A New Lattice-Based Cryptosystem Mixed with a Knapsack

Yanbin Pan and Yingpu Deng and Yupeng Jiang and Ziran Tu
Key Laboratory of Mathematics Mechanization
Academy of Mathematics and Systems Science,Chinese Academy of Sciences
Beijing 100190, China
{panyanbin,dengyp}@amss.ac.cn
jiangyupeng08@mails.gucas.ac.cn, naturetu@gmail.com

2009.6.30

Abstract

In this paper, we present a new lattice-based public-key cryptosystem mixed with a knapsack, which has reasonable key size and quick encryption and decryption. The module strategy in our cryptosystem can also be used to construct a framework for some GGH-type cryptosystems to improve their security.

Keywords: Lattice, Public-key Cryptosystem, Knapsack

1 Introduction

Since the seminal work of Ajtai [1] connecting the average-case complexity of lattice problems to their complexity in the worst case, cryptographic constructions based on lattices have drawn considerable attention. Ajtai and Dwork [3] proposed the first lattice-based public-key cryptosystem whose security is based on the worst-case hardness assumptions. After their results, several lattice-based cryptosystems [7, 10, 5, 17, 18, 2, 8, 16] have been proposed.

Lattice-based cryptosystems have many advantages: first, the computations involved are very simple and usually require only modular addition; second, by now they resist the cryptanalysis by quantum algorithms while there already exist the efficient quantum algorithms[19] for factoring integers and computing discrete logarithms.

However, most of the presented lattice-based cryptosystems which are efficient have no security proofs based on the worst-case hardness while most of those which

have security proofs are not efficient. Recently, some efficient lattice-based cryptosystems [8, 16] with security proofs have been presented.

In Crypto'97, Goldreich, Goldwasser and Halevi [7] proposed a public key cryptosystem based on the closest vector problem, which is NP-hard. Although The cryptosystem GGH has not a security proof, it has efficient encryption and decryption. Moreover, it has a natural signature scheme. However, Nguyen [14] showed there is a major flaw in it, and it can't provide sufficient security without being impractical.

The NTRU cryptosystem proposed by Hoffstein, Pipher, Silverman[10] is the most practical scheme known to date. It features reasonably short, easily created keys, high speed, and low memory requirements. By the results of Coppersmith and Shamir [6], the security of NTRU can be based on, but not equivalent to, the hardness of some lattice problems. To date, the chosen-ciphertext attacks against NTRU may be the most dangerous and most of the ciphertext-only attacks [6, 12, 9] against NTRU relies on the special cyclical structure.

Although the Ajtai-Dwork cryptosystem was thought to be secure if a particular lattice problem is difficult in the worst-case, Nguyen and Stern [13] gave a heuristic attack to show that in order to be secure, the implementations of the Ajtai-Dwork cryptosystem would require very large keys, making it impractical in a real-life environment. In 1998, Cai and Cusick [5] proposed another efficient lattice-based public-key cryptosystem with much less data expansion by mixing the Ajtai-Dwork cryptosystem with a knapsack. However, an efficient ciphertext-only attack presented by Pan and Deng [15] shows that it's not secure.

In this paper, we also propose a new lattice-based public key cryptosystem mixed with a knapsack.

Like GGH, the system use a matrix $H \in \mathbb{Z}^{m\times m}$ as a public key to encrypt a message $t \in \{0,1\}^m$, but to recover the message using direct lattice reduction, we need solve a CVP for a $2m$-dimensional lattice instead of m-dimensional in GGH. This may allow us to use small dimensional matrix as public key to provide sufficient security. Moreover, all the entries of H is bounded by a small positive integer p, so the public key size is usually smaller than in GGH.

The size of key is bigger than in NTRU which is most practical. However, there is not an obvious lattice attack to obtain the private key in our cryptosystem while the private key of NTRU can be obtained by finding the short vector of NTRU-lattice. Moreover, the lattice we use looks more random, no special cyclical structure like NTRU, this makes our scheme resist some similar attacks which are based on the special cyclical structure against NTRU.

We call a cryptosystem has GGH-type if it recovers the message t from such a vector $Bt + r$, where $B \in \mathbb{Z}^{m\times m}$, and $t \in \mathbb{Z}^m$, $r \in \mathbb{Z}^m$. The module strategy in our cryptosystem can also be used to construct a framework for such GGH-type

cryptosystems in which the entries of B are small and the entries of t and r have the same distribution. The module strategy can help the cryptosystems hide the private key and improve the security against lattice attack if the module is small enough.

The remainder of the paper is organized as follows. In Section 2, we give some preliminaries needed. In Section 3, we describe our lattice-based public key cryptosystem. Section 4 presents some details in the practical implementation and in Section 5, we give the security analysis and some experimental evidence. Finally, we give a short conclusion in Section 6.

2 Preliminaries

KNAPSACK PROBLEM. Given positive integers $N_1, N_2, \cdots, N_n$ and s, the knapsack or subset sum problem is to find variables $a_1, a_2, \cdots, a_n$, with $a_i \in \{0, 1\}$, such that

$$\sum_{i=1}^{n} a_i N_i = s.$$

The problem is known to be NP-complete. However, if $N_1, N_2, \cdots, N_n$ construct a superincreasing sequence, i.e. $N_i > \sum_{j=1}^{i-1} N_j$ for $i = 2, 3, \ldots, n$, we can find $a_1, a_2, \cdots, a_n$ efficiently by the following greedy algorithm. If $s \geq N_n$, then $a_n = 1$, otherwise, $a_n = 0$. We then substitute s by $s - a_n N_n$ and find a_{n-1} similarly. It is easy to see that the process can be continued until all a_i's are found.

LATTICE. An integer lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{Z}^n$ and can be also defined as below:

Let $b_1, b_2 \cdots, b_d \in \mathbb{Z}^n$ be linearly independent vectors, the lattice $\mathcal{L}$ spanned by them is

$$\mathcal{L}(b_1, b_2, \cdots, b_d) = \{\sum_{i=1}^{d} a_i b_i | a_i \in \mathbb{Z}\}.$$

$B = (b_1, b_2 \cdots, b_d)$ is called the basis of $\mathcal{L}$. A lattice is full rank if $d = n$. If $\mathcal{L}$ is full rank, the determinant $det(\mathcal{L})$ is equal to the absolute value of determinant of the basis B. If A is a matrix with d linearly independent columns, we denote A_i the $i - th$ column of A and $\mathcal{L}(A)$ the lattice spanned by $A_1, A_2, \cdots, A_d$.

Denote $\|v\|$ the Euclidean l_2-norm of a vector v and $\lambda_1(\mathcal{L})$ the length of the shortest non-zero vector in the lattice $\mathcal{L}$. The shortest vector problem (SVP) refers the question to find such shortest non-zero vector while the closest vector problem (CVP) is to find a lattice vector minimizing the distance to a given vector. The

celebrated LLL algorithm [11] runs in polynomial time and approximates the shortest vector within a factor of $2^{n/2}$. Babai [4] also gave an polynomial-time algorithm that approximates the closest vector by a factor of $(3/\sqrt{2})^n$, which is called Babai's Nearest Plane Algorithm.

By the Gaussian Heuristic, $\lambda_1(\mathcal{L}) \approx \sqrt{\frac{n}{2\pi e}} det(\mathcal{L})^{\frac{1}{n}}$ for an $n-dimensional$ random lattice $\mathcal{L}$. Similarly, most closest vector problems for $\mathcal{L}$ have a solution whose size is approximately $\sqrt{\frac{n}{2\pi e}} det(\mathcal{L})^{\frac{1}{n}}$. If we want to find a short vector v in $\mathcal{L}$, or a vector v such that $t-v$ is the vector in $\mathcal{L}$ close to the target vector t, then experiences tell us the smaller $\frac{\|v\|}{\sqrt{\frac{n}{2\pi e}} det(\mathcal{L})^{\frac{1}{n}}}$, the more easily we can find v in practice.

3 Description of Our Cryptosystem

Parameter: m
Key Generation:
Let $n = 2m$.

Step 1 Choose a superincreasing sequence $N_1, N_2, \cdots, N_n$ where $N_1 = 1$.

Step 2 Randomly choose a permutation τ on n letters with $\tau^{-1}(1) \leq m$.

Step 3 For $i = m+1, m+2, \cdots, n$, write $N_{\tau(i)}$ as

$$N_{\tau(i)} = \sum_{j=1}^{m} b_{i-m,j} N_{\tau(j)}$$

where $b_{i-m,j} \in \mathbb{Z}$ and we expect them to be as small as possible.
Define $A \in \mathbb{Z}^{m \times n}$ as belows:

$$A = \begin{pmatrix} 1 & 0 & \cdots & 0 & b_{1,1} & b_{2,1} & \cdots & b_{m,1} \\ 0 & 1 & \cdots & 0 & b_{1,2} & b_{2,2} & \cdots & b_{m,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{1,m} & b_{2,m} & \cdots & b_{m,m} \end{pmatrix}.$$

So, $(N_{\tau(1)}, N_{\tau(2)}, \cdots, N_{\tau(m)})A = (N_{\tau(1)}, \cdots, N_{\tau(m)}, N_{\tau(m+1)}, \cdots, N_{\tau(n)})$.

Step 4 Let p be the smallest prime greater than $2q+1$, where

$$q = \max_{i=1,\cdots,m} \max\{ \sum_{\substack{j=1,\cdots,n \\ A_{i,j}>0}} |A_{i,j}|, \sum_{\substack{j=1,\cdots,n \\ A_{i,j}<0}} |A_{i,j}| \}.$$

Step 5 Randomly choose a permutation σ on n letters, such that the matrix $S = [A_{\sigma(1)}, A_{\sigma(2)}, \cdots, A_{\sigma(m)}]$ is invertible in $\mathbb{Z}_p^{m\times m}$.

Step 6 Let $H = S^{-1}[A_{\sigma(m+1)}, A_{\sigma(m+2)}, \cdots, A_{\sigma(n)}] \bmod p$.

Public Key H, p.

Private Key $N_1, N_2, \cdots, N_n$, S, τ, σ.

Encryption: For any message $t \in \{0,1\}^m$, first,we uniformly choose a vector r from $\{0,1\}^m$, then compute the ciphertext:

$$c = Ht + r \bmod p$$

Decryption: Let $v = \begin{pmatrix} r \\ t \end{pmatrix}$, and $v' = \begin{pmatrix} v_{\sigma^{-1}(1)} \\ \vdots \\ v_{\sigma^{-1}(n)} \end{pmatrix}$, then we first compute:

$$\begin{aligned} c' &= Sc && \bmod p \\ &= S[I|H]v && \bmod p \\ &= B^{-1}CC^{-1}BAv' && \bmod p \\ &= Av' && \bmod p \end{aligned}$$

Choosing the entries of c' in the interval from $-\frac{p}{2}$ to $\frac{p}{2}$, for the choice of p, we get $c' = Av'$, then we compute

$$(N_{\tau(1)}, N_{\tau(2)}, \cdots, N_{\tau(m)})Av' = \sum_{i=1}^{n} v_{\sigma^{-1}(i)} N_{\tau(i)} = \sum_{i=1}^{n} v_{\sigma^{-1}\tau^{-1}(i)} N_i.$$

We can easily get $v_{\sigma^{-1}\tau^{-1}(i)}$ by greedy algorithm, and by τ and σ, we can get the message t and r.

Remark 1 *we give an example to show how to use the module strategy to construct a framework for some GGH-type cryptosystems which recover the message t from such a vector $Bt + r$, where $B \in \mathbb{Z}^{m\times m}$ has small entries, and $t \in \mathbb{Z}^m$, $r \in \mathbb{Z}^m$ have the same distribution for their entries. For example, suppose the entries of t and r are uniformly randomly selected in $[a, b]$ in the GGH-type cryptosystems. We first Let $A = [I|B]$, then let p be the smallest prime greater than $2q + 1$, where $q = \max\limits_{i=1,\cdots,m} \max\{|\max\limits_{x_j\in[a,b]} \sum\limits_{j=1}^{m} x_j A_{i,j}|, |\min\limits_{x_j\in[a,b]} \sum\limits_{j=1}^{m} x_j A_{i,j}|\}$. Then, we use the same method in Step 5 and Step 6 to generate the private key σ, S, and the public key H. We use the similar method to encrypt a message t, and to decrypt the message c, we first use the similar method to get $Bt + r$ by the choice of p, then we can recover t from $Bt + r$. The security analysis in Section 5 shows that if p is small enough, the cryptosystem maybe secure against the lattice attack to obtain the message.*

4 Implementations of Our Cryptosystem

4.1 About the Choice of N_i and τ

To generate a superincreasing sequence, we can first give a bound $d \in \mathbb{Z}^+$, then we select $N_1 = 1$ and generate N_i $(2 \leq i \leq n)$ inductively as follows: after having $N_1, N_2, \cdots, N_k$, we uniformly randomly choose $e \in (0, d)$ and let $N_{k+1} = \sum_{j=0}^{k} N_j + e$.

We don't uniformly randomly choose a permutation τ on n letters directly, but use the following way:

1. For $1 \leq i \leq m$, we uniformly randomly choose a permutation ρ on $\{1, 2, \cdots, m\}$, and let $\tau(i) = 2\rho(i) - 1$.
2. For $m + 1 \leq i \leq n$, we independently uniformly randomly choose another permutation ρ on $\{1, 2, \cdots, m\}$, and let $\tau(i) = 2\rho(i - m)$.

The reason to set $N_1 = 1$ and $\tau^{-1}(1) \leq m$ is mainly to ensure that $N_{\tau(i)}(m+1 \leq i \leq n)$ can be represented as the integer linear combination of $N_{\tau(1)}, N_{\tau(2)}, \cdots, N_{\tau(m)}$. Notice that even we don't let $N_1 = 1$, the probability that $N_{\tau(i)}(m+1 \leq i \leq n)$ can be represented as the integer linear combination of $N_{\tau(1)}, N_{\tau(2)}, \cdots, N_{\tau(m)}$ must be very large.

4.2 How to Realize Step 3

First, we give an algorithm to represent an integer y as an integer linear combination of $T_1, T_2, \cdots, T_k \in \mathbb{Z}$ with small coefficients by using the lattice reduction algorithm.

Algorithm $LS(y, T_1, T_2, \cdots, T_k)$

Input: $y, T_1, T_2, \cdots, T_k$.

Output: $b_1, b_2, \cdots, b_k \in \mathbb{Z}$ which are small and $y = \sum_{i=1}^{k} b_i T_i$

1. Choose a solution $b'_1, b'_2, \cdots, b'_k \in \mathbb{Z}$, such that $y = \sum_{i=1}^{k} b'_i T_i$.
2. Let $\mathcal{L}$ be the lattice $\{(x_1, x_2, \cdots, x_k)^T \in \mathbb{Z}^k | \sum_{i=1}^{k} x_i T_i = 0\}$, and use Babai's Nearest Plane Algorithm to find $(x'_1, x'_2, \cdots, x'_k)^T \in \mathcal{L}$ close to $b'_1, b'_2, \cdots, b'_k$.
3. Let $b_i := b'_i - x'_i$ for $1 \leq i \leq k$.
4. Output $b_1, b_2, \cdots, b_k$.

We don't use the algorithm LS to find $b_{i-m,j}(1 \leq j \leq m)$, because it costs too much time when m is large. To make the algorithm more efficient, a greedy strategy is involved.

Let $T_i = N_{\tau(i)}$, $i = 1, 2, \cdots, m$, we first sort $T_1, T_2, \cdots, T_m$ in ascending order, we can assume $T_{\varphi(1)} < T_{\varphi(2)} < \cdots < T_{\varphi(m)}$, where φ is a permutation on m letters. Notice that if we choose τ like in Subsection 4.1, we can easily get the order and φ, because $T_{\varphi(i)} = N_{2i-1}$.

Then we use the algorithm below to find $b_{i-m,j} (1 \leq j \leq m)$ for any $N_{\tau(i)} (m+1 \leq i \leq n)$ with additive parameters δ and k.

Input: δ, k, $T_{\varphi(1)}, T_{\varphi(2)}, \cdots, T_{\varphi(m)}$, φ and $N_{\tau(i)}$ where $i > m$

Output: $b_{i-m,1}, \cdots, b_{i-m,m} \in \mathbb{Z}$ which are small and $N_{\tau(i)} = \sum_{j=1}^{m} b_{i-m,j} T_j$

---

for j from 1 to m do
  uniformly choose an integer $a \in [-\delta, \delta]$
  $N_{\tau(i)} := N_{\tau(i)} - aT_{\varphi(j)}$
  $b_{i-m,\varphi(j)} := a$
end for
for $j = m$ to $k + 1$ do
  compute $N_{\tau(i)} = qT_{\varphi(j)} + r$, where $q, r \in \mathbb{Z}$ and $|r| \leq \frac{T_{\varphi(j)}}{2}$
  $N_{\tau(i)} := r$
  $b_{i-m,\varphi(j)} := b_{i-m,\varphi(j)} + q$
end for
Compute $(x_1, x_2, \cdots, x_k)^T := LS(N_{\tau(i)}, T_{\varphi(1)}, T_{\varphi(2)}, \cdots, T_{\varphi(k)})$, and let
  $b_{i-m,\varphi(i)} := b_{i-m,\varphi(i)} + x_i$ for $1 \leq i \leq k$.
Output $b_{i-m,1}, \cdots, b_{i-m,m}$

Experiments show that the choice of δ and k affects the size of p, the probability that S is invertible and the running time of the algorithm. So we expect δ and k are as small as possible.

4.3 Some Experimental Results

We implemented the cryptosystem on an AMD Athlon(tm) 64 Processor 2800+ 1.81 GHz PC using Shoup's NTL library version 5.4.1[20]. In all our experiments, we let the bound d in Subsection 4.1 be 40, $\delta = 2$, and we used the function LatticeSolve in NTL directly instead of implementing the Algorithm LS. For $m = 100, 200, 300, 400, 500$, we let $k = 10, 20, 30$ respectively. For each m and k, 10 instances were tested. The results are stated as below.

How Large p can Be? Since p decides the size of the key directly, it is necessary to study how large it can be. For each m and k, we give the minimum, maximum and the average of p's in our 10 instances.

m	100			200			300		
k	10	20	30	10	20	30	10	20	30
min	173	173	173	337	347	359	557	521	509
max	251	227	223	421	433	431	683	631	577
average	211.2	196.6	203.4	381.6	384.8	388.4	591.0	582.0	553.0

m	400			500		
k	10	20	30	10	20	30
min	673	701	701	863	853	821
max	877	821	863	1091	1103	1061
average	777.0	743.8	759.4	952.0	946.0	923.8

It is reasonable to assume that $p \approx 2m$. Of course, p is not necessary to be a prime, but to be a prime is to increase the probability that S is invertible.

The Probability that S is Invertible. In our experiments, we uniformly chose a permutation σ on n letters, and S was always invertible. Hence, it is also reasonable to believe that S is invertible with very high probability when we uniformly choose a permutation σ on n letters.

The Key Size and Speed. Since $p \approx 2m$, the public key size is $m^2 \log 2m$, and the private key size is $O(m^2 \log 2m)$. To encrypt an m bits message, we need $O(m^2)$ modular addition, and the ciphertext is at most $m \log 2m$ bits, so the message expansion is $\log 2m$. To decrypt a ciphertext, we also need $O(m^2)$ operations.

5 Security Analysis

Message Security. The direct lattice attack against our cryptosystem to recover the message is to solve the CVP for the lattice spanned by

$$B = \left(\begin{array}{c|c} \alpha I & 0 \\ \hline H & pI \end{array}\right)$$

with the target vector $\begin{pmatrix} 0 \\ \hline c \end{pmatrix}$, because there exists a vector $u \in \mathbb{Z}^m$, such that

$$\left(\begin{array}{c|c} \alpha I & 0 \\ \hline H & pI \end{array}\right)\begin{pmatrix} t \\ \hline u \end{pmatrix} - \begin{pmatrix} 0 \\ \hline c \end{pmatrix} = \begin{pmatrix} \alpha t \\ \hline -r \end{pmatrix},$$

where t is the corresponding message and r is the random vector selected in encryption and $\| \begin{pmatrix} \alpha t \\ \hline -r \end{pmatrix} \|$ is small.

By the Gaussian Heuristic, the size of the solution of the closest vector problems is approximately $\sqrt{\frac{n}{2\pi e}} det(\mathcal{L}(B))^{\frac{1}{n}} = \sqrt{\frac{\alpha pm}{\pi e}}$. For any message t and random vector r, to minimize

$$c(t,r) = \frac{\sqrt{\alpha^2\|t\|^2 + \|r\|^2}}{\sqrt{\frac{\alpha pm}{\pi e}}} = \sqrt{\frac{(\alpha^2\|t\|^2 + \|r\|^2)\pi e}{\alpha pm}},$$

we get $\alpha = \|r\|/\|t\|$ and

$$c(t,r) = \sqrt{\frac{2\pi e\|t\|\|r\|}{pm}}.$$

$c(t,r)$ gives a measure of the vulnerability of an individual message to a lattice attack. An encrypted message is most vulnerable is $c(t,r)$ is small, and becomes less so as $c(t,r)$ gets closer to 1.

Notice that $\|t\|\|r\| \approx \frac{m}{2}$, $c(t,r)$ is approximately $\sqrt{\frac{\pi e}{p}}$. The tables below give the values of $c_{msg} = \sqrt{\frac{\pi e}{p}}$ in our experiments when we use the average of p's.

m	100			200			300		
k	10	20	30	10	20	30	10	20	30
c_{msg}	0.201	0.208	0.205	0.150	0.149	0.148	0.120	0.121	0.124

m	400			500		
k	10	20	30	10	20	30
c_{msg}	0.105	0.107	0.106	0.095	0.095	0.096

Key Security. There is not an obvious attack to obtain the whole private key in our cryptosystem.

However, using the direct lattice attack to obtain A, we need find m short vectors for the lattice $\left(\begin{array}{c|c} I & 0 \\ \hline H^T & pI \end{array}\right)$, since every column of $A' = [A_{\sigma(1)}, A_{\sigma(2)}, \cdots, A_{\sigma(n)}]^T$ is in the lattice. We denote l the average of $\|A'_i\|$'s for $1 \le i \le m$. By the Gaussian Heuristic, the size of the solution of the shortest vector problems is approximately $\sqrt{\frac{n}{2\pi e}} det(\mathcal{L}(B))^{\frac{1}{n}} = \sqrt{\frac{pm}{\pi e}}$. So we get the value of $c_{key} = \frac{l}{\sqrt{\frac{pm}{\pi e}}}$. The smaller c_{key} is, the more easily A'_i's may be found. As it gets closer to 1, to find A'_i's may be more difficult. The tables below give the values of l's and c_{key}'s in our experiments when we use the average of p's.

m	100			200			300		
k	10	20	30	10	20	30	10	20	30
l	17.76	17.73	17.84	24.83	24.95	24.97	30.46	30.52	30.55
c_{key}	0.357	0.370	0.366	0.263	0.263	0.262	0.211	0.213	0.219

m	400			500		
k	10	20	30	10	20	30
l	35.17	35.21	35.22	39.32	39.34	39.36
c_{key}	0.184	0.189	0.187	0.167	0.167	0.169

Remark 2 *Comparing with GGH, to recover the message using direct lattice reduction, we need solve a CVP for a $2m$-dimensional lattice instead of m-dimensional in GGH. This may allow us to use small dimensional matrix as public key to provide sufficient security.*

Comparing with NTRU, there is not an obvious attack to obtain the private key in our cryptosystem while the private key of NTRU can be obtained by finding the short vector of NTRU-lattice. Moreover, it seems that we use a more random lattice with no special cyclical structure like NTRU, this makes our scheme resist some similar attacks against NTRU which are based on the cyclical structure.

6 Conclusion

We have presented a new lattice-based public-key cryptosystem mixed with a knapsack, and we have shown it has reasonable key size and quick encryption and decryption. The constant c shows that it may resist the ordinary lattice attack. Moreover, we can use the same module strategy to construct a framework for some GGH-type cryptosystems to improve their security.

References

[1] M. Ajtai, "Gennerating hard instances of lattice problems," in *Proc. of 28th STOC*, New York, USA: ACM, 1996, pp. 99–108.

[2] M. Ajtai, "Representing hard lattices with $O(n \log n)$ bits," in *Proc. of 37th STOC*, D.S. Johnson, U. Feige, Eds. New York, USA: ACM, 2005, pp. 94–103.

[3] M. Ajtai, C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. of 29th STOC*, New York, USA: ACM, 1997, pp. 284–293.

[4] L. Babai, "On Lovasz lattice reduction and the nearest lattice point problem", Combinatorica 6, 1C13 (1986)

[5] J.-Y. Cai, T.W. Cusick, "A lattice-based public-key cryptosystem," in *Proc. of SAC'98* (*Lecture Notes in Computer Science*), S. Tavares, H. Meijer, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1556, pp. 219–233.

[6] D. Coppersmith, A. Shamir, "Lattice attacks on NTRU", in *Proc of EuroCrypt'97* (*Lecture Notes in Computer Science*), W. Fumy, Ed. Berlin, Germany: Springer, 1997, Vol. 1233 pp. 52C-61.

[7] O. Goldreich, S. Goldwasser, S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Crypto'97* (*Lecture Notes in Computer Science*), B.S. Kaliski Jr., Ed. Berlin, Germany: Springer-Verlag, 1997, vol. 1294, pp. 112–131.

[8] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions", In *Proc. of 40th STOC*, New York, USA: ACM, 2008, pp 197–206.

[9] N. Howgrave-Graham, J.H. Silverman, W. Whyte, "A Meet-In-The-Meddle Attack on an NTRU Private Key", available at http://www.ntru.com/cryptolab/tech notes.htm#004

[10] J. Hoffstein, J. Pipher, J.H. Silverman, "NTRU: A Ring-Based Public Key Cryptosystem," in *Proc. of Algorithmic Number Theory* (*Lecture Notes in Computer Science*), J.P. Buhler, Ed. Berlin, Germany: Springer-Verlag, 1998, vol. 1423, pp. 267–288.

[11] A. K. Lenstra, H. W. Lenstra, and L. Lovász: Factoring polynomials with rational coeffcients. Math. Ann., 261:515-534, 1982

[12] A. May, J.H. Silverman, "Dimension Reduction Methods for Convolution Modular Lattices", In *Proc of Cryptography and Lattices* (*Lecture Notes in Computer Science*), J.H. Silverman, Ed. Berlin, Germany: Springer-Verlag, 2001, vol. 2146, pp. 110–125.

[13] P. Nguyen, J. Stern, "Cryptanalysis of the Ajtai-Dwork cryptosystem," in *Crypto'98* (*Lecture Notes in Computer Science*), H. Krawczyk, Ed. Berlin, Germany: Springer-Verlag, 1998, vol. 1462, pp. 223–242.

[14] P. Nguyen, "Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto'97", in *Proc. of Crypto'99* (*Lecture Notes in Computer Science*), Berlin/Heidelberg, Germany: Springer-Verlag, 1999, vol. 1666, pp. 288–2304.

[15] Y. Pan, Y. Deng: "Cryptanalysis of the Cai-Cusick Lattice-based Public-key Cryptosystem," Cryptology ePrint Archive, Report 2008/204, available at http://eprint.iacr.org/2008/204

[16] C. Peikert, "Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem", In *Proc. of 41th STOC*, New York, USA: ACM, 2009, pp 333–342 .

[17] O. Regev, "New lattice-based cryptographic constructions," Journal of the ACM, 51(2004), 899–942.

[18] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proc. of 37th STOC*, D.S. Johnson, U. Feige, Eds. New York, USA: ACM, 2005, pp. 84–93.

[19] P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," *In Proc. of 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, IEEE Computer Science Press, 1994, pp. 124-134.

[20] V. Shoup, NTL: A library for doing number theory. Available at http://www.shoup.net/ntl/