Attack on Fully Homomorphic Encryption over the Integers

Gu Chunsheng
School of Computer Engineering
Jiangsu Teachers University of Technology
Changzhou, China, 213001
guchunsheng@gmail.com

Abstract: This paper presents a heuristic attack on the fully homomorphic encryption over the integers by using lattice reduction algorithm. Our result shows that the FHE in [DGHV10] is not secure for some parameter settings. We also present an improvement scheme to avoid the lattice attack in this paper.

Keywords: Fully Homomorphic Encryption, Cryptanalysis, Lattice Reduction

1. Introduction

Rivest, Adleman and Dertouzos [RAD78] introduced a notion of privacy homomorphism. But until 2009, Gentry [Gen09] constructed the first fully homomorphic encryptions based on ideal lattice, all previous schemes are insecure. Following the breakthrough of [Gen09], there is currently great interest on fully-homomorphic encryption [SV10, vDGHV10, SS10, GH11a, GH11b, BV11a, BV11b, BGV11, CJMNT11, CMNT11]. In these schemes, the simplest one is certainly the one of van Dijk, Gentry, Halevi and Vaikuntanathan [DGHV10]. The public key of this scheme is a list of approximate multiples \( \{x_i = q_i p + 2r_i\}_{i=1}^{T} \) for an odd integer \( p \), where \( q_i, r_i \) is the uniform random integers over \( \mathbb{Z} \) such that \( |r| < 2^{\lambda - 1} \). The secret key is \( p \). To encrypt a message bit \( m \), the ciphertext is evaluated as

\[ c = \sum_{i=1}^{T} x_i + 2r + m, \]

where \( |r| < 2^{\lambda - 1} \). To decrypt a ciphertext, compute the message bit \( m = [c]_p \mod 2 \), where \([c]_p \) is an integer in \((-p/2, p/2)\).

To conveniently compare, we simply describe the known attacks considering in the Section 5 and appendix B in [DGHV10]. Section 5 in [DGHV10] considered known attacks on the AGCD problem for two numbers \((x_0, x_1)\) and many numbers \((x_0, \cdots, x_t)\). These attacks mainly discussed how to solve approximate GCD problem, i.e. the secret key \( p \).

The appendix B.1 in [DGHV10] analyzed Nguyen and Stern’s orthogonal lattice attack. Given
\(\tilde{x} = (x_0, \ldots, x_t) = pq + \tilde{r}\), where \(\tilde{q} = (q_0, \ldots, q_t)\) and \(\tilde{r} = (r_0, \ldots, r_t)\), consider the \(t\)-dimensional lattice \(L_q^t\) of integer vectors orthogonal to \(\tilde{x}\). It is easy to verify that any vector that is orthogonal to both \(\tilde{q}\) and \(\tilde{r}\), that is, is in the lattice \(L_{qr}^t\), it is also in \(L_q^t\).

According to [DGHV10], the idea of the attack is to reduce \(L_q^t\) to recover \(t-1\) linearly independent vectors of \(L_{qr}^t\), and further recover \(\tilde{q}\) and \(\tilde{r}\), and \(p\). Then Dijk et al. discussed that when \(t > \gamma / (\eta - \rho)\), lattice reduction algorithms can not find a \(2^{\gamma - \rho}\) approximate short vectors in \(L_{qr}^t\) on the worst-case.

Dijk et al. also analyzed a similar above attack by using the constraint \(x_i - r_i = 0 \mod p\), which also paid close attention to how to solve for the \(\tilde{r}\). They considered a lattice as follows.

\[
M = \begin{pmatrix}
    x_1 & R_1 \\
    x_2 & R_2 \\
    \vdots & \ddots \\
    x_t & R_t 
\end{pmatrix}.
\]

But one needs to find \(t\) linearly independent short vectors of the lattice \(M\) to obtain the success of this attack. That is, each \(l_1\) norm among \(t\) vectors is at most \(p/2\). When \(t\) is large, solving these vectors is very difficult by using lattice reduction algorithm.

In addition, instead of applying linear system \(x_i - r_i = 0 \mod p\), Coppersmith’s method looks at quadratic system \((x_i - r_i)^2 = 0 \mod p^2\) and \((x_i - r_i)(x_j - r_j) = 0 \mod p^2\), etc, and finds one of the \(r_i\) and thereof \(p\) and all other \(r_j\)'s by solving some small vectors in new lattice.

In a word, the attacks considering in the Section 5 and appendix B in [DGHV10] is how to recover the secret key \(p\), and their security analysis depends on the worst-case performance of the currently known lattice reduction algorithms.

The lattice we constructed is very similar to the lattice \(M\). However, our attack only requires find one short vectors with certain condition, and not to solve \(t\) short vectors. Moreover, our attack merely recovers the plaintext from a ciphertext and depends upon the average-case performance of the lattice reduction algorithms. On the other hand, if suppose \(\tilde{x} = (c, x_0, \ldots, x_t) = pq + 2\tilde{r} + m\) with a ciphertext \(c\), then our attack in some sense is similar to solving a short vector of orthogonal lattice \(L_q^t\), which is different from the lattices
Our Contribution. Our main observation is that one can directly obtain the plaintext from a ciphertext and the public key without using the secret key for some parameter settings of the FHE in [DGHV10]. The attack in this paper is different from the known attacks considering in [DGHV10]. Because our method is how to recover the plaintext from a ciphertext, whereas the attacks they considered is how to solve the secret key in the scheme. So, our result shows that the FHE in [DGHV10] is not secure for some practical parameters.

Organization of This Paper. Section 2 gives some notations and definitions, and the lattice reduction algorithms. Section 3 constructs a new lattice based on the public key, and presents a polynomial time algorithm to directly obtain plaintext from ciphertext. Section 4 presents an improvement scheme. Section 5 concludes this paper.

2. Preliminaries

2.1 Notations

In this paper, we follow the parameter setting of [DGHV10]. Let \( \lambda \) be a security parameter, \( \lambda = \{1, \ldots, \lambda\} \) be a set of integers. Let \( \gamma \) be bit-length of the integers in the public key, \( \eta \) the bit-length of the secret key, \( \rho \) the bit-length of the noise, and \( \tau \) the number of integers in the public key. To conveniently describe, we concretely set \( \rho = \lambda \), \( \eta = 4\lambda^2 \), \( \gamma = \lambda^5 \), and \( \tau = \gamma + \lambda \) throughout this paper.

Let \( w \sim S \) denote to choose an element \( w \) in \( S \) according to the distribution \( \Psi \).

2.2 Lattices

A lattice in \( \mathbb{R}^n \) is the set of all integral combination of \( n \) linearly independent vectors \( b_1, \ldots, b_n \) in \( \mathbb{R}^m \) (\( m \geq n \)), namely \( L = L(b_1, \ldots, b_n) = \{ \sum_{i=1}^{n} x_i b_i, x_i \in \mathbb{Z} \} \), usual denoted as a matrix \( B \). Any such \( n \)-tuple of vectors \( b_1, \ldots, b_n \) is called a basis of the lattice \( L \). Every lattice has an infinite number of lattice bases. Two lattice bases \( B_1, B_2 \in \mathbb{R}^{nxn} \) are equivalent if and only if \( B_1 = UB_2 \) for some unimodular matrix \( U \in \mathbb{Z}^{nxn} \). The volume of a lattice \( L \) is the determinant of any basis of \( L \), namely \( \text{vol}(L) = \det(L) = \sqrt{B^T B} \).
2.3 Lattice Reduction Algorithm

Given a basis of the lattice \( b_1, ..., b_n \), one of the most famous problems of the algorithm theory of lattices is to find a short nonzero vector. Currently, there is no polynomial time algorithm for solving a shortest nonzero vector in a given lattice. The most celebrated LLL reduction finds a vector whose approximating factor is at most \( 2^{(n-1)/2} \). In 1987, Schnorr [Sch87] introduced a hierarchy of reduction concepts that stretch from LLL reduction to Korkine-Zolotareff reduction which obtains a polynomial time algorithm with \( (4k^2)^{o(2/k)} \) approximating factor for lattices of any rank. The running time of Schnorr’s algorithm is \( \text{poly(size of basis)} \times \text{HKZ}(2k) \), where HKZ(2k) is the time complexity of computing a 2k-dimensional HKZ reduction, and equal to \( O(k^{k/2+o(k)}) \). If we use the probabilistic AKS algorithm [AKS01], HKZ(2k) is about \( O(2^{2k}) \).

**Theorem 2.1 (Sch87 Theorem 2.6)** Every block \( 2k \)-reduced basis \( b_1, ..., b_{m^k} \) of lattice \( L \) satisfies \( \|b_1\| \leq \sqrt{\gamma_k} \beta_k^{1/2} \lambda(L) \), where \( \beta_k \) is another lattice constant using in Schnorr’s analysis of his algorithm.

Schnorr [Sch87] showed that \( \beta_k \leq 4k^2 \), and Ajtai improved this bound to \( \beta_k \leq k^\epsilon \) for some positive number \( \epsilon > 0 \). Recently, Gama Howgrave, Koy and Nguyen [GHKN06] improved the approximation factor of the Schnorr’s \( 2k \)-reduction to \( \|b_1\| / \lambda(L) \leq \sqrt{\gamma_k} (4/3)^{(3k-1)/4} \beta_k^{o(2/k-1)} \), and proved the following result via Rankin’s constant.

**Theorem 2.2 (GHKN06 Theorem 2, 3)** For all \( k \geq 2 \), Schnorr’s constant \( \beta_k \) satisfies:

\[
\frac{k}{12} \leq \beta_k \leq (1 + k/2)^{2ln2+1/k}.
\]

Asymptotically it satisfies \( \beta_k \leq 0.1 \times k^{2ln2+1/k} \). In particular, \( \beta_k \leq k^{1.1} \) for all \( k \leq 100 \).

**Observation 2.3 (NS06).** For lattice \( L \), the first vector \( b_1 \) output by LLL is satisfied to the ratio \( \|b_1\| / \lambda(L) \approx (1.02)^n \) on the average.

3. Attack on FHE Scheme

To describe simplicity, we first refer the FHE scheme in [DGHV10], then construct a new lattice based on the public key and recover the plaintext bit from a ciphertext by applying LLL lattice reduction algorithm.
3.1 Fully Homomorphic Encryption

\textbf{KeyGen(\lambda\tau)}. The secret key is a random odd \(\eta\)-bit integer: \(p \leftarrow (2\mathbb{Z}+1)\cap[2^{\eta-1},2^\eta]\).

Select \(q_0,\ldots,q_r \leftarrow \mathbb{Z}\cap[0,2^\eta/p)\) with the largest odd integer \(q_0\). Select\(r_0,\ldots,r_r \leftarrow \mathbb{Z}\cap[-2^\rho,2^\rho]\), compute \(x_0 = q_0p + 2r_0\) and \(x_i = [q_ip + 2r_i]_{\eta_0}\) for \(i \in [r]\).

Output the public key \(pk = <x_0,x_1,\ldots,x_r>\) and the secret key \(sk = <p>\).

\textbf{Encrypt( pk,m \in \{0,1\})}. Select a random subset \(T \subseteq [r]\) and \(r \leftarrow \mathbb{Z}\cap[-2^\rho,2^\rho]\), and output ciphertext \(c = \left[m + 2r + \sum_{i \in T} x_i\right]_{\eta_0}\).

\textbf{Decrypt( sk,c ).} Output \(m' = \left[\left[c\right]_p\right]_2\).

To implement fully homomorphic encryption scheme, one applies to it the standard Gentry’s bootappable technique.

3.2 Lattice Attack Based on the Public Key

Given a list of approximate multiples of \(p\):

\[
\{x_i = q_ip + r_i : q_i \in \mathbb{Z}\cap[0,2^\eta/p), r_i \in \mathbb{Z}\cap(-2^\rho,2^\rho)\}_{i=0}^r, \text{ find } p.
\]

Dijk et al. [DGHV10] showed that the security of their FHE scheme is equivalent to solving the approximate GCD problem. Chen and Nguyen [CN11] presented a new AGCD algorithm running in \(2^{3\rho/2}\) polynomial-time operations, which is essentially the \(3/4\)-th root of that of GCD exhaustive search.

According to FHE, we know that an arbitrary ciphertext has general form \(c = qp + 2r + m\).

The ideal of our attack is very simple, that is, one is how to remove \(qp\) in a ciphertext \(c\) by adding small noise value. When completing this, it is easy to recover the plaintext bit \(m\) in \(c\). To do this, we, we define following Diophantine inequality equation problem.

\textbf{Definition 3.1. (Diophantine Inequality Equation (DIE)).} Given a list of integers \(\{x_i = q_ip + r_i : q_i \in \mathbb{Z}\cap[0,2^\eta/p), r_i \in \mathbb{Z}\cap(-2^\rho,2^\rho)\}_{i=0}^r\), solve the Diophantine inequality equation \(\sum_{i=0}^r y_ix_i < p/8\) subject to \(|y_i| < p/(8\pi 2^\rho)\) and at least one non-zero \(y_i\).

Suppose there is an oracle to solve the above DIE problem, then one can obtain the plaintext bit in an arbitrary ciphertext of FHE [DGHV10]. Since \(|y_i| < p/(8\pi 2^\rho)\), \(\sum_{i=0}^r y_ir_i < p/8\), that is, \(\sum_{i=0}^r y_ix_i\) is only the sum of noise terms, without non-zero multiple of \(p\). So, one
can correctly decide the plaintext bit of a ciphertext in FHE according to the parity of \(\sum_{i=0}^{r} y_i x_i\).

However, it is not difficult to see that the Diophantine inequality equation is a generalization of the knapsack problem. So, there is unlikely an efficient algorithm for general DIE unless P=NP. But, this does not demonstrate that there is not a polynomial time algorithm for special DIE.

To be concrete, we construct a new lattice based on the public key of the FHE [DGHV10]. Given the public key \(pk = \langle x_0, x_1, \ldots, x_r \rangle\) and ciphertext \(c\), we randomly choose a subset \(T\) from \([r]\) such that \(|T| = \lambda^3\). Without generality of loss, assume \(T = [\lambda^3]\) and 
\[
c = qp + 2r + m \quad \text{with} \quad |2r| \leq 2^\rho.
\]
We construct a new lattice as follows:
\[
L = \begin{pmatrix}
c & 0 & \cdots & 0 \\
-x_0 & 1 & \cdots & 0 \\
-x_1 & 0 & \cdots & 0 \\
& \vdots & \vdots & \vdots \\
-x_{\lambda^3} & 0 & \cdots & 1 \\
-x_0 & 0 & \cdots & 1
\end{pmatrix}, \quad L_1 = \begin{pmatrix}
c & 1 & 0 & \cdots & 0 & 0 \\
-x_0 & 0 & 1 & \cdots & 0 & 0 \\
& \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-x_{\lambda^3} & 0 & 0 & \cdots & 1 & 0 \\
-x_0 & 0 & 0 & \cdots & 1 & 0
\end{pmatrix}.
\]

On the one hand, the size of the shortest vector of lattice \(L\) is less than 
\[
\sqrt{2^{3(\lambda^3 + 2)}} \approx 2^{\lambda^2} \quad \text{according to the parameter setting.}
\]
On the other hand, there is a non-zero solution 
\[
\sum_{i=0}^{\lambda^3} y_i x_i + yc \leq 2^{\lambda^2} \quad \text{with} \quad |y_i| \leq 2^{\lambda^2} \quad \text{and} \quad |y| \leq 2^{\lambda^2}
\]
by using pigeon hole principle. This is because 
\[
|\tilde{y}\|, |\tilde{x}| \leq 2^{\lambda^2}, \quad \text{the number of all distinct} \quad y_i, y \quad \text{subject to}
\]
\[
|y_i|, |y| \leq 2^{\lambda^2} \quad \text{is} \quad (2^{\lambda^2})^{\lambda^3 + 2} > 2^{\lambda^2},
\]
that is, there is at least a non-zero solution for the equation 
\[
\sum_{i=0}^{\lambda^3} y_i x_i + yc \leq 2^{\lambda^2}.
\]
Thus, if one finds a non-zero small solution vector, then one gets the plaintext bit with probability at least \(1/2\) (\(y\) is an odd integer).

To conveniently decide, we use a variant lattice \(L_1\) of \(L\), and call LLL algorithm for lattice \(L_1\). Assume \(b = (b_0, b_1, \ldots, b_{\lambda^3+1})\) is the first vector of the \(L_1\)'s basis output by LLL. If 
\[
\|b\|_\infty < p / (8\lambda^3 2^{\lambda}) \quad \text{and} \quad \mod(b_1, 2) = 1, \quad \text{then} \quad m = \mod(b_0, 2).
\]
In our experiment, we notice that \(b_1\) may be an even integer, but the several vectors following the first vector (such as the second vector, or the third vector, et al.) often satisfy the above condition. That is, the first coordinate of vector is odd and its norm is small. So, as long as one gets one solution of the above form, one can correctly decide plaintext bit. In fact, LLL can also be called many times for distinct subset \(T\).
So, we have the following result by applying the block lattice reduction.

**Theorem 3.1.** Suppose the parameters of FHE [DGHV10] $\lambda \leq 100$, $\rho = \lambda$, $\eta = 5\lambda^2$, $\gamma = \lambda^5$, and $\tau = \gamma + \lambda$, then there is a running time $2^{\Omega\lambda}, (\theta \leq 1)$ algorithm recovering plaintext from ciphertext.

**Proof:** According to Theorem 2.1, 2.2, we know $\|h\| \leq \frac{\lambda(L)}{\sqrt{\gamma} (4/3)^{(3k-1)/4}} \beta_k^{n/2k-1}$ and $\beta_k \leq k^{1/2}$ for all $k \leq 100$. If we choose $k = \lambda, n = \lambda^3$, then we get $\|h\| \leq \lambda^{1.15^{1/2} \lambda^5} \times \lambda(L) \approx 2^{3.06\lambda^2} \lambda(L) \leq 2^{4.66\lambda^2} \ll < 2^{\eta}$. By using AKS [AKS01, MV10] algorithm, solving each block sub-lattice costs time $2^{\Omega\lambda}, \delta < 1$, and the times solving block is at most $\lambda^{O(1)}$. So, the total running time of algorithm is $2^{\Omega\lambda}, \theta \leq 1$.

**Theorem 3.2** Suppose the average-case performance of LLL is true, that is, Observation 2.3 holds. Then, for the parameters $\lambda \leq 100$, $\rho = \lambda$, $\eta = 4\lambda^2$, $\gamma = \lambda^5$, and $\tau = \gamma + \lambda$, the FHE scheme in [DGHV10] is insecure.

**Proof:** For the above lattice $L_4$, we have $\|h\| \leq (1.02)^{1.2} \lambda(L) \leq (1.02)^{100.12} \lambda(L) \approx 7.2^{3.2} \lambda(L) << 2^{4.1^2}$. ■

### 3.3 Computational Experiment

In the appendix, we present a toy example to show that our attack method is how to work.

### 4. Improvement

The reason the above lattice attack is successful is that the secret key $p$ is a large integer. If we replace $p$ by a matrix, then the above attack does not work.

#### 4.1 Construction

**Key Generating Algorithm (KeyGen):**

1. Select a random matrix $T \in \mathbb{Z}^{2 \times 2}$ with $\|T\|_{\infty} = 2^{O(\lambda^2)}$ such that $p = \det(T) = 2^{O(\lambda^2)}$ and $p \mod 2 = 1$. Compute $A \in \mathbb{Z}^{2 \times 2}$ with $AT = pI$, where $I$ is identity matrix.

2. Generate $\tau = O(\lambda \log \lambda)$ matrices ${B_i = (R_i A + 2R_i \cdot I) \mod p}^\tau_{i=1}$, where $R_i \in \mathbb{Z}^{2 \times 2}_p$.
is an uniformly random matrix and \(|r_i| \leq 2^{\lambda}\) and \(r_i\) is integer.

(3) Output the public key \(pk = (p, B_i, i \in [\tau])\) and the secret key \(sk = (p, T)\).

**Encryption Algorithm (Enc).** Given the public key \(pk\) and a bit \(m \in \{0,1\}\). Evaluate ciphertext \(C = (\sum_{i \in [\tau]} k_i B_i + (m + 2r)I) \mod p\) where \(|k_i| \leq 2^{\lambda}\) and \(r\) is integer.

**Add Operation (Add).** Given the public key \(pk\) and ciphertexts \(C_1, C_2\), output new ciphertext \(C = (C_1 + C_2) \mod p\).

**Multiplication Operation (Mul).** Given the public key \(pk\) and ciphertexts \(C_1, C_2\), output new ciphertext \(C = (C_1 \times C_2) \mod p\).

**Decryption Algorithm (Dec).** Given the secret key \(sk\) and ciphertext \(C\), decipher \(M = (C \times T) \mod p \mod 2\), and the plaintext \(m\) is the element \(m = M_{1,1}\) of the first row and the first column of \(M\).

It is not difficult to verify that the above scheme is a somewhat homomorphic encryption. Now, one can use Gentry’s standard bootstrappable technique to implement fully homomorphic encryption.

In addition, we can choose two random primes \(p, q = 2^{O(\lambda^2)}\) with \(p = a^2 + b^2\) i.e. \(p \equiv 1 \mod 4\). Set \(n = pq\) and \(T = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}\), \(A = \begin{pmatrix} a & -b \\ -b & a \end{pmatrix}\) with \(AT = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix} = pI\).

Now, we can replace \(p\) with \(n = pq\) in the above scheme, and use the new matrix \(A\) to generate the public key \(pk = (n, B_i, i \in [\tau])\). We observe that the security of this modification depends on the hardness of factoring \(n = pq\).

**4.2 Efficiency and Security.**

**Efficiency:** The size of the public key is \(O(\lambda^3 \log \lambda)\), the size of the secret key is \(O(\lambda^2)\), the expansion rate of ciphertext to plaintext is \(O(\lambda^2)\). To implement FHE, one only needs to add ciphertexts of the secret key to the public key.

**Security:** It is not feasible to use brute force attack by guessing noise term \(r\) because \(|r| = O(2^\lambda)\). A possible attack is to solve the following equation...
\[
\begin{align*}
TB_1 &= r_1 T \mod p \\
TB_2 &= r_2 T \mod p \\
&\vdots \\
TB_r &= r_r T \mod p
\end{align*}
\]

However, this system consists of quadratic equations when \( r_i \) is unknown. So, to solve this equation, we also require to guess \( r_i \). As well as we know, attacking the above scheme is not feasible by using algebraic equation method.

At the same time, the above scheme can avoid the lattice attack of this paper because the matrix \( B_i \) is approximate multiple of the corresponding secret key \( A \).

The above improvement scheme has more efficient, but we currently can not reduce its security to solving the secret key.

5. Conclusion

This paper presents a heuristic attack for the FHE in [DGHV10] by directly calling LLL algorithm. Our method concentrates on recovering the plaintext in a ciphertext, whereas the attacks considering in [DGHV10] mainly discussed how to avoid to recovering the secret key. Moreover, our attack applies the average-case performance of lattice reduction algorithm, whereas the security of their scheme depends upon the worst-case performance of lattice reduction algorithm.

Our result shows that the FHE scheme in [DGHV10] is not secure for some parameter settings. According to our experiment, one can avoid the above lattice attack by setting parameter \( \gamma = \lambda^6 \). But, the scheme is less practical in this case.

In addition, we also design an improvement scheme to avoid the above lattice attack.

Reference

[AKS01] Miklos Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In STOC, pages 601–610. ACM, 2001.
[Coh93] H. Cohen, A Course in Computational Algebraic Number Theory, Graduate Texts in Mathematics, 138. Springer, 1993.
[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011, http://eprint.iacr.org/2011/277.pdf.
[CJMNT11] J.-S. Coron, A. Joux, A. Mandal, D. Naccache, and M. Tibouchi. Cryptanalysis of the rsa subgroup assumption from TCC 2005. In Public Key Cryptography - Proc. PKC 2011, volume 6571 of Lecture Notes in Computer Science, pages 147–155. Springer, 2011.
[CMNT11] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic
encryption over the integers with shorter public-keys. In Advances in Cryptology - Proc.
CRYPTO 2011, volume 6841 of Lecture Notes in Computer Science. Springer, 2011.

[CN11] Y. Chen and P. Q. Nguyen. Faster algorithms for approximate common divisors:
Breaking fully homomorphic encryption challenges over the integers, Cryptology ePrint
Archive, Report 2011/436, 2011, http://eprint.iacr.org/2011/436.

[DGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic
encryption over the integers. In Proc. of Eurocrypt, volume 6110 of LNCS, pages 24-43.
Springer, 2010.

[Gen01] C. Gentry. Key Recovery and Message Attacks on NTRU-Composite. Eurocrypt’01,
LNCS 2045, pages 182-194.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In Proc. of STOC,
pages 169-178, 2009.

[GH11a] Craig Gentry and Shai Halevi. Implementing Gentry’s fully-homomorphic
encryption scheme. In Kenneth Paterson, editor, Advances in Cryptology — EUROCRYPT
2011, volume 6632 of Lecture Notes in Computer Science, pages 129–148, Berlin,
Heidelberg, New York, 2011. Springer Verlag. Cryptology ePrint Archive: Report 2010/520:
http://eprint.iacr.org/2010/520.

[GH11b] C. Gentry and S. Halevi, Fully homomorphic encryption without squashing using
depth-3 arithmetic circuits, Cryptology ePrint Archive, Report 2011/279.

[GHKN06] N. Gama, N. Howgrave-Graham, H. Koy, and P. Q. Nguyen. Rankin’s constant
and blockwise lattice reduction. In CRYPTO, pages 112–130, 2006.

[GN08a] N. Gama and P. Nguyen, Finding Short Lattice Vectors within Mordell’s Inequality,
In Proc. of the ACM Symposium on Theory of Computing STOC’08, pp. 208–216, 2008.

[GN08b] N. Gama and P.Q. Nguyen, Predicting lattice reduction, in Proc. EUROCRYPT
2008, LNCS 4965, Springer-Verlag, pp. 31–51, 2008.

[GS02] C. Gentry, M. Szydlo. Cryptanalysis of the Revised NTRU Signature Scheme.
Eurocrypt’02, LNCS 2332, pages 299-320.

[HPS98] J. Hoffstein, J. Pipher, J. H. Silverman. NTRU: A Ring-Based Public Key
Cryptosystem. LNCS 1423, pages 267-288, 1998.

[LLL82] H.W. Lenstra Jr., A.K. Lenstra and L. Lov’asz, Factoring polynomials with rational
coefficients, Mathematische Annalen 261, pp. 515–534, 1982.

[Mic07] D. Micciancio, Generalized compact knapsaks, cyclic lattices, and efficient one-way
functions. Computational Complexity, 16(4):365-411.

[MV10] D. Micciancio P. Voulgari, Faster exponential time algorithms for the shortest vector
problem, SODA 2010, Electronic Colloquium on Computational Complexity, Report No. 65
(2009).

[NS00] P. Nguyen and J. Stern, Lattice Reduction in Cryptology: An Update, in Proc. of
Algorithm Number Theory (ANTS IV), LNCS 1838, pages 85-112. Springer-Verlag, 2000.

[NS06] P.Q. Nguyen and D. Stehle, LLL on the average, proc. Of ANTS VII, 2006, LNCS
4076, pp. 238-256.

[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy
homomorphisms. In Foundations of Secure Computation, pages 169-180, 1978.

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms.
Theoretical Computer Science, 53, pp. 201–224, 1987.
Appendix

Here we present a toy example to show the attack processing in this paper.

Assume $\lambda = 3$, $\rho = 3$, $\eta = 3 \times 3^2$, $\gamma = 3^3$, and $\tau = 246$. The secret key is

\[ p = 134217729. \]

The public key $pk$ is

\[ p = \text{[long hexadecimal number]} \]

Appendix

Here we present a toy example to show the attack processing in this paper.

Assume $\lambda = 3$, $\rho = 3$, $\eta = 3 \times 3^2$, $\gamma = 3^3$, and $\tau = 246$. The secret key is

\[ p = 134217729. \]

The public key $pk$ is

\[ p = \text{[long hexadecimal number]} \]
The lattice $L_1$ is as follows.

$C = \begin{bmatrix}
-1968487892819738359724658441513555737250551194506972917051476635657242373 \\
-3250624492796816435996931888772512141226431955536420737938610559094381472 \\
6166801825361929544406619112907511579350724887303183802895393056412266189 \\
-389271416239973016638701950598010040551090821895222204461457546946680171 \\
-72971459577060219139405473893395667801987076741569193382391788378629 \\
301404573446266049075794956008230722522321993597035398126086339308590421 \\
1980397573955218078865266608072264557468205157356780229452025195354817820 \\
61712523654282593146930940068120821614184871336369883430435009695024807072 \\
31434472445604705240734304340884466636969263877303414205945241278676736 \\
873650667075946688621744141808551517576371375520393505618087627422587406 \\
56611687170316270078505583620838090355878886794176445353579682688910349 \\
-3033204216689057328397894213621146520765827553518578073761359734805150017 \\
305963962927470110087578377425382281832377424563292183331667391617245338 \\
-3775189926926650332603202627969313026841111458725949522776856445989730200 \\
6025056718432827055604293681097250135829510209439555808422352974191399388 \\
-6763448644568304758244751149882262168427403425012104198632216541146186520 \\
-432916318934822418964493132156872690912140128638518271106678567196791969 \\
00000000000000100000000000
\end{bmatrix}$
By calling LLL algorithm, the reduced basis of $L_1$ is

$$B = \begin{bmatrix} -86 & 122 & -65 & -175 & -90 & -182 & 113 & 79 & 41 & 46 & -225 & 99 & -72 & 164 & -66 & -376 & 5 & 55 & 167 & -159 & 94 & 96 & 33 & -63 & -1 & -42 & -39 & -92 & 0 \\ -87 & -49 & 65 & -321 & -209 & 49 & 11 & -30 & 29 & 48 & -149 & 181 & 12 & 109 & -153 & -237 & -43 & -83 & 10 & 79 & 177 & -120 & -127 & 171 & 170 & -89 & -52 & 4 \\ 175 & 75 & -43 & -80 & 36 & -86 & 14 & -147 & -111 & -180 & -60 & -5 & -181 & 308 & -98 & -114 & 115 & -96 & 150 & -151 & 184 & 293 & 48 & -39 & 2 & 85 & 52 & -4 \\ -153 & -21 & -61 & 172 & 138 & 198 & -31 & -188 & -3 & 107 & 61 & 47 & 260 & 42 & 30 & -55 & -82 & 64 & -91 & -52 & -31 & 179 & -59 & -104 & -113 & 72 & -25 & -6 & 9 \\ 77 & 149 & -12 & 60 & 242 & 89 & 212 & 23 & 90 & 126 & 73 & -40 & 56 & -135 & 91 & -49 & -68 & -8 & 116 & 103 & 100 & 91 & 100 & 80 & 55 & -114 & 57 & -45 & -5 \\ -169 & 55 & -115 & 362 & 140 & 102 & -157 & 23 & -69 & 84 & -9 & 4 & 145 & 4 & 5 & 97 & 110 & -113 & -22 & 76 & -59 & -83 & 34 & -88 & -71 & 107 & 9 & 39 & 14 \\ -23 & -143 & -137 & 54 & -184 & 7 & -209 & 32 & -67 & 234 & -9 & 179 & 345 & 6 & -7 & -109 & -143 & 40 & -2 & 89 & -164 & -110 & -109 & -11 & -80 & 128 & -48 & 79 & 18 \\ -62 & -66 & -64 & -232 & 64 & 131 & 1 & -175 & -42 & -107 & -145 & 170 & 263 & 234 & -154 & -95 & 119 & 124 & -128 & -281 & 211 & 111 & 55 & -82 & -7 & 91 & -68 & -87 & -38 \\ -155 & 167 & -110 & 86 & -19 & -102 & 96 & 108 & 120 & 178 & -113 & 33 & -161 & -32 & -9 & -187 & -33 & -62 & 145 & 66 & 87 & -149 & -39 & -96 & 176 & 62 & -115 & -206 & 10 \\ -31 & -5 & 56 & 2 & 97 & 146 & -42 & -213 & -88 & -2 & -173 & -99 & 74 & 214 & -64 & -53 & -50 & -156 & -16 & -51 & 21 & 96 & -244 & 150 \\ 
\end{bmatrix}$$
When calling LLL algorithm, generating matrix $U$ is as follows.

$$U = \begin{bmatrix}
\begin{array}{cccccccccc}
122 & -65 & -159 & 113 & 100 & 360 & 214 & 2 & 63 & 90 & 23 & 68 & 87 & 53 & 157 & 148 & 181 & 31 & 100 & 28 & 87 & 130 & -87 & -111 & -22 & 46 & 7 & 146 & -32 & -99 \\
-201 & -65 & -109 & -13 & -128 & -179 & -83 & 50 & -60 & 56 & 109 & 105 & -12 & 51 & 35 & -111 & -18 & 242 & 19 & -119 & -109 & 230 & 2 & 3 & 1 & -33 & -85 & -11 & -12 \\
32662 & 1532013 & 35166 & -334620 & -492845 & 319870 & -62472 & -112310 & -73327 & -101190 & -187515 & 444100 & 363631 & 224003 & 356632 & 512681 & 263715 & 351591 & -34152 & 266919 & -280216 & 127712 & 299356 & 168344 & 363922 & -258533 & 45283 & 138299 & -195047 \\
\end{array}
\end{bmatrix}$$

When calling LLL algorithm, generating matrix $U$ is as follows.

$$U = \begin{bmatrix}
\begin{array}{cccccccccc}
122 & -65 & -159 & 113 & 100 & 360 & 214 & 2 & 63 & 90 & 23 & 68 & 87 & 53 & 157 & 148 & 181 & 31 & 100 & 28 & 87 & 130 & -87 & -111 & -22 & 46 & 7 & 146 & -32 & -99 \\
-201 & -65 & -109 & -13 & -128 & -179 & -83 & 50 & -60 & 56 & 109 & 105 & -12 & 51 & 35 & -111 & -18 & 242 & 19 & -119 & -109 & 230 & 2 & 3 & 1 & -33 & -85 & -11 & -12 \\
32662 & 1532013 & 35166 & -334620 & -492845 & 319870 & -62472 & -112310 & -73327 & -101190 & -187515 & 444100 & 363631 & 224003 & 356632 & 512681 & 263715 & 351591 & -34152 & 266919 & -280216 & 127712 & 299356 & 168344 & 363922 & -258533 & 45283 & 138299 & -195047 \\
\end{array}
\end{bmatrix}$$
The above three matrices is satisfied to equality $U^*C = B$. Moreover, $U$ is equal to $B$ except for the first column.

Now, we can decide the plaintext bit in the ciphertext

$-1968487892819738597274658441513155537250551194506972917051476635676242373$

according to the parity of the first column of $U$ and $B$.

It is easy to check that they are respectively

$$[0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1],$$

$$[0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0].$$

So, the plaintext is “1” for the above ciphertext. This is because the first columns in $U$ and $B$ have same parity if the plaintext is “1” in a ciphertext and $\|U\|_\infty, \|B\|_\infty < 2^{\lambda}$.

Notice that the last row vector in $U$ is too large (that is $|y|, |y' > 2^{\lambda}$), so the last terms in the parity vectors is not satisfied the above condition.

On the other hand, suppose the ciphertext is

$-1968487892819738597274658441513155537250551194506972917051476635676242374$

then calling LLL generates the matrices $B, U$ as follows.

$B=\begin{bmatrix} -110 & 112 & -87 & -84 & 7 & 1 & 161 & -66 & 239 & 63 & -181 & -146 & -205 & 80 & -74 & -63 & 37 & -41 & -18 & 34 & 85 & 75 & 7 & 106 & 122 & 158 & -27 & 45 & 1 \\ -2 & 5 & 73 & 41 & 131 & -131 & -125 & 153 & -181 & 217 & 62 & 166 & 201 & -63 & -140 & 5 & 42 & 36 & 60 & 8 & -148 & -1 & 96 & 122 & -24 & -46 & 149 & 170 & 1 \\ 102 & -33 & -62 & 9 & 73 & -207 & 127 & -42 & -273 & 170 & 1 & 130 & 185 & 164 & -30 & -172 & -66 & 22 & 20 & -128 & -109 & -132 & -110 & -184 & 59 & -71 & 84 & 122 & 1 \\ 68 & -152 & 186 & 187 & -215 & -37 & 129 & 59 & 14 & -153 & 180 & 40 & -52 & 203 & 6 & 88 & 139 & 96 & -195 & 70 & -129 & -308 & -57 & -56 & 139 & 78 & -65 & -48 & 0 \\ 48 & 62 & -104 & -173 & 250 & -14 & -52 & -73 & -173 & -23 & 173 & -12 & 145 & 44 & -217 & -93 & -62 & 152 & -74 & 44 & 210 & 26 & -25 & -155 & -149 & -166 & 172 & 171 & 1 \\ 88 & -57 & 169 & 30 & -189 & 7 & 168 & 125 & 26 & 188 & -254 & -7 & -79 & 60 & -104 & -38 & 133 & 121 & -103 & -52 & -127 & 29 & -138 & 318 & 52 & 188 & -111 & 58 & -1 \\ -84 & 76 & 222 & 155 & -108 & -26 & -197 & 25 & -224 & 297 & 19 & -53 & 77 & -58 & 5 & 66 & -51 & -106 & 88 & -73 & -166 & -13 & 37 & 22 & -175 & 26 & -41 & -158 & -4 \\ 76 & 124 & -50 & 4 & 26 & 50 & 49 & 11 & -199 & 159 & -151 & -101 & -27 & 6 & -104 & -149 & -14 & -201 & 66 & -222 & -130 & 73 & -150 & -68 & 33 & -27 & 4 & -273 & 0 \\ -94 & -37 & -17 & -71 & -51 & -45 & -65 & -68 & -89 & -85 & 68 & 209 & 52 & -21 & 85 & -166 & -81 & 111 & -100 & -162 & 43 & -4 & -175 & 83 & -53 & 150 & -106 & 143 & -1 \\ 52 & -22 & 64 & -80 & -114 & -107 & -63 & 231 & 71 & -89 & -26 & 108 & -215 & 163 & -112 & -141 & 7 & 10 & -78 & 36 & -188 & -41 & -64 \end{bmatrix}$
Similarly, the above three matrices is satisfied to equality $U \cdot C = B$.

It is easy to check that the parity of the first columns of $B$ and $U$ are respectively

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Thus, the plaintext bit is “0” in the ciphertext. Because the parity of the first column of $B$ is “0” except its last row and is different from the parity of the first column of $U$.

Similarly, the last row vector in $U$ is too large (that is $|y|_1, |y|_2 > 2^k$), so the last terms in the parity vectors is not satisfied the above condition.