MaTRU: A New NTRU-Based Cryptosystem

Michael Coglianese[1] * and Bok−Min Goi[2] **

[1] Macgregor, 321 Summer Street
Boston, MA 02210, USA
mcoglian@comcast.net

[2] Centre for Cryptography and Information Security (CCIS)
Faculty of Engineering, Multimedia University
63100 Cyberjaya, Malaysia
bmgoi@mmu.edu.my

Abstract. In this paper, we propose a new variant of the NTRU public key cryptosystem − the MaTRU cryptosystem. MaTRU works under the same general principles as the NTRU cryptosystem, except that it operates in a different ring with a different linear transformation for encryption and decryption. In particular, it operates in the ring of k by k matrices of polynomials in $\mathbf{R} = \mathbb{Z}[X]/(X^n - 1)$, whereas NTRU operates in the ring $\mathbb{Z}[X]/(X^N - 1)$. Note that an instance of MaTRU has the same number of bits per message as an instance of NTRU when $nk^2 = N$. The improved efficiency of the linear transformation in MaTRU leads to respectable speed improvements by a factor of $O(k)$ over NTRU at the cost of a somewhat larger public key.

Keywords: Public key cryptosystems, NTRU, lattice based cryptography, lattice attacks, partial polynomial evaluation.

1 Introduction

Since the concept of public key cryptography was first introduced by Diffie and Hellman [4] in 1976, there has been steadily increasing interest in cryptographic studies; many public key cryptosystems have been proposed, i.e. RSA [22] based on integer factorization problem, the McEliece systems [15] based on algebraic coding theory, the ECC systems [12] based on the intractability of elliptic curve DLP and the variants of Matsumoto-Imai cryptosystems [14, 3] based on the systems of multivariable polynomials. Unfortunately, in practice many of these algorithms are costly in terms of computational and space complexity. These costs inhibit the ability of these algorithms to be substituted for symmetric key cryptosystems, and it prevents some of them (e.g. RSA) from running effectively

* The first author initially performed this work at Brown University, USA.

** The second author acknowledges the Malaysia IRPA grant (04-99-01-00003-EAR).

on low power computing devices such as low-cost smart cards, RFID devices, and cell phones. As a result, cryptographers continue to look for new, fast public key cryptosystems, especially those which are based on different hard problems. Since 1996, researchers from NTRU Cryptosystems [21] have proposed a group of fast public key cryptosystems based on partial evaluation of constrained polynomials over polynomial rings. These cryptosystems include the NTRU public key encryption algorithm [8] and the digital signature scheme NTRUSign [7].

Next, let us briefly describe one of these cryptosystems, NTRU. NTRU is a public key cryptosystem that operates in the ring $\mathbb{Z}[X]/(X^N - 1)$. Encryption and decryption of a message corresponds to applying a linear transformation to a ring element. Since this linear transformation performs the multiplication of two polynomials, the cost of applying it is $O(N^2)$ operations (assuming Fast Fourier Transforms are not used). In addition, these operations are on small integers, allowing for further speed optimizations. For these reasons, the speed of NTRU is one of its strongest features. NTRU operates considerably faster than both RSA and ECC at relatively the same security levels [13, 8]. However, the speed of NTRU can be further improved by choosing a different ring and applying a more efficient linear transformation [9, 10]. The hard problem underlying this cryptosystem is related to finding short vectors in a lattice due to the properties of short polynomials used in the system [2, 16, 20]. Since NTRU was proposed, it has been cryptanalyzed heavily by the cryptographic community, and some interesting results can be found in [5, 6, 11, 17, 19]. Meanwhile, some variants of NTRU encryption schemes have also been proposed, such as the generalized NTRU schemes [1].

The MaTRU cryptosystem, described in this paper, uses a more efficient linear transformation while providing a security level comparable to that of NTRU. MaTRU operates in the ring of k by k matrices of polynomials in $\mathbf{R} = \mathbb{Z}[X]/(X^n - 1)$. Note that an instance of MaTRU has the same number of bits per message as an instance of NTRU when $nk^2 = N$. While NTRU involves performing a one-sided multiplication for encryption and decryption [8], the linear transformation applied in MaTRU is a two-sided matrix multiplication. This means that the private key in MaTRU has two ring elements, as opposed to one ring element in NTRU. This is essential because multiplying on one side just gives a search space of size, say S, for the private key and the effect would be linear. Then, a lattice attack could be mounted very similar to the one on NTRU. However, multiplying on both sides will amplify the space of all linear transformations to S^2. The lattice attack will be extremely hard, due to the high dimension lattice matrix. Another difference between the two cryptosystems is that the ring in MaTRU is not commutative. This means that the matrices in the private key and the random matrices applied during encryption must specifically be constructed so that they commute with each other.

Since applying the linear transformation in MaTRU involves matrix multiplications, the encryption and decryption processes only require $O(n^2k^3)$ operations. This results in a speed increase by a factor of $O(k)$ over NTRU. In practice, this increase is approximately $\frac{k}{2}$ since MaTRU uses two matrix multiplications for every polynomial multiplication in NTRU. The private and public key lengths for MaTRU are $O(nk^2)$, making them comparable to key lengths in NTRU.

In the next section, we describe our proposed MaTRU cryptosystem in detail. Then, we discuss the parameter selection for MaTRU in Section 3. In Section 4, we present the details of the security analysis of the proposed scheme. We show its security strength based on certain parameter choices and compare it with standard NTRU in Section 5. Finally, we summarize our conclusions in Section 6.

2 The MaTRU Algorithm

2.1 Notation

The MaTRU cryptosystem operates in the ring $\mathbf{M}$ of k by k matrices of elements in the ring $\mathbf{R} = \mathbb{Z}[X]/(X^n-1)$. The ring $\mathbf{R}$ consists of polynomials with degree at most $(n-1)$ having integer coefficients. Multiplication and addition of polynomials in $\mathbf{R}$ is done in the usual manner, but exponents of X are reduced modulo n. Matrix multiplication in $\mathbf{M}$ is denoted using the $*$ symbol.

Besides n and k, MaTRU also uses the parameters $p, q \in \mathbb{N}$. The numbers p and q may or may not be prime, but they must be relatively prime. In general, p is much smaller than q; in this paper, for ease of explanation, we stick to $p = 2$ or $p = 3$ and q in the range of 2^8 to 2^{11}. When we say we perform a matrix multiplication modulo p (or q), we mean that we reduce the coefficients of the polynomials in the matrices modulo p (or q).We define the *width* of an element $M \in \mathbf{M}$ to be $|M|_\infty = (\max_{\text{polys. } m \text{ in } M} \text{coeff. in } m) - (\min_{\text{polys. } m \text{ in } M} \text{coeff. in } m)$. The width of M is the maximum coefficient in any of its k^2 polynomials minus the minimum coefficient in any of its polynomials. We say a matrix $M \in \mathbf{M}$ is *short* if $|M|_\infty \leq p$. When short matrices are multiplied together, we get a matrix which has a width which may be greater than p but is still almost certainly smaller than q; we call this matrix *pretty short*. The definitions for width and shortness apply similarly to polynomials in $\mathbf{R}$. For $r \in \mathbf{R}$, $|r|_\infty = (\text{max coeff. in r}) - (\text{min coeff. in r})$. The polynomial r is said to be short if $|r|_\infty \leq p$. We also define the *size* of an element $M \in \mathbf{M}$ to be $|M| = \sqrt{\sum_{\text{polys. } m \text{ in } M} \sum (\text{coeff. in } m)^2}$.

When defining some of the sets of short matrices below, we use the notation

$$\mathcal{L}(d) = \left\{ M \in \mathbf{M} \;\middle|\; \begin{array}{l} \text{for } i = \left\lceil -\frac{p-1}{2} \right\rceil \ldots \left\lceil \frac{p-1}{2} \right\rceil, i \neq 0, \text{ each polynomial} \\ \text{in } M \text{ has on average } d \text{ coefficients equal to } i, \\ \text{with the rest of the coefficients equal to } 0. \end{array} \right\}.$$

For example, if $p = 3$ and $n = 5$, then $\mathcal{L}(2)$ consists of all matrices of polynomials where on average each polynomial has 2 coefficients equal to 1, 2 coefficients equal to -1, and 1 coefficient equal to zero. Or, if we had $p = 2$ and $n = 5$, then $\mathcal{L}(2)$ consists of all matrices of polynomials where on average each polynomial has 2 coefficients equal to 1 and 3 coefficients equal to zero.

The parameters for MaTRU consist of the four integers (n, k, p, q) described above and the five sets of matrices $(\mathcal{L}_f, \mathcal{L}_\Phi, \mathcal{L}_A, \mathcal{L}_w, \mathcal{L}_m) \subset \mathbf{M}$. These sets have the following meanings and compositions:

Set	Elements	Description	Composition
$\mathcal{L}_f$	f, g	Compose private key	Short; see (2) below
$\mathcal{L}_\Phi$	Φ, Ψ	Random matrices applied for each encryption	Short; see (2) below
$\mathcal{L}_A$	A, B	Used to construct f, g, Φ, Ψ	Short; see (1) below
$\mathcal{L}_w$	w	Used to construct public key	Short
$\mathcal{L}_m$	m	Messages	Short; see (3) below

1. $\mathcal{L}_A$ consists of all matrices $C \in \mathbf{M}$ such that $C^0, C^1, \ldots, C^{k-1}$ are linearly independent modulo q; and for short $c_0, \ldots, c_{k-1} \in \mathbf{R}$, $\sum_{i=0}^{k-1} c_i C^i$ is short. Section 3.2 describes the exact nature of $\mathcal{L}_A$ that satisfies these conditions.
2. $\mathcal{L}_f$ and $\mathcal{L}_\Phi$ consist of all matrices $D \in \mathbf{M}$ constructed such that, for $C \in \mathcal{L}_A$ and short $c_0, \ldots, c_{k-1} \in \mathbf{R}$, $D = \sum_{i=0}^{k-1} c_i C^i$. Additionally, matrices in $\mathcal{L}_f$ must satisfy the requirement that they have inverses modulo p and modulo q.

3. The set of messages $\mathcal{L}_m$ consists of all matrices of polynomials with coefficients modulo p. We therefore express

$$\mathcal{L}_m = \left\{ M \in \mathbf{M} \;\middle|\; \begin{array}{l} \text{polynomials in } M \text{ have coefficients} \\ \text{between } \left\lceil -\frac{p-1}{2} \right\rceil \text{ and } \left\lceil \frac{p-1}{2} \right\rceil \end{array} \right\} .$$

This means that each message contains $nk^2 \log_2 p$ bits of information.

2.2 Key Creation

To create a public/private key pair, Bob chooses two k by k matrices $A, B \in \mathcal{L}_A$. Next, Bob randomly selects short polynomials $\alpha_0, \alpha_1, \ldots \alpha_{k-1} \in \mathbf{R}$ and $\beta_0, \beta_1, \ldots \beta_{k-1} \in \mathbf{R}$. Bob then constructs the matrices $f, g \in \mathcal{L}_f$ by taking

$$f = \sum_{i=0}^{k-1} \alpha_i A^i \qquad \text{and} \qquad g = \sum_{i=0}^{k-1} \beta_i B^i \ .$$

As noted above in Section 2.1, the matrices f and g must have inverses modulo p and modulo q. This will generally be the case, given suitable parameter choices. We denote the inverses as F_p, F_q and G_p, G_q, where

$$\begin{aligned} F_q * f \equiv I (\bmod\ q) \qquad &\text{and} \qquad F_p * f \equiv I (\bmod\ p); \\ G_q * g \equiv I (\bmod\ q) \qquad &\text{and} \qquad G_p * g \equiv I (\bmod\ p). \end{aligned}$$

Note that I is a k by k identity matrix. Bob now has his private key, (f, g), although in practice he will want to store the inverses F_p and G_p as well. Bob now selects a random matrix $w \in \mathcal{L}_w$, and constructs the matrix $h \in \mathbf{M}$ by taking

$$h \equiv F_q * w * G_q \qquad (\bmod\ q) \ .$$

Bob's public key consists of the three matrices, (h, A, B).

2.3 Encryption

To encrypt a message to send to Bob, Alice randomly generates the short polynomials $\phi_0, \phi_1, \ldots \phi_{k-1} \in \mathbf{R}$ and $\psi_0, \psi_1, \ldots \psi_{k-1} \in \mathbf{R}$. Alice then constructs the matrices $\Phi, \Psi \in \mathcal{L}_\Phi$ by taking

$$\Phi = \sum_{i=0}^{k-1} \phi_i A^i \qquad \text{and} \qquad \Psi = \sum_{i=0}^{k-1} \psi_i B^i \ .$$

Alice then takes her message $m \in \mathcal{L}_m$, and computes the encrypted message

$$e \equiv p(\Phi * h * \Psi) + m \qquad \pmod q \ .$$

Alice then sends e to Bob.

2.4 Decryption

To decrypt, Bob computes

$$a \equiv f * e * g \qquad \pmod q \ . \tag{1}$$

Bob translates the coefficients of the polynomials in the matrix a to the range $-q/2$ to $q/2$ using the centering techniques as in the original NTRU paper [8]. Then, treating these coefficients as integers, Bob recovers the message by computing

$$d \equiv F_p * a * G_p \qquad \pmod p \ .$$

2.5 Why Decryption Works

In decryption, from Eq. [1] Bob has

$$\begin{aligned} a &\equiv f * (p(\Phi * h * \Psi) + m) * g && \pmod q \\ &\equiv p(f * \Phi * F_q * w * G_q * \Psi * g) + f * m * g && \pmod q \end{aligned}$$

Although matrix multiplication is not generally commutative, f and Φ here do indeed commute:

$$\begin{aligned} f * \Phi &\equiv (\textstyle\sum_{i=0}^{k-1} \alpha_i A^i) * (\sum_{i=0}^{k-1} \phi_i A^i) && \pmod q \\ &\equiv \textstyle\sum_{i=0}^{k-1} \sum_{i \equiv j+\ell \pmod{\mathrm{k}}} \alpha_j A^j \phi_\ell A^\ell && \pmod q \\ &\equiv \textstyle\sum_{i=0}^{k-1} \sum_{i \equiv j+\ell \pmod{\mathrm{k}}} \phi_\ell A^{j+\ell} \alpha_j && \pmod q \\ &\equiv \textstyle\sum_{i=0}^{k-1} \sum_{i \equiv j+\ell \pmod{\mathrm{k}}} \phi_\ell A^\ell \alpha_j A^j && \pmod q \\ &\equiv (\textstyle\sum_{i=0}^{k-1} \phi_i A^i) * (\sum_{i=0}^{k-1} \alpha_i A^i) \equiv \Phi * f && \pmod q \end{aligned}$$

Similarly, $g * \Psi \equiv \Psi * g \pmod q$. So, Bob now has that

$$a \equiv p(\Phi * w * \Psi) + f * m * g \qquad \pmod q$$

For appropriate parameter choices, $|a|_\infty \leq q$. Then, treating the polynomials in this matrix as having coefficients in $\mathbb{Z}$, Bob can take those coefficients modulo p, leaving $f * m * g \pmod p$. The original message is then recovered by left-multiplying by F_p and right-multiplying by G_p.

3 Parameter Selection

3.1 Selection of pairs (f, g) and (Φ, Ψ)

We define d_f and d_ϕ such that

$$\mathcal{L}_f = \mathcal{L}(d_f) \qquad \text{and} \qquad \mathcal{L}_\Phi = \mathcal{L}(d_\phi) \ .$$

Since the matrices A and B are public, the security of f, g, Φ, and Ψ necessarily depends on the difficulty of discovering the short polynomials α_i, β_i, ϕ_i, and ψ_i. For this reason, we want to maximize the number of possible choices for these polynomials. We therefore commonly select

$$d_f \approx \frac{n}{p} \qquad \text{and} \qquad d_\phi \approx \frac{n}{p} \ .$$

See section 4.1 for precise brute force security calculations.

Remark 1. A matrix f in the ring $\mathbf{M}$ will be invertible modulo p and q, only if the correspond matrix determinant det_f, which is in the ring $\mathbf{R}$, is also invertible modulo p and q. In practice, this is impossible if $det_f(1) = 0$ (the sum of the coefficient values of the determinant polynomial is equal to 0). So we must re-select one or more of the polynomial elements in f if this condition was not fulfilled.

3.2 Selection of A and B

A main concern in generating the matrices f and Φ (and likewise, g and Ψ) is that they must not only commute, but they should also be short. Shorter matrices ensure that $|p(\Phi * w * \Psi) + f * m * g|_\infty$ will be smaller, which will allow us to reduce q and valid ciphertexts will be decipherable.

To achieve this, we select A and B to be *permutation matrices*. A permutation matrix is a binary matrix (i.e. consisting of only the scalars 0 and 1) such that there is exactly one 1 in each row and column with all 0s elsewhere. Since A and B have the additional requirement that the sets $A^0, \ldots, A^{k-1}$ and $B^0, \ldots, B^{k-1}$ are both linearly independent, we have that

$$\sum_{i=0}^{k-1} A^i = \sum_{i=0}^{k-1} B^i = \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{pmatrix} \ .$$

This implies that each row and column of f will contain some permutation of $\alpha_0, \ldots, \alpha_{k-1}$, meaning that each α_i will appear k times in f. An analogous situation exists for g, Φ, and Ψ.

Using the common choice of $d_f \approx d_\phi \approx \frac{n}{p}$, we have that

$$|f| \approx \sqrt{k^2|\alpha_i|^2} \approx \sqrt{\frac{(p-1)nk^2}{p}} \approx |g| \approx |\Phi| \approx |\Psi| \ .$$

3.3 Selection of w

Like f and g, w should also be chosen to be short in order to keep $|p(\Phi * w * \Psi) + f * m * g|_\infty$ small. For security reasons, it is important that w remain secret from an attacker. Therefore, in order to maximize the space of w we make

$$\mathcal{L}_w = \mathcal{L}\left(\left\lfloor \frac{n}{p} \right\rfloor\right) \ .$$

The size of w is then given by

$$|w| = \sqrt{\frac{(p-1)nk^2}{p}} \ .$$

Remark 2. Note that when w is chosen in this manner, on average $|w| \approx |m|$. This means that $|\Phi * w * \Psi| \approx |f * m * g|$.

4 Security Analysis

4.1 Brute Force Attacks

To find a private key by brute force, an attacker must try all possible short pairs of matrices (f, g) to find one such that $f * h * g$ is also short. Since the matrices A and B are public, f and g are determined by the $2k$ polynomials $\alpha_0, \ldots, \alpha_{k-1}, \beta_0, \ldots, \beta_{k-1}$. Each of these polynomials has degree $n-1$, so the number of possible (f, g) pairs is

$$(\text{key security}) \ = \left(\frac{n!}{(n-(p-1)d_f)!d_f!^{(p-1)}}\right)^{2k} . \qquad (2)$$

Similarly, the encryption of a particular message is determined by the $2k$ polynomials $\phi_0, \ldots, \phi_{k-1}, \psi_0, \ldots, \psi_{k-1}$, so we have the same message security as Eq. [2] with replacing d_f by d_ϕ. Using a meet-in-the-middle attack, such as the method due to Odlyzko [18] used on the standard NTRU algorithm, assuming sufficient memory storage, the key and message security would be equal to the square root of the above values. Note that for the standard NTRU algorithm with the suggested parameters, the meet-in-the-middle attack is the most effective known attack.

4.2 Lattice Attacks

Key security. Message decryption, which left-multiplies the encrypted message by f and right-multiplies it by g, amounts to the application of a linear transformation $T : \mathbf{M} \to \mathbf{M}$ such that both T and $T(h)$ are short. If this was the case, then it is likely that either T is the transformation corresponding to the one given by the actual private key, or that T will work as a substitute for the private key in decrypting messages.

Let $T_{f,g} : \mathbf{M} \to \mathbf{M}$ be the linear transformation corresponding to decryption with the actual private key. Then $T_{f,g}$ is defined by $T_{f,g}(J) : J \to f * J * g$. What does the transformation $T_{f,g}$ look like? To see this, we look at where $T_{f,g}$ takes the basis matrices for the space of all possible matrices J. The basis consists of the k^2 matrices $\delta_{i,j}$, where $\delta_{i,j}$ has a 1 in position (i,j) and 0s elsewhere. We then have that $T_{f,g} = \begin{pmatrix} f\delta_{0,0}g & f\delta_{0,1}g & \dots & f\delta_{k-1,k-1}g \end{pmatrix}$. This describes how $T_{f,g}$ maps the basis matrices for the space of possible J's: $\delta_{0,0} \to f\delta_{0,0}g$, $\delta_{0,1} \to f\delta_{0,1}g$, and so on.

Since $f = \sum_{i=0}^{k-1} \alpha_i A^i$ and $g = \sum_{i=0}^{k-1} \beta_i B^i$, we can express $T_{f,g}$ as a combination of the k^2 linear transformations T_{A^i,B^j}, where $T_{A^i,B^j} = \begin{pmatrix} A^i\delta_{0,0}B^j & A^i\delta_{0,1}B^j & \dots & A^i\delta_{k-1,k-1}B^j \end{pmatrix}$. We then have that $T_{f,g}(J) = \sum_{i,j} \gamma_{i,j} T_{A^i,B^j}(J)$. In this formula, each polynomial $\gamma_{i,j}$ is the multiple of T_{A^i,B^j} needed to produce the particular transformation $T_{f,g}$. Therefore, we have precisely that $\gamma_{i,j} = \alpha_i\beta_j$.

Now, since the α_i's and β_j's are short, the polynomials $\gamma_{i,j}$ will be pretty short. In addition, $T_{f,g}(h) = w$ is short. So, the linear transformation $T_{f,g}$ corresponds to a short target vector $(\gamma_{0,0}, \gamma_{0,1}, \dots, \gamma_{k-1,k-1}, w)$ in the lattice $L = \{(T, T(h))\}$. This lattice L is generated by the rows of the following $2nk^2$ by $2nk^2$ matrix composed of four nk^2 by nk^2 blocks:

$$\left(\begin{array}{cccc|cccc}
1 & 0 & \dots & 0 & h_{0,0} & h_{0,1} & \dots & h_{k-1,k-1} \\
0 & 1 & \dots & 0 & h_{k-1,k-1} & h_{0,0} & \dots & h_{k-1,k-2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 1 & h_{0,1} & h_{0,2} & \dots & h_{0,0} \\
\hline
0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & q
\end{array}\right)$$

In the above matrix, the n by n matrix $h_{i,j}$ represents the n coefficients of the polynomial at position (i,j) in h. Note that $\det L = q^{nk^2}$ and $\dim L = 2nk^2$.

As noted earlier, each $\gamma_{i,j} = \alpha_i \beta_j$, so $|\gamma_{i,j}| = |\alpha_i \beta_j| \approx |\alpha_i||\beta_j| \approx (p-1)d_f$. There are k^2 $\gamma_{i,j}$ polynomials, so the size of the target vector $(\gamma_{0,0}, \gamma_{0,1}, \ldots, \gamma_{k-1,k-1}, w)$ is given by

$$|\text{target}| \approx \sqrt{((p-1)d_f)^2 k^2 + |w|^2}.$$

Using the suggested $d_f \approx \frac{n}{p}$ and $|w| = \sqrt{\frac{(p-1)nk^2}{p}}$ yields

$$|\text{target}| \approx \sqrt{\frac{(p-1)nk^2((p-1)n+p)}{p^2}}.$$

By the Gaussian heuristic, the expected shortest vector in L is

$$|\text{exp. shortest}| = \sqrt{\frac{\dim L}{2\pi e}}(\det L)^{\frac{1}{\dim L}} = \sqrt{\frac{qnk^2}{\pi e}}.$$

Let c_h equal the ratio of the target vector to the expected shortest vector. If c_h is near 0, the target vector will likely be much smaller than any other vectors in the lattice, and will therefore be easier to find. If c_h is near 1, then there will likely be many vectors near the size of the target, making the target difficult to find. In our case,

$$c_h \approx \frac{|\text{target}|}{|\text{exp. shortest}|} \approx \sqrt{\frac{\pi e(p-1)((p-1)n+p)}{p^2 q}}.$$

For example, the MaTRU parameters suggested in section 5 give values for c_h around 0.2. This means that if the LLL algorithm finds a vector in the lattice L around two tenth the size of the expected shortest vector, then the algorithm has most likely found $T_{f,g}$ or another suitable linear transformation.

Message security. A lattice attack can also be used to try to discover a particular message. The way this is done is very similar to the lattice attack on a key. Since we selected the parameters $d_f \approx d_\phi$ and $|w| \approx |m|$, we have that $|\Phi * w * \Psi| \approx |f * m * g|$. So the lattice security of a message will be the same as that of the key, meaning $c_m \approx c_h$. The constant c_m indicates how difficult it will be to discover a particular message. Finding the message will be more difficult when c_m is close to 1 and easier when c_m is close to 0.

Remark 3. The above lattice matrix for MaTRU can be further optimized by multiplying the top-left nk^2 by nk^2 identity submatrix by a scaling factor, α, as in [8]. Also, by using zero-forcing technique [16], we can reduce the dimension of the lattice matrix and increase the performance of lattice attacks. These considerations will be taken into account in a future revision of the parameter choices for MaTRU.

5 Discussion

5.1 Parameter Choices

Table 1 shows some possible parameter choices for MaTRU along with their brute force and lattice security levels. Key and message securities listed below are for a meet-in-the-middle attack; these values should be squared for a regular brute force attack.

Table 1. Possible parameter choices for MaTRU

n	k	p	q	d_f	d_ϕ	key security	msg. security	c_h	$\dim L$
6	15	3	2048	2	2	$2^{97.4}$	$2^{97.4}$	0.118	2700
8	9	3	1024	2	2	$2^{78.4}$	$2^{78.4}$	0.188	1296
11	6	3	1024	3	3	$2^{79.0}$	$2^{79.0}$	0.215	792
16	8	2	379	8	8	$2^{109.2}$	$2^{109.2}$	0.318	2048
18	5	2	251	9	9	$2^{77.8}$	$2^{77.8}$	0.412	880

5.2 Comparison with Standard NTRU

Here we compare the theoretical operating characteristics of MaTRU with those of NTRU, as shown in Table 2. The properties are listed in terms of the parameters (N, p, q) for NTRU and the parameters (n, k, p, q) for MaTRU. These should be compared by setting $N = nk^2$, since this equates to plain text message blocks of the same size.

As indicated by the table, the total time for encryption and decryption is $\frac{k}{2}$ times faster for MaTRU than for NTRU. MaTRU has a larger public key length as a result of needing to store the matrices A and B, but a smaller private key length due to the particular nature of the private keys f and g.

For example, compare the NTRU "high" security level of $(N, p, q) = (263, 3, 128)$ with the MaTRU parameter choices of $(n, k, p, q) = (18, 5, 2, 251)$.

Table 2. Comparison between MaTRU with NTRU

Characteristic	NTRU [8]	MaTRU [this paper]
Plain Text Block	$N \log_2 p$ bits	$nk^2 \log_2 p$ bits
Encrypted Text Block	$N \log_2 q$ bits	$nk^2 \log_2 q$ bits
Encryption Speed	$O(N^2)$ operations	$O(n^2k^3)$ operations[1]
Decryption Speed	$O(N^2)$ operations	$O(n^2k^3)$ operations[1]
Message Expansion	$\log_p q$-to-1	$\log_p q$-to-1
Private Key Length	$2N \log_2 p$ bits	$2nk^2 \log_2 p$ bits[2]
Public Key Length	$N \log_2 q$ bits	$3nk^2 \log_2 q$ bits[3]
Key Security (Message Security)[4]	$\frac{N!}{d_g!^2(N-2d_g)!}$	$(\frac{n!}{((n-2)d_f)!d_f!^2})^{2k}$
Lattice Security, c_h (c_m)[5]	$2(\frac{\pi^2 de^2}{3Nq^2})^{\frac{1}{4}}$	$\frac{1}{3}\sqrt{\frac{2\pi e(2n+3)}{q}}$

[1] Since MaTRU performs two-sided multiplications, the constant factor will be about twice that of standard NTRU.
[2] A key length of $2nk \log_2 p + 2k^2 \log_2 k$ bits can be achieved by storing f and g not as matrices but as the $2k$ polynomials found in the matrices along with their positions in the matrices.
[3] A key length of $nk^2 \log_2 q + 2k \log_2 k$ bits can be achieved by storing A and B not as matrices but as the positions of each of the k 1s in the two matrices.
[4] For message security, d_g is replaced by d for NTRU whereas d_f is replaced by d_ϕ for MaTRU. For ease of comparison, we fix $p = 3$. We refer the readers to [8] for the definition of d_g and d used in NTRU.
[5] Note that $c_h \approx c_m$, so that we have equivalent security level at key and message. For ease of comparison, we fix $p = 3$.

NTRU in this case would have a plain text block size of 417 bits, a private key length of 834 bits, and a public key length of 1841 bits. MaTRU would have a plain text block size of 450 bits, a private key length of 297 bits, and a public key length of 3611 bits. MaTRU would theoretically be 2.5 times faster at encryption/decryption than the instance of NTRU in this case.

6 Conclusion

We have presented the MaTRU cryptosystem in detail, and we have shown that its security level is comparable to NTRU with respect to several well-known attacks, including brute force attacks, lattice attacks and meet-in-the-middle attacks. However, the security analysis of MaTRU is heuristic because there may be a better attack on it than on the original NTRU. Further research on the lattice attack on MaTRU may yield new techniques (e.g. subdividing the lattice) that are more effective. We have also suggested several parameter choices for MaTRU that provide a significant speed improvement over NTRU with relatively similar security levels. Future work to obtain precise running times and lattice attack times will allow for further refinements to the list of suggested MaTRU parameters in Table 1. Additionally, the introduction of the commutative family (using permutation matrices) has been given a reasonable scrutiny but would benefit from further analysis. Finally, we believe that the continued study of optimization, improvement and cryptanalysis of MaTRU based on the previously proposed techniques used with the original NTRU, especially the impact of imperfect decryption [17], presents interesting challenges to explore.

Acknowledgement

We would like to thank Jeffrey Hoffstein for his advice and suggestions, Wee-Keong Lim for the discussion in matrix theory and anonymous referees for their constructive and detailed comments that have greatly improved this paper.

References

1. William D. Banks and Igor E. Shparlinski. A variant of NTRU with non-invertible polynomials. In *Proceeding of Indocrypt '02*, LNCS, vol. 2551, Springer-Verlag, pp.62-70, 2002.
2. D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Proceeding of Eurocrypt '97*, LNCS, vol. 1233, Springer-Verlag, pp.52-61, 1997.
3. J. Ding. A new variant of the Matsumoto-Imai cryptosystem through perturbation. In *Proceeding of PKC '04*, LNCS, vol. 2947, Springer-Verlag, pp.305-318, 2004.
4. W. Diffie and M.E. Hellman. New directions in cryptography. In *IEEE Trans. On Information Theory*, vol. 22, pp.644-654, 1976.
5. C. Gentry. Key recovery and message attacks on NTRU-composite. In *Proceeding of Eurocrypt '01*, LNCS, vol. 2045, Springer-Verlag, pp.182-194, 2001.
6. Daewan Han, Jin Hong, Jae Woo Han and Daesung Kwon. Key recovery attacks on NTRU without ciphertext validation routine. In *Proceeding of ACISP '03*, LNCS, vol. 2727, Springer-Verlag, pp.274-284, 2003.
7. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J.H. Silverman and W. Whyte. NTRUSign: Digital Signatures Using the NTRU Lattice. In *Proceeding of CT-RSA '03*, LNCS, vol. 2612, Springer-Verlag, pp.122-140, 2003.
8. J. Hoffstein, J. Pipher and J.H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In *Proceeding of ANTS III*, LNCS, vol. 1423, Springer-Verlag, pp. 267-288, 1998.
9. J. Hoffstein and J.H. Silverman. Optimizations for NTRU. In *Public-key Cryptography and Computational Number Theory*, DeGruyter, 2000. Available at [21].
10. J. Hoffstein and J.H. Silverman. Random small hamming weight products with applications to cryptography. Discrete Applied Mathematics, vol. 130, Issue 1 - special issue on the 2000 com2MaC workshop on cryptography, pp. 37 - 49, 2003. Available at [21].

11. E. Jaulmes and A. Joux. A Chosen Ciphertext Attack on NTRU. In *Proceeding of CRYPTO '00*, LNCS, vol. 1880, Springer-Verlag, pp. 20-35, 2000.
12. N. Koblitz. Elliptic curves cryptosystems. Math of Comp. vol. 48, pp. 203-209, 1987.
13. P. Karu and J. Loikkanen. Practical comparison of fast public-key cryptosystems. Seminar on Network Security, Telecommunications Software and Multimedia Laboratory, Kelsinki University of Technology. Available at http://www.tml.hut.fi/Opinnot/Tik-110.501/2000/papers.html.
14. T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Proceeding of Eurocrypt '88*, LNCS, vol. 330, Springer-Verlag, pp.419-453, 1988.
15. R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. JPL DSN Progress Report 42-44, pp. 114-116, 1978.
16. A. May and J.H. Silverman. Dimension Reduction Methods for Convolution Modular Lattices. In *Proceeding of CaCL '01*, LNCS, vol. 2146, Springer-Verlag, pp. 110-125, 2001.
17. N. Howgrave-Graham, P.Q. Nguyen, D. Pointcheval, J. Proos, J.H. Silverman, A. Singer and W. Whyte. The Impact of Decryption Failures on the Security of NTRU Encryption. In *Proceeding of CRYPTO '03*, LNCS, vol. 2729, Springer-Verlag, pp. 226-246, 2003.
18. N. Howgrave-Graham, J.H. Silverman, W. Whyte, *NTRU Cryptosystems Technical Report #004, Version 2: A Meet-In-The-Middle Attack on an NTRU Private Key*, `www.ntru.com`.
19. P.Q. Nguyen and D. Pointcheval. Analysis and Improvements of NTRU Encryption Paddings. In *Proceeding of CRYPTO '02*, LNCS, vol. 2442, Springer-Verlag, pp. 210-225, 2002.
20. P.Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptology. In *Proceeding of CaCL '01*, LNCS, vol. 2146, Springer-Verlag, pp. 148-180, 2001.
21. NTRU Cryptosystems. Technical reports available at http://www.ntru.com/cryptolab/tech_notes.htm.
22. R.L. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public key cryptosystem. *Communications of the ACM*, vol. 21, pp. 120-126, 1978.