A Study on NTRU Public Key Cryptosystem

Peroly Natesan
Department of Electrical & Computer Engineering,
Oregon State University, Corvallis, Oregon 97331 -USA.
E-mail: *natesan@engr.orst.edu*

Abstract— **This paper is a study on the new public key cryptosystems, NTRU. This paper gives an introduction to NTRU, its requirements, its advantage and its application. This paper throws light on how NTRU features reasonably short, easily created keys, high speed, and low memory requirements. The NTRU security which comes from the interaction of the polynomial mixing system with the independence of reduction modulo two relatively prime integers p and q is studied. Some of the methods that may be used to increase the speed and efficiency of the NTRU is also studied.**

KEYWORDS: NTRU, RING-BASED PUBLIC KEY CRYPTOSYSTEM, SECURITY ANALYSIS, LATTICE ATTACK

I. Introduction

With the exponential growth of computing power and electronic system, there has been considerable interest and research going on in the creation of efficient and computationally inexpensive public key cryptosystems since Diffie and Hellman [1] explained how such systems could be created using one-way functions. The most widely used public key system currently is RSA, created by Rivest, Shamir and Adelman in 1978 [4] and is based on the difficulty of factoring large numbers. Other cryptosystems include McEliece system [2] which relies on error correcting codes. Another recent system of Goldreich, Goldwasser, and Halevi [3] is based on the difficulty of lattice reduction problems.

The NTRU Public Key Cryptosystem is based on ring theory [5] and relies for its security on the difficulty of solving certain lattice problems. A ring is a mathematical object which has two algebraic operations, addition and multiplication.

In NTRU, the encryption procedure uses a mixing system based on polynomial algebra and reduction modulo two numbers p and q. The decryption procedure uses an unmixing system whose validity depends on elementary probability theory.

The two main reason for the security of NTRU public key cryptosystem are
1. The security comes from the interaction of the polynomial mixing system with the independence of reduction modulo p and q.
2. It is very difficult to find extremely short vectors for most lattices.

Also NTRU fits into the general framework of a probabilistic cryptosystem which means that encryption includes a random element, so each message has many possible encryptions. One of the major advantage of NTRU is Encryption and decryption are extremely fast, and key creation is fast and easy.

Comparison of NTRU and RSA Speed and Key Length: NTRU takes $O(N^2)$ operations to encrypt or decrypt a message block of length N, making it considerably faster than the $O(N^3)$ operations required by RSA. NTRU key lengths are $O(N)$, which is better than $O(N^2)$ key lengths as required by other "fast" public keys systems such as [2],[3].

Author is a graduate student at the Department of Electrical & Computer Engineering, Oregon State University, Corvallis, Oregon 97331. E-mail: *natesan@engr.orst.edu*

II. Description of NTRU algorithm

A. An Overview of NTRU

A general formulation of the NTRU Public Key Cryptosystem uses a ring R and two (relatively prime) ideals p and q in R. A rough outline of the key creation, encryption, and decryption processes is as follows:

- Key Creation

Bob creates a public key h by choosing elements $f, g \in R$, computing the mod q inverse f_q^{-1} of f, and setting
$h \equiv f_q^{-1} * g(modq)$.

Bob's private key is the element f. Bob also precomputes and stores the mod p inverse f_q^{-1} of f.

- Encryption

In order to encrypt a plaintext message $m \in R$ using the public key h, Alice selects a random element $r \in R$ and forms the ciphertext
$e \equiv r * h + m(modq)$.

- Decryption

In order to decrypt the ciphertext e using the private key f, Bob first computes
$a \equiv f * e(modq)$.
He chooses $a \in R$ to satisfy this congruence and to lie in a certain prespecified subset Ra of R. He next does the mod p calculation
$f_q^{-1} * a(modp)$,
and the value he computes is equal to m modulo p.

III. An Example Implemented Using MATLAB

An Example that was implemented and verified using MATLAB is given below

NTRU ALGORITHM
$N = 11 \quad p = 3 \quad q = 32$

KEY GENERATION:

• Bob first randomly chooses two "small" polynomials f(x) and g(x) in the ring of truncated polynomials R.

• For the purposes of this section, we take $d_f = 4$ $d_g = 3$.

$f = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$

$g = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$

• Compute the inverse of f modulo q and the inverse of f modulo p.

$f * f_q = 1(moduloq)$ and $f * f_p = 1(modulop)$

$f_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9(modulo3)$

$f_q = 5 + 9X + 6X^2 + 16X^3 + 4X^4 + 15X^5 + 16X^6 + 22X^7 + 20X^8 + 18X^9 + 30X^10(modulo32)$

• Bob computes the Public Key

$h = pf_q * g(modulo32)$

$h = 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 + 19X^7 + 12X^8 + 19X^9 + 16X^10(modulo32)$

ENCRYPTION:

• Alice uses the message m, her randomly chosen polynomial r, and Bob's public key h to Compute the polynomial

$e = r * h + m(moduloq)$

The polynomial e is the encrypted message which Alice sends to Bob.

– Blinding value $rdr = 3$

$r = -1 + X^2 + X^3 + X^4 - X^5 - X^7$

– Now, suppose Alice wants to send the message

$m = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$

– Encrypted message e is

$e = r * h + m(moduloq)$

$e = 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10}(modulo32)$.

DECRYPTION:

– Bob uses his private polynomial f to compute the polynomial

$a = f * e(moduloq)$.

$a = 3 - 7X - 10X^2 - 11X^3 + 10X^4 + 7X^5 + 6X^6 + 7X^7 + 5X^8 - 3X^9 - 7X^{10}(modulo32)$.

– Next Bob computes

$b = a(modulop)$.

$b = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10}(modulo3)$.

– Bob uses f_p, the other part of his private key, to compute

$c = f_p * b(modulop)$

$c = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}(modulo3)$

$m = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}(modulo3)$

IV. STANDARDS

The IEEE P1363 project develops Standard Specifications For Public-Key Cryptography, towards the goal of issuing a series of IEEE standards documents. The core NTRU algorithms are currently being standarized in the ieee P1363 wroking group[8]. IEEE P1363.1 will specify cryptographic techniques based on hard problems over lattices. These techniques may offer tradeoffs in operating characteristics when compared with the methods already specified in IEEE 1363-2000 and draft P1363a. The purpose is to provide: (1) a reference for specification of a variety of techniques from which applications may select, (2) the relevant number-theoretic background, and (3) extensive discussion of security and implementation considerations so that a solution provider can choose appropriate security requirements for itself.

V. COMPARISION OF NTRU WITH OTHER SYSTEMS

• Reason for NTRUEncrypt much faster than RSA, El Gamal, and ECC:

The NTREncrypt cryptosystem is much faster than exponentiation systems such as RSA, El Gamal, and ECC. One reason is that the basic operations used by NTRUEncrypt involve manipulation of small numbers, generally numbers less than 255. Exponentiation systems, on the other hand, require numbers with hundreds of digits. A private key for any cryptosystem can be thought of as a long list of bits, for example key = (1,1,0,1,0,0,1,0,0,1,1,1,1,0,0,0,1,0,1,0, ...,0,0,1,0,1,1,1,0,0,1,0,1). The number of 0's and 1's in the list is called the bit-length of the key. A typical key for NTRUEncrypt, RSA, or El Gamal might be between 1000 and 2000 bits long, so it's not a difference in key lengths that makes NTRUEncrypt faster. ECC uses even shorter keys. Instead, its how NTRUEncrypt uses the key. When RSA, El Gamal, or ECC want to encrypt a message, they need to do lots of computations involving every single bit in the key; so they need to manipulate the entire long key. By way of contrast, NTRUEncrypt breaks the key up into small chunks, generally consisting of 7 or 8 bits each. For example, NTRUEncrypt key = (1,1,0,1,0,0,1,0) — (0,1,1,1,1,0,0,0,) —....— (1,1,1,0,0,1,0,1). When encrypting a message, NTRUEncrypt works with the key chunks one at a time, so it never needs to manipulate extremely long lists of bits. This makes for speedy computations, even on low power processors. A careful mathematical analysis shows that for keys consisting of around N bits, the RSA, El Gamal, and ECC systems require on the order of N3 operations to encrypt or decrypt a message, while NTRUEncrypt requires only on the order of N2 operations to encrypt or decrypt a message.

• Performance figures for NTRU Encryption, Decryption, Signing, Verification and Key Generation relative to RSA and ECC (for equivalent of 1024 and 2048 RSA):

Here are the benchmarks for NTRU-251 against RSA 1024 and ECC over GF ($2^1$63):

TABLE I

NTRU-251 v RSA-1024 on 800 MHz Pentium III:

	NTRU	*RSA*	*NTRU*	*Advantage*
Encrypt	Blocks/sec	22727	1280	17 to 1
Decrypt	Blocks/sec	10869	110	99 to 1

In the real world situation, a single application of the public key cryptosystem consists of three operations:

Generate Key / Encrypt One Block / Decrypt One Block

If we compare NTRUEncrypt N=263 with RSA 1024 in this situation, we find that NTRUEncrypt is far more that 100 times faster. Indeed, NTRUEncrypt key pair generation takes about 3.0 milliseconds, while Crypto++ takes over 1 second to generate an RSA key pair (on our 300 MHz machines), so the NTRUEncrypt speed advantage tops 300. The following table gives timing comparisons for NTRUEncrypt and RSA at comparable security levels.

System	*Blocks/Second*	*NTRUEn vs RSA*
NTRUEncrypt 263	645.6	27.6 times faster
RSA 1024	23.4	27.6 times faster
NTRUEncrypt 503	246.8	68.6 times faster
RSA 2048	3.6	68.6 times faster

VI. SECURITY ANALYSIS

An NTRU cryptosystem depends upon three integer parameters (N,p,q) and four sets L_f, L_g, L_ϕ, L_m of polynomials of degree N-1 with integer coefficients. We work in the ring $R = Z[X]/(X^N - 1)$. The paper describes the following possible attacks and the security of NTRU,

1. Brute force attack: Under Brute force attack, An attacker can recover the private key by trying all possible $f \in L_f$ and testing if $f * h^{-1}(modq)$ has small entries. Similarly, an attacker can recover a message by trying all possible $\phi \in L_\phi$ and testing if $e - \phi * h(modq)$ has small entries. The security level is given by

$$(keysecurity) = \sqrt{\#L_g} = \frac{1}{d_g}\sqrt{\frac{N}{(N-2d_g)}}$$

$$(Messagesecurity) = \sqrt{\#L_\phi} = \frac{1}{d}\sqrt{\frac{N}{(N-2d)}}$$

2. Meet-in-the-middle attacks. An encrypted message looks like $e \equiv \phi * h + m(modq)$. If one splits f in half, say $f = f_1 + f_2$, and then one matches $f1 * e$ against $-f_2 * e$, looking for (f_1, f_2) so that the corresponding coefficients have approximately the same value. Hence, in order to obtain a security level of 2^{80}, one must choose f, g and ϕ from sets containing around 2^{160} elements.

3. Lattice Based attack: Consider a 2N-by-2N matrix composed of four N-by-N blocks. Let L be the lattice generated by the rows of this matrix. The determinant of L is $q^N\alpha^N$. Since, the public key is $h = g * f^{-1}$, the lattice L will contain the vector $\tau = (\alpha f, g)$, by which we mean the $2N$ vector consisting of the N coefficients of f multiplied by α, followed by the N coefficients of g.

An implementation of a lattice reduction algorithm will have the best chance of locating τ, or another vector whose length is close to τ, if the attacker chooses α to maximize the ration $s\tau_2$.

c_h is the ratio of length of the target vector to the length of the expected shortest vector and is given by

$$c_h = \sqrt{\frac{2\pi e \pmod{f_2} \pmod{g_2}}{N_q}}$$

c_h may be viewed as a measure of how far the associated lattice departs from a random lattice, for a given pair (f, g) used to set up the cryptosystem.

Another important constant c_m gives a measure of the vulnerability of an individual message to a lattice attack.

$$c_m = \sqrt{\frac{2\pi e \pmod{m_2} \pmod{\phi_2}}{N_q}}$$

As c_h and c_m becomes closer to 1, the vulnerability of attack for private key and message becomes very less. The authors also made c_m approximately equal to c_h, to make attacks on h and m equally difficult.

Measurement of security of NTRUEncrypt:

The hard problem underlying the NTRUEncrypt Public Key Cryptosystem is that of finding a very short vector in a lattice of very high dimension. The best way known to attack NTRUEncrypt is to use the LLL lattice reduction method to search for the target vector. In order to test the security of NTRUEncrypt, they used LLL to run numerous tests on NTRUEncrypt lattices of various dimensions and graphed the amount of time it took to find the target vector. From this graph they extrapolated the running time for lattices of higher dimension and used these figures to select appropriate parameters for NTRUEncrypt. The following table gives estimated breaking times for NTRUEncrypt and RSA at various security levels. (Times are rounded to the nearest 10 MIPS-years.)

Cryptosystem	*Security Level*	*EBT*
RSA	512 bits	10^5 MIPS yrs
NTRUEncrypt	N = 167	10^6 MIPS yrs
RSA	1024 bits	10^{12} MIPS yrs
NTRUEncrypt	N = 263	10^{14} MIPS yrs
RSA	4096 bits	10^{33} MIPS yrs
NTRUEncrypt	N = 503	10^{35} MIPS yrs

VII. Practical implementation of NTRU

The authors presented three distinct sets of parameters which yielded three different levels of security.

Case A: Moderate security This case is suitable for situations in which the intrinsic value of any individual message is small and in which keys will be changed with reasonable frequency. The examples are television, pager, and cellular transmissions. $(N, p, q) = (107, 3, 64)$ $L_f = L(15, 14), L_g = L(12, 12), L_\phi = L(5, 5)(i.e., d = 5)$.

The above things give key sizes Private Key = 340 bits and Public Key = 642 bits,

And security levels $Keysecurity = 2^{50}$ and $MessageSecurity = 2^{26.5}$

Substituting the above values into the appropriate formulas yields lattice values $C_h = 0.257, c_m = 0.258$, and $s = 0.422q$.

Case B:

$(N, p, q) = (167, 3, 128)$

$L_f = L(61, 60), L_g = L(20, 20), L_\phi = L(18, 18)(i.e., d = 18)$.

The above things give key sizes Private Key = 530 bits and Public Key = 1169 bits,

And security levels Key security = $2^{82.9}$ and Message Security = $2^{77.5}$

Substituting the above values into the appropriate formulas yields lattice values $C_h = 0.236, c_m = 0.225$, and $s = 0.296q$.

Case C:

$(N, p, q) = (503, 3, 256)$

$L_f = L(216, 215), L_g = L(72, 72), L_\phi = L(55, 55)(i.e., d = 55)$.

The above things give key sizes Private Key = 1595 bits and Public Key = 4024 bits,

And security levels Key security = 2^{285} and Message Security = 2^{170}

Substituting the above values into the appropriate formulas yields lattice values $c_h = 0.182, c_m = 0.160$, and $s = 0.365q$.

Optimization of NTRU:

The following optimization methods were used to increase the speed and efficiency of the NTRU PKCS[6],

1. To guard against the chosen ciphertext attacks padding techniques of Fujisaki and Okamoto and scrambling of message is done.
2. The most time consuming part of NTRU encryption process involves one product $r * hmodq$ and decryption requires computation of two products $f * emodq$ and $f_p^{-1} * amodp$ and key creation process involves computation of the inverses f_p^{-1} and f_q^{-1}. If we choose the element f to have the form $f = 1 + P * f_1$ with $f_1 \in R$, then no need to compute inverse modulo p and second computation in decryption process also disappears as an f of this form has the property $f_p^{-1} = 1$.
3. Through the use of Low Hamming weight polynomials the encryption and decryption can be speeded up [7].

VIII. Application

The growth of consumer and wireless networking computing devices is set to explode over the next decade as consumers and enterprises increasingly demand access to previously desktop-bound applications and services across a wide range of handheld devices. Consumers will use smart cards, mobile phones, RFID tags/labels, and wireless networking devices on a regular basis for payments, digital media download and playback, home networking, and securing physical and logical access. Secure communications and trusted devices are a fundamental building block for these applications and services. NTRU security is designed to meet the clear and fundamental need for small, fast, and power-efficient security technology for all of tomorrow's mobile computing devices, including:

- Smart Cards
- RFID tags and labels
- Mobile phones
- Wireless networking products, including 802.11 and 802.15 devices
- Media players
- PDAs

IX. CONCLUSION

Thus as observed NTRU is Easy to program, Easy to build into hardware, Ideal for Digital Signal Processors (DSPs) and NTRU Requires Less memory (RAM) in software,Less storage in software, Fewer gates in hardware. NTRU easily fits into Low power smart cards, Handheld devices, Cellular telephones, Set top boxes. As seen from the section, comparison of NTRU with other current systems, NTRU has great advantage of being Fast and easy key generation and security is also superior then another comparable systems.

References

[1] W. Diffie, M.E. Hellman "New directions in cryptography," in , *IEEE Trans. On Information Theory 22 (1976), 644-654*

[2] R.J. McEliece "A public-key cryptosystem based on algebraic coding theory," in *,JPL Pasadena, DSN Progress Reports 43-44 (1978), 114-116*

[3] O. Goldreich, S. Goldwasser, S. Halevi "Public-key cryptosystems from lattice reduction problems," in , *MIT - Laboratory for Computer Science preprint, November 1996*

[4] R. L. Rivest, A. Shamir, L. Adleman "A method for obtaining digital signatures and public key cryptosystems," in , *Communications of ACM 21 (1978), 120-126.*

[5] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman 'NTRU: A Ring-Based Public Key Cryptosystem," in *,Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.),*

[6] J. Hoffstein, J. Silverman "Optimizations for NTRU," in , *Public-Key Cryptography and Computational Number Theory (Warsaw, September 11-15, 2000)*

[7] J. Hoffstein, J. Silverman "Random Small Hamming Weight Products With Applications to Cryptography,"

[8] IEEE P1363 Working Group "http://grouper.ieee.org/groups/1363/,"