



Research Experience for Undergraduates, Summer
2004

Presented by: Chau Ngo–Thu Trinh

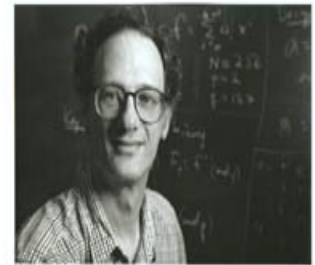
NTRU Public Key Cryptosystem

Part I

- History of the NTRU Public Key Cryptosystem
- How NTRU Public Key Cryptosystem (PKCS) Works
 - NTRU PKCS Parameters
 - NTRU Key Generation
 - NTRU Encryption
 - NTRU Decryption
 - Why NTRU Works
- Special Techniques to Speed Up NTRU Operations

Brief History of NTRU

- Invented and developed by three Brown University mathematicians: J. Hoffstein, J. Pipher, and J. Silverman
- Presented by Hoffstein at the Crypto '96 conference



NTRU PKCS Parameters

- NTRU Multiplication
- Small polynomials and polynomials mod q
- Public parameters

NTRU Multiplication

- NTRU uses polynomial of degree $N-1$ with integer coefficients (including 0's):

$$\mathbf{a} = a_0 + a_1X + a_2X^2 + a_3X^3 + \dots + a_{N-2}X^{N-2} + a_{N-1}X^{N-1}$$

- Polynomials are added in the usual way

$$\mathbf{a} + \mathbf{b} = (a_0+b_0) + (a_1+b_1)X + \dots + (a_{N-1}+b_{N-1})X^{N-1}$$

- The multiplication of \mathbf{a} and \mathbf{b} with

$$\mathbf{a} = a_0 + a_1X + a_2X^2 + a_3X^3 + \dots + a_{N-2}X^{N-2} + a_{N-1}X^{N-1}$$

$$\mathbf{b} = b_0 + b_1X + b_2X^2 + b_3X^3 + \dots + b_{N-2}X^{N-2} + b_{N-1}X^{N-1}$$

is

$$\mathbf{a}^*\mathbf{b} = c_0 + c_1X + c_2X^2 + c_3X^3 + \dots + c_{N-2}X^{N-2} + c_{N-1}X^{N-1}$$

where

$$c_k = a_0b_k + a_1b_{k-1} + \dots + a_kb_0 + a_{k+1}b_{N-1} + a_{k+1}b_{N-2} + \dots$$

$$a_{N-1}b_{k+1}$$

NTRU Key Generation

Bob wants to create a Public/Private Key pair for the NTRU PKCS.

- 1) Bob randomly chooses 2 small polynomials $f(x)$ and $g(x)$.
- 2) Bob computes the inverses of f modulo q and modulo p . Thus he computes polynomials f_q and f_p with the property that

$$f * f_q = 1 \text{ (modulo } q) \text{ and } f * f_p = 1 \text{ (modulo } p).$$

(Bob uses the Euclidean Algorithm to compute these inverses).

- 3) Bob computes the product

$$h = p * f_q * g \text{ (modulo } q).$$

- 4) Bob's Private Key: the pair of polynomials f and f_p .
Bob's Public Key: the polynomial h .

Key Generation Example

$$N = 11, \mathbf{q} = 32, \mathbf{p} = 3$$

Bob chooses

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

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

Bob computes the inverse f_p of \mathbf{f} modulo \mathbf{p} and the inverse f_q of \mathbf{f} modulo

\mathbf{q} .

$$\mathbf{f}_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$$

$$\mathbf{f}_q = 5 + 9X + 6X^2 + 16X^3 + 4X^4 + 15X^5 + 16X^6 + 22X^7 + 20X^8 + 18X^9 + 30X^{10}$$

Bob computes

$$\mathbf{h} = \mathbf{p} * \mathbf{f}_q * \mathbf{g} \text{ (modulo } \mathbf{q}\text{)}$$

$$= 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 + 19X^7 + 12X^8 + 19X^9 + 16X^{10} \text{ (modulo } 32\text{)}.$$

Bob's Private Key: the pair of polynomials \mathbf{f} and \mathbf{f}_p

Bob's Public Key: the polynomial \mathbf{h} .

NTRU Encryption

Alice wants to send a message to Bob using Bob's public key $h(x)$.

- 1) Alice first puts her message in the form of a small polynomial $m(x)$ mod p .
- 2) Alice randomly chooses another small polynomial $r(x)$.
- 3) Alice uses $m(x)$, $r(x)$, and Bob's Public Key $h(x)$ to compute the polynomial

$$e = r * h + m \text{ (modulo } q\text{)}.$$

Alice encrypted message: $e(x)$.

- 4) Alice sends the encrypted message $e(x)$ to Bob.

NTRU Encryption Example

Alice's message in form of \mathbf{m}

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

Alice randomly chooses blinding value \mathbf{r}

$$\mathbf{r} = -1 + X^2 + X^3 + X^4 - X^5 - X^7$$

Alice uses \mathbf{m} , \mathbf{r} , and Bob's Public Key \mathbf{h} to compute

$$\mathbf{e} = \mathbf{r} * \mathbf{h} + \mathbf{m} \text{ (modulo } \mathbf{q})$$

$$= 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10} \\ \text{(modulo 32).}$$

Alice's encrypted message: the polynomial \mathbf{e} .

NTRU Decryption

Bob has received Alice's encrypted message $e(x)$ and he wants to decrypt it.

1) Bob uses his Private Key $f(x)$ to compute

$$\mathbf{a} = \mathbf{f} * \mathbf{e} \text{ (modulo } q\text{)}.$$

2) Bob chooses coefficients of $\mathbf{a}(x)$ lie between $-q/2$ and $q/2$.

3) Bob next reduces each of the coefficients of the polynomial $\mathbf{a}(x)$ mod p

$$\mathbf{b} = \mathbf{a} \text{ (modulo } p\text{)}.$$

4) Finally Bob uses his other Private Key polynomial $\mathbf{f}_p(x)$ to compute

$$\mathbf{c} = \mathbf{f}_p * \mathbf{b} \text{ (modulo } p\text{)}.$$

The polynomial \mathbf{c} will be the same as Alice's original message \mathbf{m} .

NTRU Decryption Example

Bob has received from Alice the encrypted message

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

Bob uses his private key \mathbf{f} and chooses values between $[-15, 16]$ to compute

$$\begin{aligned} \mathbf{a} &= \mathbf{f} * \mathbf{e} \\ &= 3 - 7X - 10X^2 - 11X^3 + 10X^4 + 7X^5 + 6X^6 + 7X^7 + 5X^8 - 3X^9 - 7X^{10} \\ &\text{(modulo 32)}. \end{aligned}$$

Next Bob reduces the coefficients of \mathbf{a} modulo \mathbf{p} to get

$$\begin{aligned} \mathbf{b} &= \mathbf{a} \text{ (modulo } \mathbf{p}) \\ &= -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \text{ (modulo 3)}. \end{aligned}$$

Finally Bob uses \mathbf{f}_p , the other part of his private key, to compute

$$\mathbf{c} = \mathbf{f}_p * \mathbf{b} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10} \text{ (modulo 3)}.$$

The polynomial \mathbf{c} is Alice's message \mathbf{m} .

So Bob has successfully decrypted Alice's message.

Why NTRU Works

$$\begin{aligned}
 \mathbf{a} &= \mathbf{f} * \mathbf{e} && (\text{mod } \mathbf{q}) && \mathbf{f} \text{ is one of Bob's Private Key} \\
 &= \mathbf{f} * (\mathbf{r} * \mathbf{h} + \mathbf{m}) && (\text{mod } \mathbf{q}) && \mathbf{r} \text{ is Alice's message blinding value} \\
 &= \mathbf{f} * (\mathbf{r} * \mathbf{p} * \mathbf{f}_q * \mathbf{g} + \mathbf{m}) && (\text{mod } \mathbf{q}) && \mathbf{p} * \mathbf{g} * \mathbf{f}_q \text{ is Bob's Public Key} \\
 &= \mathbf{r} * \mathbf{p} * \mathbf{g} + \mathbf{f} * \mathbf{m} && (\text{mod } \mathbf{q}) && \text{since } \mathbf{f} * \mathbf{f}_q = 1(\text{mod } \mathbf{q}).
 \end{aligned}$$

The coefficients of polynomials \mathbf{r} , \mathbf{g} , \mathbf{f} , and \mathbf{m} are all quite small.

The prime \mathbf{p} is also small compared to \mathbf{q} .

This means the coefficients of the product polynomial

$$\mathbf{r} * \mathbf{p} * \mathbf{g} + \mathbf{f} * \mathbf{m}$$

are also quite small compared to \mathbf{q} , so reducing coefficients to mod \mathbf{q} has no effects at all. Therefore,

$$\mathbf{a} = \mathbf{r} * \mathbf{p} * \mathbf{g} + \mathbf{f} * \mathbf{m}.$$

The NTRU Public Key Cryptosystem

Part II

- Review
- Choosing the form of f
- Taking $p = 2 + X$
- Centering the polynomial a
- An example

Review

- f** A small polynomial, part of the Private Key.
- f_p** The inverse of **f** mod **p**.
- f_q** The inverse of **f** mod **q**.
- g** A small polynomial, used in generating the Public Key.
- h** The public key. $\mathbf{h} = \mathbf{f}_q * \mathbf{g} \text{ mod } \mathbf{q}$.
- m** The message, a small polynomial.
- r** The random blinding value, used when encrypting. A small polynomial.
- e** The encrypted message.
- a** The partially decrypted message. $\mathbf{a} = \mathbf{f} * \mathbf{e} \text{ mod } \mathbf{q}$.

Choosing the form of f

Properties of f :

- f is invertible mod p .
- f is invertible mod q .
- f is small.

In commercial applications, we use an alternative way of choosing f .

We take

$$f = 1 + p * F,$$

where F is a small polynomial.

This choice means that

$$f = 1 \pmod{p}$$

Advantages in choosing f

- Speeds up key generation
 f is always invertible mod p (in fact, $f^{-1} = 1 \pmod{p}$), we don't have to explicitly calculate the inverse mod p .
- Speeds up decryption considerably
Since $f^{-1} = 1 \pmod{p}$, we no longer have to compute
$$c = f_p * b \pmod{p}$$
when decrypting. So only one multiplication
$$b = a \pmod{p}$$
 where
$$a = f * e \pmod{q}.$$
is needed, not two.
- It also means that we don't need $f_p = f^{-1} \pmod{p}$ as part of the Private Key.

Taking $\mathbf{p} = 2 + X$

- Since all polynomials \mathbf{r} , \mathbf{g} , \mathbf{f} , and \mathbf{m} are small polynomials with small coefficients and \mathbf{p} is always chosen to be small (all compared to \mathbf{q}), the success of decryption depends on the coefficients of \mathbf{a} where

$$\mathbf{a} = (\mathbf{r} * \mathbf{p} * \mathbf{g} + \mathbf{f} * \mathbf{m}) \text{ (modulo } \mathbf{q}\text{)}$$

being unchanged when reduced modulo \mathbf{q} .

- So if we can reduce the size of \mathbf{p} , we make it easier to pick parameter sets such that decryption will succeed.
- \mathbf{p} and \mathbf{q} are relatively prime. Therefore, we may take \mathbf{p} to be a small polynomial, such as

$$\mathbf{p} = 2 + X.$$

- It makes a little more complicated to recover the message polynomial \mathbf{m} from its value modulo \mathbf{p} .

Centering the Polynomial a

- When Bob decrypts, he computes the following values:

$$\begin{aligned}
 \mathbf{a} &= \mathbf{f} * \mathbf{e} && \text{(modulo } \mathbf{q}) \\
 &= \mathbf{f} * (\mathbf{r} * \mathbf{h} + \mathbf{m}) && \text{(modulo } \mathbf{q}) \quad \text{since } \mathbf{e} = \mathbf{r} * \mathbf{h} + \mathbf{m} \\
 &\text{(modulo } \mathbf{q}) \\
 &= \mathbf{f} * (\mathbf{r} * \mathbf{p} * \mathbf{f}_q * \mathbf{g} + \mathbf{m}) && \text{(modulo } \mathbf{q}) \quad \text{since } \mathbf{h} = \mathbf{p} * \mathbf{f}_q * \mathbf{g} \text{ (modulo } \mathbf{q}) \\
 &= \mathbf{p} * \mathbf{r} * \mathbf{g} + \mathbf{f} * \mathbf{m} && \text{(modulo } \mathbf{q}) \quad \text{since } \mathbf{f} * \mathbf{f}_q = 1 \text{ (modulo } \mathbf{q})
 \end{aligned}$$

- \mathbf{r} , \mathbf{g} and \mathbf{m} are centered around zero (in that they have equal numbers of +1s and -1s), and \mathbf{f} is nearly centered around zero (in that \mathbf{f} had one more +1 than -1). Another way of putting this is to say that

$$\begin{aligned}
 \mathbf{r}(1) &= \mathbf{g}(1) = \mathbf{m}(1) = 0; \\
 \mathbf{f}(1) &= 1.
 \end{aligned}$$

Centering the Polynomial a

- Now that we're using $\mathbf{f} = 1 + \mathbf{p} * \mathbf{F}$, the values of \mathbf{r} , \mathbf{g} , \mathbf{m} and \mathbf{f} are no longer centered at zero (their coefficients won't lie within the specific range $(-q/2, q/2)$).
- So before carrying out the reduction modulo \mathbf{p} when decrypting, we have to find what the true range for the coefficients of $\mathbf{p} * \mathbf{r} * \mathbf{g} + \mathbf{f} * \mathbf{m}$. We don't know how many 1s and 0s there are in \mathbf{m} before we decrypt it; but we can extract it from \mathbf{a} using the following method.
 1. Set $\mathbf{I} = \mathbf{f}_q(1) \cdot (\mathbf{a}(1) - \mathbf{p}(1) \cdot \mathbf{r}(1) \cdot \mathbf{g}(1)) \pmod{\mathbf{q}}$
 2. Set $\mathbf{Avg} = (\mathbf{p}(1) \cdot \mathbf{r}(1) \cdot \mathbf{g}(1) + \mathbf{f}(1)) / \mathbf{N}$
 3. The expected range of the coefficients will be between $\mathbf{Avg} - \mathbf{q}/2$ and $\mathbf{Avg} + \mathbf{q}/2$. \mathbf{Avg} will generally not be an integer, so the actual expected range will be the \mathbf{q} integers that lie between $\mathbf{Avg} - \mathbf{q}/2$ and $\mathbf{Avg} + \mathbf{q}/2$.

Advanced Topic Example



Ntrū

The NTRU Public Key Cryptosystem

Part III

- Review
- Low Hamming Weight Polynomials
- Products of Low Hamming Weight Polynomials
- Security Considerations

Review

- The NTRU Cryptosystem is parameterized by three values \mathbf{N} , \mathbf{p} , and \mathbf{q} .
- Multiplications of polynomials use the convolution product rule.
- \mathbf{p} and \mathbf{q} are moduli; multiplications and additions are generally followed by reduction modulo \mathbf{p} or modulo \mathbf{q} .
- The most time-consuming operations in the NTRU cryptosystem are the convolution multiplications.

Low Hamming Weight Polynomials

- Instead of writing $X + X^3 + X^5 + X^8 + X^9 + X^{10}$ we'll represent the polynomial in vector form $[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1]$.
- A small example of convolution multiplication for polynomials of degree 3.

The convolution product

$$\begin{aligned}
 [4, 5, 7] * [5, 3, 2] &= 4 \cdot [5, 3, 2] \\
 &\quad + 5 \cdot [2, 5, 3] \\
 &\quad + 7 \cdot [3, 2, 5] \\
 &= [20, 12, 8] \\
 &\quad + [10, 25, 15] \\
 &\quad + [21, 14, 35] \\
 &= [20+10+21, 12+25+14, 8+15+35] \\
 &= [51, 51, 58].
 \end{aligned}$$

- So this requires nine multiplies and six adds.

Low Hamming Weight Polynomials (continued)

- Now let's look at what happens if one of the polynomials has only 1s and 0s:

$$\begin{aligned} [1, 0, 0, 1, 0] * [5, 3, 2, 9, 10] &= 1 * [5, 3, 2, 9, 10] \\ &\quad + 0 * [10, 5, 3, 2, 9] \\ &\quad + 0 * [9, 10, 5, 3, 2] \\ &\quad + 1 * [2, 9, 10, 5, 3] \\ &\quad + 0 * [3, 2, 9, 10, 5] \\ &= 1 * [5, 3, 2, 9, 10] \\ &\quad + 1 * [2, 9, 10, 5, 3] \\ &= [5+2, 3+9, 2+10, 9+5, 10+3] \\ &= [7, 12, 12, 14, 13] \end{aligned}$$

- This requires no multiplications at all, and each coefficient in the result is simply the sum of two numbers.

Low Hamming Weight Polynomials (continued)

- In general, we can say that if \mathbf{i} is a small polynomial with d_i coefficients equal to 1 and the rest equal to zero, and if \mathbf{a} is an arbitrary polynomial, then each coefficient of the product ($\mathbf{i} * \mathbf{a}$) is obtained by summing d_i of the coefficients of \mathbf{a} .

- Using the previous example we have

$$\mathbf{i} = [1, 0, 0, 1, 0]$$

$$\mathbf{a} = [5, 3, 2, 9, 10]$$

$$\mathbf{i} * \mathbf{a} = [5, 3, 2, 9, 10]$$

$$+ [9, 10, 5, 3, 2]$$

$$= [5+2, 3+9, 2+10, 9+5, 10+3]$$

$$= [7, 12, 12, 14, 13].$$

Security Considerations

Selections of security parameters:

- Dimension N
- Large modulus q
- Small modulus p

Other selections:

- Target vectors
- Private keys
- Encryption blinding elements and message elements
- Generation of private keys and blinding elements using small Hamming weight products

Selection of Dimension Security Parameter N

- N is required to be prime

Selection of Large Modulus q

- For each prime divisor q_0 of q , the polynomial $X^N - 1$ modulo q_0 should have no factors of small degree (aside from the obvious factor $X - 1$).
- If N is prime, then $X^N - 1$ modulo q_0 factors as $(X - 1)A_1(X)\dots A_e(X)$, where each $A_i(X)$ has degree equal to the multiplicative order of q_0 modulo N . The following table lists some security parameters satisfying this recommendation.

N	q	q_0	Order(q_0 mod N)
167	64	2	83
251	128	2	50
263	128	2	131
347	128	2	346
503	256	2	251

Selection of Small Modulus p

- If the small modulus p divides the large modulus q , then reduction modulo p of an expression $p^*r^*h + m$ modulo q will immediately recover m .
- More generally, if p and q are not relatively prime, then reduction modulo a common factor will reveal information about m . For this reason it is required that the large modulus q and the small modulus p be relatively prime. This is equivalent to the condition that the three quantities q , p , and $X^N - 1$ must generate the unit ideal in the ring $\mathbf{Z}[X]$.
- p needs not to be an integer.



NTRU parameters for various levels of security

These parameters are purported to be used in NTRU Cryptosystems commercial applications.

	N	q	p	d_f	d_g
Low Security	107	64	3	15	5
Moderate Security	167	128	3	61	18
High Security	263	128	3	50	16
Highest Security	503	256	3	216	55

References

- Ntru Cryptosystems, The NTRU Public Key Cryptosystem : Basic Tutorial
- Ntru Cryptosystems, The NTRU Public Key Cryptosystem : Advanced Topics
- Ntru Cryptosystems, The NTRU Public Key Cryptosystem : Further Topics in Fast Implementation
- IEEE P1363.1 /D2 Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices
- Ntru Cryptosystem, Implementation and Comparative Analysis