fused multiply add (FMA) units that can compute in one clock the vector multiplication operation $(a_1, \ldots, a_8) \cdot (b_1, \ldots, b_8) \mapsto (a_1 \cdot b_1, \ldots, a_8 \cdot b_8)$ and vector sum operation $(a_1, \ldots, a_8) \mapsto \sum_{i=1}^{8} a_i$ over eight 64-bit floating point numbers. Thus, the sum (14) at depth k can be computed in $2\lceil k/8 \rceil$ cycle clocks in theory that neglects the speed of memory transfer. Assuming the computation of squaring and comparison can be done in 2 cycle clocks, the total clock cycles need to process the node at depth k is at least $2\lceil k/8 \rceil + 2$. Furthermore, since recent common CPUs can work in about 5GHz clock at maximum in a standard environment, the timing cost lower bound would be

$$Cost_{time}(k) = 2.0 \cdot 10^{-10} \cdot (\lceil k/8 \rceil + 2) \tag{16}$$

This is about 10 times faster than the known fastest implementation for dimensions satisfying $(\lceil k/8 \rceil + 2) = 10$.

Hence, substituting it to (15), we get

$$EC_{Model,single,time}(B; params) > 10^{-10} \cdot \sum_{k=1}^{m} \frac{c^k \mathrm{vol}(C_k) \cdot (\lceil k/8 \rceil + 2)}{\prod_{i=m-k+1}^{m} \|\mathbf{b}_i^*\|} \text{ [sec]}. \tag{17}$$

and will give our theoretical lower bound by providing inequality for $\mathrm{vol}(C_k)$.

Number of logical gates: By the argument in Section A.3.2, we can claim the minimum dimension n used in the lattice reduction, which is slightly smaller than the whole lattice dimension m in enumeration. The experimental heuristic analysis by Nguyen-Stehlé [34, Heuristic. 4] claims that $d = 0.25n + o(n)$ bits are necessary for the mantissa in the floating point operation [4] to carry out the LLL algorithm. Since it has a small order term, considering a margin, we assume that $d = \lfloor 0.2n \rfloor$ bits is the lower bound of necessary precision.

We estimate the number of necessary logical gates to treat this precision of floating point numbers by the following rule. To carry out the addition (resp. the multiplication) of floating

[4] This is not the same as the bit-size of floating point numbers. For example, the standard 64-bit floating point number has 53-bit length mantissa.

point numbers with d-bit mantissa, the necessary size of circuit can be bounded by the d-bit adder (resp. multiplier) respectively. A simple d-bit adder consists of d full-adders and each one has typically 5 logical gates. Also, a simple d-bit times d'-bit multiplier also consists of $d \cdot d'$ full adders. Since the coefficients y_i in the sum (14) are typically small integers, we assume they can be represented by 8 bits. Hence, a possible logical gate to compute summing computation (14) includes k multiplication of d-bit numbers and $d' = 8$ bit integers, and $k-1$ addition of d-bit numbers. Totally, it requires

$$Cost_{gates}(k) := k \cdot d \cdot 8 \cdot 5 + (k-1) \cdot d \cdot 5 = 45kd - 5d = 9kn - n \tag{18}$$

logical gates at least. Again, substituting its local cost to (15), we obtain

$$EC_{Model,single,gates}(B; params) > \frac{1}{2} \sum_{k=1}^{m} \frac{c^k \mathrm{vol}(C_k) \cdot (9km - m)}{\prod_{i=m-k+1}^{m} \|\mathbf{b}_i^*\|}. \tag{19}$$

and the theoretical lower bound will be provided via our lower bound for $\mathrm{vol}(C_k)$.

A.1.6 Our complexity models

We define several models to discuss the complexity of enumeration algorithm. From now on, we use $EC_{Model,Alg,Measure}(B; params)$ to denote the optimal enumeration cost (15) where the target model $Model \in \{prob, many, LWE\}$, usage of enumeration algorithm $Alg \in \{single, multi\}$, and complexity measurement $Measure \in \{nodes, gates, time\}$, which will be defined in this section. Moreover, we will use $UB_{Model,Alg,Measure}(B; params)$ and $LB_{Model,Alg,Measure}(B; params)$ for the upper bounds (in Section A.5.1) and the lower bounds (in Section A.2), respectively.

Particularly, we will use $LB_{many,multi,Measure}(B; N = 1, c)$ for the lower bound of lattice reduction algorithm and use $LB_{LWE,multi,Measure}(B; p, s)$ for the lower bound of attacking LWE problem and our parameter setting.

In *the approximation setting* such as the approximate SVP, there exist many target points. By the Gaussian heuristic, the number of lattice points shorter than c within the searching area $c \cdot C_k$ is about $N = c^m \mathrm{vol}(C_m)/\mathrm{vol}(L)$. Thus, the best enumeration algorithm expected to find N lattice points is given by best combination of $(R_1, \ldots, R_m)$ that minimizes (15) subject to that $\mathrm{vol}(C_m) \geq N\mathrm{vol}(L)/c^m$. We denote this cost by $EC_{many,single,Measure}(B; N, c)$. where *single* denotes the single usage of enumeration subroutine, B is the input lattice basis, and $Measure \in \{nodes, gates, time\}$ means that we use $Cost_{Measure}(k)$ to measure the complexity.

The multiple usage of enumeration subroutine is considered by Gama-Nguyen-Regev [22]. For a given number N of wanted target points, one can run M trials of enumeration with a small target number N/M which might not be an integer. The total cost is the sum of $(M-1)$ randomizations, $(M-1)$ lattice reductions, and M enumerations with the low probability. We denote such cost by

$$\begin{aligned} &EC_{many,multi,Measure}(B; N, c) \\ &:= \min_M [(M-1) \cdot Cost(LatticeReduction) + M \cdot EC_{many,single,Measure}(B; N/M, c)]. \end{aligned}$$

Hence, we finished the definition of $EC_{many,Alg,Measure}(B; N, c)$.

Another interesting situation is the *unique setting*. It is a special situation of BDD problem with instance $(B, \boldsymbol{t})$ where the distribution $\mathcal{D}$ of error vector $\boldsymbol{e} = \boldsymbol{t} - \boldsymbol{v}$ is known. In the situation where we want to recover the error, searching radius is not fixed since the range of error distribution is not limited in general. Suppose we have an optimized radius and bounding coefficients which derive the searching space

$$U_m = \left\{ (x_1, \ldots, x_m) : \sum_{i=1}^{\ell} x_i^2 < T_i^2 = (cR_i)^2 \text{ for } \forall\ \ell \in [m] \right\}.$$

Then, the success probability of the enumeration is given by the integral of $f_{\mathcal{D}}(\boldsymbol{x})$ over the searching space:

$$\Pr_{\boldsymbol{x} \leftarrow \mathcal{D}} [\boldsymbol{x} \in U_m] = \int_{U_m} f_{\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{20}$$

To analyse the LWE problem, we assume the error distribution is given by the *continuous Gaussian* whose probability density function is $f_{LWE}(\boldsymbol{x}) = \exp(-\pi \|\boldsymbol{x}\|^2/s^2)/s^m$ while a typical definition uses the *discrete Gaussian*. Such assumption makes us the discussion much analytic since the continuous density function is rotation invariant and matches the requirement of our geometric lemma used for analysis. In other words, with using the radial basis function $\phi_e(r) = e^{-\pi r^2/s^2}/s^m$, we can write $f_{LWE}(\boldsymbol{x}) = \phi_e(\|\boldsymbol{x}\|)$, and discuss a lower bound of cost (13) subject to that the success probability (20) is larger than the given threshold.

As the approximation setting, we use the notations $EC_{LWE,single,Measure}(B; p, s)$ and $EC_{LWE,multi,Measure}(B; p, s)$ to denote the costs about the LWE problem.

Finally, we introduce the probability model by Gama-Nguyen-Regev [22]. Under the reasonable assumption ([22, Assumption 3]), they assumed that the probability to find a vector $\boldsymbol{v}$ with using searching radius $c = \|\boldsymbol{v}\|$ is given by

$$\Pr_{(x_1,\ldots,x_m) \leftarrow S_m(\|\boldsymbol{v}\|)} \left[\sum_{i=1}^{\ell} x_i^2 < \|\boldsymbol{v}\|^2 \cdot R_\ell^2 \text{ for } \forall\ \ell \in [m] \right]. \tag{21}$$

Hence, the best enumeration algorithm of success probability p_0 is given by best combination of $(R_1, \ldots, R_m)$ that minimizes (13) subject to that (25) is larger than p_0. We denote such optimized cost by $EC_{prob,single,Measure}(B; p_0, c)$. Also, as the approximation setting, the cost under the multiple usage of enumeration algorithm is

$$\begin{aligned} &EC_{prob,multi,Measure}(B; p, c) \\ &:= \min_M [(M-1) \cdot Cost(LatticeReduction) + M \cdot EC_{prob,single}(B; p/M, c)]. \end{aligned}$$

A.2 Bounding cost for pruned lattice vector enumeration

A.2.1 Single usage of the enumeration algorithm

In order to bound the cost (15, we need to bound each volume factor $\mathrm{vol}(C_k)$ in (13). The following geometric lemma has a crucial role.

Lemma 4 *Let C_k be a finite k-dimensional object, i.e., the k-dimensional volume* $\mathrm{vol}(C_k) < \infty$. *Let τ_k be the radius such that $V_k(\tau_k) = \mathrm{vol}(C_k)$. Fix a radial basis function $r(\boldsymbol{x}) = \phi(\|\boldsymbol{x}\|)$ where $\phi(\|\boldsymbol{x}\|)$ is a positive decreasing function on the radius: $\phi(x) \geq \phi(y) \geq 0$ for any $0 \leq x \leq y$. Then we have*

$$\int_{C_k} r(\boldsymbol{x})d\boldsymbol{x} \leq \int_{B_k(\tau_k)} r(\boldsymbol{x})d\boldsymbol{x}. \tag{22}$$

Noting that similar results where $\phi(x)$ is a Gaussian function is already known [24, Lemma 7.1] and [46, Sect. 14.8] in another contexts and they also proved their claim by similar strategies. For future discussions, we give the proof for generic $\phi(x)$.

Proof. By $V_k(\tau_k) = \mathrm{vol}(C_k)$, it holds that $V := \mathrm{vol}(C_k \setminus B_k(\tau_k)) = \mathrm{vol}(B_k(\tau_k) \setminus C_k)$. Since $\phi(\|\boldsymbol{x}\|)$ is decreasing function, we have the inequalities

$$\int_{C_k \setminus B_k(\tau_k)} r(\boldsymbol{x})d\boldsymbol{x} \leq V \cdot \phi(\tau_k) \leq \int_{B_k(\tau_k)\setminus C_k} r(\boldsymbol{x})d\boldsymbol{x}.$$

Hence,

$$\int_{C_k} = \int_{C_k \cap B_k(\tau_k)} + \int_{C_k \setminus B_k(\tau_k)} \leq \int_{C_k \cap B_k(\tau_k)} + \int_{B_k(\tau_k)\setminus C_k} = \int_{B_k(\tau_k)}.$$

ending the proof. □

By using this lemma, we provide the lower bound complexity of the single and multiple usages of Gama et al.'s pruned enumeration [22] for the approximation setting and unique (LWE) settings.

Approximation setting: Fixing the pruning coefficients $R_1, \ldots, R_m$, the intermediate searching areas C_k are fixed by (12). Suppose we have an optimal set of pruning coefficients. Then, for any $k \leq m-2$,

$$\begin{aligned}\frac{N \cdot \mathrm{vol}(L)}{c^m} \leq \mathrm{vol}(C_m) &= \int_{C_k} \mathrm{vol}\{\boldsymbol{z} \in C_m : (z_1, \ldots, z_k) = \boldsymbol{x})\}d\boldsymbol{x} \\ &\leq \int_{C_k} \mathrm{vol}\{\boldsymbol{z} \in B_m(1) : (z_1, \ldots, z_k) = \boldsymbol{x})\}d\boldsymbol{x}\end{aligned}$$

where the volumes are the $(m-k)$-dimensional ones defined on the coordinates $(z_{k+1}, \ldots, z_m)$.

The latter integrating function

$$r(\boldsymbol{x}) = \mathrm{vol}\{\boldsymbol{z} \in B_m(1) : (z_1, \ldots, z_k) = \boldsymbol{x})\} = \begin{cases} B_{m-k}(\sqrt{1-\|\boldsymbol{x}\|^2}) & \text{if } \|\boldsymbol{x}\| \leq 1 \\ 0 & \text{if otherwise} \end{cases}$$

satisfies the requirement of Lemma 4. Thus, we have

$$\mathrm{vol}(C_m) \leq \int_{B_k(\tau_k)} r(\boldsymbol{x})d\boldsymbol{x} = V_m(1) \cdot I_{\tau_k^2}\left(\frac{k}{2}, \frac{m+2-k}{2}\right).$$

By the condition $\frac{N\cdot\mathrm{vol}(L)}{c^m} \leq \mathrm{vol}(C_m)$, we have the lower bound of the radius

$$\tau_k \geq \sqrt{I^{-1}_{N\cdot\mathrm{vol}(L)/V_m(c)}\left(\frac{k}{2}, \frac{m+2-k}{2}\right)}.$$

Substituting this bound with the equation $V_k(\tau_k) = \mathrm{vol}(C_k)$ in Lemma 4 to (13), we obtain our lower bound for the enumeration to find N vectors shorter than c is given as follows:

$$LB_{many,single,Measure}(B; N, c) = \frac{1}{2}\sum_{k=1}^{m} \frac{c^k V_k(1)\left[I^{-1}_{\frac{N\mathrm{vol}(L)}{V_n(c)}}\left(\frac{k}{2}, \frac{m+2-k}{2}\right)\right]^{k/2} \cdot Cost_{Measure}(k)}{\prod_{i=n-k+1}^{n}\|\mathbf{b}_i^*\|}. \tag{23}$$

This is valid for the parameters satisfying $N\mathrm{vol}(L) \leq V_m(c)$.

Learning with errors setting: Since each C_k is the k-dimensional projection of C_m, we have the inequality on the success probability. When the distribution is continuous Gaussian $\mathcal{C}_s$, we have the relation

$$(20) = \Pr_{\boldsymbol{x}\leftarrow\mathcal{C}_s^m}[\boldsymbol{x} \in C_m] < \Pr_{\boldsymbol{x}\leftarrow\mathcal{C}_s^m}[(x_1, \ldots, x_k) \in C_k] = \Pr_{\boldsymbol{x}\leftarrow\mathcal{C}_s^k}[(x_1, \ldots, x_k) \in C_k].$$

The RHS is $\int_{C_k}\phi_e(\|\boldsymbol{x}\|)d\boldsymbol{x}$ where $\phi_e(\|\boldsymbol{x}\|) = \exp(-\pi\|\boldsymbol{x}\|^2/s^2)/s^k$ and the integral function meets the requirement of Lemma 4. Thus, combining with (6), we have the inequality

$$p < \int_{C_k}\phi_e(\|\boldsymbol{x}\|)d\boldsymbol{x} = P\left(\frac{k}{2}, \frac{\pi\tau_k^2}{s^2}\right)$$

which implies the lower bound of the radius

$$\tau_k \geq \frac{s}{\sqrt{\pi}}\sqrt{P^{-1}\left(\frac{k}{2}, p\right)}.$$

Hence, substituting this into (13) we obtain our lower bound as

$$LB_{LWE,single,Measure}(B; p, s) = \frac{1}{2}\sum_{k=1}^{m} \frac{(s/\sqrt{\pi})^k V_k(1)\left[P^{-1}\left(\frac{k}{2}, p\right)\right]^{\frac{k}{2}} \cdot Cost_{Measure}(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|}. \tag{24}$$

Probability setting: In [22], they assume the probability model for the shortest vector problem. We introduce this model to use in the computer experiments to justify our new assumption (Assumption 2) although this model is not used for the LWE parameter setting. Under the reasonable assumption ([22, Assumption 3]), the probability to find a vector $\boldsymbol{v}$ with using searching radius $c = \|\boldsymbol{v}\|$ is given by

$$\Pr_{(x_1,\ldots,x_m)\leftarrow S_m(\|\boldsymbol{v}\|)}\left[\sum_{i=1}^{\ell} x_i^2 < \|\boldsymbol{v}\|^2 \cdot R_\ell^2 \text{ for } \forall\ \ell \in [m]\right]. \tag{25}$$

Then, the probability (25) is bounded upper as

$$\begin{aligned}(25) &\leq \Pr_{\boldsymbol{x}\leftarrow S_m(\|\boldsymbol{v}\|)}\left[\sum_{i=1}^{\ell} x_i^2 < \|\boldsymbol{v}\|^2 \cdot R_\ell^2 \text{ for } \forall\ \ell \in [m-2]\right] \\ &\quad \text{(by relaxed condition)} \\ &= \Pr_{\boldsymbol{x}\leftarrow B_{m-2}(1)}\left[\sum_{i=1}^{\ell} x_i^2 < R_\ell^2 \text{ for } \forall\ \ell \in [m-2]\right] = \frac{\mathrm{vol}(C_{m-2})}{V_{m-2}(1)}.\end{aligned} \tag{26}$$

This probability have a relation to the volume of cylinder intersection by

$$\tau_k \geq \sqrt{I^{-1}_{\mathrm{vol}(C_{m-2})/V_{m-2}(1)}\left(\frac{k}{2}, \frac{m-k}{2}\right)} \geq \sqrt{I_p^{-1}\left(\frac{k}{2}, \frac{m-k}{2}\right)}.$$

Thus, as the same argument to the situation of approximation setting, we obtain our lower bound for the enumeration of probability p and radius c:

$$LB_{prob,single,Measure}(B;p,c) = \frac{1}{2}\sum_{k=1}^{m} \frac{c^k V_k(1)\left[I_p^{-1}\left(\frac{k}{2},\frac{m-k}{2}\right)\right]^{\frac{k}{2}} \cdot Cost_{Measure}(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|}. \tag{27}$$

A.2.2 Multiple usage of the enumeration algorithm

Using our lower bound for single usage of enumeration algorithm, we can bound the cost of Gama-Nguyen-Regev's extreme pruning that uses multiple random bases. As mentioned in Section A.1.4, the cost of multiple usage is given by

$$\begin{aligned}&EC_{Model,multi,Measure}(B;p) = \\ &\min_M[(M-1)\cdot Cost(LatticeReduction) + M \cdot EC_{Model,single,Measure}(B;p/M)]\end{aligned}$$

where p stands for the success probability or the number of found vectors depending on the model. We estimate the lower bound cost by neglecting the cost for the lattice reduction

$$EC_{Model,multi,Measure}(B;p) > \min_M[M \cdot LB_{Model,single,Measure}(B;p/M)] \tag{28}$$

Remark that as we will show in the next section, the lower bound cost of lattice reduction is not zero if we start with a random (LLL-reduced) basis. However, in the situation of extreme pruning, the complete randomization, for instance, multiplying random unimodular matrix to the whole basis can achieve it, is too strong for our goal. Thus, in the sense of algorithm optimization, the quality of basis after randomization may not far from the original reduced basis if we have a good procedure as [2, Appendix. A] and [17]. This is the reason that we neglect the lattice reduction cost here. As we see in below, we do not need to consider the number M of used bases in the lower bounds.

In (29), (30) and (31) below, we use the notation $C(k)$ for $Cost_{Measure}(k)$ for simplicity.

Approximation setting: For the target number N of lattice points that we want to find if we use M randomized bases, at least N/M target numbers are necessary for each basis. It should be larger than N/M because of duplication of the found vectors. For these parameters, by a similar argument as above, the cost is bounded lower by using (23), and by the inequality (8), we have

$$\begin{aligned}
&M \cdot LB_{many,single,Measure}(B; N/M, c) > \frac{M}{2}\sum_{k=1}^{m} \frac{c^k V_k(1)\left[I^{-1}_{\frac{N\mathrm{vol}(L)}{MV_m(c)}}\left(\frac{k}{2}, \frac{m+2-k}{2}\right)\right]^{k/2} \cdot C(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|} \\
&> \frac{N\mathrm{vol}(L)}{4V_m(c)}\sum_{k=1}^{m} \frac{V_k(c)\cdot k \cdot B\left(\frac{k}{2}, \frac{m+2-k}{2}\right)\cdot C(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|} \\
&= \frac{N}{4}\sum_{k=1}^{m}\left[\prod_{i=1}^{m-k}\frac{\|\mathbf{b}_i^*\|}{c}\right]\cdot k \cdot \frac{V_k(1)}{V_m(1)}\cdot B\left(\frac{k}{2}, \frac{m+2-k}{2}\right)\cdot C(k) \\
&= \frac{N}{2}\sum_{k=1}^{m}\left[\prod_{i=1}^{m-k}\frac{\|\mathbf{b}_i^*\|}{c\sqrt{\pi}}\right]\cdot \Gamma\left(\frac{m+2-k}{2}\right)\cdot C(k) = \frac{N}{2}\sum_{k=1}^{m}\left[\prod_{i=1}^{k}\frac{\|\mathbf{b}_i^*\|}{c\sqrt{\pi}}\right]\cdot \Gamma\left(\frac{k+2}{2}\right)\cdot C(m-k) \\
&:= LB_{many,multi,Measure}(B; N, c).
\end{aligned} \tag{29}$$

Remark that if N is fixed, increasing of c implies the decreasing cost and increasing the length of found vectors.

Learning with Errors setting: We can show the cost lower bound of the primal attack by Liu-Nguyen [28] as follows:

$$\begin{aligned}
&M \cdot LB_{LWE,single,Measure}(B; p/M, s) > \frac{M}{2}\sum_{k=1}^{m} \frac{(s/\sqrt{\pi})^k V_k(1)\left[P^{-1}\left(\frac{k}{2}, \frac{p}{M}\right)\right]^{\frac{k}{2}} \cdot C(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|} \\
&> \frac{p}{4}\sum_{k=1}^{n} \frac{(s/\sqrt{\pi})^k V_k(1)\cdot k \cdot \Gamma(k/2)\cdot C(k)}{\prod_{i=n-k+1}^{n}\|\mathbf{b}_i^*\|} = \frac{p}{2}\sum_{k=1}^{n} \frac{s^k \cdot C(k)}{\prod_{i=n-k+1}^{n}\|\mathbf{b}_i^*\|} \\
&:= LB_{LWE,multi,Measure}(B; p, s).
\end{aligned} \tag{30}$$

Probability setting: For the lower bound for Gama-Nguyen-Regev's probabilistic model, we can show the lower bound by using (27) and (8):

$$\begin{aligned}
&M \cdot LB_{prob,single,Measure}(B; p/M, c) > \frac{M}{2}\sum_{k=1}^{m}\frac{V_k(c)\left[I^{-1}_{p/M}\left(\frac{k}{2},\frac{m-k}{2}\right)\right]^{\frac{k}{2}}\cdot C(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|}\\
&> \frac{p}{4}\sum_{k=1}^{m}\frac{V_k(c)\cdot k\cdot B\left(\frac{k}{2},\frac{m-k}{2}\right)\cdot C(k)}{\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|} = \frac{p}{2}\sum_{k=1}^{m}\frac{c^k\pi^{k/2}\Gamma(\frac{m-k}{2})\cdot C(k)}{\Gamma(\frac{m}{2})\prod_{i=m-k+1}^{m}\|\mathbf{b}_i^*\|}\\
&= \frac{p}{2\Gamma(\frac{m}{2})\mathrm{vol}(L)}\sum_{k=1}^{m}\prod_{i=1}^{k}\|\boldsymbol{b}_i^*\|\cdot c^{m-k}\pi^{\frac{m-k}{2}}\Gamma\left(\frac{k}{2}\right)\cdot C(m-k)\\
&= \frac{p\cdot c^m\pi^{m/2}}{2\Gamma(\frac{m}{2})\mathrm{vol}(L)}\sum_{k=1}^{m}\left[\prod_{i=1}^{k}\frac{\|\mathbf{b}_i^*\|}{c\sqrt{\pi}}\right]\cdot\Gamma\left(\frac{k}{2}\right)\cdot C(m-k) := LB_{prob,multi,Measure}(B; p, c)
\end{aligned} \tag{31}$$

where the equality from the second line to the third line holds by swapping index $m-k$ by k.

Concluding these inequalities, we have two remarks. First, we do not need to consider the number of randomized bases in the calculation of lower bounds. Second, since they are linear functions of probability or the number of target points, the speeding up of extreme pruning strategy is bounded by a constant independent from the number of bases and probabilities.

A.3 Bounding cost of lattice problems

We introduce our formula to bound the cost of lattice reduction that achieves the root Hermite factor δ_0. Unlike the argument of enumeration algorithm in the above section, we need to add a new heuristic assumption that was supported by our experiments.

The cost for attackers is typically given by

$$Cost(Problem) = \min\frac{Cost(LatticeReduction) + Cost(PointSearch)}{\text{Success probability of PointSearch}}.$$

Here, the minimum is over all typical lattice reduction algorithms and the pruned lattice enumeration algorithm. Parameters in each step are optimized via suitable preliminary simulations.

In this paper, we will argue the attacker's cost to solve a lattice problem in the unique setting by

$$\begin{aligned}
&Cost(Problem)\\
&\geq \min\left[Cost(LatticeReduction) + \frac{Cost(PointSearch, prob = p)}{p}\right]\\
&= \min_{\delta_0, B\in LR(\delta_0,m)}\left[CostLR(\delta_0, m) + \frac{EC_{Model,multi,Measure}(B; p)}{p}\right].
\end{aligned} \tag{32}$$

where p is the success probability of point search and $LR(\delta_0, m)$ denotes the set of output of lattice reduction algorithms that achieve the root Hermite factor δ_0 in m-dimensional lattice.

Our goal is to establish lower bounds of the both factors that depend on only δ_0 and dimension m. Concretely, we provide two lower bounds

- $\min_{\delta_0, B \in LR(\delta_0,m)}[EC_{Model,multi,Measure}(B;p)] \geq L_1(\delta_0, m, p)$ (in Section A.3.1), and
- $CostLR(\delta_0, m) \geq L_2(\delta_0, m)$ (in Section A.3.2)

in simple closed formulas. Then the minimizing problem in the last line of (32) is to be the following form:

$$Cost(Problem) \geq \min_{\delta_0} \left[L_2(\delta_0, m) + \frac{L_1(\delta_0, m, p)}{p} \right]. \tag{33}$$

Again we remark that the probability p is cancelled out when we analyze under the Gaussian model (30). Also, the range of δ_0 should be bounded as follows: $V_m^{-1/m^2} \lessapprox \delta_0 \lessapprox 1.022$ is from the Gaussian heuristic and the experimental observation in [21].

A.3.1 Bounding enumeration cost over a reduced basis

Fix the root Hermite factor δ_0. The cost we want to bound in this section is

$$CostEnum(LR(\delta_0, m)),$$

that is, the cost (13) for a given radius and success probability, and the Gram-Schmidt lengths of output of a lattice reduction algorithm; $\|\boldsymbol{b}_1\| = \delta_0^m \mathrm{vol}(L)^{1/m}$ is known but other values are unknown.

To give a simple formula for the lower bound cost, we performed the experiments that compare the enumeration cost (13) and the lower bound cost (27) for the practical reduced basis B, and the corresponding δ_0-GSA basis B_{δ_0}. Figures 3 and 4 in the experimental section show an example of cost comparison. From the observation, we find the costs of GSA bases are typically lower than that of the original basis.

Assumption 2 *Let $Model$ be in $\{prob, many, LWE\}$. For parameters (success probability p, searching radius c, Gaussian parameter s) in a reasonable range and an output basis B of a typical lattice reduction algorithm, the cost of enumeration $EC_{Model,single,Measure}(B; params)$ is larger than the lower bound cost of δ_0-GSA basis, $LB_{Model,single}(B_{\delta_0}; params)$ where $\delta_0 = (\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/m})^{1/m}$.*

Additional explanation on this assumption is necessary. First, this assumption is a combination of experimental observation that $EC_{prob,single}(B; p, c) \geq EC_{prob,single}(B_{\delta_0}; p, c)$ holds in many situations (see Figure 3) and our theory that proves

$$EC_{Model,single,Measure}(B_{\delta_0}; params) \geq LB_{Model,single,Measure}(B_{\delta_0}; params).$$

Second, the assumption on the single usage derives the same relation on the multiple usage. Actually, for a suitable M,

$$\begin{aligned} &EC_{Model,multi,Measure}(B; params) \geq M \cdot EC_{Model,single,Measure}(B; params/M) \\ &\geq M \cdot LB_{Model,single,Measure}(B_{\delta_0}; params/M) = LB_{Model,multi,Measure}(B_{\delta_0}; params). \end{aligned} \tag{34}$$

Accepting the assumption, we can treat lattice bases by only two parameters δ_0 and m that makes high experimental reproducibility for security estimation.

Noting that it is possible to consider an artificial counter-example to break this assumption. For example, for an LLL-reduced basis, one can apply the strong BKZ algorithm to its projected sublattice $\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_m)$. However, we think that typical reduction algorithms used in lattice-based attacks do not have such behaviours.

A.3.2 Bounding Cost for Lattice Reduction

This section provides our lower bound formula for the cost $CostLR(\delta_0, m)$ of lattice reduction algorithms that achieve the root Hermite factor δ_0. Clearly,

$$CostLR(\delta_0, m) > CostLR(\delta_0, n) \text{ holds for } m > n. \tag{35}$$

However, there exists the lower bound on the dimension deduced from the Gaussian heuristic, i.e., it needs to hold $\delta_0^m > V_m^{-1/m}$. If not, it is hard to exist a vector shorter than $\delta_0^m \mathrm{vol}(L)^{1/m}$ and the cost bound is not valid. Throughout this section, we fix m as the smallest integer satisfying this inequality, thus,

$$\delta_0^n < V_n^{-1/n} \text{ holds for integers } n < m. \tag{36}$$

Except for the LLL algorithm, all the known lattice reduction algorithms for finding a vector shorter than $c = \delta_0^m \mathrm{vol}(L)^{1/m}$ must have at least one calling of an enumeration subroutine working over a first sublattice $B_n = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ with the radius c and target volume $\mathrm{vol}(C_n) \cdot c^n \geq \mathrm{vol}(B_n)$. Explicitly written, the cost relation is

$$CostLR(\delta_0, m) = \min_{n, \tilde{\ell_n}} \Big[CostLR(m, \tilde{\ell_n}) + EC_{many,multi,Measure}(B_n; 1, c) \Big] \tag{37}$$

where $\tilde{\ell_n} := (\|\boldsymbol{b}_1\|, \ldots, \|\boldsymbol{b}_n\|)$ is the possible sequence of Gram-Schmidt lengths satisfying $\ell_1 \geq \delta_0^m \mathrm{vol}(L)^{1/m}$, and $CostLR(m, \tilde{\ell_n})$ is the minimum cost to find an m-dimensional lattice basis B_m such that $\|\boldsymbol{b}_i^*\| = \ell_i$ for all $i \in [n]$. Also, B_n is the sublattice of B_m having such Gram-Schmidt lengths.

Proposition 1 *The subdimension n satisfies $n = m$ under Assumption 1 in the cost model (37).*

Proof. Suppose $n < m$ and the enumeration subroutine runs over the sublattice $B_n = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ and it finds a vector shorter than $\delta_0^m \mathrm{vol}(L)^{1/m}$. We show a contradiction. By Assumption 1 over the sublattice B_{n+1}, we have $\|\boldsymbol{b}_{n+1}^*\| < V_{n+1}^{\frac{1}{n+1}} \mathrm{vol}(B_{n+1})^{\frac{1}{n+1}}$. Thus, $\|\boldsymbol{b}_{n+1}^*\|^{\frac{n}{n+1}} < V_{n+1}^{\frac{1}{n+1}} \prod_{i=1}^{n+1} \|\boldsymbol{b}_i^*\|^{\frac{1}{n+1}}$. Multiplying $\prod_{i=1}^{n} \|\boldsymbol{b}_i^*\|^{\frac{n}{n+1}}$ and taking n-th root for both sides, we have $\prod_{i=1}^{n+1} \|\boldsymbol{b}_i^*\|^{\frac{1}{n+1}} < V_{n+1}^{\frac{1}{n(n+1)}} \cdot \prod_{i=1}^{n} \|\boldsymbol{b}_i^*\|^{1/n}$. Since the volume function V_k is strictly decreasing

for large k, the relation $V_{n+1}^{\frac{1}{n(n+1)}} < V_n^{\frac{1}{n(n+1)}} = (V_n^{1/n})^{\frac{1}{n+1}}$ holds and it is bounded upper by $\delta_0^{-\frac{n+1}{n}}$ by using (37). Hence, it derives the induction formula

$$\mathrm{vol}(B_{n+1})^{\frac{1}{n+1}}\delta_0^{\frac{n+1}{n}} < \mathrm{vol}(B_n)^{\frac{1}{n}}. \tag{38}$$

Thus,

$$GH(B_n) > \delta_0^n \mathrm{vol}(B_n)^{1/n} > \delta_0^{n+\frac{n+1}{n}+\cdots+\frac{m}{m-1}}\mathrm{vol}(B_m)^{1/m} > \delta_0^m \mathrm{vol}(B_m)^{1/m}.$$

Therefore, it is difficult to exist a vector shorter than $\delta_0^m \mathrm{vol}(B_m)^{1/m}$ in the sublattice B_n if $n < m$. □

Neglecting cost for lattice reduction in (37), we have $CostLR(\delta_0, m) > CostENUM(m, \tilde{\ell_m})$ where $\tilde{\ell_m}$ is from a reduced basis so that $\ell_1 \geq c$. Using Assumption 2 and its multiple version (34), bounded lower by $LB_{many,multi,Measure}(B_{\delta_0}; N, c)$ with $N = 1$ and $\|\boldsymbol{b}_i^*\| = r^{(i-1)/2}$. In conclusion, our lower bound for lattice reduction to find a short vector is

$$CostLR(\delta_0, m) > \frac{1}{2}\sum_{k=1}^{m} r^{\frac{k(k-1)}{4}} \cdot \pi^{-k/2} \cdot \Gamma\left(\frac{k}{2}\right) := LBLR(\delta_0) \tag{39}$$

for the smallest integer n such that $\delta_0^n > V_n^{-1/n}$ and $r = \delta_0^{\frac{-4n}{n-1}}$.

Remark that this is the dimension independent formula as the formulas in previous works [4,27]. However, the reason of independence is completely different from them since we use the relation (35) while the existing works have used the interpolation of the experimental computing time in the fixed dimensions.

A.4 Our parameters for the LWE problem

In this section we provide our LWE parameter selection.

A.4.1 Our parameter setting

For the LWE problem parameter (n, q, s) and the lattice dimension m (= number of samples) to be optimized, the attack consists of the lattice reduction part to achieve a root Hermite factor δ_0 and the enumeration step to solve the BDD. By the above argument, our cost lower bound under the model (33) is given by setting $L_1(\delta_0, m, p) =$(30) with the GSA basis, and $L_2(\delta_0, m) =$(39). Therefore, the minimum cost is achieved by $\min_{m,\delta_0}[(39) + (30)]$ with $Measure \in \{nodes, gates, time\}$. For readability, the explicit formula is provided as follows

$$\begin{aligned} &CostLWE_{Measure}(n, q, s) = \\ &\min_{m,\delta_0}\left[\frac{1}{2}\sum_{k=1}^{n'} r^{\frac{k(k-1)}{4}} \cdot \pi^{-k/2} \cdot Cost_{Measure}(n'-k) \cdot \Gamma\left(\frac{k}{2}\right) + \frac{1}{2}\sum_{k=1}^{m} \frac{s^k \cdot Cost_{Measure}(k)}{q^{\frac{nk}{m}} r^{\frac{mk-2m-k^2}{4}}}\right]. \end{aligned} \tag{40}$$

Table 8. Example estimations for LWE in the Lindner-Peikert parameter [27] by using (40) with cost measures $Cost_{nodes}(k) = 1$, $Cost_{time}(k) = 2.0 \cdot 10^{-10} \cdot (\lceil k/8 \rceil + 2)$ and $Cost_{gates}(k) = 45kd - 5d = 9kn - n$.

n	q	s	$\log_2$ of Time [sec.]	$\log_2(\sharp nodes)$	$\log_2(\sharp gates)$	Comments	[28]	[5]
128	2053	6.77	-24.0	6.11	21.46	Toy param.	23.6	17
192	4093	8.87	1.78	30.73	47.48	Low param.	62.8	59
256	4093	8.35	26.86	55.34	73.15	Medium param., claimed as 128-bit security	105.5	102
320	4093	8.00	56.00	84.12	102.73	High param.	-	147
136	2003	13.01	-3.78	25.32	41.67	Example param. in [31]	-	-
214	16381	7.37	-8.14	21.07	37.18	Example param. in [31]	-	-

for each $Measure \in \{nodes, time, gates\}$. Here, n' and r in the bracket are fixed by that the smallest integer n' such that $\delta_0^{n'} > V_{n'}^{-1/n'}$ and $r = \delta_0^{\frac{-4n'}{n'-1}}$ respectively.

Recall that $Cost_{nodes}(k) = 1$ for counting nodes, $Cost_{time}(k) = 2.0 \cdot 10^{-10} \cdot (\lceil k/8 \rceil + 2)$ for measuring single thread time (16), and $Cost_{gates}(k) = 45kd - 5d = 9kn - n$ for counting number of logical gates (18).

Table 8 shows revisited cost and parameters introduced in Lindner-Peikert [27]. Since they are lower bounds, they are much smaller than the other works for average complexities such as [28].

Table 9 shows the lower bound complexity for our parameters: $s = 3$, $q = 8192$ and various LWE dimensions n, which includes the parameters for LOTUS.

A.4.2 Comparison on the cost of lattice reduction with previous models

Many previous works have given models of the relation between computing time and the achieved root Hermite factor δ_0. We provide a short survey together with our comments below.

Lindner-Peikert [27] estimated $\log_2(t_{BKZ}[\text{sec}.]) = \frac{1.8}{\log_2(\delta)} - 110$ from their experiments using NTL-BKZ for q-ary lattices derived from random LWE instances. They claimed it as a practical lower bound line from their curve fitting. However, this model may hard to believe since it derives a subexponential algorithm for the LWE problem [4].

Albrecht et al. [3] estimated $\log_2(t_{BKZ}[\text{sec}.]) = \frac{0.009}{\log_2(\delta)^2} - 27$ that is an extrapolation of the points from BKZ 2.0 simulation for the expecting time in [28]. However, this model also contradicts to the theoretical upper bound result by Schnorr [41] as we mention below.

Albrecht et al. [2, 4] proposed $\log_2(t_{BKZ}[\text{sec}.]) = \Theta\left(\frac{\log(1/\log\delta)}{\log\delta}\right)$ under the assumption that we have a β-dimensional SVP oracle that works in time $2^{\Theta(\beta)}$.

Table 9. Our lower bound cost estimation for LWE to set the parameters used in our proposals by using (40) with cost measures $Cost_{nodes}(k) = 1$ and $Cost_{gates}(k) = 45kd - 5d = 9kn - n$.

n	q	s	$\log_2(\sharp nodes)$	$\log_2(\sharp gates)$	Comments
544	8192	3.0	122.56	141.92	
576	8192	3.0	136.19	155.76	Our `lotus-params128` parameter (AES-128 and SHA3-256 strength)
608	8192	3.0	150.24	170.01	
640	8192	3.0	164.71	184.66	
672	8192	3.0	179.55	199.67	
704	8192	3.0	194.77	215.06	Our `lotus-params192` parameter (AES-192 and SHA3-384 strength)
736	8192	3.0	210.34	230.78	
768	8192	3.0	226.26	245.85	
800	8192	3.0	242.49	263.23	
832	8192	3.0	259.04	279.92	Our `lotus-params256` parameter (AES-256 strength)

The comparison among these estimations and our lower bound (39) is summarized in Figure 1. Furthermore, experimental estimations can be derived from the SVP challenge contest [40]. Most of the current records are achieved by Kashiwabara and Teruya. We plot δ_0 derived from their approximation factor and computing single core time that was simply adds their claimed number of CPU cores and days. It shows that how danger an extrapolation from a small range of experiments is.

Problem in the estimation by Albrecht et al. [3]: Besides a lack of the dimension factor, we find their model contradicts to Schnorr's result [41]. His block-type lattice reduction algorithm finds a lattice vector shorter than $(6k^2)^{n/2k}\lambda_1(L)$ in time $O(n^2(k^{k/2+o(k)} + n^2)\log\max||\boldsymbol{b}_i||)$. Here k is a parameter in $[n]$ for time-approximation trade off. Letting $k = n/2$, we have a $\frac{3}{2}n^2$ approximation algorithm working in time $2^{O(n\log n)}$. On the other hand, the approximation factor $\frac{3}{2}n^2$ corresponds to $\delta_0^n = \frac{3}{2}n^2 V_n(1)^{-1/n} = \Theta(n^{2.5})$ and thus $\log^2(\delta_0) = \Theta(\frac{\log^2 n}{n^2})$. For this approximation factor, Albrecht et al.'s model predicts that time is $2^{\Theta(1/\delta_0^2)} = 2^{\Theta(n^2/\log_2^2 n)}$. This is exactly larger than Schnorr's theoretical bound. Therefore, the hardness estimation based on this in large dimensions would be overestimate and the parameter is smaller than the necessary one for security requirement though it is very simple and clearly reproducible.

The same problem has been occurred in [15] since they have used the estimation of [3]. For example, our `lotus-param128` parameter will have the strength around 2^{268} seconds to attack if using [15]. Similarly, `lotus-param192` and `lotus-param256` parameters will have the strength around 2^{379} and 2^{504} seconds to attack, respectively. In other words, our parameters are very conservative.

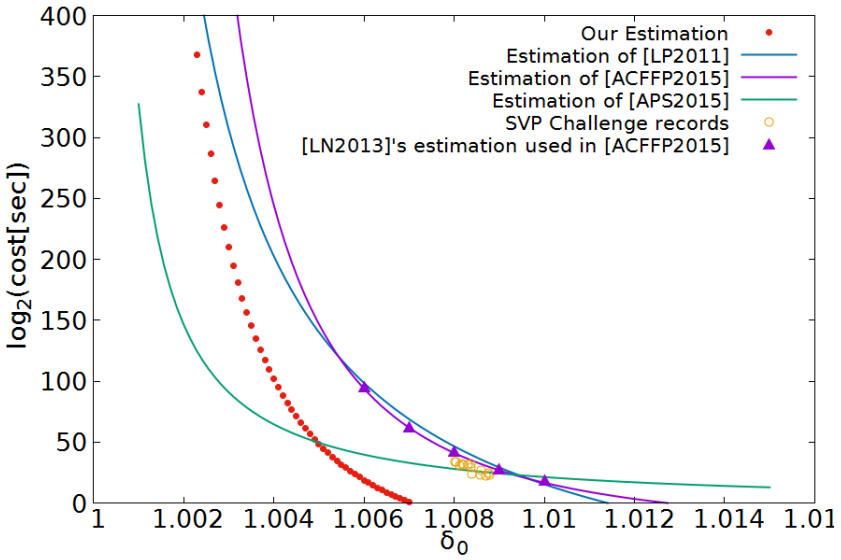

Fig. 1. Comparison among several models to achieve the root Hermite factor δ_0. Our estimation $LBLR(\delta_0)$ in (39) computing with $Cost_{time}(k)$; [LP2011] is [27]; [ACFFP2015] is [3]; [APS2015] is [4] with the constant $c = 0.05$. Also, the orange circles indicates the recent records in 130 to 150 dimensions in SVP challenge.

Space realizability of estimation by Albrecht et al. [2, 4]: In this estimation, they assume that one can solve the β-dimensional shortest vector problem in $2^{\Theta(\beta)}$. This is a reasonable assumption in "time complexity" since the latest sieve algorithm by Bai-Laarhoven-Stehlé [9] works in $2^{0.4812m+o(m)}$ times and $2^{0.1887m+o(m)}$ spaces. However, considering space complexity, it is hard to believe to attack a 256-bit security of the LWE problem. That is, if $0.4812m = 256$, the space requirement becomes about 2^{100}. One cannot treat such gigantic data storage even in the post quantum era. For example, the total amount of the data in the world is about 16.1 zettabytes $\approx 2^{74}$ bytes in 2016, and it grows to be $10 \approx 2^{3.3}$ times larger in the next 10 years [1]. From this viewpoint there could not exist an entity or organization to treat 2^{100} bytes to attack lattice based schemes.

A.5 Computer experiments

A.5.1 Sharpness of our bound

We perform our computer experiment to compare our lower bound, the estimation by the method in [22] and an upper bound by setting artificial bounding function. Recall that the paper by Gama-Nguyen-Regev [22] provided an efficient method to compute good upper and lower bounds for the cost (13) and probability (25) for given parameters. We use them as subroutines.

Systematic Upper bound: To show the sharpness of our lower bound, and to show that our subroutine to find $EC_{prob,single,nodes}(B;p,c)$ works well, we give a method to compute upper bound cost $UB_{prob,single,nodes}(B;p,c)$. In contrast to the lower bound situation, a possible upper bound can be computed by setting feasible bounding coefficients. Thus, using a finite set of bounding coefficients whose probability is larger than p, an upper bound is given by the minimum cost among the coefficients. For this purpose, we define the pruning coefficients in dimension m parametrized by $\alpha \in \mathbb{R}$ and $j \in [m]$ by $R_i(\alpha, j) = \min((i/j)^\alpha, 1)$.

For given parameters $(||\mathbf{b}_1^*||, \ldots, ||\mathbf{b}_n^*||, c, p)$, and for each integer j, we can compute α so that the lower bound probability is p by the binary search. Then, by computing the minimum of upper bound cost among all j, we can obtain an upper bound of enumeration cost of probability p.

Comparison: Figure 2 shows the comparison among the systematic upper bound

$$UB_{prob,single,nodes}(B;p,c),$$

cost of pruned enumeration $EC_{prob,single}(B;p,c)$, lower bound cost of single usage

$$LB_{prob,single,nodes}(B;p,c)$$

and the lower bound for multiple usage $LB_{prob,multi,nodes}(B;p,c)$. We use the radius $c = GH(L)$ and various success probability. We used two bases: a PBKZ-60 reduced 180 dimensional basis and a PBKZ-20 reduced 120 dimensional basis. The lattice bases are published instances of SVP challenge, and PBKZ-β stands for Aono et al.'s progressive BKZ [7] with target blocksize β. In 180 dimension, for $p < 10^{-4}$, we can see ratio

$$EC_{prob,single,nodes}(B;p,c)/LB_{prob,single,nodes}(B;p,c)$$

is less than 10^4.

A.5.2 Evidence for our Assumption 2

Recall that Assumption 2 claims that

$$EC_{Model,single,Measure}(B;params) \geq LB_{Model,single,Measure}(B_{\delta_0};params)$$

for each $Model \in \{prob, many, LWE\}$ and $Measure \in \{nodes, gates, time\}$, reasonable parameters, and $\delta_0 = (\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/m})^{1/m}$.

Figure 3 shows that the comparison among $EC_{prob,single,node}(B;p_0 = 10^{-3}, c = GH(L))$, $EC_{prob,single,node}(B_{\delta_0};p,c)$ and $LB_{prob,single,node}(B_{\delta_0};p,c)$ for many reduced basis. We can see the assumption holds in the all cases.

Additional experiments over several probabilities and dimensions are shown in Figure 4. These graphs show the ratios

$$\frac{EC_{prob,single}(B;p_0,c)}{EC_{prob,single}(B_{\delta_0};p_0,c)} \quad \text{and} \quad \frac{EC_{prob,single}(B;p_0,c)}{LB_{prob,single}(B_{\delta_0};p_0,c)} \tag{41}$$

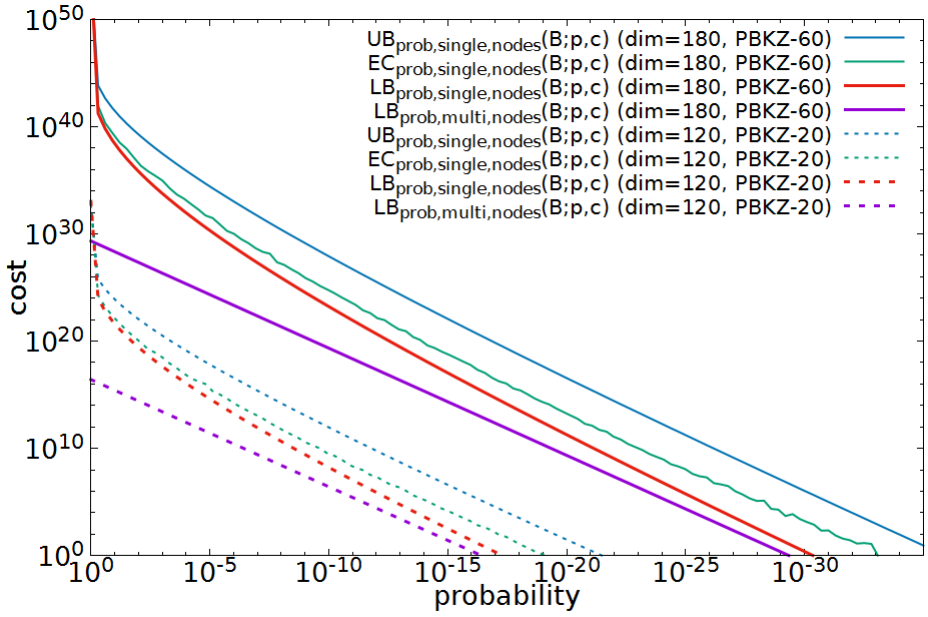

Fig. 2. For two lattice bases, compare the systematic upper bound $UB_{prob,single,nodes}(B;p,c)$, cost of pruned enumeration $EC_{prob,single,nodes}(B;p,c)$, lower bound cost of single usage $LB_{prob,single,nodes}(B;p,c)$, and the lower bound for multiple usage $LB_{prob,multi,nodes}(B;p,c)$. We used a PBKZ-60 (resp. PBKZ-20) bases of random 180-dim. (resp. 120-dim) lattice.

in red and blue curves respectively. Assumption 2 claims that the right values (blue curves) always above 1. For a majority of situations, $EC_{prob,single}(B;p_0,c) > EC_{prob,single}(B_{\delta_0};p_0,c)$ holds and for all situations in our experiments, $EC_{prob,single}(B;p_0,c) > LB_{prob,single}(B_{\delta_0};p_0,c)$ holds. In the bottom-right picture, we used our version of primal-dual progressive BKZ algorithm that is a naïve combination of the primal-dual BKZ [32] and the progressive BKZ [7].

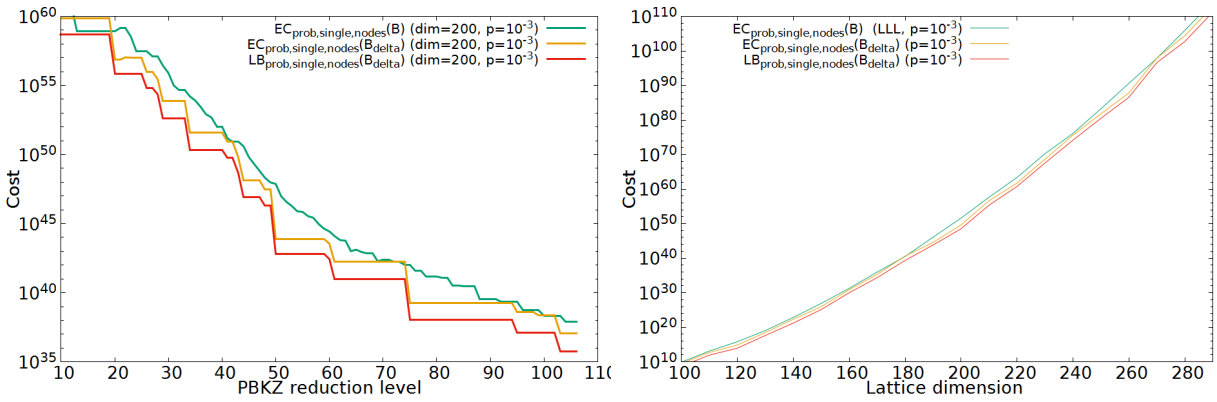

Fig. 3. Comparison of various simulated costs between real reduced basis and equivalent GSA basis. (**Left**) 200-dimensional bases reduced by progressive BKZ [7] with many block-sizes; (**Right**) For LLL reduced basis of many dimensions; In all experiments, parameters are $p = 10^{-3}$ and $c = \mathrm{GH}(L)$.

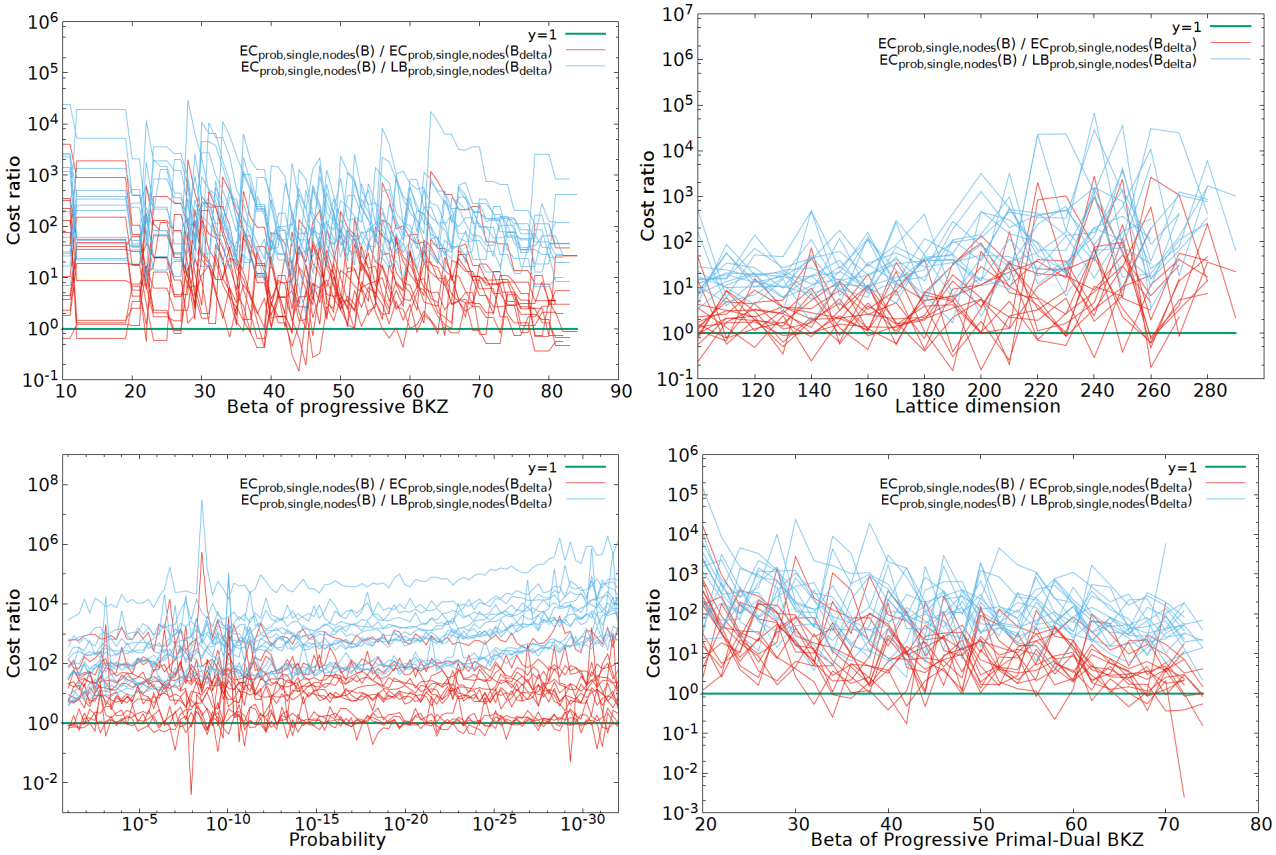

Fig. 4. Experiments to compare the ratios (41) for various lattice dimension m, searching radius c, success probability p and target reduced level β of progressive BKZ. (**Top-Left**) Experiments using 12 bases, $m = 200, c = \mathrm{GH}(L), p = 10^{-3}$ and various β-reduced bases; (**Top-Right**) Experiments in using bases, $c = \mathrm{GH}(L), p = 10^{-3}, \beta = 40$ and various dimensions; (**Bottom-Left**) Experiments using 16 bases, $m = 200, c = \mathrm{GH}(L), \beta = 40$ and various success probabilities; (**Bottom-Right**) Experiments using 16 bases, $m = 200, c = \mathrm{GH}(L), p = 10^{-3}$. The bases are the outputs of our preliminary version of primal-dual progressive BKZ.