

# Introduction to NTRU Public Key Cryptosystem

NTRUEncrypt

Hosein Hadipour<sup>†</sup>

May 27, 2021

---

<sup>†</sup>hsn.hadipour@gmail.com



# Outline

1. Introduction
2. Convolution Polynomial Rings
3. Operations of Convolution Polynomial Rings
4. Multiplicative Inverse
5. NTRUEncrypt
6. NTRUEncrypt-Overview
7. NTRUEncrypt with SageMath
8. Security
9. NTRUEncrypt - Lattice Reduction with SageMath
10. Speed
11. Conclusion

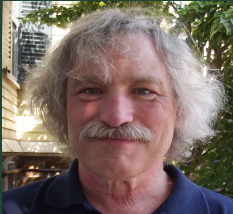


# NTRU

- NTRU: *N*th-degree *TRU*ncated polynomial ring (pronounced *en-trū*)

# NTRU

- NTRU: *N*th-degree *TRU*ncated polynomial ring (pronounced *en-trū*)
- A public key cryptosystem [HPS98] invented in early 1996 by



Hoffstein



Pipher



Silverman

# Ring of Convolution Polynomials

## Definition

The ring of convolution polynomials of rank  $N^1$  is the quotient ring

$$R = \frac{\mathbb{Z}[x]}{\langle x^N - 1 \rangle}$$

---

<sup>1</sup>a.k.a.  $N$ -th truncated polynomial ring

# Ring of Convolution Polynomials

## Definition

The ring of convolution polynomials of rank  $N^1$  is the quotient ring

$$R = \frac{\mathbb{Z}[x]}{\langle x^N - 1 \rangle}$$

## Definition

The ring of convolution polynomials modulo  $q$  of rank  $N$  is the quotient ring

$$R_q = \frac{\mathbb{Z}_q[x]}{\langle x^N - 1 \rangle}$$

---

<sup>1</sup>a.k.a.  $N$ -th truncated polynomial ring

# The Elements of Convolution Polynomial Rings

How does the elements of convolution polynomial rings look?

# The Elements of Convolution Polynomial Rings

How does the elements of convolution polynomial rings look?

- Every element of  $R$  or  $R_q$  has a unique representation of the form

$$a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1}$$

with the coefficients in  $\mathbb{Z}$  or  $\mathbb{Z}_q$ , respectively.



# The Elements of Convolution Polynomial Rings

How does the elements of convolution polynomial rings look?

- Every element of  $R$  or  $R_q$  has a unique representation of the form

$$a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1}$$

with the coefficients in  $\mathbb{Z}$  or  $\mathbb{Z}_q$ , respectively.

- For every term  $x^k$ , if  $k = r \pmod N$ , then

$$x^k = x^r.$$

## The Elements of Convolution Polynomial Rings

How does the elements of convolution polynomial rings look?

- Every element of  $R$  or  $R_q$  has a unique representation of the form

$$a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1}$$

with the coefficients in  $\mathbb{Z}$  or  $\mathbb{Z}_q$ , respectively.

- For every term  $x^k$ , if  $k = r \pmod N$ , then

$$x^k = x^r.$$

- A polynomial  $a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1} \in R$  can also be identified with its vector of coefficients:

$$(a_0, a_1, a_2, \cdots, a_{N-1}) \in \mathbb{Z}^N.$$

## The Elements of Convolution Polynomial Rings

### How does the elements of convolution polynomial rings look?

- Every element of  $R$  or  $R_q$  has a unique representation of the form

$$a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1}$$

with the coefficients in  $\mathbb{Z}$  or  $\mathbb{Z}_q$ , respectively.

- For every term  $x^k$ , if  $k = r \pmod N$ , then

$$x^k = x^r.$$

- A polynomial  $a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1} \in R$  can also be identified with its vector of coefficients:

$$(a_0, a_1, a_2, \cdots, a_{N-1}) \in \mathbb{Z}^N.$$

- Polynomials in  $R_q$  can also be uniquely identified in the same way.

## Operations of Convolution Polynomial Rings

Every ring has two operations, i.e, addition and multiplication.

- **Addition** of polynomials correspond to the usual addition of vectors,

$$a(x) + b(x) \leftrightarrow (a_0 + b_0, a_1 + b_1, a_2 + b_2, \dots, a_{N-1} + b_{N-1}).$$

## Operations of Convolution Polynomial Rings

Every ring has two operations, i.e, addition and multiplication.

- **Addition** of polynomials correspond to the usual addition of vectors,

$$a(x) + b(x) \leftrightarrow (a_0 + b_0, a_1 + b_1, a_2 + b_2, \dots, a_{N-1} + b_{N-1}).$$

- **Multiply** two polynomials mod  $x^N - 1$ , i.e., replace  $x^k$  with  $x^{k \bmod N}$ .
- Polynomial multiplication in  $R_q$  can be computed using the same way, except that the coefficients are reduced modulo  $q$ .

## Example

**Example.** Let  $N = 5$  and  $a(x) = 1 - 2x + 4x^3 - x^4$ , and  $b(x) = 3 + 4x - 2x^2 + 5x^3 + 2x^4$ . Then

$$\begin{aligned}a(x) \star b(x) &= 3 - 2x - 10x^2 + 21x^3 + 5x^4 - 16x^5 + 22x^6 + 3x^7 - 2x^8 \\&= 3 - 2x - 10x^2 + 21x^3 + 5x^4 - 16 + 22x + 3x^2 - 2x^3 \\&= -13 + 20x - 7x^2 + 19x^3 + 5x^4 \in R = \frac{\mathbb{Z}[x]}{\langle x^5 - 1 \rangle}.\end{aligned}$$

## Example

**Example.** Let  $N = 5$  and  $a(x) = 1 - 2x + 4x^3 - x^4$ , and  $b(x) = 3 + 4x - 2x^2 + 5x^3 + 2x^4$ . Then

$$\begin{aligned} a(x) \star b(x) &= 3 - 2x - 10x^2 + 21x^3 + 5x^4 - 16x^5 + 22x^6 + 3x^7 - 2x^8 \\ &= 3 - 2x - 10x^2 + 21x^3 + 5x^4 - 16 + 22x + 3x^2 - 2x^3 \\ &= -13 + 20x - 7x^2 + 19x^3 + 5x^4 \in R = \frac{\mathbb{Z}[x]}{\langle x^5 - 1 \rangle}. \end{aligned}$$

If we work instead in the ring  $R_{11}$ , then we reduce the coefficients modulo 11:

$$a(x) \star b(x) = 9 + 9x + 4x^2 + 8x^3 + 5x^4 \in R_{11} = \frac{\mathbb{Z}_q[x]}{\langle x^5 - 1 \rangle}.$$

# Convolution Polynomial Rings in Sage I

- Generate  $R = \frac{\mathbb{Z}[x]}{\langle x^7-1 \rangle}$ :

```
N = 7
ZX.<X> = PolynomialRing(ZZ)
R.<x> = ZX.quotient(X^N - 1); R
Univariate Quotient Polynomial Ring in x over
Integer Ring with modulus X^7 - 1
```

- Generate  $R_3 = \frac{\mathbb{Z}_3[x]}{\langle x^7-1 \rangle}$

```
N, q = 7, 3
ZqX.<X> = PolynomialRing(Zmod(q))
Rq.<x> = ZqX.quotient(X^N - 1); Rq
Univariate Quotient Polynomial Ring in x over
Ring of integers modulo 3 with modulus X^7 + 2
```



## Convolution Polynomial Rings in Sage II

- Choose two elements at random from  $R_3$ , and multiply them:

```
[f, g] = [Rq.random_element() for _ in range(2)]
print("(f, g) = ", (f, g))
print("f*g = ", f*g)
(f, g) = (2*x^6 + 2*x^4 + x^3, 2*x^6 + x^2 + 2*x)
f*g = 2*x^6 + 2*x^4 + x^3 + 2*x^2 + 2*x + 1
```

- Lift  $f \in R_3 = \frac{\mathbb{Z}_3[X]}{\langle X^7-1 \rangle}$  into  $\mathbb{Z}_3[X]$

```
print(f.parent())
Univariate Quotient Polynomial Ring in x over
Ring of integers modulo 3 with modulus X^7 + 2
```

```
f = f.lift()
print(f.parent())
Univariate Polynomial Ring in X over
Ring of integers modulo 3
```

# Multiplicative Inverse I

Very few polynomials in  $R$  have multiplicative inverse, but the situation is quite different in  $R_q$ .

## Theorem

*Let  $q$  be prime. Then  $a(x) \in R_q$  has a multiplicative inverse if and only if*

$$\gcd(a(x), x^N - 1) = 1 \in \mathbb{Z}_q[x].$$

*If so, then the inverse  $a(x)^{-1} \in R_q$  can be computed using the extended Euclidean algorithm to find polynomials  $u(x), v(x) \in \mathbb{Z}_q[x]$  satisfying*

$$a(x)u(x) + (x^N - 1)v(x) = 1.$$

*Then  $a(x)^{-1} = u(x) \in R_q$ .*

- What if  $q$  is not prime?

## Multiplicative Inverse II

- You can simply compute the inverse via SageMath[The21] (if it exists!)

```
reset()
N, q = 7, 4
Zx.<X> = ZZ[]
f = X^6 - X^4 + X^3 + X^2 - 1
Zq.<a> = PolynomialRing(Zmod(q))
f = Zq(f) # Moving f from Zx[x] into Zq[a]
print("gcd(f, a^N - 1) = ", f.gcd(a^N - 1))
f_inv = f.inverse_mod(a^N - 1); f_inv(a=X)

gcd(f, a^N - 1) = 1
X^5 + 3*X^4 + 3*X^3 + 2*X^2
```

- Check to see if the multiplication of  $f \star f^{-1} = 1 \pmod q$

```
Zq(f*f_inv).mod(a^N - 1)
```

```
1
```

## NTRUEncrypt

- Three prominent sub-algorithms of NTRUEncrypt:
  - ▶ **Key-Generation:** It produces the private and public keys taking the security parameter  $1^n$  as input
  - ▶ **Encryption:** It takes as input a public key and message from some message space (that may depend on public key), and outputs a ciphertext (it might be a probabilistic)
  - ▶ **Decryption:** A deterministic algorithm takes as input the private key and a ciphertext and outputs a message or a special symbol  $\perp$  denoting failure

We need the following notation before describing NTRUEncrypt:

### Definition

For any positive integers  $d_1$  and  $d_2$ , the set of *ternary* polynomials  $L(d_1, d_2)$  is defined by:

$$L(d_1, d_2) := \{a(x) = \sum_{i=1}^k c_i x^i \in R \mid \#\{c_i = 1\} = d_1, \#\{c_i = -1\} = d_2, \\ \#\{c_i = 0\} = k - d_1 - d_2\}.$$

## NTRUEncrypt - Key-Generation

- A trusted party choose public parameters  $(N, p, q, d)$  with  $N$  and  $p$  prime,  $\gcd(p, q) = \gcd(N, q) = 1$ , and  $q > (6d + 1)p$ .

## NTRUEncrypt - Key-Generation

- A trusted party choose public parameters  $(N, p, q, d)$  with  $N$  and  $p$  prime,  $\gcd(p, q) = \gcd(N, q) = 1$ , and  $q > (6d + 1)p$ .
- Alice perform the following operation to create her keys:
  - ▶ Choose private  $f(x) \in L_f = L(d + 1, d)$  that is unit (invertible) in  $R_q$  and  $R_p$
  - ▶ Choose private  $g(x) \in L_g = L(d, d)$
  - ▶ Compute  $f_q$ , the inverse of  $f$  in  $R_q$
  - ▶ Compute  $f_p$ , the inverse of  $f$  in  $R_p$
  - ▶ Publish the public key  $h(x) = pf_p \star g(x)$

## NTRUEncrypt - Key-Generation

- A trusted party choose public parameters  $(N, p, q, d)$  with  $N$  and  $p$  prime,  $\gcd(p, q) = \gcd(N, q) = 1$ , and  $q > (6d + 1)p$ .
- Alice perform the following operation to create her keys:
  - ▶ Choose private  $f(x) \in L_f = L(d + 1, d)$  that is unit (invertible) in  $R_q$  and  $R_p$
  - ▶ Choose private  $g(x) \in L_g = L(d, d)$
  - ▶ Compute  $f_q$ , the inverse of  $f$  in  $R_q$
  - ▶ Compute  $f_p$ , the inverse of  $f$  in  $R_p$
  - ▶ Publish the public key  $h(x) = pf_p \star g(x)$
- Condition  $q > (6d + 1)p$  ensures the correctness of decryption algorithm.

## NTRUEncrypt - Encryption

- Bob wants to encrypt a message to Alice!



## NTRUEncrypt - Encryption

- Bob wants to encrypt a message to Alice!
- Bob selects plaintext  $m \in R_p$

## NTRUEncrypt - Encryption

- Bob wants to encrypt a message to Alice!
- Bob selects plaintext  $m \in R_p$
- He chooses a random  $r(x) \in L_r = L(d, d)$

## NTRUEncrypt - Encryption

- Bob wants to encrypt a message to Alice!
- Bob selects plaintext  $m \in R_p$
- He chooses a random  $r(x) \in L_r = L(d, d)$
- He uses Alice's public key  $h(x)$  to compute

$$e(x) = r(x) \star h(x) + m(x) \in R_q$$

## NTRUEncrypt - Encryption

- Bob wants to encrypt a message to Alice!
- Bob selects plaintext  $m \in R_p$
- He chooses a random  $r(x) \in L_r = L(d, d)$
- He uses Alice's public key  $h(x)$  to compute

$$e(x) = r(x) \star h(x) + m(x) \in R_q$$

- The ciphertext is the polynomial  $e(x) \in R_q$

## NTRUEncrypt - Decryption

- Alice receives  $e(x)$  from Bob

---

<sup>2</sup>Coefficients of  $a(x)$  are taken in  $(-\frac{q}{2}, \frac{q}{2}]$

## NTRUEncrypt - Decryption

- Alice receives  $e(x)$  from Bob
- Using her private key  $(f, g)$  computes

$$a(x) = f(x) \star e(x) = pg(x) \star r(x) + f(x) \star m(x) \in R_q$$

---

<sup>2</sup>Coefficients of  $a(x)$  are taken in  $(-\frac{q}{2}, \frac{q}{2}]$

## NTRUEncrypt - Decryption

- Alice receives  $e(x)$  from Bob
- Using her private key  $(f, g)$  computes

$$a(x) = f(x) \star e(x) = pg(x) \star r(x) + f(x) \star m(x) \in R_q$$


- Alice center-lifts<sup>2</sup>  $a(x)$  to  $a(x) \in R$  and compute

$$m(x) = f_p \star a(x) \in R_p$$

---

<sup>2</sup>Coefficients of  $a(x)$  are taken in  $(-\frac{q}{2}, \frac{q}{2}]$

# NTRUEncrypt - Overview

	<b>Public parameter creation</b>	
	A trusted party chooses public parameters $(N, p, q, d)$ with $N$ and $p$ prime, $\gcd(p, q) = \gcd(N, q) = 1$ , and $q > (6d + 1)p$ .	
	<b>Alice</b>	<b>Bob</b>
	<b>Key creation</b>	
	Choose private $f \in L(d + 1, d)$ that is invertible in $R_q$ and $R_p$ . Choose private $g \in L(d, d)$ . Compute $f_q$ , the inverse of $f$ in $R_q$ . Compute $f_p$ , the inverse of $f$ in $R_p$ . Publish the public key $h = pf_q \star g$ .	
	<b>Encryption</b>	
		Choose plaintext $m \in R_p$ . Choose a random $r \in L(d, d)$ . Use Alice's public key $h$ to compute $e = r \star h + m(\text{mod } q)$ . Send ciphertext $e$ to Alice.
	<b>Decryption</b>	
	Compute $f \star e = pg \star r + f \star m(\text{mod } q)$ . Center-lift to $a \in R$ and compute $m = f_p \star a(\text{mod } p)$ .	





## NTRUEncrypt - The Correctness of Decryption

- $q$ , and  $p$ , as well as  $d$  have been chosen such *s.t.*  $q > (6d + 1)p$

## NTRUEncrypt - The Correctness of Decryption

- $q$ , and  $p$ , as well as  $d$  have been chosen such *s.t.*  $q > (6d + 1)p$
- $r(x), g(x) \in L(d, d)$ . If in  $g(x) \star r(x)$ , all of their 1's match up and all of their  $-1$ 's match up, the largest coefficients of  $g(x) \star r(x)$  is  $2d$ .

## NTRUEncrypt - The Correctness of Decryption

- $q$ , and  $p$ , as well as  $d$  have been chosen such *s.t.*  $q > (6d + 1)p$
- $r(x), g(x) \in L(d, d)$ . If in  $g(x) \star r(x)$ , all of their 1's match up and all of their  $-1$ 's match up, the largest coefficients of  $g(x) \star r(x)$  is  $2d$ .
- $f(x) \in L(d + 1, d)$  and the coefficients of  $m(x)$  are in  $(-\frac{p}{2}, \frac{p}{2}]$ .  
Hence, the largest possible coefficient of  $f(x) \star m(x)$  is  $(2d + 1) \cdot \frac{p}{2}$ .

## NTRUEncrypt - The Correctness of Decryption

- $q$ , and  $p$ , as well as  $d$  have been chosen such *s.t.*  $q > (6d + 1)p$
- $r(x), g(x) \in L(d, d)$ . If in  $g(x) \star r(x)$ , all of their 1's match up and all of their  $-1$ 's match up, the largest coefficients of  $g(x) \star r(x)$  is  $2d$ .
- $f(x) \in L(d + 1, d)$  and the coefficients of  $m(x)$  are in  $(-\frac{p}{2}, \frac{p}{2}]$ .  
Hence, the largest possible coefficient of  $f(x) \star m(x)$  is  $(2d + 1) \cdot \frac{p}{2}$ .
- Even if the largest possible coefficient of  $g(x) \star r(x)$  happens to coincide with the largest coefficient of  $r(x) \star m(x)$ , the largest coefficient of  $a(x) = pg \star r + f \star m$  is at most

$$p \cdot 2d + (2d + 1) \cdot \frac{p}{2} = \left(3d + \frac{1}{2}\right) p.$$

- Hence,  $a(x) = f(x) \star e(x)$  is in  $R_q$  by default and moving to  $R_q$  (or reducing modulo  $q$ ) has no effect, and  $f_p \star a(x) \in R_p$  is equal to the original plaintext

## NTRUEncrypt - SageMath Example I

- Compute ciphertext:

```
reset()
N, p, q, d = 7, 3, 41, 2
assert(q > (6*d + 1)*p)
Zx.<x> = ZZ[]
Zp.<s> = PolynomialRing(Zmod(p))
Zq.<t> = PolynomialRing(Zmod(q))
f = x^6 - x^4 + x^3 + x^2 - 1
g = x^6 + x^4 - x^2 - x
fp = Zp(f).inverse_mod(s^N - 1); fp = fp(s=x)
fq = Zq(f).inverse_mod(t^N - 1); fq = fq(t=x)
h = Zq(p*fq*g).mod(t^N - 1)
h = h(t=x)
m = -x^5 + x^3 + x^2 - x + 1
r = x^6 - x^5 + x - 1
e = Zq(h*r + m).mod(t^N - 1); e = e(t=x)
print('ciphertext:\n%s' % e)

ciphertext:
31*x^6 + 19*x^5 + 4*x^4 + 2*x^3 + 40*x^2 + 3*x + 25
```

## NTRUEncrypt - SageMath Example II

- Recovering the plaintext:

```
a = Zq(f*e).mod(t^N - 1); a = a(t=x)
center_lift = lambda c, p: (ZZ(c)%p - p) if ZZ(c)%p > p//2\
else ZZ(c)%p
a_coeffs = [center_lift(c, q) for c in \
a.coefficients(sparse=False)]
lifted_a = sum([a_coeffs[i]*x^i for i in \
range(len(a_coeffs))])
b = Zq(fp*lifted_a).mod(t^N - 1); b = b(t=x)
b_coeffs = [center_lift(c, p) for c in \
b.coefficients(sparse=False)]
lifted_b = sum([b_coeffs[i]*x^i for i in \
range(len(b_coeffs))])
print('plaintext:\n%s' % lifted_b)
plaintext:
-x^5 + x^3 + x^2 - x + 1
```

# The NTRU Key Recovery Problem

- What is the hard math problem behind NTRU?

# The NTRU Key Recovery Problem

- What is the hard math problem behind NTRU?
- **Lattice reduction**
  - ▶ Same problem that breaks the knapsack!



## The NTRU Key Recovery Problem

- What is the hard math problem behind NTRU?
- **Lattice reduction**
  - ▶ Same problem that breaks the knapsack!
- If attacker can determine  $f(x)$  or  $f_q(x)$ , from  $h(x)$ , she gets the private key

## The NTRU Key Recovery Problem

- What is the hard math problem behind NTRU?
- **Lattice reduction**
  - ▶ Same problem that breaks the knapsack!
- If attacker can determine  $f(x)$  or  $f_q(x)$ , from  $h(x)$ , she gets the private key
- Recall  $h(x) = p \cdot f_q(x) \star g(x) \bmod q$

## The NTRU Key Recovery Problem

- What is the hard math problem behind NTRU?
- **Lattice reduction**
  - ▶ Same problem that breaks the knapsack!
- If attacker can determine  $f(x)$  or  $f_q(x)$ , from  $h(x)$ , she gets the private key
- Recall  $h(x) = p \cdot f_q(x) \star g(x) \bmod q$
- Equivalently,  $h(x) \star f(x) = p \cdot g(x) \bmod q$

## The NTRU Key Recovery Problem[HPSS08]

Given  $h(x)$ , find **ternary** polynomials  $f(x)$  and  $g(x)$  satisfying  $f(x) \star h(x) = p \cdot g(x) \bmod q$

- The solution of NTRU key recovery problem is not unique (why?)

## Why NTRU Key Recovery Problem is Hard?

- Solving NTRU key recovery problem is (almost certainly) equivalent to solving **SVP** problem in a certain class of lattices.

## Why NTRU Key Recovery Problem is Hard?

- Solving NTRU key recovery problem is (almost certainly) equivalent to solving **SVP** problem in a certain class of lattices.
- Denote  $h(x) = h_0 + h_1x + \dots + h_{N-1}x^{N-1}$

## Why NTRU Key Recovery Problem is Hard?

- Solving NTRU key recovery problem is (almost certainly) equivalent to solving **SVP** problem in a certain class of lattices.
- Denote  $h(x) = h_0 + h_1x + \dots + h_{N-1}x^{N-1}$
- Define

$$\mathbf{H} = \begin{pmatrix} h_0 & h_{N-1} & h_{N-2} & \dots & h_1 \\ h_1 & h_0 & h_{N-1} & \dots & h_2 \\ \vdots & & \ddots & & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \dots & h_0 \end{pmatrix}$$

## Why NTRU Key Recovery Problem is Hard?

- Solving NTRU key recovery problem is (almost certainly) equivalent to solving **SVP** problem in a certain class of lattices.
- Denote  $h(x) = h_0 + h_1x + \dots + h_{N-1}x^{N-1}$
- Define

$$\mathbf{H} = \begin{pmatrix} h_0 & h_{N-1} & h_{N-2} & \dots & h_1 \\ h_1 & h_0 & h_{N-1} & \dots & h_2 \\ \vdots & & \ddots & & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \dots & h_0 \end{pmatrix}$$

- Let  $\mathbf{h}$  be the coefficients of  $h(x)$  as a column and similarly for  $\mathbf{f}$  and  $\mathbf{g}$  corresponding to  $f(x)$ , and  $g(x)$ , respectively.

## NTRUEncrypt as A Lattice Cryptosystem I

- According to the definition of  $\star$ , we have

$$\mathbf{H}\mathbf{f} = p\mathbf{g} \mod q$$

- Equivalently to block matrix equation

$$M \cdot V = \begin{pmatrix} \mathbf{I}_{N \times N} & \mathbf{0}_{N \times N} \\ \mathbf{H}_{N \times N} & q\mathbf{I}_{N \times N} \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ p\mathbf{g} \end{pmatrix} = W \mod q$$

- That is  $\mathbf{f} = \mathbf{f}$ , and  $\mathbf{H}\mathbf{f} + q\mathbf{s} = p\mathbf{g} \mod q$



## NTRUEncrypt as A Lattice Cryptosystem II

- Attacker can find private key from  $V$  or  $W$ 
  - ▶  $W$  is in lattice spanned by columns of  $M$
  - ▶  $W$  has special form (number of +1's and -1's and 0's)
  - ▶  $W$  is a **short** vector
- Lattice reduction attack
  - ▶ Just like the knapsack?
- But NTRU lattice is hard to break!
  - ▶ As far as anybody knows ...

# Lattice Reduction Attack Using SageMath I

```
N, p, q = 7, 3, 41
Zx.<X> = ZZ[]
f=X^6 - X^4 + X^3 + X^2-1
g=X^6 + X^4 - X^2 - X
h=19*X^6 + 38*X^5 + 6*X^4 + 32*X^3 + 24*X^2 + 37*X + 8
M = matrix(2*N)
for i in [0..N-1]: M[i,i] = 1
for i in [N..2*N-1]: M[i,i] = q
for i in [0..N-1]:
for j in [0..N-1]:
M[i+N,j] = ((Zx(GF(q)(1/p)*h)*X^i)%(X^N-1))[j]
pretty_print(M)
pretty_print(M.transpose().LLL())
pretty_print(f.coefficients(sparse=False))
pretty_print(g.coefficients(sparse=False))
```

# Lattice Reduction Attack Using SageMath II

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 30 & 26 & 8 & 38 & 2 & 40 & 20 & 41 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20 & 30 & 26 & 8 & 38 & 2 & 40 & 0 & 41 & 0 & 0 & 0 & 0 & 0 \\ 40 & 20 & 30 & 26 & 8 & 38 & 2 & 0 & 0 & 41 & 0 & 0 & 0 & 0 \\ 2 & 40 & 20 & 30 & 26 & 8 & 38 & 0 & 0 & 0 & 41 & 0 & 0 & 0 \\ 38 & 2 & 40 & 20 & 30 & 26 & 8 & 0 & 0 & 0 & 0 & 41 & 0 & 0 \\ 8 & 38 & 2 & 40 & 20 & 30 & 26 & 0 & 0 & 0 & 0 & 0 & 41 & 0 \\ 26 & 8 & 38 & 2 & 40 & 20 & 30 & 0 & 0 & 0 & 0 & 0 & 0 & 41 \end{pmatrix}$$

## Lattice Reduction Attack Using SageMath III

After applying LLL algorithm on  $M$ :

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & -1 & 0 & 1 & -1 & 0 & -1 & 0 & 1 & 1 & 0 \\ 1 & -1 & -1 & 0 & 1 & -1 & 0 & -1 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & -1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 2 & 0 \\ 0 & 1 & -1 & 0 & 1 & -1 & -1 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \\ 1 & -2 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\ 1 & 0 & 1 & -1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 2 & 0 & 0 \\ -10 & -1 & 0 & 0 & 8 & 1 & -1 & -3 & -5 & 8 & -5 & -1 & 5 & 1 \\ 1 & -2 & -9 & 0 & 0 & -1 & 9 & 5 & 2 & -3 & -6 & 7 & -5 & 0 \\ 0 & 0 & 8 & 1 & -1 & -10 & -1 & 8 & -5 & -1 & 5 & 1 & -3 & -5 \\ -1 & 9 & 1 & -2 & -9 & 0 & 0 & -5 & 0 & 5 & 2 & -3 & -6 & 7 \\ 9 & 1 & -2 & -9 & 0 & 0 & -1 & 0 & 5 & 2 & -3 & -6 & 7 & -5 \\ -1 & 0 & 0 & 8 & 1 & -1 & -10 & -5 & 8 & -5 & -1 & 5 & 1 & -3 \\ -3 & -1 & 6 & -2 & 6 & -2 & -2 & -6 & -11 & -3 & -4 & -9 & 1 & -9 \end{pmatrix}$$

$$\mathbf{f} = [-1, 0, 1, 1, -1, 0, 1]$$

$$\mathbf{g} = [0, -1, -1, 0, 1, 0, 1]$$

# Known Attacks on NTRUEncrypt

- Lattice reduction
  - ▶ Generic attack (like factoring for RSA)
- Meet-in-the-middle
  - ▶ Square root of *exhaustive search* work
  - ▶ Inherent due to use of polynomials
- Multiple transmission
  - ▶ Encrypt  $m(x)$  multiple times with different  $r(x)$
  - ▶ Complex padding can prevent it
- Chosen ciphertext
  - ▶ Broke earlier version of NTRU

## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication

## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication
- Each coefficient is essentially the dot product of two vectors

## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication
- Each coefficient is essentially the dot product of two vectors
- A polynomial multiplication of two polynomial of length  $N$  requires  $N^2$  multiplications



## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication
- Each coefficient is essentially the dot product of two vectors
- A polynomial multiplication of two polynomial of length  $N$  requires  $N^2$  multiplications
- The multiplication required by NTRUEncrypt have the form  $r \star h$ ,  $f \star e$ , and  $f_p \star a$ , where  $r$ ,  $f$ , and  $f_p$  are ternary polynomials (+1, 0, -1)

## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication
- Each coefficient is essentially the dot product of two vectors
- A polynomial multiplication of two polynomial of length  $N$  requires  $N^2$  multiplications
- The multiplication required by NTRUEncrypt have the form  $r \star h$ ,  $f \star e$ , and  $f_p \star a$ , where  $r$ ,  $f$ , and  $f_p$  are ternary polynomials (+1, 0, -1)
- Hence, multiplications can be computed without any multiplications

## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication
- Each coefficient is essentially the dot product of two vectors
- A polynomial multiplication of two polynomial of length  $N$  requires  $N^2$  multiplications
- The multiplication required by NTRUEncrypt have the form  $r \star h$ ,  $f \star e$ , and  $f_p \star a$ , where  $r$ ,  $f$ , and  $f_p$  are ternary polynomials (+1, 0, -1)
- Hence, multiplications can be computed without any multiplications
- Multiplications require approximately  $\frac{2}{3}N^2$  additions and subtractions

## How Fast is NTRUEncrypt?

- The most time consuming part of encryption and decryption is the polynomial multiplication
- Each coefficient is essentially the dot product of two vectors
- A polynomial multiplication of two polynomial of length  $N$  requires  $N^2$  multiplications
- The multiplication required by NTRUEncrypt have the form  $r \star h$ ,  $f \star e$ , and  $f_p \star a$ , where  $r$ ,  $f$ , and  $f_p$  are ternary polynomials  $(+1, 0, -1)$
- Hence, multiplications can be computed without any multiplications
- Multiplications require approximately  $\frac{2}{3}N^2$  additions and subtractions
- Hence, NTRUEncrypt encryption and decryption take  $\mathcal{O}(N^2)$  steps, where each step is extremely fast.




# Conclusion

- A lattice-based public key cryptosystem
- Its security relies on difficulty of SVP problem
- Has evolved since its introduction
- Considered theoretically sound
- Unlike RSA and ECC, NTRU is **not** known to be vulnerable against quantum computer based attack
- Its open source implementations in Java and C are available
- It has been standardized (IEEE Std 1363.1, X9.98)

Thanks for your attention!

Question?

## References I

-  Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman, *Ntru: A ring-based public key cryptosystem*, International Algorithmic Number Theory Symposium, Springer, 1998, pp. 267–288.
-  Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman, *An introduction to mathematical cryptography*, vol. 1, Springer, 2008.
-  The Sage Developers, *Sagemath, the Sage Mathematics Software System (Version 9.2.0)*, 2021, <https://www.sagemath.org>.