Hindawi
Security and Communication Networks
Volume 2019, Article ID 8319508, 6 pages
https://doi.org/10.1155/2019/8319508

Research Article

Improved Cryptanalysis of a Fully Homomorphic Symmetric Encryption Scheme

Quanbo Qu,[1] **Baocang Wang**,[1,2] **Yuan Ping**,[2] **and Zhili Zhang**[2]

[1]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China*
[2]*School of Information Engineering, Xuchang University, Xuchang 461000, China*

Correspondence should be addressed to Zhili Zhang; zlzhangxc@163.com

Received 5 March 2019; Accepted 22 April 2019; Published 2 June 2019

Guest Editor: Mingwu Zhang

Homomorphic encryption is widely used in the scenarios of big data and cloud computing for supporting calculations on ciphertexts without leaking plaintexts. Recently, Li et al. designed a symmetric homomorphic encryption scheme for outsourced databases. Wang et al. proposed a successful key-recovery attack on the homomorphic encryption scheme but required the adversary to know some plaintext/ciphertext pairs. In this paper, we propose a new ciphertext-only attack on the symmetric fully homomorphic encryption scheme. Our attack improves the previous Wang et al.'s attack by eliminating the assumption of known plaintext/ciphertext pairs. We show that the secret key of the user can be recovered by running lattice reduction algorithms twice. Experiments show that the attack successfully and efficiently recovers the secret key of the randomly generated instances with an overwhelming probability.

1. Introduction

With the rapid development of big data, the significance of privacy and security issues was highly regarded. A series of cryptoghraphic applications, such as fair electronic transaction [1], outsourcing data classification [2], lightweight security system of Internet of Things [3], mobile Ecommerce [4], and data mining based on homomorphic encryption, have been proposed.

Homomorphic encryption schemes allow users to meaningfully calculate ciphertexts without knowing the underlying plaintexts. For example, the RSA cryptosystem [5] (Pallier cryptosystem [6], respectively) only supports homomorphic multiplications (additions, respectively) on ciphertexts. In 2009, Gentry [7] designed the first fully homomorphic encryption scheme with ideal lattices. Thereafter, significant efforts had been performed to improve the efficiency of homomorphic encryption schemes [8–10]. However, all the known fully homomorphic encryptions are criticized for the high ciphertext expansion and ciphertext refreshing costs and hence cannot be directly used in practice. So researchers designed some cryptographic schemes with homomorphic properties dedicated to some concrete computing scenarios [11–15].

Recently, Li et al. [16] designed a symmetric homomorphic encryption scheme for outsourced databases that allow multiple data owners to efficiently share their data securely without compromising the privacy of the data. However, Wang et al. [17] observed that if some plaintext/ciphertext pairs were succesfully overdropped, one can efficiently recover the corresponding secret key of the scheme from the obtained plaintext/ciphertext pairs.

In practical scenarios, it may be difficult for the adversary to capture plaintext/ciphertext pairs. In this paper, we propose a new efficient cryptanalytic attack on Li et al.'s homomorphic encryption scheme. The attack consists of two stages. In the first stage, we separate the parts of the ciphertexts, which contain no secret key s. In the second stage, we separate the parts of the ciphertexts, which contain neither secret key s nor q. Thus s and q can be calculated during an acceptable time. The whole attack needs only several ciphertexts without corresponding plaintexts.

This paper is organised as follows. In Section 2, we review Li et al.'s symmetric homomorphic encryption scheme and

introduce the concept of lattice. In Section 3, we propose our attack and give the experimental results. In Section 4, we conclude our work.

2. Preliminaries

2.1. Notations. In this paper, the symbol $\mathbb{Z}$ is used to denote the ring of integer. Matrices are represented with bold upper-case characters like $\mathbf{B}$, while vectors are represented with bold lower-case characters like $\mathbf{v}$. All of the vectors in this paper are represented as row vectors. The symbol $\|\mathbf{v}\|$ means the length of vector $\mathbf{v}$ under the Euclidean norm, while the symbol $|p|_2$ means the bit length of integer p.

The symbol $\ll$ means "much less than", *i.e.* if $a \ll b$, the ratio a/b is a negligible function $negl(\lambda)$ of the security parameter λ. In mathematics, a negligible function $\mu(x)$ means that for any polynomial function $poly(x)$, there exists an integer N_c such that for any $x > N_c$,

$$|\mu(x)| < \frac{1}{poly(x)}. \tag{1}$$

2.2. Symmetric Homomorphic Encryption. The symmetric homomorphic encryption scheme proposed by Li et al. comprises these three algorithms as follows:

(i) *Key generation algorithm* $KeyGen()$

$$(s, q, p) \longleftarrow KeyGen(\lambda). \tag{2}$$

Input a security parameter λ, this algorithm outputs a secret key $SK = (s, q)$ and a public parameter p, where $q \ll p$.

(ii) *Encryption algorithm* $E()$

$$c = E(SK, m, d) = s^d (rq + m) \pmod p. \tag{3}$$

Input a secret key SK, a plaintext $m \in F_q$ and a parameter d, this algorithm outputs a ciphertext $c = E(SK, m, d)$. Notice that the parameter r should satisfy $|r|_2 + |q|_2 < |p|_2$.

(iii) *Decryption algorithm* $D()$

$$m = D(SK, c, d) = \left(c \times s^{-d} \pmod p\right) \pmod q. \tag{4}$$

Input a secret key SK, a ciphertext $c \in F_p$ and the ciphertext's degree d, this algorithm outputs a ciphertext $c \longleftarrow E(SK, m, d)$. The proof of the correctness is simple:

$$\begin{aligned} D(SK, c, d) &= \left(c \times s^{-d} \pmod p\right) \pmod q \\ &= \left(\left(s^d (rq + m) \pmod p\right) \times s^{-d} \pmod p\right) \pmod q \\ &= (rq + m) \pmod q \\ &= m. \end{aligned} \tag{5}$$

Notice that the correctness of $(rq + m) \pmod q = m$ requires $q \ll p$ and $|r|_2 + |q|_2 < |p|_2$.

The symmetric homomorphic encryption scheme proposed by Li et al. supports homomorphic addition and multiplication and is used to construct their secure outsourced comparison scheme and privacy-preserving mining solutions. Though our attack needs no homomorphic properties, we still list a brief proof, for the reason that it implies the setting of parameters.

(i) *Homomorphic addition*: For the ciphertext c_1, c_2 of two plaintexts m_1, m_2, we have

$$\begin{aligned} (c_1 + c_2) \pmod p &= s^{d_1} (r_1 q + m_1) \pmod p \\ &\quad + s^{d_2} (r_2 q + m_2) \pmod p \\ &= s^{d_1} ((r_1 + r_2) q + m_1 + m_2) \pmod p, \ d_1 = d_2. \end{aligned} \tag{6}$$

The correct decryption of $c_1 + c_2$ requires $d_1 = d_2$ and $(r_1 + r_2)q + m_1 + m_2 < p$.

(ii) *Homomorphic multiplication*: For the ciphertext c_1, c_2 of two plaintexts m_1, m_2, we have

$$\begin{aligned} (c_1 \times c_2) \pmod p &= s^{d_1} (r_1 q + m_1) \pmod p \\ &\quad \times s^{d_2} (r_2 q + m_2) \pmod p \\ &= s^{d_1 + d_2} \left(r_1 r_2 q^2 + r_1 q m_2 + r_2 q m_1 + m_1 \times m_2\right) \pmod p \\ &= s^{d_1 + d_2} \left((r_1 r_2 q + r_1 m_2 + r_2 m_1) q + m_1 \times m_2\right) \pmod p. \end{aligned} \tag{7}$$

The correct decryption of $c_1 \times c_2$ requires $(r_1 r_2 q + r_1 m_2 + r_2 m_1)q + m_1 \times m_2 < p$.

2.3. Lattice. An m-dimension lattice $\mathscr{L}$ can be regarded as a set of all integer coefficient linear combinations of basis vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m\}$, that is $\mathscr{L}(\mathbf{B}) = \{\mathbf{v} = \sum_{i=1}^{m} a_i \mathbf{b}_i \mid a_i \in \mathbb{Z}, \ \mathbf{b}_i \in \mathbb{R}^n\}$. If $m = n$, we call that $\mathscr{L}$ is a full-rank lattice.

One of the most famous problems involving lattice is the shortest vector problem (SVP). Given a basis of a lattice, the goal is to find one non-zero vector, which has the shortest length $\|\mathbf{v}\|$. Some approximation algorithms are usually used for solving SVP as oracles, such as LLL and BKZ algorithms. The LLL algorithm is developed by A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovasz [18] in 1982. Given a basis of a lattice, the LLL algorithm outputs a reduced basis, which has a smaller size by Gram-Schmidt orthogonalization. It has various applications in cryptanalysis and other fields, such as integer programming and finding integer relations.

3. The Proposed Attack

3.1. Main Idea. Define $r_i q + m_i$ as a_i, and $c_i c_1^{-1}$ as f_i. Our attack has two stages. In the first stage, we construct a lattice $\mathscr{L}(\mathbf{F})$

Input: k ciphertexts $c_1, \cdots, c_k$
Output: private key (s^d, q)
for $i = 2$ **to** k **do**
$f_i = c_i c_1^{-1}$
end for
Set $\mathscr{L}(\mathbf{F})$ with f's
$\mathbf{f}$ = LLL_reduction($\mathscr{L}(\mathbf{F})$)
$a_1 = |(\mathbf{f})_k|$
for $i = 1$ **to** $k - 1$ **do**
$a_{i+1} = |(\mathbf{f})_i|$
end for
Set $\mathscr{L}(\mathbf{A})$ with a's
$\mathbf{a}$ = LLL_reduction($\mathscr{L}(\mathbf{A})$)
$r_1 = |(\mathbf{a})_k|$
for $i = 1$ **to** $k - 1$ **do**
$r_{i+1} = (|(\mathbf{a})_i| + r_1 * a_{i+1})/a_1$
end for
Compute $s^d = c_1 a_1^{-1} \bmod p$
Compute $q = a_1/r_1 \bmod p$
return (s^d, q)

Algorithm 1: The Attack Algorithm.

with f's and run the LLL algorithm to obtain a short vector, which contains a's. In the second stage, we construct a lattice $\mathscr{L}(\mathbf{A})$ with a's and run the LLL algorithm again to obtain a short vector which contains r's. It is obvious that the secret key (s^d, q) can be computed as $s^d = ca^{-1} \bmod p$ and $q = a/r \bmod p$. Notice that there is no need for the plaintexts m's in the attack.

3.2. Details. In this part, we give a specification of the attack in Algorithm 1. The input of the attack algorithm contains a set of ciphertexts $c = \{c_1, \ldots, c_k\}$ and the modular p of the encryption scheme without plaintexts m's. The output of the attack algorithm contains s^d and q which can be used to decrypt ciphertexts.

In the first stage, the lattice $\mathscr{L}(\mathbf{F})$ is constructed as

$$\mathbf{F} = \begin{pmatrix} p & 0 & \cdots & 0 & 0 \\ 0 & p & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & p & 0 \\ -f_2 & -f_3 & \cdots & -f_k & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_k \end{pmatrix}. \quad (8)$$

Thus a short vector $\mathbf{v} \in \mathscr{L}(\mathbf{F})$ could be expressed as

$$\begin{aligned} \mathbf{v} &= x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + \cdots + x_{k-1}\mathbf{v}_{k-1} + x_k\mathbf{v}_k \\ &= (x_1 p - x_k f_2, x_2 p - x_k f_3, \ldots, x_{k-1} p - x_k f_k, x_k). \end{aligned} \quad (9)$$

Through LLL's algorithm, we could obtain a short vector $\mathbf{f} \in \mathscr{L}(\mathbf{F})$.

Claim 1. Parameters $a_2, \ldots, a_k$ are close to $|(\mathbf{f})_1|, \ldots, |(\mathbf{f})_{k-1}|$, while a_1 is close to $|(\mathbf{f})_k|$, where $(\mathbf{f})_i$ means the i'th entry of the output vector $\mathbf{f}$.

Sketch of Proof. Suppose that $f_i a_1 = a_i + t_i p$, then we have $f_i/p - t_i/a_1 = a_i/(a_1 p)$. Because $|a| \ll |p|$, $a_i/(a_1 p) \approx 1/p < negl(\lambda)$. Hence, values of $f_2/p, \ldots, f_k/p$ are close to those of $t_2/a_1, \ldots, t_k/a_1$. Because $|x_{i-1} p - x_k f_i|$ is small, we can obtain that $x_{i-1}/x_k - f_i/p < negl(\lambda)$. Hence, values of $x_1/x_k, \ldots, x_{k-1}/x_k$ are close to those of $f_2/p, \ldots, f_k/p$. Since $t_2/a_1, \ldots, t_k/a_1$ and $x_1/x_k, \ldots, x_{k-1}/x_k$ are both close to $f_2/p, \ldots, f_k/p$, we obtain that values of $t_2/a_1, \ldots, t_k/a_1$ are close to $x_1/x_k, \ldots, x_{k-1}/x_k$. With a non-negligible probability, $x_1, \ldots, x_{k-1}$ equal $t_2, \ldots, t_k$, and x_k is equal to a_1.

As $|a_1|$ is close to $|x_k|$, we have $x_k \approx |a_1|$ or $x_k \approx -|a_1|$. Considering $|a|, |x| \ll p$, we believe that $|a| < p/2$, thus $a_1 \approx |x_k|$. Similarly, as $|x_{i-1} p - x_k f_i|$ is small, we have $|x_{i-1} p - x_k f_i| = (x_{i-1} p - x_k f_i) \bmod p = -x_k f_i \bmod p$ or $|x_{i-1} p - x_k f_i| = -(x_{i-1} p - x_k f_i) \bmod p = x_k f_i \bmod p$, i.e. $|x_{i-1} p - x_k f_i| = |x_k| f_i \bmod p$, thus $a_{i+1} = a_1 f_{i+1} \bmod p \approx |x_k| f_{i+1} \approx (\mathbf{f})_i$, $i = 1, \ldots, k-1$.

In the second stage, the lattice $\mathscr{L}(\mathbf{A})$ is constructed as

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 & \cdots & 0 & 0 \\ 0 & a_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_1 & 0 \\ -a_2 & -a_3 & \cdots & -a_k & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_k \end{pmatrix}. \quad (10)$$

Thus a short vector $\mathbf{w} \in \mathscr{L}(\mathbf{A})$ could be expressed as

$$\begin{aligned} \mathbf{w} &= y_1\mathbf{w}_1 + y_2\mathbf{w}_2 + \cdots + y_{k-1}\mathbf{w}_{k-1} + y_k\mathbf{w}_k \\ &= (y_1 a_1 - y_k a_2, y_2 a_1 - y_k a_3, \ldots, y_{k-1} a_1 - y_k a_k, y_k). \end{aligned} \quad (11)$$

Through LLL's algorithm, we could obtain a short vector $\mathbf{a} \in \mathscr{L}(\mathbf{A})$.

Claim 2. Parameters $r_1, r_2, \ldots, r_k$ are close to $|(\mathbf{a})_k|, (|(\mathbf{a})_1| + r_1 * a_2)/a_1, \ldots, (|(\mathbf{a})_{k-1}| + r_1 * a_k)/a_1$, where $(\mathbf{a})_i$ means the i'th entry of the output vector $\mathbf{a}$.

Sketch of Proof. Considering $a_i = r_i q + m_i$, we have $a_i/a_1 = (r_i q + m_i)/(r_1 q + m_1) \approx (r_i q)/(r_1 q) = r_i/r_1$. Hence, values of $a_2/a_1, \ldots, a_k/a_1$ are close to those of $r_2/r_1, \ldots, r_k/r_1$.

Because $|y_{i-1} a_1 - y_k a_i|$ is small, we can obtain that $y_{i-1}/y_k - a_i/a_1 < negl(\lambda)$. Hence, values of $y_1/y_k, \ldots, y_{k-1}/y_k$ are close to those of $a_2/a_1, \ldots, a_k/a_1$. Since $r_2/r_1, \ldots, r_k/r_1$ and $y_1/y_k, \ldots, y_{k-1}/y_k$ are both close to $a_2/a_1, \ldots, a_k/a_1$, we obtain that values of $r_2/r_1, \ldots, r_k/r_1$ are close to $y_1/y_k, \ldots, y_{k-1}/y_k$. With a non-negligible probability, $y_1, \ldots, y_{k-1}$ equal $r_2, \ldots, r_k$, and y_k is equal to r_1.

Likewise, we have $r_1 = |(\mathbf{a})_k|$ and $r_{i+1} = (|(\mathbf{a})_i| + r_1 * a_{i+1})/a_1$, $i = 1, \ldots, k-1$.

Since we have recovered all the a's and r's, the secret key (s^d, q) could be simply computed as $s^d = c_i a_i^{-1}$ and $q = a_i/r_i$. Parameters c_1, a_1 and r_1 are used to compute (s^d, q) in the algorithm.

Table 1: Results in experiments with different parameters.

$\|p\|_2$	$\|q\|_2$	$\|r\|_2$	m	k	Instances	Successes	Average Time
256	64	64	$\leq q/256$	20	100	98	28.65s
256	64	64	$\leq q/64$	20	100	10	29.20s
256	64	96	$\leq q/256$	20	100	23	39.70s
256	64	48	$\leq q/256$	20	100	2	25.09s
256	48	48	$\leq q/32$	20	100	51	20.56s

Table 2: Example for $|p|_2 = 256$.

The Encryption Scheme	KeyGen	p	96676600900811618538103427778952879986194318084870939447026698628675235799451
		q	17957200991146257161
		s	5710889004555322387
	E()	d	1
		m_1	47863226783593508
		r_1	13683909070104700313
		c_1	1403306518881428241485832704008411265962802817946162878687
		...	...
		...	...
		...	...
		m_{20}	8614814073974658
		r_{20}	15727429030749794270
		c_{20}	1612872723066650760854493964947953343890747803002904919536
The Cryptanalytic Algorithm	The First Stage	v_1	-19946395763615474623999402780809496189 3
		...	...
		v_{20}	24572470551643938262638513685031878 4901
	The Second Stage	w_1	-23371144323647625378158005673135813 3
		...	...
		w_{20}	13683909070104700313
		s	5710889004555322387
		q	17957200991146257161

3.3. Experiments. We run our proposed cryptanalytic algorithm on a personal computer using NTL library [19]. The environment is listed as follows:

(i) CPU: Intel(R) Core (TM) i3-7100 3.90GHz

(ii) RAM: 4.00GB

(iii) OS: Windows 10 64bit

Notice that the output of the attack algorithm is s^d. In [16], the parameter d is called *ciphertext degree* and is believed to be a small positive integer. It means that we could collect enough *d-degree ciphertexts* we need, and it is not difficult to recover s from s^d. For the *d-degree ciphertexts*, the encryption and decryption algorithms only require s^d rather than s. Thus, it is sufficient to break the scheme if we can recover s^d. For convenience, we suppose the parameter $d = 1$ in the encryption algorithm. When $d \neq 1$, our algorithm still works correctly.

The results are given in Table 1. As a result of the approximation, the chance of success is relavant to $|q|_2$, $|r|_2$, and $|m|_2$. The best situation is when $|q|_2 \approx |r|_2$ and $|m| \ll |rq|$. To make it easier to understand our proposed attack, we give an example to illustrate the procedure of the algorithm in Table 2. The parameters are set as $|p|_2 = 256$, $|q|_2 = 64$, $|r|_2 = 64$, and $m \leq q/256$.

Firstly, we compute all the f's with the input ciphertexts $(c_1, \ldots, c_k)$. Secondly, we use LLL algorithm to obtain a short vector for solving all a's. Thirdly, we use LLL algorithm again to obtain a short vector for solving all r's. Finally, we compute secret key (s, q) with a's and r's.

In practice, the first row $\mathbf{f}$ ($\mathbf{a}$, respectively) of the reduced basis of $\mathscr{L}(\mathbf{F})$ ($\mathscr{L}(\mathbf{A})$, respectively) which is a row vector with a short norm; thus we regard it as the short vector $\mathbf{v}$ ($\mathbf{w}$, respectively) we need.

The chance of success depends on the bit lengths of r, q, and m. In the first stage, $|a| \ll |p|$ requires $|rq| \ll |p|$. In the second stage, $a_i/a_1 = (r_i q + m_i)/(r_1 q + m_1) \approx (r_i q)/(r_1 q) = r_i/r_1$ requires $|m| \ll |rq|$. Thus we need to hold $|m| \ll |rq| \ll |p|$. Besides, the recovery of q from a also limit the setting of parameters. Notice that $a = rq + m$, where $0 \leq m < q$. If $m <$

Table 3: Comparision of Wang et al.'s Attack and Ours.

	$\|p\|_2$	$\|q\|_2$	$\|r\|_2$	Chance of Success	Average Time	Plaintexts Needed?
Attack in [17]	241	80	40	98%	0.1292s	Yes
Our Attack	256	64	64	98%	28.65s	No

r, the result of a/r is equal to q. However, when $|r|_2 < |q|_2$, we cannot confirm that $0 \le m < r$. In conclusion, the best situation is when $|r| \approx |q|$ and $|r|$ is slightly greater than $|q|$.

3.4. Complexity Analysis. We start with some simple conclusions about computational complexity.

(1) The computational complexity of modular inverse modulo p is $O(\log^3 p)$.

(2) The computational complexity of modular multiplication modulo p is $O(\log^2 p)$.

(3) The computational complexity of the LLL algorithm is $O(N^5(\log^2 B))$ [20], where N is the dimension of the lattice, and B is the maximum length of input basis under the Euclidean norm.

Combining (1) and (2), we can conclude that the computational complexity of calculating f's, r's, s, and q is $O(\log^3 p)$. In our attack, $N = k$ and $B \le \sqrt{(k-1)p^2 + 1}$. We can obtain the computational complexity of the LLL algorithm

$$\begin{aligned} O\left(N^5\left(\log^2 B\right)\right) &= O\left(k^5 \log^2 \sqrt{(k-1)\,p^2 + 1}\right) \\ &= O\left(k^5 \log^2 \sqrt{kp^2}\right) = O\left(k^5\left(\frac{1}{2}\log k + \log p\right)^2\right) \\ &= O\left(\frac{1}{4}k^5 \log^2 k + k^5 \log k \log p + k^5 \log^2 p\right). \end{aligned} \tag{12}$$

Because $k < p$, we can obtain $\log k < \log p$, thus $O(N^5(\log^2 B)) = O(k^5 \log^2 p)$.

In practice, the computational complexity of our attack is mainly dependent on that of the LLL algorithm. For example, suppose that $k^5 \log^2 p \le \log^3 p$ we can obtain $\log p \ge k^5$. If we set $k = 20$, then $\log p \ge 20^5 \approx 2^{21}$. It means that p is a 2^{21}-bit-length prime, while the bit length of the prime we usually use is $2048 = 2^{11}$ or $4096 = 2^{12}$.

Above all, the computational complexity of our attack algorithm is $O(k^5 \log^2 p)$. Obviously, it is worse than the complexity $O(\log^4 p)$ of Wang et al.'s attack [17]; however, our attack eliminates the assumption of known plaintext/ciphertext pairs.

3.5. Discussions. Notice that in the attack algorithm, the output of the LLL algorithm is a vector, such as $\mathbf{f}$ and $\mathbf{a}$, rather than a reduced basis. We regard the first row vector of an LLL-reduced basis as the goal short vector. We explain the reason below.

An δ-LLL-reduced $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ has two important properties:

(1) $\forall 1 \le i \le n,\ j < i,\ |\mu_{i,j}| \le 1/2$,

(2) $\forall 1 \le i \le n,\ \delta\|\tilde{\mathbf{b}}_i\|^2 \le \mu_{i+1,i}^2\|\tilde{\mathbf{b}}_i\|^2 + \|\tilde{\mathbf{b}}_{i+1}\|^2$,

where $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2, \ldots, \tilde{\mathbf{b}}_n\}$ is the Gram-Schmidt orthogonalization of $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$, and the coefficient $\mu_{i,j} = \langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle / \langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle$.

From these two properties, we can conclude that

$$\left\|\tilde{\mathbf{b}}_1\right\| \le 2^{(n-1)/2} \lambda_1(\mathscr{L}), \tag{13}$$

where $\lambda_1(\mathscr{L})$ is the length of the shortest non-zero vector in $\mathscr{L}(\mathbf{B})$. Please refer to [21] for more detailed introduction and proof.

The efficiency of our attack algorithm is mainly subject to the parameter k. Smaller k implies a greater chance of success for the reason that a's and r's can be recovered from the LLL algorithm easier while the runtime of the LLL algorithm rises rapidly. In our experiment, we recommend that k should be set as 20 considering both the chance of success and the runtime. In addition, the chance of success is also limited by sizes of r, q, and m.

Table 3 gives a comparision of Wang et al's attack and ours. Although the bit lengths of the parameters $|p|_2$, $|q|_2$ and $|r|_2$ are close but different, the average time of Wang et al's is much less than ours. However, the improved attack algorithm eliminates the assumption of known plaintext/ciphertext pairs, thus a ciphertext-only adversary can break the encryption scheme through this way.

4. Conclusion

In this paper, we propose a new attack algorithm on the symmetric homomorphic encryption scheme presented by Li et al. Our attack can recover the secret key pair from several ciphertexts without plaintexts. In our experiment, the attack an be finished during an acceptable period of time with recovering most of the secret key in the generated instances. For the cases $|p|_2 = 256$, $|q|_2 = |r|_2 = 64$, $m \le q/256$, the key-recovery cryptanalytic algorithm only takes about 29 seconds. Although the running time and the opportunity of success depend on the sizes of parameters, the attack algorithm can still be used in real practice to recover secret key pairs.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Key R&D Program of China under Grant No. 2017YFB0802000, the National Natural Science Foundation of China under Grant Nos. 61572390, U1736111, the National Cryptography Development Fund under Grant No. MMJJ20180111, the Plan For Scientific Innovation Talent of Henan Province under Grand no. 184100510012, the Program for Science & Technology Innovation Talents in Universities of Henan Province under Grant No. 18HASTIT022, and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province, Science & technology planning project in Henan Province (182102210124).

References

[1] M. Zhang, Y. Zhang, Y. Jiang, and J. Shen, "Obfuscating EVES algorithm and its application in fair electronic transactions in public clouds," *IEEE Systems Journal*, pp. 1–9, 2019.

[2] X. Li, Y. Zhu, J. Wang, Z. Liu, Y. Liu, and M. Zhang, "On the soundness and security of privacy-preserving SVM for outsourcing data classification," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 906–912, 2018.

[3] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo, and L. Zhou, "On emerging family of elliptic curves to secure internet of things: ECC comes of age," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 3, pp. 237–248, 2017.

[4] M. Zhang, Y. Yao, Y. Jiang, B. Li, and C. Tang, "Accountable mobile E-commerce scheme in intelligent cloud system transactions," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 6, pp. 1889–1899, 2018.

[5] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, pp. 169–179, 1978.

[6] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology —EUROCRYPT '99*, J. Stern, Ed., vol. 1592, pp. 223–238, Springer, Berlin, Germany, 1999.

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of Computing (STOC '09)*, pp. 169–178, ACM, New York, NY, USA, 2009.

[8] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS '11)*, pp. 97–106, Palm Springs, Calif, USA, October 2011.

[9] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Advances in Cryptology – CRYPTO 2011*, R. Phillip, Ed., vol. 6841, pp. 505–524, Springer, Berlin, Germany, 2011.

[10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in cryptology—EUROCRYPT 2010*, H. Gilbert, Ed., vol. 6110, pp. 24–43, Springer, Berlin, Germany, 2010.

[11] F. Armknecht and T. Strufe, "An efficient distributed privacy-preserving recommendation system," in *Proceedings of the 2011 the 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net'2011*, pp. 65–70, Italy, June 2011.

[12] C. Bosch, A. Peter, P. Hartel, and W. Jonker, "SOFIR: Securely outsourced Forensic image recognition," in *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2014*, pp. 2694–2698, Italy, May 2014.

[13] A. Jeckmans, A. Peter, and P. Hartel, "Efficient privacy-enhanced familiarity-based recommender system," in *Computer Security – ESORICS 2013*, J. Crampton, S. Jajodia, and K. Mayes, Eds., vol. 8134 of *Lecture Notes in Computer Science*, pp. 400–417, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[14] M. Naehrig, L. Kristin, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the ACM Cloud Computing Security Workshop, Ccsw 2011*, pp. 113–124, Chicago, Ill, Usa, October 2011.

[15] Z. Yang, S. Zhong, and R. N. Wright, "Privacy-preserving classification of customer data without loss of accuracy," in *Proceedings of the SDM*, pp. 92–102, 2005.

[16] L. Li, R. Lu, K.-K. R. Choo, A. Datta, and J. Shao, "Privacy-preserving-outsourced association rule mining on vertically partitioned databases," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1547–1861, 2016.

[17] B. Wang, Y. Zhan, and Z. Zhang, "Cryptanalysis of a symmetric fully homomorphic encryption scheme," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1460–1467, 2018.

[18] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[19] V. Shoup, *Ntl: A Library for Doing Number Theory*, vol. 01, 2003.

[20] Y. Park and J. Park, "Analysis of the upper bound on the complexity of LLL algorithm," *Journal of the Korean Society for Industrial and Applied Mathematics*, vol. 20, no. 2, pp. 107–121, 2016.

[21] O. Regev, "Lattices in computer science," in *Proceedings of the Lecture notes of a course given in Tel Aviv University*, vol. 31, 2004.