NTRU PKCS Tutorial

This tutorial describes how the NTRU Public Key Cryptosystem (PKCS) works.

1. NTRU PKCS PARAMETERS

The basic collection of objects used by the NTRU Public Key Cryptosystem is the ring **R** that consists of all truncated polynomials of degree *N*-1 having integer coefficients:

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

Polynomials are added in the usual way. They are also multiplied more-or-less as usual, except that X^N is replaced by 1, X^{N+1} is replaced by X, X^{N+2} is replaced by X^2, and so on. For more details about the ring **R**, see Section 2 of the Algebra Tutorial.

A full implementation of the NTRU Public Key Cryptosystem is specified by a number of parameters. However, for the purposes of this overview we'll concentrate on the three most important:

N - the polynomials in the truncated polynomial ring have degree *N*-1.
q - large modulus: usually, the coefficients of the truncated polynomials will be reduced mod *q*.
p - small modulus. As the final step in decryption, the coefficients of the message are reduced mod *p*.

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

	N	*q*	*p*
Moderate Security	167	128	3
Standard Security	251	128	3
High Security	347	128	3
Highest Security	503	256	3

These values are provided to give you some idea of the quantities used in commercial applications (in fact, for commerical applications we take *p* to have a slightly different form; more details are in the Advanced Topics section). For the examples in this tutorial we will use much smaller parameter values:

	N	*q*	*p*
Small Illustration Parameters	11	32	3

2. KEY GENERATION

Key Generation Overview

Bob wants to create a public/private key pair for the NTRU Public Key Cryptosystem. He first randomly chooses two "small" polynomials **f** and **g** in the ring of truncated polynomials **R**.

- A "small" polynomial is small relative to a random polynomial mod *q*. In a random polynomial, the coefficients will in general be randomly distributed mod *q*; in a small polynomial, the coefficients are much smaller than *q*.

Bob must keep the values of the polynomials **f** and **g** private, since anyone who knows the value of either one of them will be able to decrypt messages sent to Bob.

Bob's next step is to compute the inverse of **f** modulo *q* and the inverse of **f** modulo *p*. Thus he computes polynomials $\mathbf{f}_q$ and $\mathbf{f}_p$with the property that

$$\mathbf{f}*\mathbf{f}_q = 1 \text{ (modulo } q\text{) and } \mathbf{f}*\mathbf{f}_p = 1 \text{ (modulo } p\text{)}.$$

(If by some chance these inverses do not exist, Bob will need to go back and choose another **f**. For information about computing inverses in the ring of truncated polynomials, see Section 3 of the Algebra Tutorial.) Now Bob computes the product

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

Bob's private key is the pair of polynomials **f** and f_p. Bob's public key is the polynomial **h**.

Key Generation: Example

As described in Section 1, the example parameters are:

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

We also need to define a "small" polynomial more precisely. For the purposes of this example, we do this using the quantities d_f and d_g.

- The polynomial **f** has d_f coefficients equal to +1, (d_f-1) coefficients equal to -1, and the rest equal to 0.
- The polynomial **g** has dg coefficients equal to +1, d_g coefficients equal to -1, and the rest equal to 0.

(The reason for the slight difference in form between **f** and **g** is that **f** has to be invertible, while **g** doesn't).

For the purposes of this tutorial, we take

$$d_f = 4 \quad d_g = 3.$$

So Bob needs to choose a polynomial **f** of degree 10 with four 1's and three -1's, and he needs to choose a polynomial g of degree 10 with three 1's and three -1's. Suppose he chooses:

$$\mathbf{f} = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$
$$\mathbf{g} = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Next Bob computes the inverse f_p of **f** modulo **p** and the inverse f_q of **f** modulo q. He finds that:

$f_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$
$f_q = 5 + 9X + 6X^2 + 16X^3 + 4X^4 + 15X^5 + 16X^6 + 22X^7 + 20X^8 + 18X^9 + 30X^{10}$

The final step in key creation is to compute the product

$\mathbf{h} = p\mathbf{f}_q*\mathbf{g} = 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 + 19X^7 + 12X^8 + 19X^9 + 16X^{10}$ (modulo 32).

Bob's private key is the pair of polynomials **f** and f_p, and his public key is the polynomial **h**.

3. ENCRYPTION

Encryption: Overview

Alice wants to send a message to Bob using Bob's public key **h**. She first puts her message in the form of a polynomial **m** whose coefficients are chosen modulo *p*, say between *-p/2* and *p/2* (in other words, **m** is a small polynomial mod *q*). Next she randomly chooses another small polynomial, **r**. This is the "blinding value", which is used to obscure the message (similar to the way that the ElGamal algorithm uses a one-time random value when encrypting).

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

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

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

Encryption: Example

As before, we need to specify what we mean by saying that **r** is a "small" polynomial. We do this using the quantity d_r.

- **r** has d_r of its coefficients equal to 1, it has d_r of its coefficients equal to -1, and it has all of the rest of its coefficients equal to 0.

For the purposes of this tutorial, we take

$$d_r = 3.$$

Now, suppose Alice wants to send the message

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

to Bob using Bob's public key

$$\mathbf{h} = 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 + 19X^7 + 12X^8 + 19X^9 + 16X^{10}.$$

She first chooses a random polynomial **r** of degree 10 with three 1's and three -1's. Say she chooses

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

Then her encrypted message **e** is

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

Alice sends the encrypted message **e** to Bob.

4. DECRYPTION

Decryption: Overview

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

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

Since Bob is computing **a** modulo *q*, he can choose the coefficients of **a** to lie between *-q/2* and *q/2*. (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. See the Advanced Topics tutorial for details.) It is very important that Bob does this before performing the next step. Bob next computes the polynomial

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

That is, he reduces each of the coefficients of **a** modulo p. Finally Bob uses his other private polynomial $\mathbf{f}_p$ to compute

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

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

Decryption: Example

Bob has received the encrypted message

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

from Alice. He uses his private key f to compute

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

Note that when Bob reduces the coefficients of **f*e** modulo 32, he chooses values lying between -15 and 16, not between 0 and 31. It is very important that he choose the coefficients in this way. Next Bob reduces the coefficients of **a** modulo 3 to get

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

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

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

The polynomial **c** is Alice's message **m**, so Bob has successfully decrypted Alice's message.

5. WHY IT WORKS

Alice's encrypted message **e** looks like **e = r*h + m** (modulo q), but of course Bob doesn't initially know the values of **r** and **m**. Bob's first step is to compute **f*e** and reduce the coefficients modulo q. Remember that Bob's public key **h** was actually formed by multiplying pf_q*g and reducing its coefficients modulo q. So although Bob doesn't know **r** and **m**, when he computes **a = f*e** (modulo q), he is actually performing the following computation:

a= f*e (modulo q)	
= f*(r*h + m) (modulo q)	[since e = r*h + m (modulo q)]
= f*(r*$pf_{q*g + m)}$ (modulo q)	[since h = pf_q*g (modulo q)]
= pr*g + f*m (modulo q)	[since f*f_q = 1 (modulo q)]

Now look back at the sizes of the various parameters. The polynomials **r, g, f,** and **m** all have coefficients that are quite small. This means that the coefficients of the products **r*g** and **f*m** are also quite small, at least in comparison to q. Since the prime p is also small compared to q, this means (assuming that the parameters have been properly chosen) that the coefficients of the polynomial $p\mathbf{r}*\mathbf{g} + \mathbf{f}*\mathbf{m}$ already lie between $-q/2$ and $q/2$, so reducing the coefficients modulo q has no effect at all!

In other words, when Bob computes a by first multiplying $\mathbf{f}*\mathbf{e}$ and then reducing the coefficients modulo q, the polynomial a that he ends up with is exactly equal to the polynomial $p\mathbf{r}*\mathbf{g} + \mathbf{f}*\mathbf{m}$. When Bob next reduces the coefficients of a modulo p to form the polynomial **b**, he is really reducing the coefficients of$p\mathbf{r}*\mathbf{g} + \mathbf{f}*\mathbf{m}$ modulo p, so the **b** that he ends up with is equal to

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

Keep in mind that Bob still doesn't know the value of **m**, but he now knows the value of **b**. So his final step is to multiply **b** by $\mathbf{f}_p$ and use the fact that $\mathbf{f}_p*\mathbf{f} = 1$ (modulo p) to compute

$$c = fp* \ b = fp* \ f*m = m \text{ (modulo } p\text{)},$$

which allows him to recover Alice's message **m**.

NEXT STEPS

This page has outlined the main concepts behind the NTRU cryptosystem. As described, it is already highly efficient. However, there are some parameter and implementation enhancements that we can use to speed up operations even more. These additional techniques are explained in the Advanced Topics tutorial and in the Low Hamming Weight Polynomials tutorial.

FURTHER READING

A complete description of the NTRU Public Key Cryptosystem with full technical details is given in the paper

NTRU: A Ring Based Public Key Cryptosystem, Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, in*Algorithmic Number Theory* (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.), Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998, 267-288.

Further enhancements to NTRU, including the use of $\mathbf{f}=1+p*\mathbf{F}$ and p=X+2, are described in

Optimizations for NTRU, J. Hoffstein, J. Silverman, Public-Key Cryptography and Computational Number Theory (Warsaw, September 11-15, 2000), DeGruyter, to appear.

A description of how to speed up NTRU and other cryptosystems by the use of quantities with small Hamming weight can be found in

Random Small Hamming Weight Products with Applications to Cryptography, J. Hoffstein, J. Silverman, Com2MaC Workshop on Cryptography (Pohang, Korea, June 2000), Discrete Mathematics, to appear.

The following are some additional sources to learn about algebra, number theory, algorithms, and cryptography.

- *A Course in Computational Algebraic Number Theory*, H. Cohen, GTM 138, Springer-Verlag, Berlin, 1993.
- *A Friendly Introduction to Number Theory*, J.H. Silverman, Prentice-Hall, New Jersey, 1997.
- *Cryptography: Theory and Practice*, D. Stinson, CRC Press, Boca Raton, 1995.
- *Handbook of Cryptography*, S. Vanstone, P. Van Oorschot, A. Menezes, CRC Press, Boca Raton, 1996.