Ciphertext-Only Attacks against Compact-LWE Submitted to NIST PQC Project

Haoyu Li[1,2], Renzhang Liu[3], Yanbin Pan[1], Tianyuan Xie[1,2]

[1]Key Laboratory of Mathematics Mechanization, NCMIS,
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
Beijing 100190, China
[2] School of Mathematical Sciences, University of Chinese Academy of Sciences,
Beijing 100049, China
[3] Westone Cryptologic Research Center, Westone Information Industry INC. Beijing

Abstract. In 2017, Liu, Li, Kim and Nepal submitted a new public-key encryption scheme Compact-LWE to NIST as a candidate of the standard of post-quantum cryptography. Compact-LWE features its structure similar to LWE, but with different distribution of errors. Liu, Li, Kim and Nepal thought that the special error distribution they employed would protect Compact-LWE from the known lattice-based attacks. Furthermore, they recommended a set of small parameters to improve the efficiency of Compact-LWE and claimed it can offer 192 bits of security. However, in this paper, we show that Compact-LWE is not secure with recommended parameters by presenting two efficient ciphertext-only attacks against it.

- The first one is to recover the equivalent private keys just from the public keys. By exploiting the special structure of Compact-LWE, employing some known skills such as orthogonal-lattice technique, and also developing some new techniques, we finally recovered the equivalent private keys for more than 80% of the random generated instances in our experiments.
- The second one is to recover the corresponding message given the public keys and a ciphertext. Note that any short enough solutions of corresponding inhomogeneous linear systems can be used to decrypt a ciphertext equivalently. We recovered all the messages without knowing the private keys in our experiments.

Keywords: Post-quantum encryption · LWE · ciphertext-only attack · lattice.

1 Introduction

Since Shor's algorithms [15, 16] that solve integer factorization and discrete logarithm problem in polynomial time with quantum computers, and Grover's algorithm [8] that improves the time complexity of search problems, quantum computing has been getting more attractive and shows its advantages over classical algorithms in solving some computational problems. Meanwhile, there has been

a substantial amount of researches on quantum computers or machines, which are imperiling modern public key cryptosystems such as RSA, DSA, ECDSA based on number-theoretical hard problems. Therefore there is an urgent need for the quantum-resistant public-key cryptographic replacements to cope with the challenge of quantum attacks.

In December 2016, the U.S. National Institute of Standards and Technology (NIST) launched a post-quantum cryptography project to call for proposals of efficient quantum-resistant public-key cryptographic algorithms in order to draft new standards for next-generation cryptography. The deadline of submissions is Nov. 3, 2017 and NIST plans to issue guidance on candidate post-quantum schemes in five years. In December 2017, NIST published the Round 1 candidates for the post-quantum cryptography.

All the candidates can be classified into several classes according to their underlying hard problems, such as lattice-based schemes, code-based schemes, hash-based schemes, and multivariate schemes. Most of the schemes submitted to NIST PQC Project are based on lattice since lattice-based cryptography is widely believed to be one of the most promising candidates as it enjoys worst-case hardness security guarantees, versatile functionalities and relatively simple basic operations.

In general, a lattice is a discrete additive subgroup in $\mathbb{R}^n$, and the shortest vector problem (SVP) is one of the most famous underlying problem for lattice-based cryptosystems, which refers to the question of finding a shortest nonzero lattice vector in a given lattice. It has been proved that SVP is NP-hard under random reductions [1]. Approximating-SVP is a variant of SVP, whose goal is to find a non-zero lattice vector not longer than the length of shortest nonzero lattice vector multiplying some approximation factor. It is well-known that approximating-SVP is hard for constant factor but easy for exponential factor due to some lattice basis reduction algorithms like LLL algorithm [9]. As for the polynomial factor, approximation-SVP is conjectured and believed hard.

However, it seems not easy to construct public-key cryptosystems with SVP and its variants directly. The widely used intractable underlying problem for lattice-based cryptosystems is the LWE (Learning With Errors) problem, that is, solving a modulo linear system with errors drawn from certain distribution. In 2005, Regev [13] first defined LWE, and showed that it has average-case hardness by presenting a quantum reduction from approximating-SVP to LWE. Besides, Regev also constructed an LWE-based public-key encryption scheme. From then on, LWE and its descendants have served as a simple and extraordinarily important basic building block for the construction of some cryptographic primitives, especially the public-key encryption schemes.

Recently, Liu, Li, Kim and Nepal [10] proposed to NIST a lattice-based encryption scheme called Compact-LWE. Compact-LWE features its structure similar to LWE [13], but requires smaller number of samples and somewhat bigger errors. Liu, Li, Kim and Nepal presented some cryptanalysis on the security of Compact-LWE, and thought that the special structure they employed can protect Compact-LWE from the known lattice-based attack, which is different from

the classical LWE. Finally they claimed that "*even if the hard problems in lattice, such as CVP and SIS, can be efficiently solved, the secret values or private key in Compact-LWE still cannot be efficiently recovered. This allows Compact-LWE to choose very small dimension parameters, such as $n = 8$ in our experiment*" [10]. Based on the optimistic assess, Liu, Li, Kim and Nepal recommended a set of small parameters to improve the efficiency of Compact-LWE and claimed it can offer 192 bits of security.

As a variant of LWE, the security of Compact-LWE should be studied carefully, since LWE lies at the heart of constructions of efficient lattice-based cryptosystems. However, we have to say that the current version of Compact-LWE is incompetent in NIST competition.

Our Contributions. In this paper, we present two ciphertext-only attacks against Compact-LWE: the first one is to recover an equivalent private key only using the public key, and the second one is to recover the corresponding message given the public key and a ciphertext.

Key Recovery Attack. The private key recovery attack is the most technical part in this paper. By exploiting the special structure of Compact-LWE, employing some known skills such as orthogonal-lattice technique, and also developing some new techniques, we finally recover an equivalent private key for more than 80% of the randomly generated instances in our experiments.

Roughly speaking, the public key of Compact-LWE consists of several sample-pairs:

$$(\boldsymbol{a_i}, \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + k^{-1} \cdot (sk \cdot u_i + \Delta_i) \bmod q)$$
$$(\boldsymbol{a_i}, \langle \boldsymbol{a_i}, \boldsymbol{s'} \rangle + k'^{-1} \cdot (sk' \cdot u_i + \Delta'_i) \bmod q),$$

where u_i's are public, while $\boldsymbol{s}$, $\boldsymbol{s'}$, k, k', sk, sk', Δ_i, Δ'_i are private and Δ_i, Δ'_i satisfy $ck \cdot \Delta_i + ck' \cdot \Delta_i \equiv 0 \bmod p$ for some private ck, ck' and p.

Employing the orthogonal-lattice technique, we can first recover the private k and k', and some integer coefficient combinations of Δ_i, Δ'_i. In our experiments, we can recover both k and k' for more than 80% of the randomly generated instances.

By building the linear relation between Δ_i and Δ'_i, we can then recover the correct p, hence we will solve a pair of equivalent candidates ck, ck'.

By the orthogonal-lattice technique again, we can recover the value $ck \cdot sk + ck' \cdot sk' \bmod p$, then by solving a CVP in a 2-dimensional lattice, we can always recover sk and sk' in our experiments.

With all known parameters, we can transform the public key pairs into the following form:

$$(\boldsymbol{a_i}, \langle \boldsymbol{a_i}, k\boldsymbol{s} \rangle + \Delta_i \bmod q)$$
$$(\boldsymbol{a_i}, \langle \boldsymbol{a_i}, k'\boldsymbol{s'} \rangle + \Delta'_i \bmod q).$$

Since Δ_i and Δ_i' are slightly big, and $\boldsymbol{a_i}$'s are very short, we can not recover Δ_i and Δ_i' by the existing lattice attacks. However, we develop a new technique to reduce it to the following lattice decomposition problem: given $\boldsymbol{t} \in \mathcal{L}_1 + \mathcal{L}_2$ where $\mathcal{L}_1$ and $\mathcal{L}_2$ are two lattices, find $\boldsymbol{t}_1 \in \mathcal{L}_1$ and SHORT $\boldsymbol{t}_2 \in \mathcal{L}_2$ such that $\boldsymbol{t} = \boldsymbol{t}_1 + \boldsymbol{t}_2$. We propose a lattice-based algorithm to solve this problem and finally recover equivalent Δ_i and Δ_i'.

We believe that the new techniques may be employed somewhere else.

Message Recovery Attack. We can reduce the task to recover the message to the problem of recovering $\sum_{i=1}^{m} l_i \cdot u_i$ given

$$\begin{aligned}
\mathbf{la} &= \sum_{i=1}^{m} l_i \cdot \boldsymbol{a}_i, \\
lpk &= \sum_{i=1}^{m} l_i \cdot (\langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + k^{-1} \cdot (sk \cdot u_i + \Delta_i)) \bmod q, \\
lpk' &= \sum_{i=1}^{m} l_i \cdot (\langle \boldsymbol{a_i}, \boldsymbol{s}' \rangle + k'^{-1} \cdot (sk' \cdot u_i + \Delta_i')) \bmod q,
\end{aligned}$$

where l_i's are some very small integers. The key observation is that if we can find another short enough vector $(l_1', l_2', \ldots, l_m')$, not necessarily the exact $(l_1, l_2, \ldots, l_m)$, such that

$$\begin{aligned}
\mathbf{la} &= \sum_{i=1}^{m} l_i' \cdot \boldsymbol{a}_i, \\
lpk &= \sum_{i=1}^{m} l_i' \cdot (\langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + k^{-1} \cdot (sk \cdot u_i + \Delta_i)) \bmod q, \\
lpk' &= \sum_{i=1}^{m} l_i' \cdot (\langle \boldsymbol{a_i}, \boldsymbol{s}' \rangle + k'^{-1} \cdot (sk' \cdot u_i + \Delta_i')) \bmod q,
\end{aligned}$$

then we must have

$$\sum_{i=1}^{m} l_i' \cdot u_i = \sum_{i=1}^{m} l_i \cdot u_i,$$

due to the fact that the decryption algorithm is deterministic.

It is obvious that the equivalent vector $(l_1', l_2', \ldots, l_m')$ can be found by finding a short solution for a system of inhomogeneous linear equations. Therefore if we can solve CVP (closest vector problem) in lattices to obtain short solutions, we can break Compact-LWE, thus overturn the claim in [10].

Since the parameters in Compact-LWE are small, it is not hard to find a short solution using lattice basis reduction algorithms such as LLL algorithm [9], hence we can always recover all the messages without knowing the private keys in our experiments.

It seems that Compact-LWE should enlarge its parameters to resist our attacks, which will result in the sacrifice of efficiency to some extent, the security of Compact-LWE should be reevaluated more carefully.

Related Work. We have to point out that a previous version of Compact-LWE was presented in [11], in which Liu, Li, Kim and Nepal employed m pairs $(\boldsymbol{a}_i, pk_i)$ as the public key, which satisfy

$$pk_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \cdot sk_q^{-1} \cdot p \bmod q.$$

To encrypt a small message $v \in \mathbb{Z}_t$, one first chooses w indices $i_1, \ldots, i_w$ in $\{1, \ldots, m\}$ uniformly and independently at random, then computes

$$(\boldsymbol{a}, b) = \sum_{k=1}^{w} (\boldsymbol{a}_{i_k}, pk_{i_k}).$$

It is obvious that $(\boldsymbol{a}, b)$ can be also written as $\sum_{i=1}^{m} l_i(\boldsymbol{a}_i, pk_i)$ where l_i is a small integer. At last the ciphertext $\boldsymbol{c}$ is computed as

$$\boldsymbol{c} = (\boldsymbol{a}, d = v - b \bmod q).$$

To decrypt a ciphertext $\boldsymbol{c} = (\boldsymbol{a}, d)$, one can compute $v = sk_p^{-1} \cdot (sk \cdot (\langle \boldsymbol{a}, \boldsymbol{s} \rangle + d \bmod q)$ to recover v.

Since the message v is small, $(\boldsymbol{a}, d)$ is very close to $(\boldsymbol{a}, -b)$ which is in the known lattice spanned by $(\boldsymbol{a}_i, pk_i)$'s and qI. Hence, v may be recovered by finding the lattice vector close to $(\boldsymbol{a}, d)$. By this key observation, together with the restriction that $\sum_{i=1}^{m} l_i \boldsymbol{a}_i = \boldsymbol{a}$ and the coefficients l_i's are small, Bootle et al. [4, 5] presented a lattice-based attack to recover the message v. Some detailed analysis for various cases can also be found in [4, 5].

Moreover, Bootle et al. [5] also presented an attack to recover an equivalent private key. The same orthogonal-lattice technique was employed to recover $sk_q^{-1} \cdot p \bmod q$, but then they tried to recover p by enumerating all possible values. Finally they recovered equivalent parameters sk, $\boldsymbol{s}$ and $\boldsymbol{e}$.

To resist Bootle et al.'s attacks, Liu, Li, Kim and Nepal made some changes to the version of Compact-LWE submitted to NIST. They used the collection of sample-pairs with some linear relation instead of the collection of single samples, added more parameters to the error terms, such as $sk \cdot u_i + r_i$, which makes the scheme more complicated. Instead of encrypting a message by computing $v - d$ simply, they first encrypted some ephemeral key $\sum_{i=1}^{m} l_i \cdot u_i$, then used it to encrypt the message. By doing so, we can not recover the message or the ephemeral key by solving some approximating-CVP as in Bootle et al.'s message recovery attack. Moreover, the new involved parameters makes Bootle et al.'s private key recovery attack infeasible.

As stated before, our message recovery attack against Compact-LWE submitted to NIST is based on the key observation that $\sum_{i=1}^{m} l'_i \cdot u_i = \sum_{i=1}^{m} l_i \cdot u_i$, if l'_i's satisfy the conditions above, whereas Bootle et al.'s attack against the previous

version is based on the observation that $(\boldsymbol{a}, d)$ is very close to $(\boldsymbol{a}, -b)$. Although both of the two attacks need to find a short vector in some lattice, the motivation to construct the lattice and the way to use the computed short lattice vector are very different. We believe that our key observation is non-trivial since even the designers, Liu, Li, Kim and Nepal, thought that the exact $(l_1, l_2, \ldots, l_m)$ should be recovered to complete the attack (see Section 3.3 in [10]).

For the key recovery attacks, both of our attack and Bootle et al.'s attack employ the same technique to recover $k(k')$ in the first step. Then a enumeration process was employed in Bootle et al.'s attack. However, since the new Compact-LWE involves more parameters, it is infeasible to apply Bootle et al.'s enumeration strategy directly. We have to say that recovering $k(k')$ is just the beginning in our attack, far from the end of the whole attack. To recover the equivalent key for the new Compact-LWE, we have to do more work and involve some new techniques.

Roadmap. The remainder of the paper is organized as follows. In Section 2, we give some preliminaries needed. In Section 3, we describe Compact-LWE encryption scheme. In Section 4, we describe our private key recovery attack and in Section 5, we give a message recovery attack. Finally, we give a short conclusion in Section 6.

2 Notations and Preliminaries

Notations. Row vectors are denoted by bold lowercase letters. We use $\|\cdot\|$ to denote the length of a vector and $\langle\cdot,\cdot\rangle$ to denote the inner product of two vectors. Matrices are written in bold capital letters, and row representation is used. For a positive integer q, let $\mathbb{Z}_q$ be the residue class ring module q. For a finite set S, we use the notation $a \xleftarrow{\$} S$ to represent randomly sampling an element a from S.

2.1 Lattices

Let $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\} \subset \mathbb{R}^m$ be a set of n linearly independent vectors. The lattice generated by $\boldsymbol{B}$ is defined as

$$\mathcal{L}(\boldsymbol{B}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i |\ x_i \in \mathbb{Z} \right\}.$$

We call $\boldsymbol{B}$ the basis of $\mathcal{L}(\boldsymbol{B})$, n and m the rank and dimension of the lattice respectively.

The fundamental parallelepiped spanned by $\boldsymbol{B}$ is defined as $\mathcal{P}(\boldsymbol{B}) = \{\boldsymbol{x}\boldsymbol{B}|\ \boldsymbol{x} \in [0,1)^n\}$, and the centered parallelepiped spanned by $\boldsymbol{B}$ is defined as $\mathcal{P}_c(\boldsymbol{B}) = \{\boldsymbol{x}\boldsymbol{B}|\ \boldsymbol{x} \in [-1/2, 1/2)^n\}$. Given a basis $\boldsymbol{B}$, denote by $\det(\mathcal{L}(\boldsymbol{B}))$ the determinant of lattice $\mathcal{L}(\boldsymbol{B})$, which is defined as the volume of $\mathcal{P}(\boldsymbol{B})$. We have $\det(\mathcal{L}(\boldsymbol{B})) = \sqrt{\det(\boldsymbol{B}\boldsymbol{B}^T)}$. When $n = m$, $\det(\mathcal{L}(\boldsymbol{B})) = |\det(\boldsymbol{B})|$.

For an ordered lattice basis $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$, the Gram-Schmidt orthogonalization $\boldsymbol{b}_1^\star, \ldots, \boldsymbol{b}_n^\star$ can be efficiently computed by the recursion

$$\begin{cases} \boldsymbol{b}_1^\star = \boldsymbol{b}_1, \\ \boldsymbol{b}_i^\star = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{b}_j^\star \text{ for i=2,...,n,} \end{cases}$$

where $\mu_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j^\star \rangle / \langle \boldsymbol{b}_j^\star, \boldsymbol{b}_j^\star \rangle$. This procedure depends on the order of the basis. Hereafter, the Gram-Schmidt vectors corresponding to $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ are denoted by $\boldsymbol{B}^\star = \{\boldsymbol{b}_1^\star, \ldots, \boldsymbol{b}_n^\star\}$. When $\boldsymbol{B}$ is a basis for $\mathcal{L}(\boldsymbol{B})$, it is easy to obtain that $\det(\mathcal{L}(\boldsymbol{B})) = \prod_{i=1}^{n} \|\boldsymbol{b}_i^\star\|$.

2.2 SVP and CVP: Problems and Algorithms

SVP. The shortest vector problem (SVP) is one of the most famous hard problems in lattice, which refers to the question of finding a nonzero shortest lattice vector in a given lattice.

Denote by $\lambda_1(\mathcal{L})$ the length of a shortest nonzero lattice vector. For a "random" lattice $\mathcal{L}$, Gauss Heauristic predicts that

$$\lambda_1(\mathcal{L}) \approx \Big(\frac{\det \mathcal{L}}{V_n(1)}\Big)^{\frac{1}{n}},$$

where $V_n(1)$ is the volume of an n-dimension ball with radius 1.

The approximating-SVP with factor γ, denoted by SVP_γ, asks for a short nonzero lattice vector $\boldsymbol{v}$ such that $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$. The hardness of SVP_γ depends on the factor γ heavily. For constant γ, this problem is known to be NP-hard under randomized reduction [1]. However, for exponential factor γ with respect to the rank n, there do exist polynomial-time algorithms to find a short vector, such as the famous LLL algorithm [9].

Though the length of the short vector found by LLL algorithm can only be proved to be less than $\gamma \cdot \lambda_1(\mathcal{L})$ with exponential γ, we would like to point out that for low dimensional lattices, LLL algorithm usually find the shortest vector in practice.

CVP. The closest vector problem (CVP) is another famous hard problem in lattice, which refers to the problem of finding a lattice point that is closest to a given target $\boldsymbol{t}$. for Similarly, the CVP_γ problem can be defined as finding a close vector to $\boldsymbol{t}$ up to a factor $\gamma \geq 1$.

The CVP is known to be NP-hard [3], even with an almost polynomial approximation factor [6]. However, there do exist polynomial-time algorithms to approximate CVP within exponential factors, such as Babai's rounding-off algorithm, nearest plane algorithm [2] and the GPV algorithm [7], all of which start with an LLL-reduced basis [9].

Given an LLL-reduced basis $\boldsymbol{B}$ and a target $\boldsymbol{t}$, Babai's rounding-off algorithm outputs a lattice vector $\boldsymbol{v}_{\text{roa}} = \lceil \boldsymbol{t}\boldsymbol{B}^{-1} \rfloor \boldsymbol{B}$. It can be easily seen that $\boldsymbol{v}_{\text{roa}} - \boldsymbol{t} \in \mathcal{P}_c(\boldsymbol{B})$.

The nearest plane algorithm uses a more complicated method as described in Algorithm 1. Denote by $\boldsymbol{v}_{npa}$ the lattice vector outputted by the nearest plane algorithm. It can be easily concluded that $\boldsymbol{v}_{\mathrm{npa}} - \boldsymbol{t} \in \mathcal{P}_c(\boldsymbol{B}^\star)$.

Algorithm 1 Babai's Nearest Plane Algorithm.

Input: A reduced lattice basis $\boldsymbol{B}$ and a target vector $\boldsymbol{t}$;
Output: A lattice vector close to $\boldsymbol{t}$;
1: $\boldsymbol{b} = \boldsymbol{t}$;
2: for $j = n$ to 1 do
3: $\quad \boldsymbol{b} = \boldsymbol{b} - c_j \cdot \boldsymbol{b}_j$ where $c_j = \lceil \frac{\langle \boldsymbol{b}, \boldsymbol{b}_j^\star \rangle}{\langle \boldsymbol{b}_j^\star, \boldsymbol{b}_j^\star \rangle} \rfloor$;
4: **return** $\boldsymbol{t} - \boldsymbol{b}$;

The GPV algorithm is a randomized version of Babai's nearest plane algorithm. For Step 3 in Algorithm 1, GPV algorithm chooses a random c_j from a "discrete Gaussian distribution" and produces a random lattice vector which is close to the target. The readers are referred to [7] for more details.

2.3 Kernel Lattice

Given a matrix $\boldsymbol{B} \in \mathbb{Z}^{m \times n}$ with rank r, its kernel lattice or orthogonal lattice is defined as

$$\mathcal{L}(\boldsymbol{B})^\perp = \{\boldsymbol{x} \in \mathbb{Z}^m | \ \boldsymbol{x}\boldsymbol{B} = \boldsymbol{0}\}.$$

The rank of $\mathcal{L}(\boldsymbol{B})^\perp$ is $m - r$.

Furthermore, Nguyen and Stern [12] presented a way to compute an LLL-reduced basis for $\mathcal{L}(\boldsymbol{B})^\perp$ from the matrix $\boldsymbol{B}$. The main idea is to construct a new matrix

$$\boldsymbol{B}' = \left(c\boldsymbol{B}, \boldsymbol{I}_m\right)$$

with big enough c first, and then to apply LLL algorithm on $\boldsymbol{B}'$. The outputted matrix must have the following form:

$$\begin{pmatrix} \boldsymbol{0}_{(m-r)\times n} & \boldsymbol{U} \\ * & * \end{pmatrix}$$

Then $\boldsymbol{U}$ will be an LLL-reduced basis for $\mathcal{L}(\boldsymbol{B})^\perp$. More details can be found in [12].

Another important kernel lattice is defined as

$$\mathcal{L}(\boldsymbol{B})_p^\perp = \{\boldsymbol{x} \in \mathbb{Z}^m | \ \boldsymbol{x}\boldsymbol{B} \equiv 0 \bmod p\}.$$

Let

$$\boldsymbol{B}_p = \begin{pmatrix} \boldsymbol{B}^T \\ p\boldsymbol{I}_n \end{pmatrix}.$$

Similarly, to compute an LLL-reduced basis for $\mathcal{L}(\boldsymbol{B})_p^{\perp}$, we can construct a matrix

$$\boldsymbol{B}'_p = \begin{pmatrix} c\boldsymbol{B} & \boldsymbol{I}_m \\ cp\boldsymbol{I}_n & \boldsymbol{0} \end{pmatrix},$$

with big enough c first, then use LLL algorithm to $\boldsymbol{B}'_p$. The outputted matrix must have the following form:

$$\begin{pmatrix} \boldsymbol{0}_{m\times n} & \boldsymbol{U} \\ * & * \end{pmatrix}.$$

It can be concluded that $\boldsymbol{U}$ is an LLL-reduced basis for $\mathcal{L}(\boldsymbol{B})_p^{\perp}$.

2.4 Inhomogeneous Short Integer Solution.

Now consider the ISIS (Inhomogeneous Short Integer Solution) problem [7]

$$\boldsymbol{x} \cdot \boldsymbol{B} \equiv \boldsymbol{t} \bmod p,$$

with $\boldsymbol{x} \in \mathbb{Z}^m$, $\boldsymbol{B} \in \mathbb{Z}^{m\times n}$, p an positive integer and nonzero $\boldsymbol{t} \in \mathbb{Z}^n$. We want to obtain a short $\boldsymbol{x}$.

Although it seems hard to find a shortest integer solution, we can find an approximately short solution by the following algorithm.

Note that any solution $\boldsymbol{x}$ can be written into

$$\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{y}\boldsymbol{K},$$

where $\boldsymbol{x}_0 \in \mathbb{Z}^m$ is any fixed solution for $\boldsymbol{x} \cdot \boldsymbol{B} \equiv \boldsymbol{t} \bmod p$, $\boldsymbol{K}$ is a basis for $\mathcal{L}(\boldsymbol{B})_p^{\perp}$ and $\boldsymbol{y}$ is some integer vector. Hence finding short solution $\boldsymbol{x}$ is reduced to the problem of finding a lattice vector in $\mathcal{L}(\boldsymbol{K})$ close to the target vector $-\boldsymbol{x}_0$. By employing some algorithms for approximating-CVP such as Babai's nearest plane algorithm, a small solution $\boldsymbol{x}$ is expected to be found.

Similar method can be used to obtain a short integer solution for $\boldsymbol{x} \cdot \boldsymbol{B} = \boldsymbol{t}$.

We summarize these two solving algorithms as Algorithm 2.

Algorithm 2 Finding Short Integer Solution to $\boldsymbol{x}\cdot\boldsymbol{B} \equiv \boldsymbol{t} \bmod p$ ($\boldsymbol{x}\cdot\boldsymbol{B} = \boldsymbol{t}$ *resp.*)

Input: A matrix $\boldsymbol{B} \in \mathbb{Z}^{m\times n}$ and $\boldsymbol{t} \in \mathbb{Z}^n$;
Output: A short solution $\boldsymbol{x} \in \mathbb{Z}^m$ satisfying $\boldsymbol{x} \cdot \boldsymbol{B} \equiv \boldsymbol{t} \bmod p$ ($\boldsymbol{x} \cdot \boldsymbol{B} = \boldsymbol{t}$ *resp.*);
1: Find a solution $\boldsymbol{x}_0 \in \mathbb{Z}^m$ such that $\boldsymbol{x}_0 \cdot \boldsymbol{B} \equiv \boldsymbol{t} \bmod p$ ($\boldsymbol{x}_0 \cdot \boldsymbol{B} = \boldsymbol{t}$ *resp.*);
2: Compute a basis $\boldsymbol{K}$ for the kernel lattice $\mathcal{L}(\boldsymbol{B})_p^{\perp}$ ($\mathcal{L}(\boldsymbol{B})^{\perp}$ *resp.*);
3: Use some lattice basis reduction algorithms like LLL algorithm to reduce $\boldsymbol{K}$ to obtain a better basis $\boldsymbol{K}'$;
4: Use Babai's Rounding algorithm or Nearest Plane Algorithm with lattice basis $\boldsymbol{K}'$ and the target vector $-\boldsymbol{x}_0$ to find a close lattice vector $\boldsymbol{v}$;
5: **return** $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{v}$.

3 The Compact-LWE Public-key Encryption Scheme

We describe Compact-LWE as follows. For more details, please see [10].

3.1 The Description of Basic Compact-LWE

Public Parameters: The following parameters are all positive integers.

1. Moduli: q, t, b';
2. Dimension: n;
3. Bounds: w, w', b;
4. Number of samples: m.

Private Parameters: The parameters p_size, s_max, e_min, e_max are all positive parameters and used to bound other parameters.

For the sake of correctness of the basic decryption algorithm and the security of the cryptosystem, [10] requires that

1. $m > n + 2$;
2. $w > w'$, $(w - w') \cdot b' > t$;
3. t is a power of 2;
4. $sk\_max \cdot b' + p + e\_max \cdot p < q/(w + w')$.

Key Generation

The private key is generated in the following steps.

1. Generate two vectors $\boldsymbol{s}, \boldsymbol{s}' \xleftarrow{\$} \mathbb{Z}_q^n$;
2. Generate two integers $k, k' \xleftarrow{\$} \mathbb{Z}_q$ with k, k' coprime with q respectively;
3. Choose an integer $p \xleftarrow{\$} \{(w + w') \cdot b', \ldots, (w + w') \cdot b' + p\_size\}$ such that p is coprime with q;
4. Randomly generate $ck, ck' \xleftarrow{\$} \mathbb{Z}_p$ with ck' coprime with p;
5. Randomly generate $sk, sk' \xleftarrow{\$} \mathbb{Z}_{sk\_max}$ such that $sk \cdot ck + sk' \cdot ck'$ is coprime with p.

The private key is

$$\boldsymbol{SK} = (\boldsymbol{s}, \boldsymbol{s}', k, k', p, ck, ck', sk, sk').$$

For $i \in \{1, 2, \ldots, m\}$, the generation of the public key is to collect m random Compact-LWE samples.

1. Generate $\boldsymbol{a}_i \xleftarrow{\$} \mathbb{Z}_b^n$;
2. Randomly generate $e_i, e_i' \xleftarrow{\$} [e\_min, e\_max]$;
3. Randomly sample $u_i \xleftarrow{\$} \mathbb{Z}_{b'}$;
4. Choose a random $r_i' \xleftarrow{\$} \mathbb{Z}_p$ and compute r_i such that $ck \cdot r_i + ck' \cdot r_i' = 0 \bmod p$;
5. Compute k_q^{-1}, ${k'}_q^{-1} \in \mathbb{Z}_p$ such that $k_q^{-1} \cdot k = 1 \bmod q$ and ${k'}_q^{-1} \cdot k' = 1 \bmod q$;

6. Compute

$$pk_i = \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + k_q^{-1} \cdot (sk \cdot u_i + r_i + e_i \cdot p) \bmod q \tag{1}$$

$$pk'_i = \langle \boldsymbol{a_i}, \boldsymbol{s'} \rangle + {k'}_q^{-1} \cdot (sk' \cdot u_i + r'_i + e'_i \cdot p) \bmod q \tag{2}$$

Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)$, $\boldsymbol{pk} = (pk_1, pk_2, \ldots, pk_m)$, $\boldsymbol{pk'} = (pk'_1, pk'_2, \ldots, pk'_m)$.

The public key is

$$\boldsymbol{PK} = (\boldsymbol{a_1}, \boldsymbol{a_2}, \cdots, \boldsymbol{a_m}, \boldsymbol{u}, \boldsymbol{pk}, \boldsymbol{pk'}).$$

Encryption

To encrypt a message $v \in \mathbb{Z}_t$, we compute the ciphertext $\boldsymbol{c}$ as follows:

1. Generate the m-dimensional random vector $\boldsymbol{l}$ such that the sum of all positive entries of $\boldsymbol{l}$ lies between w and $w + w'$, the sum of all negative entries of $\boldsymbol{l}$ between $-w'$ and 0.
2. Compute $lu = \sum_{i=1}^{m} l_i \cdot u_i$, if $lu \leq 0$, then regenerate $\boldsymbol{l}$, otherwise calculate $u_t = lu \mod t$, take u'_t the smallest integer equal to or greater than lu/t coprime with t, encrypt v by computing

$$d = f_{lu}(v)$$

   where f is an efficiently computable function including two basic operations: XOR operation and bit shift operation. For more details about f see [10].
3. Compute $\boldsymbol{la} = \sum_{i=1}^{m} l_i \cdot \boldsymbol{a_i}$, $lpk = \sum_{i=1}^{m} l_i \cdot pk_i$ and $lpk' = \sum_{i=1}^{m} l_i \cdot pk'_i$, the ciphertext is

$$\boldsymbol{c} = (\boldsymbol{la}, d, lpk, lpk').$$

Decryption

Given the ciphertext $\boldsymbol{c} = (\boldsymbol{la}, d, lpk, lpk')$, the decryption algorithm recovers the message v in the following steps.

1. Compute $d_1 = k \cdot (lpk - \langle \boldsymbol{la}, \boldsymbol{s} \rangle) \bmod q$, $d'_1 = k' \cdot (lpk' - \langle \boldsymbol{la}, \boldsymbol{s'} \rangle) \bmod q$;
2. Compute $d_2 = ck \cdot d_1 + ck' \cdot d'_1 \bmod p$ and $sckInv$ satisfying $sckInv \cdot (sk \cdot ck + sk' \cdot ck') = 1 \bmod p$;
3. Compute $d_3 = sckInv \cdot d_2 \bmod p$;
4. Compute $v = f_{d_3}^{-1}(d)$.

Remark 1. Liu, Li, Kim and Nepal [10] also proposed a general encryption algorithm, which appends some padding and encodes the message first before encryption. After encoding the original message, the general encryption algorithm divides a long message into blocks and encrypts each block with the basic encryption algorithm.

3.2 Parameter Choice and Security Declaration

By considering some possible attacks to analyze the security of Compact-LWE, Liu, Li, Kim and Nepal [10] thought that the special construction of Compact-LWE makes it immune to all lattice-based attacks. They believed that even for small parameters, Compact-LWE can also achieve a high security. To make Compact-LWE more efficient, they recommended a set of small parameters as in Table 1 and Table 2, and claimed that Compact-LWE with such parameters could offer 192 bits of security.

Table 1. Public Parameters

q	t	n	m	w	w'	b	b'
2^{64}	2^{32}	8	128	224	32	16	2^{36}

Table 2. Private Parameters

sk_max	p_size	e_min	e_max
229119	2^{24}	457	3200

4 Equivalent Private Key Recovery Attack

For simplicity, we can write every public pair (pk_i, pk_i') as

$$pk_i = \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + k^{-1} \cdot (sk \cdot u_i + \Delta_i) \bmod q, \tag{3}$$

$$pk_i' = \langle \boldsymbol{a_i}, \boldsymbol{s'} \rangle + k'^{-1} \cdot (sk' \cdot u_i + \Delta_i') \bmod q, \tag{4}$$

where $\Delta_i = r_i + e_i \cdot p$ and $\Delta_i' = r_i' + e_i' \cdot p$.

Roughly speaking, the key idea of our key recovery attack is that, if we could find $\boldsymbol{s}$, $\boldsymbol{s'}$, k, k', p, sk, sk', $\boldsymbol{\Delta} = (\Delta_1, \cdots, \Delta_m)$, $\boldsymbol{\Delta'} = (\Delta_1', \cdots, \Delta_m')$, ck, ck' such that

- every public key pair (pk_i, pk_i') can be written as (3) and (4);
- sk and sk' are small enough;
- p is large enough;
- Δ_i and Δ_i' are relatively smaller than q and satisfy $ck \cdot \Delta_i + ck' \cdot \Delta_i' \equiv 0 \bmod p$;
- $sk \cdot ck + sk' \cdot ck'$ is coprime to p;

then the set of $\boldsymbol{s}$, $\boldsymbol{s'}$, k, k', p, sk, sk', ck, ck' can be used to decrypt the ciphertext, that is, it is an equivalent private key.

Since the private key consists of many parameters, we recover the equivalent key step by step:

- first, we recover k and k' by orthogonal-lattice technique in Section 4.1; Usually we can recover the real values.
- second, we recover p, ck and ck' in Section 4.2; We can always recover the real p, but recover equivalent ck and ck'.
- third, we recover sk and sk' by lattice algorithm again in Section 4.3; We can always recover the real sk and sk'.
- at last, we recover equivalent $\boldsymbol{s}$ and $\boldsymbol{s}'$ by introducing a new technique in Section 4.4.

4.1 Recovering k and k'

Denote by $\boldsymbol{A}$ the matrix

$$\begin{pmatrix} \boldsymbol{a}_1 \\ \cdots \\ \boldsymbol{a}_m \end{pmatrix},$$

and by $[\boldsymbol{A}\ \boldsymbol{u}]$ the matrix

$$\begin{pmatrix} \boldsymbol{a}_1 & u_1 \\ \cdots & \cdots \\ \boldsymbol{a}_m & u_m \end{pmatrix}.$$

Considering $\boldsymbol{x} \in \mathcal{L}([\boldsymbol{A}\ \boldsymbol{u}])^{\perp}$, we will obtain

$$k \langle \boldsymbol{x}, \boldsymbol{pk} \rangle = \langle \boldsymbol{x}, \boldsymbol{\Delta} \rangle \bmod q.$$

where $\boldsymbol{\Delta} = (r_1 + e_1 \cdot p, \ldots, r_m + e_m \cdot p)$.

Note that if $\boldsymbol{x}$ is short enough, together with the fact that $\boldsymbol{\Delta}_i (\approx 2^{56})$ is a relatively smaller than $q = 2^{64}$, then $\langle \boldsymbol{x}, \boldsymbol{\Delta} \rangle$ will be relatively smaller than q. Moreover, if we could collect some independent short vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \in \mathcal{L}([\boldsymbol{A}\ \boldsymbol{u}])^{\perp}$, then $\boldsymbol{v} = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta} \rangle, \ldots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta} \rangle)$ is expected to be a short vector in the lattice $\mathcal{L}$ spanned by

$$\begin{pmatrix} \langle \boldsymbol{x}_1, \boldsymbol{pk} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{pk} \rangle \\ q\boldsymbol{I}_M \end{pmatrix},$$

due to the relation

$$k(\langle \boldsymbol{x}_1, \boldsymbol{pk} \rangle, \ldots, \langle \boldsymbol{x}_M, \boldsymbol{pk} \rangle) = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta} \rangle, \ldots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta} \rangle) \bmod q.$$

By applying lattice reduce algorithm to $\mathcal{L}$, it is expected to find the short vector $\boldsymbol{v} = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta} \rangle)$ and with $\boldsymbol{v}$ we can therefore recover the secret k.

Similarly, with the same $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$, we can expect to recover $(\langle \boldsymbol{x}_1, \boldsymbol{\Delta}' \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta}' \rangle)$ and k'.

It remains to show how to collect short vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ in $\mathcal{L}([\boldsymbol{A}\ \boldsymbol{u}])^{\perp}$. As stated in Section 2, by the algorithm in [12], we can compute an LLL-reduced basis for $\mathcal{L}([\boldsymbol{A}\ \boldsymbol{u}])^{\perp}$ and collect some basis vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ from the LLL-reduced basis satisfying $\|\boldsymbol{x}_j\|_1 \leq B$ ($j \in \{1, \ldots, M\}$) for some bound B to make the value $\langle \boldsymbol{x}_j, \Delta \rangle$ for $1 \leq j \leq M$ relatively small to q.

We describe the procedure to recover k in algorithm 3.

Algorithm 3 Recover a candidate for k

Input: The public parameters $\boldsymbol{a_1}, \boldsymbol{a_2}, \cdots, \boldsymbol{a_m}, \boldsymbol{u}, \boldsymbol{pk}, \boldsymbol{pk'}$.
Output: a candidate $\tilde{k}$ for k

1: Compute an LLL-reduced basis of $\mathcal{L}([\boldsymbol{A}\ \boldsymbol{u}])^{\perp}$, and collect vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ satisfying $\|\boldsymbol{x}_j\|_1 \leq B$ with respect to an appropriate bound B;
2: Apply LLL algorithm on

$$\begin{pmatrix} \langle \boldsymbol{x}_1, \boldsymbol{pk} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{pk} \rangle \\ q\boldsymbol{I}_M \end{pmatrix};$$

3: Choose the first nonzero vector $\boldsymbol{v}$ from reduced basis, and compute $\tilde{k} \in \mathbb{Z}_q$ such that $\tilde{k} \cdot (\langle \boldsymbol{x}_1, \boldsymbol{pk} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{pk} \rangle) = \boldsymbol{v} \bmod q$;
4: **return** $\tilde{k}$.

Experimental Results of Recovering k, k'. All the experiments in this paper were run in C++ and Sage [14] on a computer with 8G RAM and Intel(R) Core(TM) i5-7200U CPU @2.50GHz.

As for the parameters $r_i \in [0, p-1]$, $e_i \in [457, 3200]$ and $p \leq 3200 \cdot (2^{44} + 2^{24})$ specified by the designers, it follows

$$0 \leq \Delta_i \leq 2^{56}.$$

By choosing the bound $B = 2^6$ in our recovery experiments,

$$|\langle \boldsymbol{x}_j, \Delta \rangle| \leq \|\boldsymbol{x}_j\|_1 \cdot \|\Delta\|_\infty \leq 2^6 \cdot 2^{56} = 2^{62} = \frac{q}{4}.$$

For system parameters $m = 128$ and $n = 8$, we randomly generated 1000 instances to verify our recovering steps. In all these experiments, we collected $M = 119$ short lattice vectors in $\mathcal{L}([\boldsymbol{A}\ \boldsymbol{u}])^{\perp}$.

For 817 instances, we recovered $\boldsymbol{v} = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta} \rangle)$ (or $-\boldsymbol{v}$) and $\boldsymbol{v'} = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta'} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta'} \rangle)$ (or $-\boldsymbol{v'}$) at the same time.

Note that if we find the real $\boldsymbol{v}$, then we can recover the real k, but for $-\boldsymbol{v}$, we just recover $q - k$. We can not tell whether the $\tilde{k}$ we recovered is the real k or just $q - k$ by now. The same situation happens for k'. However, there are four possible values for the pair (k, k'). We can try all the four possible values to carry out our attack to check which pair is the correct one[1].

Hence, without loss of generality, we can assume we recovered the correct k and k', together with

$$\boldsymbol{v} = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta} \rangle),$$

and

$$\boldsymbol{v'} = (\langle \boldsymbol{x}_1, \boldsymbol{\Delta'} \rangle, \cdots, \langle \boldsymbol{x}_M, \boldsymbol{\Delta'} \rangle).$$

[1] In fact, the following analysis of our attack can help us recover the correct k and k'. For simplicity, we assume we know the correct k and k'.

4.2 Recover p, ck and ck'

By the relation $ck \cdot r_i + ck' \cdot r'_i \equiv 0 \bmod p$ for Compact-LWE, it follows that

$$ck \cdot \langle \boldsymbol{x}_i, \boldsymbol{\Delta} \rangle + ck' \cdot \langle \boldsymbol{x}_i, \boldsymbol{\Delta}' \rangle \equiv 0 \bmod p,$$

which implies that the system of linear equations

$$\begin{pmatrix} \langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle \\ \langle \boldsymbol{x}_t, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_t, \boldsymbol{\Delta}' \rangle \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \equiv 0 \bmod p$$

has a non-trivial solution $(ck, ck')^T$ for any $s, t \in \{1, \ldots, M\}$. We immediately have

$$\det \begin{pmatrix} \langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle \\ \langle \boldsymbol{x}_t, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_t, \boldsymbol{\Delta}' \rangle \end{pmatrix} = 0 \bmod p.$$

Denote by Δ_{st} the determinant of

$$\begin{pmatrix} \langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle \\ \langle \boldsymbol{x}_t, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_t, \boldsymbol{\Delta}' \rangle \end{pmatrix}.$$

Then $p | \Delta_{st}$ for any $s, t \in \{1, \ldots, M\}$, which implies that p can be recovered by computing the greatest common divisor of $\Delta_{12}, \Delta_{13}, \ldots, \Delta_{M-1,M}$.

When p is recovered, we can choose an s such that $\langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle$ is coprime to p. With high probability, such s exists. Then we can solve the following linear equation

$$\langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle \cdot x_1 + \langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle \cdot x_2 \equiv 0 \bmod p$$

for equivalent ck and ck'.

For simplicity, we can even let $ck_1 = -\langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle^{-1} \langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle \mod p$ and $ck'_1 = 1$ as the equivalent ck and ck'.

We state this step in algorithm 4.

Algorithm 4 Recover p

Input: k, k', $\boldsymbol{v}$,$\boldsymbol{v}'$ obtained in algorithm 3
Output: p, ck_1, ck'_1

1: Compute

$$\Delta_{st} = \det \begin{pmatrix} \langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle \\ \langle \boldsymbol{x}_t, \boldsymbol{\Delta} \rangle & \langle \boldsymbol{x}_t, \boldsymbol{\Delta}' \rangle \end{pmatrix}$$

for $s, t \in \{1, 2, \ldots, M\}$;
2: Compute the greatest common divisor p of Δ_{st} for $s, t \in \{1, 2, \ldots, M\}$;
3: Choose an s such that $\langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle$ is coprime to p and solve the following linear equation

$$\langle \boldsymbol{x}_s, \boldsymbol{\Delta} \rangle \cdot x_1 + \langle \boldsymbol{x}_s, \boldsymbol{\Delta}' \rangle \cdot x_2 \equiv 0 \bmod p$$

for ck_1, ck'_1.
4: **return** p, ck_1, ck'_1.

Experimental Results of Recovering p. In our experiments, we can always recover the correct p and the desired ck_1, ck_1'.

Remark 2. When we obtain $q-k$ in Section 4.1, we can recover the correct p using Algorithm 4 since

$$(q-k)\cdot \boldsymbol{pk}\cdot(\boldsymbol{x}_1^T,\ldots,\boldsymbol{x}_M^T) \bmod q = -(\langle \boldsymbol{x}_1,\boldsymbol{\Delta}\rangle,\ldots,\langle \boldsymbol{x}_M,\boldsymbol{\Delta}\rangle),$$

while

$$k\cdot \boldsymbol{pk}\cdot(\boldsymbol{x}_1^T,\ldots,\boldsymbol{x}_M^T) \bmod q = (\langle \boldsymbol{x}_1,\boldsymbol{\Delta}\rangle,\ldots,\langle \boldsymbol{x}_M,\boldsymbol{\Delta}\rangle),$$

hence the output will be the same. Similar results can be obtained for the case $q-k'$. When we obtain a wrong k or k', we expect Algorithm 4 cannot output a value passing the test for p. Furthermore, when k, k' or $q-k, q-k'$ are recovered, the ratio $ck/ck' \bmod p$ or $ck'/ck \bmod p$ is recovered and for $q-k, k'$ or $k, q-k'$, we will recover $-ck/ck' \bmod p$ or $-ck'/ck \bmod p$.

4.3 Recover sk and sk'

Once recovering k, k', p and ck_1, ck_1', we continue to compute an LLL-reduced basis for the kernel lattice $\mathcal{L}(\boldsymbol{A})^{\perp}$, and find short vectors from the reduced basis as stated in Section 4.1. We denote by $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_L$ the short basis vectors in the kernel lattice $\mathcal{L}(\boldsymbol{A})^{\perp}$.

It is easy to conclude that

$$\begin{aligned} k\cdot\langle \boldsymbol{z}_i,\boldsymbol{pk}\rangle &\equiv sk\,\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle + \langle \boldsymbol{z}_i,\boldsymbol{\Delta}\rangle \bmod q,\\ k'\cdot\langle \boldsymbol{z}_i,\boldsymbol{pk}'\rangle &\equiv sk'\,\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle + \langle \boldsymbol{z}_i,\boldsymbol{\Delta}'\rangle \bmod q. \end{aligned}$$

Since sk, sk' are very small relatively to q, the values $sk\,\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle + \langle \boldsymbol{z}_i,\boldsymbol{\Delta}\rangle$ and $sk'\,\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle + \langle \boldsymbol{z}_i,\boldsymbol{\Delta}'\rangle$ are within $(-q/2, q/2)$. Hence these values can be computed by $Z_i = k\cdot\langle \boldsymbol{z}_i,\boldsymbol{pk}\rangle \bmod q$ and $Z_i' = k'\cdot\langle \boldsymbol{z}_i,\boldsymbol{pk}'\rangle \bmod q$ into $(-q/2, q/2)$ respectively.

Then we compute $ck_1\cdot Z_i + ck_1'\cdot Z_i' \bmod p$, which equals to $(ck_1\cdot sk + ck_1'\cdot sk')\cdot\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle \bmod p$. Therefore, once obtaining $\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle$ coprime to p, we will obtain the value

$$(ck_1\cdot sk + ck_1'\cdot sk') \bmod p = (ck_1\cdot Z_i + ck_1'\cdot Z_i')\cdot(\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle)^{-1} \bmod p.$$

That is

$$(sk, sk')\cdot\begin{pmatrix} ck_1\\ ck_1'\end{pmatrix} \equiv (ck_1\cdot Z_i + ck_1'\cdot Z_i')\cdot(\langle \boldsymbol{z}_i,\boldsymbol{u}\rangle)^{-1} \bmod p.$$

Again note that sk, sk' are very small relatively to q, so we can use Babai's algorithm to find a small solution (sk_1, sk_1') as stated in Section 2.2.

So this procedure consists of the following steps.

Algorithm 5 Recover sk,sk'

Input: ck_1,ck'_1 and public key
Output: sk, sk'

1: Compute an LLL-reduced basis for the kernel lattice $\mathcal{L}(A)^{\perp}$, from which we collect short vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L$;
2: Find a $\boldsymbol{z}_i$ such that $\gcd(\langle \boldsymbol{z}_i, \boldsymbol{u} \rangle, p) = 1$, then compute $Z_i = k \cdot \langle \boldsymbol{z}_i, \boldsymbol{pk} \rangle \bmod q$ and $Z'_i = k' \cdot \langle \boldsymbol{z}_i, \boldsymbol{pk}' \rangle \bmod q$ in $(-q/2, q/2)$ respectively, and finally we compute $ck_1 \cdot sk + ck'_1 \cdot sk' \equiv (ck_1 \cdot Z_i + ck'_1 \cdot Z'_i) \cdot (\langle \boldsymbol{z}_i, \boldsymbol{u} \rangle)^{-1} \bmod p$;
3: Using Babai's algorithm to recover (sk, sk');
4: **return** (sk, sk').

Experimental Results of Recovering sk and sk' Similar to Section 4.1, we set the bound to be 2^6 to obtain short $\boldsymbol{z}_i$.

Since the parameters $sk, sk' \leq sk\_max = 229119$, $|\langle \boldsymbol{z}_i, \boldsymbol{u} \rangle| \leq \|\boldsymbol{z}_i\|_1 \cdot \|\boldsymbol{u}\|_\infty \leq 2^{42}$. Hence $|sk \cdot \langle \boldsymbol{z}_i, \boldsymbol{u} \rangle| < q/4$. Together with $|\langle \boldsymbol{z}_i, \boldsymbol{\Delta} \rangle| \leq q/4$, we have

$$|sk \cdot \langle \boldsymbol{z}_i, \boldsymbol{u} \rangle + \langle \boldsymbol{z}_i, \boldsymbol{\Delta} \rangle| < \frac{q}{2}.$$

In all instances, correct sk_1, sk'_1 was recovered.

Remark 3. We would like to point out that in this step, we can tell whether $\tilde{k}$ we recovered in Section 4.1 is k or $q-k$ by the sign of the recovered sk and sk'. For example, when $q-k, k'$ are recovered in Section 4.1,

$$\begin{aligned} Z_i &= (q-k) \langle \boldsymbol{z}_i, \boldsymbol{pk} \rangle \bmod q = -(sk \langle \boldsymbol{z}_i, \boldsymbol{u} \rangle + \langle \boldsymbol{z}_i, \boldsymbol{\Delta} \rangle), \\ Z'_i &= k' \langle \boldsymbol{z}_i, \boldsymbol{pk}' \rangle \bmod q = sk' \langle \boldsymbol{z}_i, \boldsymbol{u} \rangle + \langle \boldsymbol{z}_i, \boldsymbol{\Delta}' \rangle. \end{aligned}$$

Hence solving (x, x') such that

$$ck_1 \cdot x + ck'_1 \cdot x' \bmod p = (ck_1 \cdot Z_i + ck'_1 \cdot Z'_i) \cdot (\langle \boldsymbol{z}_i, \boldsymbol{u} \rangle)^{-1} \bmod p$$

is expected to recover $(-sk, sk')$. Similar results can be obtained for the case $k, q-k'$ and $q-k, q-k'$.

4.4 Recover $\boldsymbol{s}$ and $\boldsymbol{s}'$

Now suppose we obtain the correct k, k', sk, sk', and denote

$$\begin{aligned} \boldsymbol{t} &= k \cdot \boldsymbol{pk} - sk \cdot \boldsymbol{u} \equiv \boldsymbol{A} \cdot k\boldsymbol{s} + \boldsymbol{\Delta} \bmod q, \\ \boldsymbol{t}' &= k' \cdot \boldsymbol{pk}' - sk' \cdot \boldsymbol{u} \equiv \boldsymbol{A} \cdot k'\boldsymbol{s}' + \boldsymbol{\Delta}' \bmod q. \end{aligned}$$

That is

$$(\boldsymbol{t}, \boldsymbol{t}') \equiv (k\boldsymbol{s}, k'\boldsymbol{s}') \cdot \begin{pmatrix} \boldsymbol{A}^T & 0 \\ 0 & \boldsymbol{A}^T \end{pmatrix} + (\boldsymbol{\Delta}, \boldsymbol{\Delta}') \bmod q.$$

Together with the fact that $ck_1 \cdot \boldsymbol{\Delta} + ck_1' \cdot \boldsymbol{\Delta}' \equiv \mathbf{0} \bmod p$, which implies

$$(\boldsymbol{\Delta}, \boldsymbol{\Delta}') \cdot \begin{pmatrix} ck_1 \boldsymbol{I}_m \\ ck_1' \boldsymbol{I}_m \end{pmatrix} \equiv \mathbf{0} \bmod p,$$

we have

$$(\boldsymbol{t}, \boldsymbol{t}') = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \cdot \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ q\boldsymbol{I}_{2m} \end{pmatrix},$$

where $\boldsymbol{N}$ is an LLL-reduced basis for $\mathcal{L}\begin{pmatrix} ck_1 \boldsymbol{I}_m \\ ck_1' \boldsymbol{I}_m \end{pmatrix}_p^{\perp}$, $\boldsymbol{\alpha} = (k\boldsymbol{s}, k'\boldsymbol{s}')$, $\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A}^T & 0 \\ 0 & \boldsymbol{A}^T \end{pmatrix}$ and some integer vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.

Therefore, $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0) + \boldsymbol{\theta}\boldsymbol{K}$ with $(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0)$ an arbitrary solution satisfying $(\boldsymbol{t}, \boldsymbol{t}') = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0) \cdot \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ q\boldsymbol{I}_{2m} \end{pmatrix}$, $\boldsymbol{K}$ an LLL-reduced basis for $\mathcal{L}\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ q\boldsymbol{I}_{2m} \end{pmatrix}^{\perp}$ and some integer vector $\boldsymbol{\theta}$.

Divide $\boldsymbol{K}$ into 3 blocks $(\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}_3)$ corresponding to the length of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$. We obtain that

$$\begin{cases} \boldsymbol{\alpha} = \boldsymbol{\alpha}_0 + \boldsymbol{\theta}\boldsymbol{K}_1, \\ \boldsymbol{\beta} = \boldsymbol{\beta}_0 + \boldsymbol{\theta}\boldsymbol{K}_2, \\ \boldsymbol{\gamma} = \boldsymbol{\gamma}_0 + \boldsymbol{\theta}\boldsymbol{K}_3. \end{cases}$$

Therefore,

$$\begin{aligned} (\boldsymbol{t}, \boldsymbol{t}') &= (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \cdot \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ q\boldsymbol{I}_{2m} \end{pmatrix} \\ &= (\boldsymbol{\alpha}_0 + \boldsymbol{\theta}\boldsymbol{K}_1)\boldsymbol{M} + (\boldsymbol{\beta}_0 + \boldsymbol{\theta}\boldsymbol{K}_2)\boldsymbol{N} + q(\boldsymbol{\gamma}_0 + \boldsymbol{\theta}\boldsymbol{K}_3) \end{aligned}$$

In Compact-LWE, $(\boldsymbol{\beta}_0 + \boldsymbol{\theta}\boldsymbol{K}_2)\boldsymbol{N} = (\boldsymbol{\Delta}, \boldsymbol{\Delta}')$ for some $\boldsymbol{\theta}$. Note that a short $(\boldsymbol{\beta}_0 + \boldsymbol{\theta}'\boldsymbol{K}_2)\boldsymbol{N}$ can be used to achieve equivalent decryption. It suffices to find a $\boldsymbol{\theta}'$ such that $(\boldsymbol{\beta}_0 + \boldsymbol{\theta}'\boldsymbol{K}_2)\boldsymbol{N}$ short enough.

By using lattice reduction algorithm to $\boldsymbol{K}_2\boldsymbol{N}$ and using Babai's nearest plane algorithm [2] with target vector $-\boldsymbol{\beta}_0\boldsymbol{N}$, we will find a short vector $(\boldsymbol{\beta}_0 + \boldsymbol{\theta}'\boldsymbol{K}_2)\boldsymbol{N}$, hence a corresponding $\boldsymbol{\theta}'$.

Finally, we compute $\boldsymbol{\alpha}' = (\boldsymbol{\alpha}_0 + \boldsymbol{\theta}'\boldsymbol{K}_1)\boldsymbol{M}$ and solve $(\boldsymbol{s}_1, \boldsymbol{s}_1')$ such that

$$(k\boldsymbol{s}_1, k'\boldsymbol{s}_1') \cdot \begin{pmatrix} \boldsymbol{A}^T & 0 \\ 0 & \boldsymbol{A}^T \end{pmatrix} = \boldsymbol{\alpha}' = (\boldsymbol{\alpha}_0 + \boldsymbol{\theta}'\boldsymbol{K}_1) \cdot \boldsymbol{M}.$$

If the output $(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_1') = (\boldsymbol{\beta}_0 + \boldsymbol{\theta}'\boldsymbol{K}_2)\boldsymbol{N}$ is short enough, we will complete the equivalent decryption.

To sum up all these steps, we have the following Algorithm 6.

Algorithm 6 Construct $\boldsymbol{s}, \boldsymbol{s}'$ to Implement Equivalent Decryption.

Input: Public parameters $\boldsymbol{A}$, $\boldsymbol{u}$,
Recovered parameters $k, k', sk, sk', ck_1, ck_1', p$;
Output: Equivalent private parameters $\boldsymbol{s}, \boldsymbol{s}'$;
1: Compute an LLL-reduced basis $\boldsymbol{N}$ as stated above;
2: Compute a solution $(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0)$ for $(\boldsymbol{t}, \boldsymbol{t}') = \boldsymbol{x} \cdot \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ q\boldsymbol{I}_{2m} \end{pmatrix}$;
3: Compute an LLL-reduced basis $\boldsymbol{K}$ for $\mathcal{L} \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ q\boldsymbol{I}_{2m} \end{pmatrix}^{\perp}$;
4: Find a $\boldsymbol{\theta}'$ such that $(\boldsymbol{\beta}_0 + \boldsymbol{\theta}'\boldsymbol{K}_2)\boldsymbol{N}$ is short;
5: Compute $\boldsymbol{\alpha}' = (\boldsymbol{\alpha}_0 + \boldsymbol{\theta}'\boldsymbol{K}_1)\boldsymbol{M}$ and solve $(\boldsymbol{s}_1, \boldsymbol{s}_1')$ satisfying the following fomula

$$(k\boldsymbol{s}_1, k'\boldsymbol{s}_1') \cdot \begin{pmatrix} \boldsymbol{A}^T & 0 \\ 0 & \boldsymbol{A}^T \end{pmatrix} = \boldsymbol{\alpha}' = (\boldsymbol{\alpha}_0 + \boldsymbol{\theta}'\boldsymbol{K}_1) \cdot \boldsymbol{M};$$

6: **return** $(\boldsymbol{s}_1, \boldsymbol{s}_1')$.

Experimental Results of Recovering $\boldsymbol{s}$ and $\boldsymbol{s}'$ In our experiments, we found a pair of $(\boldsymbol{s}_1, \boldsymbol{s}_1')$ such that the corresponding $(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_1')$ was slightly shorter than the correct $(\boldsymbol{\Delta}, \boldsymbol{\Delta}')$ with infinity norm.

4.5 Equivalent Decryption

Suppose the outputted $(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_1')$ has infinity norm B_1, then $l\Delta_1 = \sum_{i=1}^{m} l_i \Delta_{1i} \in [-(w+2w') \cdot B_1, (w+2w') \cdot B_1]$. Hence $sk \cdot lu + l\Delta_1 \in [-(w+2w') \cdot B_1, (w+2w') \cdot B_1 + sk\_max \cdot (w+w') \cdot b']$.

When decrypting the ciphertext in the equivalent decryption, we will compute

$$\begin{aligned} d_1 &= (lpk - \langle \boldsymbol{la}, \boldsymbol{s}_1 \rangle) \cdot k \bmod q \\ &\equiv sk \cdot lu + l\Delta_1 \bmod q \\ d_2 &= (lpk' - \langle \boldsymbol{la}, \boldsymbol{s}_1' \rangle) \cdot k' \bmod q \\ &\equiv sk \cdot lu + l\Delta_1' \bmod q \end{aligned}$$

Since $sk \cdot lu + l\Delta_1,\ sk \cdot lu + l\Delta_1' \in [B_2, B_3]$ with $B_2 = -(w+2w') \cdot B_1$ and $B_3 = (w+2w') \cdot B_1 + sk\_max \cdot (w+w') \cdot b'$, we can traverse all possible values of r_1, r_2 such that $d_1' = d_1 + r_1 \cdot q, d_2' = d_2 + r_2 \cdot q \in [B_2, B_3]$, one of which must be the correct value of $sk \cdot lu + l\Delta_1,\ sk \cdot lu + l\Delta_1'$.

Once obtaining the correct $sk \cdot lu + l\Delta_1,\ sk \cdot lu + l\Delta_1'$, we compute $ck_1 \cdot d_1' + ck_1' \cdot d_2' \bmod p = (ck_1 \cdot sk + ck_1' \cdot sk') \cdot lu \bmod p$, hence we compute $lu = (ck_1 \cdot d_1' + ck_1' \cdot d_2') \cdot (ck_1 \cdot sk + ck_1' \cdot sk')^{-1} \bmod p$.

Since the message is always padded before being encrypted, we can decide whether we recover the correct message or not. In fact, since we know the values of $(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_1')$, there is usually some better strategy to determine the correct r_1 and r_2.

Experimental Results of Equivalent Decryption In our experiments, we found $(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}'_1)$ with infinity norm between $2^{55} \sim 2^{56}$. Therefore, the lower bound for $sk \cdot lu + l\Delta_1$, $sk' \cdot lu + l\Delta'_1$ is $-2^{63.17} \sim -2^{64.17}$, and the upper bound is $2^{63.60} \sim 2^{64.60}$. Hence we only considered the interval $(-q, q)$, which would produce 4 possibilities of combinations of $sk \cdot lu + l\Delta_1$ and $sk' \cdot lu + l\Delta'_1$. We implemented our equivalent decryption for these 4 possibilities, and the outputs were compared with the randomly generated messages. We found that all the values of $sk \cdot lu + l\Delta_1$, $sk' \cdot lu + l\Delta'_1$ lie in $(0, q)$ in our experiments.

Hence for all the random instances in our experiments, we computed $d_1 = (lpk - \langle \boldsymbol{la}, \boldsymbol{s}_1 \rangle) \cdot k \bmod q$ and $d_2 = (lpk' - \langle \boldsymbol{la}, \boldsymbol{s}'_1 \rangle) \cdot k' \bmod q$ in the interval $(0, q)$, and recovered all the correct messages successfully.

5 Message Recovery Attack against Compact-LWE

Note that in the encryption algorithm of Compact-LWE, $\sum_{i=1}^{m} l_i \cdot u_i$ plays a role as the ephemeral key. If we can recover its value, we can of course recover the corresponding message. Next we will show how to find the value by lattice-based algorithm.

5.1 Key Idea of Our Attack

Note that what we need is just the value of $\sum_{i=1}^{m} l_i \cdot u_i$ instead of the coefficients l_i's. Moreover, there may be more than one set of l_i's, which yield the same value. Hence, it is enough to recover any one set of l_i's.

The key observation is that if we could find a short enough vector $\boldsymbol{l'}$ such that it yields exactly the same ciphertext that is generated by $\boldsymbol{l}$, then $\sum_{i=1}^{m} l'_i \cdot u_i$ must equal to $\sum_{i=1}^{m} l_i \cdot u_i$. More precisely, we have

Lemma 1. *Given a Compact-LWE ciphertext* $\boldsymbol{c} = (\boldsymbol{la}, d, lpk, lpk')$ *where*

$$\begin{aligned} \boldsymbol{la} &= \sum_{i=1}^{m} l_i \cdot \boldsymbol{a}_i, \\ lpk &= \sum_{i=1}^{m} l_i \cdot pk_i \bmod q, \\ lpk' &= \sum_{i=1}^{m} l_i \cdot pk'_i \bmod q, \end{aligned}$$

for any vector $\boldsymbol{l}'$ *satisfying*

$$0 \leq \sum_{i=1}^{m} l'_i(sk \cdot u_i + r_i + e_i \cdot p) < q, \tag{5}$$

$$0 \leq \sum_{i=1}^{m} l'_i(sk' \cdot u_i + r'_i + e'_i \cdot p) < q \tag{6}$$

$$0 \leq \sum_{i=1}^{m} l'_i u_i < p, \tag{7}$$

and

$$\sum_{i=1}^{m} l'_i \cdot \boldsymbol{a}_i = \boldsymbol{la}, \tag{8}$$

$$\sum_{i=1}^{m} l'_i \cdot pk_i = lpk \bmod q, \tag{9}$$

$$\sum_{i=1}^{m} l'_i \cdot pk'_i = lpk' \bmod q, \tag{10}$$

then

$$\sum_{i=1}^{m} l'_i \cdot u_i = \sum_{i=1}^{m} l_i \cdot u_i,$$

Proof. Following the steps of decryption, with $\boldsymbol{l}$ and $\boldsymbol{l}'$, we can compute

$$d_1 = \sum_{i=1}^{m} l_i \cdot k(pk_i - \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle) \bmod q = \sum_{i=1}^{m} l_i(sk \cdot u_i + r_i + e_i \cdot p) \mod q,$$

$$\tilde{d_1} = \sum_{i=1}^{m} l'_i \cdot k(pk_i - \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle) \bmod q = \sum_{i=1}^{m} l'_i(sk \cdot u_i + r_i + e_i \cdot p) \mod q.$$

Due to (8), (9), (10), we know that $d_1 = \tilde{d_1}$. Similarly $d'_1 = \tilde{d'_1}$. Therefore

$$\tilde{d_2} = ck \cdot \tilde{d_1} + ck' \cdot \tilde{d'_1} \bmod p = ck \cdot d_1 + ck' \cdot d'_1 \bmod p = d_2.$$

Because of (5) and (6), it holds that

$$\tilde{d_1} = \sum_{i=1}^{m} l'_i(sk \cdot u_i + r_i + e_i \cdot p),$$

$$\tilde{d'_1} = \sum_{i=1}^{m} l'_i(sk' \cdot u_i + r'_i + e'_i \cdot p).$$

Then we have

$$\tilde{d_2} = \sum_{i=1}^{m} l'_i \cdot u_i(ck \cdot sk + ck' \cdot sk') \bmod p,$$

and by the fact that $\tilde{d_2} = d_2$, we know

$$\tilde{d_3} = sckInv \cdot \tilde{d_2} \bmod p = sckInv \cdot d_2 \bmod p = d_3.$$

For Compact-LWE, we know

$$d_3 = \sum_{i=1}^{m} l_i \cdot u_i.$$

By (7), it holds that

$$\tilde{d_3} = \sum_{i=1}^{m} l'_i u_i,$$

and then the lemma follows.

A problem remained is that we can not check if ANY given vector $\boldsymbol{l'}$ satisfied (5)-(7) or not, since we do not know the private keys such as sk, sk'. However, it can be easily concluded that if $\boldsymbol{l'}$ is SHORT enough, then it must satisfy (5)-(7). For example, if vector $\boldsymbol{l'} = (l'_1, l'_2, \ldots, l'_m)$ satisfies that the sum of all its positive entries is between w and $w+w'$, the sum of all its negative entries of $\boldsymbol{l'}$ is between $-w'$ and 0, and $\sum_{i=1}^{m} l'_i \cdot u_i > 0$, then $\boldsymbol{l'}$ must satisfy (5)-(7).

Remark 4. In [10], the authors also considered such a ciphertext-only attack. However, they thought that one needs to guess the exact $\boldsymbol{l}$ to recover the message, which is incorrect due to our lemma.

5.2 Finding Short $\boldsymbol{l'}$ by Lattice Algorithm

By the discussion before, we only need to find short $\boldsymbol{l'}$ such that it satisfies (8), (9), (10).

Consider the following equations:

$$(l'_1, l'_2, \ldots, l'_m, x_1, x_2) \cdot \begin{pmatrix} \boldsymbol{a}_1 & pk_1 & pk'_1 \\ \cdots & \cdots & \cdots \\ \boldsymbol{a}_m & pk_m & pk'_m \\ \boldsymbol{0} & q & 0 \\ \boldsymbol{0} & 0 & q \end{pmatrix} = (\boldsymbol{la}, lpk, lpk').$$

Denote the matrix above by $\boldsymbol{A}$. Let $\boldsymbol{b} = (\boldsymbol{la}, lpk, lpk')$ and $\boldsymbol{x} = (\boldsymbol{l'}, x_1, x_2)$. Note that when $\boldsymbol{l'}$ is short, x_1 and x_2 are small too. Hence we can find $\boldsymbol{l'}$ by finding a short solution $\boldsymbol{x}$ for $\boldsymbol{x} \cdot \boldsymbol{A} = \boldsymbol{b}$ using Algorithm 2. After recovering $\boldsymbol{l'}$, we can recover the ephemeral key $\sum_{i=1}^{m} l_i \cdot u_i$ and hence the message. The whole message recovery attack is summarized as in Algorithm 7.

Algorithm 7 Ciphertext-only attack on Compact-LWE

Input: Public parameters m, n, q, t, b, b', w, w';
Function f used in basic encryption algorithm of Compact-LWE;
Public key $\boldsymbol{PK} = (\boldsymbol{a_1}, \boldsymbol{a_2}, \cdots, \boldsymbol{a_m}, \boldsymbol{u}, \boldsymbol{pk}, \boldsymbol{pk'})$;
A ciphertext $\boldsymbol{c} = (\boldsymbol{la}, d, lpk, lpk')$;
Output: Message $v \in \mathbb{Z}_t$ with respect to $\boldsymbol{c}$
1: Construct the matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b} = (\boldsymbol{la}, lpk, lpk')$ as stated above;
2: Call algorithm 2 with $\boldsymbol{A}$ and $\boldsymbol{b}$ as input to obtain a short integer solution $\boldsymbol{x}_0$;
3: Retrieve the first m entries of $\boldsymbol{x}_0$ to generate $\boldsymbol{l'} = (l'_1, \ldots, l'_m)$;
4: Compute $lu' = \sum_{i=1} l'_i u_i$ and $v' = f^{-1}_{lu'}(\boldsymbol{c})$;
5: **return** v';

Experimental Results. We tested our message recovery attack with LatticeSolve function implemented in NTL [17]. In our experiments, we randomly generated 100 Compact-LWE instances and recovered all correct messages by our attack.

We have to point out that the plausibility of our message recovery attack relies on the fact that the dimension parameters ($m = 128, n = 8$) selected by Compact-LWE designers are fairly small. The existing lattice basis reduction algorithm and approximating-CVP algorithm work very well for such low-dimensional lattices.

6 Conclusions

In this paper, we presented two efficient cipher-only attacks against Compact-LWE and revealed the weakness of Compact-LWE. The small values of n and m recommended in [10] enable us to apply existing lattice basis reduction algorithms to obtain some short solutions, which suffice to recover the equivalent private key or the message and thus violate the security of Compact-LWE. The parameters of Compact-LWE should not be set small in this way to make the scheme lightweight. The security requirement of encryption scheme is not satisfied as the Compact-LWE designers originally expected.

References

1. Ajtai, M.: The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. pp. 10–19. STOC '98, ACM, New York, NY, USA (1998), http://doi.acm.org/10.1145/276698.276705
2. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. In: Mehlhorn, K. (ed.) STACS 85: 2nd Annual Symposium on Theoretical Aspects of Computer Science Saarbrücken, January 3-5, 1985. pp. 13–20. Springer Berlin Heidelberg, Berlin, Heidelberg (1985), https://doi.org/10.1007/BFb0023990
3. Boas, P.V.E.: Another NP-complete problem and the complexity of computing short vectors in lattices. Math. Dept. Report 81-04. Univ. of Amsterdam (1981)

4. Bootle, J., Tibouchi, M.: Cryptanalysis of compact-LWE. Cryptology ePrint Archive, Report 2017/742 (2017), https://eprint.iacr.org/2017/742
5. Bootle, J., Tibouchi, M., Xagawa, K.: Cryptanalysis of compact-LWE. In: Smart, N.P. (ed.) Topics in Cryptology – CT-RSA 2018. Lecture Notes in Computer Science, vol. 10808, pp. 80–97. Springer International Publishing, Cham (2018)
6. Dinur, I., Kindler, G., Safra, S.: Approximating-CVP to within almost-polynomial factors is NP-hard. In: Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280). pp. 99–109 (Nov 1998)
7. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1374376.1374407
8. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. pp. 212–219. STOC '96, ACM, New York, NY, USA (1996), http://doi.acm.org/10.1145/237814.237866
9. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (Dec 1982), https://doi.org/10.1007/BF01457454
10. Liu, D., Li, N., Kim, J., Nepal, S.: Compact-LWE: a public key encryption scheme. NIST Post-Quantum Cryptography (2017), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/Compact_LWE.zip
11. Liu, D., Li, N., Kim, J., Nepal, S.: Compact-LWE: Enabling practically lightweight public key encryption for leveled IoT device authentication. Cryptology ePrint Archive, Report 2017/685 (2017), https://eprint.iacr.org/2017/685
12. Nguyen, P., Stern, J.: Merkle-Hellman revisited: A cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations, Lecture Notes in Computer Science, vol. 1294, pp. 198–212. Springer Berlin Heidelberg, Berlin, Heidelberg (1997), https://doi.org/10.1007/BFb0052236
13. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. pp. 84–93. STOC '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1060590.1060603
14. Stein, W., et al.: Sage Mathematics Software Version 7.5.1. The Sage Development Team (2017)
15. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE, Santa Fe, NM, USA (Nov 1994)
16. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (Oct 1997), http://dx.doi.org/10.1137/S0097539795293172
17. Shoup, V.: NTL: A library for doing number theory. http://www.shoup.net/ntl (2001)