**Lecture 10: NTRU Cryptosystem**

**LEARNING OUTCOME**

**By the end of this lesson a student will be able to:**

1. understand a concept of an NTRU cryptosystem
2. compute in input in terms of a polynomial over Fp.
3. encrypt a plaintexet and decrypt a ciphertext given a public key via NTRU cryptosystem

In general, a public key infrastructure(PKI) cryptosystem is operating on at least 2 domain rings. For instance, RSA operates on a ring modulo N. At the same time, there are 2 more smaller rings over prime P and Q fields. A ring modulo N is public and a ring modulo prime P is private.

a popular ECC is not operating in this mode. It follows a one way function.

On one hand, RSA is operating on integers. On the other hand, there are 2 ECC modes. An ECC over prime P and another operates on finite field over an irreducible polynomial. Next, NTRU will make use of polynomial prowes.

In terms of popularity, we will give an estimate RSA~60-70%, ECC~20% and NTRU~10%. A good analogy here, Microsoft Windows 60-70%, Apple 20-30% and Linux~10% but now Android is on the way.

In term of efficiency, RSA is slow, ECC is faster but NTRU is the fastest. In term of running time RSA take O(*n*3) due to a power mod operation, ECC takes O(*n*3) due to point projection but NTRU takes O(*n*2) due to multiply modulo operation. Ring size varies but popular size is RSA~2048-bit, ECC~256-bit but NTRU~6400-bit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CryptoSystem | Year | Bit Size | Speed | Running Time | Market Popularity | IP Control | Post Quantum |
| NTRU | 1995 | 6400 | Fastest | O(*n*2) | 5-10% | Open | Immune |
| ECC | 1985 | 256 | Faster | O(*n*3) | 20-30% | Strong | Vulnerable |
| RSA | 1978 | 2048 | Fast | O(*n*3) | 60-70% | Standard | Breakable |

The popular NTRU public-key algorithm competes with RSA and elliptic curve ECC. NTRUEncrypt was approved by the financial services standards body, the Accredited Standards Committee X9. The X9.98 standard specifies how to use NTRU, as it's called for short, in financial transactions.

RSA(1978-1980), ECC(1985), NTRU(1995-2000)

It was invented in the mid-1990s. Unlike RSA, NTRU is not widely used, and in fact the NTRU cryptosystem needed changes early on to improve its security by addressing weaknesses and performance. But today NTRU is recognised as faster than the widely used RSA algorithm.

NTRU at a high security level, is much faster than RSA (around five orders of magnitude) and ECC (around three orders of magnitude). This type of lattice design makes it more resistant than an algorithm like RSA to so-called quantum computing attacks.

NTRU will carry general parameters (N, p, q)

N - the polynomials in the truncated polynomial ring have degree N−1.

For example N=11, N(*x*) = *x*11 −1.

q - large modulus: usually, the coefficients of the truncated polynomials will be reduced mod q. For example a large system will opt for q = 28 = 256.

p - small modulus. As the final step in decryption, the coefficients of the message are reduced mod p. A typical example p = 3.

In order to ensure security, it is essential that p and q have no common factors, gcd(p, q) =1. The following table gives some possible values for NTRU parameters at various security levels.

Table 1. A typical NTRU parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Security Level | N | p | q |
| Small sample | 11 | 3 | 32 |
| Moderate | 167 | 3 | 128 |
| Standard | 251 | 3 | 128 |
| High | 347 | 3 | 128 |
| Highest | 503 | 3 | 256 |

In NTRU, we have 2 domains, rings modulo p and modulo q. Earlier, we take a group member *a* ∈{0, 1, 2, ..., *p*−1} mod *p*. For example, {0, 1, 2, 3, 4, 5, 6} modulo 7.

However, in this arena, NTRU takes mod *p* as *a* ∈{, ..., −2, −1, 0, +1, +2, ..., } in center lift. For examples,

{−1, 0, 1} modulo 3.

{−3, −2, −1, 0, 1, 2, 3} modulo 7.

{−15, ..., −2, −1, 0, 1, 2, ..., 16} modulo 32.

Let us take a short overview on a ternary system. Along with ternary arithmetic, a computer built of base-3 hardware can also exploit ternary logic. Unlike a bit, this time it is a trit. Consider the task of comparing two numbers.

Compare(*x*, *y*) = 

Ternary logic simplifies the process: A single comparison can yield any of three possible outcomes: "less," "equal" and "greater."

In NTRU, a number shall be written in a balanced mode.

Given a system parameter (N, p, q) = (11, 3, 32)

**Key Generation**

1. Generate a random private (preferable irreducible) polynomial *f*(*x*) and a camouflage polynomial g(*x*)
2. Compute an inverse *fp*−1 of *f* modulo *p* and an inverse *fq*−1 of *f* modulo *q* against modulo truncated polynomial N(*x*).
3. Compute public key polynomial *h*(*x*) = *p* ⋅ *fq*−1(*x*)\*g(*x*) mod q.

Given

*f*(*x*) = −*x*10 + *x*9 + *x*6 − *x*4 + *x*2 + *x* −1 = [−1,1,0,0,1,0,−1,0,1,1,−1] (mod *p*) and

*g*(*x*) = −*x*10 − *x*8 + *x*5 + *x*3 + *x*2 −1 = [−1,0,−1,0,0,1,0,1,1,0,−1] in little endian.

*fp*−1(*x*) = 2*x*9 + *x*8 + 2*x*7 + *x*5 + 2*x*4 +2*x*3 + 2*x* +1 (mod *p*)

= [ 2, 1, 2, 0, 1, 2, 2, 0, 2, 1] mod N(*x*)

= [-1, 1,-1, 0, 1,-1,-1, 0,-1, 1]

*fq*−1(*x*) = 30*x*10 + 18*x*9 + 20*x*8 + 22*x*7 + 16*x*6 + 15*x*5 + 4*x*4 + 16*x*3 + 6*x*2 + 9*x* +5 (mod *q*)

= [30, 18, 20, 22,16,15,4,16,6,9,5] mod N(*x*)

= [-2,-14,-12,-10,16,15,4,16,6,9,5]

Take a public key,

*h*(*x*) = *p* ⋅ *fq*−1(*x*)\*g(*x*) mod *q*.

= 16*x*10 + 19*x*9 + 12*x*8 + 19*x*7 + 15*x*6 + 24*x*5 + 12*x*4 + 20*x*3 + 22*x*2 + 25*x* +8

= [16,19,12,19,15,24,12,20,22,25,8] (mod 32)

= [16,−13,12,−13,15,−8,12,−12,−10,−7,8]

Let the private key be [*f*(*x*), *fp*−1(*x*)] and public key *h*(*x*) with general parameters

N=11, p=3, and q = 32.

**Encryption Process**

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

2. She first puts her message in the form of a polynomial *m*(*x*)

whose coefficients are chosen modulo p.

3. Next she randomly chooses another small polynomial, a random encrypt polynomial key *r*(*x*). This is the "blinding value", which is used to obscure the message.

Given Bob's public key *h*(*x*),

*h*(*x*) = *p* ⋅ *fq*−1(*x*)\*g(*x*) mod *q*.

= 16*x*10 + 19*x*9 + 12*x*8 + 19*x*7 + 15*x*6 +24*x*5 + 12*x*4 +20*x*3 + 22*x*2 +25*x* +8

= [16,19,12,19,15,24,12,20,22,25,8]

= [16,−13,12,−13,15,−8,12,−12,−10,−7,8]

The ciphertext *e*(*x*) = r(*x*) \**h*(*x*) + *m*(*x*) (modulo q).

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

*m*(*x*) = *x*8 + *x*7 + *x*4 = [ 1,1,0,0,1,0,0,0,0].

r(*x*) = − *x*7 − *x*5 + *x*4 − *x*3 + *x* +1 = [-1,0,-1,1,-1,0,1,1].

Then the ciphertext ***e*** is

*e*(*x*) = r(*x*) \**h*(*x*) + *m*(*x*) (modulo q).

Let us do the computation:

r(*x*) \**h*(*x*) = [-1, 0,-1, 1,-1, 0, 1, 1]

\*[16,19,12,19,15,24,12,20,22,25,8]

= -1[16,19,12, 19,15, 24,12, 20,22, 25,8]

-1[16, 19,12, 19,15, 24,12, 20,22,25,8]

+1[16,19, 12,19, 15,24, 12,20,22,25,8]

-1[16, 19,12, 19,15, 24,12,20,22,25,8]

+1[16, 19,12, 19,15,24,12,20,22,25,8]

+1[16,19, 12,19,15,24,12,20,22,25,8]

=[-16,-19,-28,-22,-24,-50,-4,-13,6,-26,12,16,31,15,34,47,33,8]

=[ 16, 13, 4, 10, 8, 14,28, 19,6, 6,12,16,31,15, 2,15, 1,8](mod q)

=[19, 6, 6,12,16,31,15, 2,15, 1, 8] since *x*11 ≡ 1 (mod *x*11 − 1)

16,13, 4,10, 8,14,28

=[19, 6, 6,12,32,44,19,12,23,15,36] (mod *x*11 − 1)

=[19, 6, 6,12, 0,12,19,12,23,15, 4](mod q)

*e*(*x*) = r(*x*) \**h*(*x*) + *m*(*x*)

=[19, 6, 6,12, 0,12,19,12,23,15, 4]

+[1, 1, 0, 0, 1, 0, 0, 0, 0]

=[19, 6, 7,13, 0,12,20,12,23,15, 4]

=[-13,6, 7,13, 0,12,-12,12,-9,15,4] in center mode.

**Decryption Process**

Now Bob has received Alice's encrypted message *e* and he wants to decrypt it. He begins by using his private polynomial *f* to compute the polynomial

*a*(*x*)= *f*(*x*)\**e*(*x*) (modulo q).

Since Bob is computing *a* modulo q, he can choose the coefficients of a to lie between  and . In general, Bob will choose the coefficients of *a* to lie in an interval of length q. The specific interval depends on the form of the small polynomials

performing the next step. Bob next computes the polynomial

*b*(*x*) = *a*(*x*) (modulo p).

That is, he reduces each of the coefficients of *a* modulo *p*. Finally Bob uses his other private polynomial Fp(*x*) = *f* −1(*x*) mod (*xN* −1, p) to compute

c(*x*) = Fp(*x*) \*b(*x*) (modulo p).

The polynomial c will be Alice's original message *m*(*x*).

0. Bob has received the encrypted message

*e*(*x*) = 19*x*10 +6*x*9 +7*x*8 +13*x*7 +12*x*5 +20*x*4 +12*x*3 +23*x*2 +15*x* +4 (mod 32).

=[19, 6, 7, 13, 0,12, 20, 12, 23, 15, 4]

from Alice.

He uses his private key

*f*(*x*) = −*x*10 + *x*9 + *x*6 − *x*4 + *x*2 + *x* −1

= [−1,1,0,0,1,0,−1,0,1,1,−1]

and compute

*a*(*x*) = *f*(*x*)\**e*(*x*) (modulo q).

= [-1, 1, 0, 0, 1, 0,-1, 0, 1, 1,-1]

\* [19, 6, 7,13, 0,12,20,12,23,15, 4]

=-[19, 6, 7,13, 0,12,20,12,23,15, 4]

+[19, 6, 7,13, 0,12,20,12,23,15, 4]

+[19, 6, 7,13, 0,12,20,12,23,15, 4]

-[19, 6, 7,13, 0,12,20,12,23,15, 4]

+[19, 6, 7,13, 0,12,20,12,23,15, 4]

+[19, 6, 7,13, 0,12,20,12,23,15, 4]

-[19, 6, 7,13, 0,12,20,12,23,15, 4]

=[-19,13,-1,-6,32,-6,-20,15,1,32,25,18, 7, 2,13, 5,11,26,-4,-11,-4]

=[25,18, 9, 2,13, 5,11, 26,-4,-11, -4]

[-19,13,-1,-6,32,-6,-20,15, 1, 32]

=[25,-1,22, 1, 7,37, 5, 6,11,-10, 28]

Since Bob is computing *a* modulo q, he can choose the coefficients of *a* to lie between −q/2 and q/2. That is, he reduces each of the coefficients of *a* modulo *p*.

1. Center Lifting: if *ai*> q/2 then *ai* = *ai* − q.
2. Center Lifting: if *ai*> p/2 then *bi* = *ai* − p.

*a*(*x*)=[25,-1, 22, 1, 7,37, 5, 6,11,-10, 28]

=[-7,-1,-10, 1, 7, 5, 5, 6,11,-10, -4] (mod 32) in center mode

*b*(*x*)≡[ 2, 2, 2, 1, 1, 2, 2, 0, 2, 2, 2] (mod 3)

*b*(*x*)≡[-1,-1,-1, 1, 1,-1,-1,0,-1,-1,-1] (mod 3)

Finally Bob uses his other private polynomial F*p* to compute

c(*x*) = *fp*−1(*x*)\**b*(*x*) (modulo p).

=[-1, 1,-1, 0, 1,-1,-1, 0,-1, 1]

\*[-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

= -1[-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

+1 [-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

-1 [-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

+1 [-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

-1 [-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

-1 [-1, 0,-1, 1, 0, 1, 1, 1,-1,-1, 0]

-1 [-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

+1 [-1,-1,-1, 1, 1,-1,-1, 0,-1,-1,-1]

= [ 1, 0, 1,-1, 0, 1, 0, 3, 4,-3, 0, 0, 1, 2, 1, 1, 2, 0, 0,-1]

= [-3, 0, 0, 1, 2, 1, 1, 2, 0, 0,-1]

+ [ 1, 0, 1,-1, 0, 1, 0, 3, 4]

= [-3, 0, 1, 1, 3, 0, 1, 3, 0, 3, 3]

= [ 1, 1, 0, 0, 1, 0, 0, 0, 0] (mod p)

The polynomial *c* will be Alice's original message *m*.

*m*(*x*) = *x*8 + *x*7 + *x*4 = [ 1,1,0,0,1,0,0,0,0].

**Tutorial 10: A small sample on NTRU**

Let M= 1000 + (ID mod 1000).

Convert M into binary.

Convert M into a polynomial in F3.

Encryp the plaintext M using the same public key h(*x*).

Decrypt M back into original plaintext.