*Int. J. Communications, Network and System Sciences*, 2012, 5, 834-838
http://dx.doi.org/10.4236/ijcns.2012.512088 Published Online December 2012 (http://www.SciRP.org/journal/ijcns)

# Cryptanalysis of the Double-Moduli Cryptosystem

**Sonia Mihaela Bogos, Serge Vaudenay**
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Email: soniamihaela.bogos@epfl.ch, serge.vaudenay@epfl.ch

Received October 2, 2012; revised November 2, 2012; accepted November 13, 2012

## ABSTRACT

In this article we present a lattice attack done on a NTRU-like scheme introduced by Verkhovsky in [1]. We show how, based on the relation between the public and private key, we can construct an attack which allows any passive adversary to decrypt the encrypted messages. We explain, step by step, how an attacker can construct an equivalent private key and guess what the original plaintext was. Our attack is efficient and provides good experimental results.

**Keywords:** Complex Modulus; Primary Residue; Plaintext Pre-conditioning; Plaintext Attack; Public-Key Scheme; Lattices; LLL Algorithm; Orthogonal Lattices

## 1. Introduction

Lattice-based cryptography has become a research topic more and more studied nowadays. It may offer a good alternative to cryptographic schemes based on classical number-theory problems (e.g. discrete logarithm, factorization) that are easily solved on quantum computers.

Lattices have proven to provide securely hard problems on which we can build cryptographic schemes but also good tools for cryptanalysis. There are several lattice attacks [2,3] done on NTRU [4]. The main tool of these attacks is the LLL algorithm [5]. In order to overcome this, there are variants of NTRU which base their security on lattice hard problems [6].

In this article we present a lattice attack done on a NTRU-like scheme introduced by Verkhovsky in [1].

Based on the relation between the public and private key, we construct an attack which allows any passive adversary to decrypt the encrypted messages. Moreover, our attack is efficient and provides good experimental results.

## 2. Preliminaries

We present in this section the essential background in lattices and Gaussian integers and the algorithms we use in our attack.

**Notation.** We use small letters and capital letters, $b$ and $B$, to denote vectors and matrices, respectively. Capital letters like $R$ are also used for Gaussian integers. In order to avoid confusion, we use the Gaussian integers in the following form $R = r_1 + r_2 \times i$. We denote by $\langle a, b \rangle$ the inner product of two vectors $a$ and $b$.

### 2.1. Background

Given $n$ linearly independent vectors $b_1, b_2, \cdots, b_n \in \mathbb{Z}^m$, a *lattice* $L$ is the set of all linear combinations of $b_i$'s with integral coefficients:

$$L(b_1, b_2, \cdots, b_n) = L(B) = \left\{ \sum x_i b_i \,\middle|\, x_i \in \mathbb{Z} \right\}.$$

We say that $b_1, b_2, \cdots, b_n$ is the *basis* $B$ of the lattice $L$, $n$ is the *rank* and $m$ is the *dimension* of the lattice $L$. If $n = m$, then the lattice is called a *full-rank lattice*. We define the norm, $|a|$, of a vector

$$a = (a_1, a_2, \cdots, a_m) \in \mathbb{Z}^m$$

to be the Euclidean norm.

The *orthogonal lattice* $L^\perp$ of $L$ is the set of vectors orthogonal with all the vectors from $L$:

$$L^\perp = \left\{ a \in \mathbb{Z}^m \,\middle|\, \langle a, b \rangle = 0, \forall b \in L \right\}.$$

It makes sense to speak about orthogonal lattice only for non full-rank lattices, where $n < m$. The lattice $L^\perp$ has dimension $m$ and rank $m - n$.

One important tool in cryptanalysis, *LLL algorithm* [5] was published in 1982 and since then couple of schemes were broken [7-9] by using it. Several improvements that reduce its complexity appeared in [10,11]. Given a basis of a lattice $L$, the aim of the LLL algorithm is to provide a *LLL reduced basis* where the first vector gives an approximation of the shortest non-zero vector of $L$, $\lambda_1(L)$. It is possible to apply the LLL algorithm for the orthogonal lattice $L^\perp$ (See Algorithm 2.1).

The notation $p_m \downarrow (b)$ denotes the last $m$ components of $b$, with $b \in \mathbb{Z}^{m+n}$. By $B^{\mathrm{T}}$ we denote the trans-

 

**Algorithm 2.1. LLL algorithm for $L^{\perp}$ [12].**

**Input:** Lattice basis $b_1, b_2, \cdots, b_n \in \mathbb{Z}^m$.

**Output:** A LLL reduced basis for $L^{\perp}$.

1. Select $c = 2^{(m-1)/2+(m-n)(m-n-1)/4} \prod_{i=1}^{n} |b_i|$.

2. Construct matrix $B_{\Omega} = \begin{pmatrix} cB^T \\ I_{m,m} \end{pmatrix}$.

3. Run LLL algorithm on the lattice spanned by $B_{\Omega}$ to obtain the LLL reduced basis $(a_1, \cdots, a_m)$.

4.Output $\left(p_m \downarrow (a_1), \cdots, p_m \downarrow (a_{m-n})\right)$.

pose of matrix *B*.

## 2.2. Gaussian Integers

Gaussian integers are represented by the set

$$\mathbb{Z}[i] = \left\{ r_1 + r_2 \times i \,|\, r_1, r_2 \in \mathbb{Z}, i^2 = -1 \right\}.$$

The *norm* of a Gaussian integer $R$, denoted by $N(R)$, is defined as $N(R) = r_1^2 + r_2^2$. The units of $\mathbb{Z}[i]$ are $\pm 1, \pm i$.

For division in $\mathbb{Z}[i]$ with unique remainder we need the following definition:

**Definition 1([1]).** Given two Gaussian integers, $A = a_1 + a_2 \times i$ and $R = r_1 + r_2 \times i$, we say that $A$ is a *primary residue* modulo $R$ if the following 4 inequalities are satisfied:

$$0 \le r_1 \times a_2 - r_2 \times a_1 \le N(R) - 1$$

$$0 \le r_1 \times a_1 + r_2 \times a_2 \le N(R) - 1.$$

All primary residues modulo $R$ are located inside the square with vertices $O, R, iR, (1+i)R$ and with sides equal to $\sqrt{N(R)}$.

This definition allows us to have the following theorem:

**Theorem 1.** For any two Gaussian integers $A, R \in \mathbb{Z}[i]$, with $R \neq 0$, there exists unique $B, C \in \mathbb{Z}[i]$ such that $A = R \times C + B$ and $B$ is a primary residue modulo $R$.

We denote $B = A \bmod R$. Note that $A = R \times C + B$ is *not* the Euclidean division.

For completeness, we provide in this section all the definitions used in the formalization of the scheme for which we construct an attack.

**Definition 2.** Primes $R$ in $\mathbb{Z}[i]$ can be expressed by one of the following forms:

- $r_1 = 0, r_2 \neq 0$ and $r_2$ is a prime number of the form $4n+3$ with $n > 0, n \in \mathbb{Z}$.
- $r_1 \neq 0, r_2 = 0$ and $r_1$ is a prime number of the form $4n+3$ with $n > 0, n \in \mathbb{Z}$.
- $r_1 \neq 0, r_2 \neq 0$ and $N(R)$ is a prime number.

**Theorem 2.** A prime number $R = r_1 + r_2 \times i$ of $\mathbb{Z}[i]$ is an irreducible element. Every irreducible element $A$ has a unique prime representative $R$ (*i.e.*, $A \times R^{-1}$ is an unit).

Two Gaussian integers, $A, R \in \mathbb{Z}[i]$, $A \neq 0$ and $R \neq 0$, are *relatively prime* if they have no prime factors in common. The greatest (in the sense of the norm) common divisor of any two elements of $\mathbb{Z}[i]$ is unique up to a unit factor. The Euclid algorithm (using Euclidean division) always returns a greatest (in the sense of norm) common divisor. A multiplicative inverse of $R$ modulo $n$, with $n \in \mathbb{Z}$, exists if and only if $n$ and $N(R)$ are relatively prime.

# 3. Double Moduli Cryptosystem

The cryptosystem introduced by Verkhovsky in [1], for which we construct an attack, is described in this section. We assume an a priori agreed large integer $n$. Apart from the value $n$, which is an integer, all the other parameters and inputs are Gaussian integers.

## 3.1. Encryption/Decryption Algorithms

**Algorithm 3.1** presents the steps followed by a participant with the aim of obtaining its public and private keys. The private key consists of two parts, $P$ and $R$, which are relatively prime. Here, $P$ is invertible modulo $n$. The public key, $U$, is obtained by multiplying $R$ with the inverse of $P$ modulo $n$.

Before encrypting a message $M = m_1 + m_2 \times i$ with the public key $U$, one has to pre-condition the plaintext so that it is a primary residue modulo $R$, where $R$ is part of the private key. Since $R$ is not known to the sender, a threshold is imposed so that the inequalities from Definition 1 hold. The pre-conditioned plaintext $W$ must be selected such that the upper bound of the real and imaginary parts is $\sqrt{n/6}$. The algorithms of pre-conditioning and recovery of a plaintext are described afterwards.

**Algorithm 3.2** shows how to encrypt a pre-conditioned plaintext $W$. Besides the public key $U$, the sender chooses periodically a new value

$$S = s_1 + s_2 \times i \in \mathbb{Z}[i].$$

After hiding the value of the public key, by multiplying it with $S$, the ciphertext is obtained by adding this new value to the plaintext.

After receiving the ciphertext and provided that it has the correct private keys, the receiver is able to decrypt the message by following the steps from **Algorithm 3.3**. After the first step of the algorithm the receiver will compute $D$ as $PW + RS$. In the second step, he is able to compute $Q$ as the inverse of $P$ modulo $R$, as $P$ and $R$ were chosen such that they are relatively prime. Finally, in the last step, the pre-conditioned plaintext is

 

**Algorithm 3.1. Key generation.**

**Input:** Large integer $n$.

**Output:** Public key $U \in \mathbb{Z}[i]$; Private key: $(P,R) \in \mathbb{Z}[i]$.

**Constraints:** $P \in \mathbb{Z}[i]: p_2 < 0 < p_1, \sqrt{n/6} \le p_1, |p_2| \le \sqrt{2n/3}$;
$R \in \mathbb{Z}[i]: r_2 < 0 < r_1, |r_2| < r_1, \sqrt{n/6} \le r_1, |r_2| \le \sqrt{2n/3}$.

1. Choose uniformly $P, R \in \mathbb{Z}[i]$ such that the constraints are respected and $gcd(p_1, p_2) = 1, gcd(r_1, r_2) = 1$, $P$ and $R$ are relatively prime, $P$ and $n$ are relatively prime.
2. Compute $F = P^{-1} \bmod n$.
3. $U = F \times R \bmod n$.

**Algorithm 3.2. Encryption.**

**Input:** Pre-conditioned plaintext $W \in \mathbb{Z}[i]$, integer $n$.

**Public key:** $U \in \mathbb{Z}[i]$.

**Output:** Encryption $C \in \mathbb{Z}[i]$.

**Constraints:** $S \in \mathbb{Z}[i]: s_1 < 0 < s_2, 0 \le |s_1|, s_2 \le \sqrt{n/6}$;
$W \in \mathbb{Z}[i]: 0 \le w_2 < w_1, 0 \le w_1, w_2 \le \sqrt{n/6}$.

1. Select uniformly $S = s_1 + s_2 \times i$ such that the constraints are respected.
2. $C = (W + SU) \bmod n$.

**Algorithm 3.3. Decryption.**

**Input:** Encryption $C \in \mathbb{Z}[i]$, large integer $n$.

**Private key:** $(P,R) \in \mathbb{Z}[i]$.

**Output:** Pre-conditioned plaintext $W \in \mathbb{Z}[i]$.

1. $D = PC \bmod n$.
2. Compute $Q = P^{-1} \bmod R$.
3. $W = QD \bmod R$.

obtained. Afterwards, the receiver will run the algorithm of plaintext recovery, algorithm illustrated later.

### 3.2. Plaintext Pre-Conditioning

As aforementioned, the plaintext is pre-conditioned before being encrypted. Similarly, a plaintext recovery algorithm is necessary in order to obtain the original message after decryption. These two transformations are illustrated in **Algorithms 3.4** and **3.5**. As $W$ must be a primary residue modulo $R$, the sender must ensure that the original plaintext $M$ is split into blocks of appropriate sizes.

## 4. Using LLL to Break the Scheme

This section presents our lattice attack. We prove that the double moduli scheme is insecure as any passive adversary that observes the encrypted messages can decrypt them with a non-negligible probability.

**Algorithm 3.4. Plaintext pre-conditioning.**

**Input:** Message $M \in \mathbb{Z}[i]$.

**Output:** Pre-conditioned plaintext $W \in \mathbb{Z}[i]$.

1. $w_1 = m_1 + m_2$.
2. **if** $m_1 \ge m_2$,
   **then** $w_2 = m_1 - m_2$;
   **else** $w_2 = m_2 - m_1 - 1$.

**Algorithm 3.5. Plaintext recovery.**

**Input:** Pre-conditioned plaintext $W \in \mathbb{Z}[i]$

**Output:** Message $M \in \mathbb{Z}[i]$.

1. **if** $w_1 \equiv w_2 \pmod 2$
   **then** $m_1 = (w_1 + w_2)/2$, $m_2 = w_1 - m_1$;
   **else** $m_1 = (w_1 - w_2 - 1)/2$
   $m_2 = w_1 - m_1 = (w_1 + w_2 + 1)/2$.

### 4.1. Lattice Attack

An attacker is a probabilistic algorithm which runs in polynomial time. From **Algorithm 3.1** an attacker can observe the following relation between $P$, $R$ and $U$ namely $P \times U \equiv R \pmod n$. Using this relation he is able to obtain an *equivalent* private key. This equivalent key is not necessary the private key $(P,R)$, but can be used to decrypt correctly the encrypted message. We write the aforementioned relation for the imaginary and real parts and separate the known parts from those unknown. We obtain the following equation:

$$\begin{pmatrix} u_1 & -u_2 & -1 & 0 & -n & 0 \\ u_2 & u_1 & 0 & -1 & 0 & -n \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ r_1 \\ r_2 \\ k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

With the design constraints of the scheme where both components of the private key are of size $\sqrt{n}$ and the public key is of size $n$, both $k_1$ and $k_2$ should be of size $\sqrt{n}$.

Vectors

$$v_1 = (u_1, -u_2, -1, 0, -n, 0)$$

and

$$v_2 = (u_2, u_1, 0, -1, 0, -n)$$

are linearly independent and are also orthogonal. They form the basis, $B$, of a lattice $L = (v_1, v_2)$ of dimension 6 and rank 2. Vector $o_1 = (p_1, p_2, r_1, r_2, k_1, k_2)$ from the above relation is orthogonal to both vectors and belongs to the orthogonal lattice $L^{\perp}$ of $L$. We can run

 

**Algorithm 2.1** to obtain a reduced basis $B'$ of $L^{\perp}$.

The four vectors from $B'$ should be small in the sense that their norm should be at most $\det(L)^{\frac{1}{4}}$ [12]. As $v_1$ and $v_2$ are orthogonal, the determinant of $L$ can be easily computed as

$$\det(L) = |v_1| \cdot |v_2| \leq \sqrt{n^2+n^2+1+n^2} \cdot \sqrt{n^2+n^2+1+n^2} = 3 \cdot n^2 + 1$$

Thus, the vectors of the reduced basis of $L^{\perp}$ should have norm at most $\left(3n^2+1\right)^{\frac{1}{4}}$ which is of order $\sqrt{n}$. Comparing this value with the norm of the vector $o_1$ which is also of order $\sqrt{n}$ indicates that $o_1$ may not be the shortest vector from $L^{\perp}$ Nevertheless, using the vectors from $B'$ we can find an equivalent key $(P', R')$ that decrypts correctly the ciphertext $C$.

With the results we have so far, we can design a decryption strategy for an attacker illustrated in **Algorithm 4.1**. **Algorithm 4.2** follows.

The following lemma proves that the experiment works correctly: given the ciphertext $C$ and the public key, the attacker is able to obtain an equivalent private key and to decrypt correctly $C$.

**Algorithm 4.1. Decrypting *C* with an equivalent private key (*P′*, *R′*).**

**Input:** Encryption $C \in \mathbb{Z}[i]$.

**Public key:** $U \in \mathbb{Z}[i]$, $n$.

**Output:** Equivalent private key $(P', R')$, pre-conditioned plaintext $W$.

1. Set the basis of lattice $L$ to be $B = (v_1, v_2)$ where, $v_1 = (u_1, -u_2, -1, 0, -n, 0), v_2 = (u_2, u_1, 0, -1, 0, -n)$.
2. Run the LLL algorithm to obtain the reduced basis, $B' = (v_1', v_2', v_3', v_4')$, of the orthogonal lattice $L^{\perp}$.
3. With $v_{j'} = \left(p_1^j, p_2^j, r_1^j, r_2^j, k_1^j, k_2^j\right)$, construct $P_j = p_1^j + p_2^j \times i$ and $R_j = r_1^j + r_2^j \times i$, with $1 \leq j \leq 4$.
4. **If** $\exists (P_j, R_j)$ such that $P_j$ and $R_j$ are relatively prime, $1 \leq j \leq 4$,
   **then** $P' = P_j$ and $R' = R_j$
   Run **Algorithm 4.2** with input *C* and private key $(P', R')$.
   **else** Fail.

**Algorithm 4.2. New decryption procedure.**

**Input:** Encryption $C \in \mathbb{Z}[i]$., large integer $n$.

**Private key:** $(P', R') \in \mathbb{Z}[i]$.

**Output:** Pre-conditioned plaintext $W \in \mathbb{Z}[i]$.

1. $D' = P'C \bmod n$.
2. Compute $Q' = P'^{-1} \bmod R'$.
3. For all 13 possible values for $x \in \mathbb{Z}[i]$ such that $N(x) < 5$, compute $W = Q' \times (D' - xn) \bmod R'$.

**Lemma 1.** The new decryption algorithm works correctly for an equivalent key $(P', R')$, given that $P'$ and $R'$ are relatively prime**.**

*Proof.* Let $C = (W + SU) \bmod n$ be the encrypted message. **Algorithm 4.1** constructs the private key $(P', R')$. From the way we obtained $L^{\perp}$, we have $P'U \equiv R' \pmod n$.

Then,

$$\begin{aligned} D' &= P'C \bmod n \\ &= \left(P' \times (W + SU)\right) \bmod n \\ &= (P'W + P'SU) \bmod n \\ &= (P'W + R'S) \bmod n \\ &= (P'W + R'S) + xn. \end{aligned}$$

We can bound the value of $x$ from equality $D' = (P'W + R'S) + xn$.

$$\begin{aligned} N(x)n^2 &\leq N(D') + N(P'W + R'S) \\ &< N(D') + \max\left(N(P'), N(R')\right) \\ &\quad \cdot 4 \max\left(N(S), N(W)\right) \\ &\leq 2n^2 + \left(\sqrt{3n^2+1}\right) \times \frac{4n}{3} < \frac{13}{3} n^2. \end{aligned}$$

So, $N(x) < 5$ which gives us 13 possibilities for $x$. Using the last equality and computing $Q' = P'^{-1} \bmod R'$, we obtain that

$$\begin{aligned} &Q' \times (D' - xn) \bmod R' \\ &= Q' \times (P'W + R'S) \bmod R' \\ &= W \bmod R'. \end{aligned}$$

By knowing the values of $Q'$, $x$ and $n$, we can find the value of $W \bmod R'$. Having, with high probability, $N(W) < N(R')$, we can guess the original preconditioned plaintext by trying all four values that are inside the circle $C\left(O, \sqrt{N(R')}\right)$ with origin $O$ and radius $\sqrt{N(R')}$. Given $W \bmod R'$, one may obtain the other three values by adding to it $R', \mathrm{i}R', -R'$ or $-\mathrm{i}R'$.

If we analyze the complexity of **Algorithm 4.1**, we easily see that each step is completed in polynomial time. By running **Algorithm 4.1**, an adversary is able to decrypt any message with a high probability. Thus, the scheme is not secure (*i.e.* not even one-way encryption secure).

## 4.2. Experimental Results

The experiments were done on an INTEL Q9550 2.83 GHz processor, running a 32-bit version of Windows 7.

The implementation of the scheme and of the attack was done in the PARI-GP environment. The structure of the scheme was respected as it is described in **Algo-**

 

**Table 1. Experimental results of the lattice attack.**

| $\log_2 n$ | Prob. of success | Running time |
|---|---|---|
| 128 | 1 | 0.09 s |
| 256 | 0.996 | 0.28 s |
| 12 | 0.999 | 0.8 s |
| 1024 | 0.996 | 1.19 s |
| 2048 | 0.997 | 2.23 s |

**rithms 3.1-3.3**. From the beginning we use preconditioned messages. For the Gaussian integers the division operation was implemented making use of the notion of primary residue (see Theorem 1). We made use of the already implemented LLL algorithm, *qflll*, from the PARI-GP library.

Regarding the attack, the implementation respects **Algorithms 4.1** and **4.2**. We run the attack and output all $W$ values that satisfy the constraints from **Algorithm 3.2**. We consider having a success when one of the outputted values is equal to the original plaintext. If no such value is displayed, then the attack fails.

We tested the attack for different sizes of the value $n$, namely values varies between 128 to 2048 bits. A thousand experiments were done for each size of $n$. The results that we obtained are displayed in **Table 1**.

In almost all the cases one of the candidates messages was the original plaintext. The very few cases when the message is not recovered is due to possible two scenarios. It may happen that all four possible values of $R'$ are smaller than the message $W$ and an attacker loses the value of the message by performing the operation $W \bmod R'$. The second scenario may be that none of the four possible values of $P'$ and $R'$ are relatively prime. This can be repaired by constructing a new private key as the linear combination of the four possible private keys, using small coefficients.

This experimental result ascertains that the double-moduli cryptosystem from [1] is insecure.

## 5. Conclusion

In this article we have presented a lattice attack done on a NTRU-like scheme. The main tool is the LLL algorithm used on the orthogonal lattice. Our attack is efficient and gives good results as, in almost all the cases, we can guess correctly the value of the plaintext. Hence, we broke the scheme.

## REFERENCES

[1] B. S. Verkhovsky, "Double-Moduli Gaussian Encryption/ Decryption with Primary Residues and Secret Controls," *IJCNS*, Vol. 4, No. 7, 2011, pp. 475-481. doi:10.4236/ijcns.2011.47058

[2] D. Coppersmith and A. Shamir, "Lattice Attacks on NTRU," *Advances in Cryptology—EUROCRYPT* '97, *International Conference on the Theory and Application of Cryptographic Techniques*, Konstanz, 11-15 May 1997, pp. 52-61.

[3] A. May, "Cryptanalysis of NTRU," 1999. http://www-stud.rbi.informatik.uni-frankfurt.de/~alex/ntru.ps

[4] J. Hoffstein, J. Pipher and J. H. Silverman, "NTRU: A Ring-Based Public Key Cryptosystem," *Algorithmic Number Theory*, *Third International Symposium*, *ANTS-III*, Portland, 21-25 June 1998, pp. 267-288.

[5] A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen*, Vol. 261, No. 4, 1982, pp. 515-534. doi:10.1007/BF01457454

[6] D. Stehlé and R. Steinfeld, "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices," *Advances in Cryptology—EUROCRYPT* 2011—30*th Annual International Conference on the Theory and applications of Cryptographic Techniques*, Tallinn, 15-19 May 2011, pp. 27-47.

[7] J. Håstad, "On Using RSA with Low Exponent in a Public Key Network," *Advances in Cryptology—CRYPTO* '85, Santa Barbara, 18-22 August 1985, pp. 403-408.

[8] J. C. Lagarias and A. M. Odlyzko, "Solving Low-Density Subset Sum Problems," *Journal of the ACM*, Vol. 32, No. 1, 1985, pp. 229-246. doi:10.1145/2455.2461

[9] C. Gentry, J. Jonsson, J. Stern and M. Szydlo, "Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001," *Advances in Cryptology—ASIACRYPT*'01, 7*th International Conference on the Theory and Application of Cryptology and Information Security*, Gold Coast, 9-13 December 2001, pp. 1-20.

[10] P. Q. Nguyen and D. Stehlé, "An LLL Algorithm with Quadratic Complexity," *SIAM Journal on Computing*, Vol. 39, No. 3, 2009, pp. 874-903. doi:10.1137/070705702

[11] C.-P. Schnorr, "A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms," *Theoretical Computer Science Journal*, Vol. 53, 1987, pp. 201-224. doi:10.1016/0304-3975(87)90064-8

[12] P. Q. Nguyen and J. Stern, "Merkle-Hellman Revisited: A Cryptanalysis of the Qu-Vanstone Cryptosystem Based on Group Factorizations," *Advances in Cryptology—CRYPTO*'97, 17*th Annual International Cryptology Conference*, Santa Barbara, 17-21 August 1997, pp. 198-212.

 