The NTRUEncrypt Cryptosystem

Anirudh Narsipur
MATH 1580 Spring 2024 Final Project

1 Introduction

NTRUEncrypt is a lattice-based public key cryptosystem that is based on the intractability of certain lattice problems [1], such as the shortest vector problem (SVP). Unlike integer factorization or the discrete logarithm problem, no known quantum algorithm exists to solve the shortest vector problem [2] - making NTRU-like cryptosystems a promising direction for post-quantum cryptography. Further, NTRUEncrypt is relatively fast and has low memory usage in comparison to other cryptosystems [3]. In this paper, I outline the NTRUEncrypt cryptosystem and discuss its connections with lattice cryptography. Finally, I compare my Python implementation of NTRUEncrypt with its theoretical runtime complexity.

2 NTRUEncrypt

Suppose Alice and Bob wish to communicate in private. First, a trusted party chooses public parameters (N, p, q, d) such that N, p are prime, $gcd(Np, q) = 1$ and $q > (6d + 1)p$.

NTRUEncrypt operates over elements of a convolution polynomial ring. Convolution polynomial rings are rings that have polynomials as elements modulo a given monic polynomial $x^N - 1$. A more detailed overview can be found on page 412 of [3]. Using the public parameters, we fix the convolution polynomial rings.

$$R = \frac{\mathbb{Z}[x]}{x^N - 1}, R_p = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^N - 1}, R_q = \frac{\mathbb{Z}/q\mathbb{Z}[x]}{x^N - 1}$$

To generate her public, private key Alice first chooses $f \in \mathcal{T}(d + 1, d)$ where $\mathcal{T}$ is defined as:

$$\mathcal{T}(d_1, d_2) = \alpha(x) \in R : \left\{ \begin{array}{ll} \alpha(x) & \text{has } d_1 \text{ coefficients equal to 1} \\ \alpha(x) & \text{has } d_2 \text{ coefficients equal to -1} \\ \alpha(x) & \text{has all other } N - (d_1 + d_2) \text{ coefficients equal to 0} \end{array} \right\}$$

Choosing a random $f(x)$ is easy. Alice simply generates a list which has d_1 1's,d_2 -1's and $N - (d_1 + d_2)$ 0's where $d_1 = d + 1, d_2 = d$. Randomly shuffling this list creates an arbitrary element $f(x) \in \mathcal{T}(d + 1, d)$. Next, Alice computes the inverses:

$$F_q(x) = f(x)^{-1} \text{ in } R_q \text{ and } F_p(x) = f(x)^{-1} \text{ in } R_p$$

If either inverse does not exist, Alice discards $f(x)$ and generates a new one. To compute the inverses, observe that as p is prime, $\mathcal{F}_p$ is a field, and R_p is an Euclidean domain. Hence, for any two polynomial a, b in R_p such that $b \neq 0$ we can write:

$$a = b \cdot k + r \quad \text{r = 0 or deg r < deg b}$$

Using this polynomial division, one can use the extended Euclidean algorithm to compute the gcd and inverse of elements in R_p analogous to the finite groups [3]. In particular to compute F_p Alice com-

putes $extended\_gcd(f(x), x^N - 1)$ in $\mathcal{F}_p[x]$. If the gcd is 1, then the inverse exists in R_p and is given by $extended\_gcd$.

Computing F_q requires computing the inverse modulo the prime powers $p_1, p_2 ... p_n$ of $q = p_1^{k_1} p_2^{k_2} ... p_n^{k_n}, k_1 : k_n \in \mathbb{Z}$ which can then be combined using the Chinese Remainder Theorem. In practice, q is chosen to be a power of 2 [4]. Thus, it suffices to be able to compute an inverse modulo a given prime power p^i, which can be accomplished using the formula below.

Lemma: Let $f(x) \in R$. Suppose there exists $F(x)$ such that: $f(x) \star F(x) \equiv 1 \bmod p^i$ for some prime p and integer i. Then $G(x) = F(x) \star (2 - (f(x) \star F(x)))$ satisfies $f(x) \star G(x) \equiv 1 \bmod p^{2i}$.

Proof.

$$\begin{aligned} f(x) \star G(x) &= f(x)(F(x) \star (2 - (f(x) \star F(x)))) \\ &= 2f(x)F(x) - (f(x) \star F(x))^2 \\ f(x)F(x) \equiv 1 \bmod p^i \implies f(x)F(x) &= 1 + p^i H, H \in R \\ &= 2(1 + p^i H) - (1 + p^i H)^2 \textit{ Substituting} \\ &= 2(1 + p^i H) - (1 + p^{2i} H^2 + 2(1 + p^i H)) \\ &= 1 + p^{2i} H^2 \textit{ Cancelling terms} \\ &= 1 \mod p^{2i} \text{ as } p^{2i} H^2 \equiv 0 \bmod p^{2i} \end{aligned}$$

□

Repeating the above procedure (implemented in Python) allows the computation of inverses for arbitrary values of i [2], allowing the computation of an inverse in R_q. Next, Alice generates $g(x) \in \mathcal{T}(d, d)$ using the same process followed to generate $f(x)$. Using these quantities, Alice computes $h(x) = F_q(x) \star g(x)$ in R_q

Alice's public key is $h(x)$. Her private key is the pair $(f(x), F_p(x))$, which will be used for decryption.

To encrypt messages, Bob first chooses a plaintext $m(x) \in R_p$. He then chooses a random $r(x) \in \mathcal{T}(d, d)$. Using Alice's public key, he computes:

$$e(x) \equiv pr(x) \star h(x) + m(x) \bmod \text{q}$$

and sends the ciphertext e to Alice.

To decrypt the message, Alice computes $\alpha(x) = f(x) \star e(x) \bmod \text{q}$:

$$\begin{aligned} \alpha(x) &\equiv f(x) \star e(x) \bmod \text{q} \\ &\equiv f(x) \star (pr(x) \star h(x) + m(x)) \\ &\equiv pf(x) \star (F_q \star g(x)) \star r(x) + f(x) \star m \text{ As } h(x) = F_q \star q \\ (\text{As } f(x) \star F_q \equiv 1 \text{ in } R_q) &\equiv p \star g(x) \star r(x) + f(x) \star m(x) \to (1) \end{aligned}$$

Now, we can turn the equivalence above into equality as the condition that $q > (6d + 1)p$ ensures that the coefficients in the polynomial in (1) are bounded. All the coefficients in $g(x)$ and $r(x)$ are in $0, 1, -1$ and there are at most $2d$ non-zero coefficients in each polynomial (as they are in $\mathcal{T}(d, d)$). Thus, if all the coefficients line up perfectly, the largest possible coefficient is $2d$. Similarly, $f(x) \in \mathcal{T}(d + 1, d)$ so there at most $2d + 1$ non-zero coefficients. The coefficients of m are in $(\frac{-1}{2}p, \frac{1}{2}p]$ so if all the coefficients align, the largest possible coefficient is $(2d + 1) \cdot \frac{1}{2}p$. Hence, the largest possible coefficient is:

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

Observe $(6d+1)p < q$ by construction

$$(3d + \frac{1}{2})p = \frac{(6d+1)p}{2} < \frac{q}{2}$$

Thus, as every coefficient has a magnitude less than $\frac{q}{2}$, the center-lift will recover the polynomial in (1) exactly:

Proposition: The center-lift of a polynomial $a(x)$ from R_q to R is the polynomial $a'(x) \in R$ such that $a'(x)$ mod q $= a(x)$ and coefficients are in the interval $\frac{-q}{2} < a'_i \leq \frac{q}{2}$. The center-lift is unique.

Proof. Suppose the center-lift is not unique. Then there exists $a'(x) \equiv c(x) = a(x)$mod q. So for all coefficients, we have $a'_i \equiv c_i$ mod q. But as $a'_i, c_i \in (\frac{-q}{2}, \frac{q}{2}]$ $a'_i \equiv c_i$ mod q $\implies$ $a'_i = c_i$. As this is true for arbitrary i, we have $a'(x) = c(x)$, and we have a contradiction. Hence, the center-lift is unique. □

Thus, upon applying the lifting map, we recover $\alpha(x) = p \star g(x) \star r(x) + f(x) \star m(x)$ and decryption proceeds:

$$\begin{aligned}
b(x) &\equiv F_p(x) \star \alpha(x) \text{ mod p} \\
&\equiv F_p(x) \star (p \star g(x) \star r(x) + f(x) \star m(x)) \text{ mod p} \\
&\equiv p \star g(x) \star F_p(x) \star r(x) + F_p(x) \star f(x) \star m(x) \text{ mod p} \\
&\equiv F_p(x) \star f(x) \star m(x) \text{ mod p . } px \equiv 0 \text{ mod p for any x} \\
&\equiv m(x) \text{ mod p} \\
b(x) &= m(x)
\end{aligned}$$

3 NTRU As A Lattice Problem

The only known feasible attack method on NTRU cryptosystems is viewing it as a lattice-based problem [2]. The task of decrypting an NTRUEncrypt ciphertext can be formulated as an instance of the closest vector problem (CVP) [5], but here I focus on the SVP formulation. Consider the block matrix:

$$M_h^{NTRU} = \begin{pmatrix} I & h \\ 0 & qI \end{pmatrix}$$

where each component is a NxN matrix with coefficients in $R = \frac{\mathbb{Z}[x]}{x^N - 1}$. I is the identity matrix, 0 is the zero matrix, and h is the cyclical permutations of the coefficients of the public key $h(x)$. Solving the SVP in the lattice generated by M_h^{NTRU} allows for the recovery of the decryption keys. First, let us show that M_h^{NTRU} generates the lattice we want.

Lemma: M_h^{NTRU} is a basis in $\Re^{2n}$

Proof: The standard basis in $\Re^{2n}$ can be thought of as a block matrix $\begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$. Take an arbitrary set of scalars $a_1 ... a_{2n}$. Let $y \in \Re^n = (a_1, a_2 ... a_N)$. Let z_i be the vector that gives the $i'th$ column of the h block in the matrix M_h^{NTRU}. That is $z_0 = (h_0, h_{N-1} ... h_1)$.Then let $\nu_i = z_i \cdot y$. We can now show that M_h^{NTRU} spans $\Re^{2n}$:

Proof.

$$[a_1, a_2 ... a_{2n}] \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} = [a_1, a_2, ... a_{2n}]$$
$$= [a_1, a_2, ... a_n, \frac{a_{n+1} - \nu_0}{q}, \frac{a_{n+2} - \nu_1}{q} ... \frac{a_{2n} - \nu_{N-1}}{q}] \begin{pmatrix} I & h \\ 0 & qI \end{pmatrix}$$

□

Thus M_h^{NTRU} is a set of $2n$ vectors that spans R^{2n} and hence is a basis.

Now M_h^{NTRU} is a basis in $\Re^{2n}$ so we can consider the lattice generated by it. For $a(x) \in R, b(x) \in R$ let $(a, b) = (a_0, a_1 ... a_{N-1}, b_0, b_1, ... b_{N-1}) \in \mathbb{Z}^{2N}$.Using this notation, we can see that M_h^{NTRU}, due to its construction, has a special relation with the decryption keys.

Lemma: The lattice generated by M_h^{NTRU}, $L(M_h^{NTRU})$ is equal to the set $L_h^{NTRU} = \{(a, b) \in \mathbb{Z}^{2N}, a \star h \equiv b \text{ mod q}\}$

Proof. We need to show that $L(M_h^{NTRU}) = L_h^{NTRU}$. Split this into two cases:

(i) $L(M_h^{NTRU}) \in L_h^{NTRU}$ Consider some arbitrary $(x, y) \in \mathbb{Z}^{2n}$.

$$(x, y) \begin{pmatrix} I & h \\ 0 & qI \end{pmatrix} = (x, x \star h + yq)$$

Now let $(a, b) = (x, x \star h + yq)$. Clearly, $b - (a \star h) = (x \star h + yq) - x \star h = yq \implies a \star h \equiv b$ mod q. Thus for any arbitrary $(a, b) \in M_h^{NTRU}, (a, b) \in L_h^{NTRU}$. Thus $L(M_h^{NTRU}) \in L_h^{NTRU}$.

(ii)$L_h^{NTRU} \in L(M_h^{NTRU})$

Take any arbitrary $(a, b) \in L_h^{NTRU}$. By definition, $(a, b) = (a, a \star h + yq)$ for some $y \in \mathbb{Z}$ as $a \star h \equiv b$ mod q . Then we can see that:

$$(a, a \star h + yq) = (a, y) \begin{pmatrix} I & h \\ 0 & qI \end{pmatrix}$$

Thus every element in L_h^{NTRU} is also in $L(M_h^{NTRU})$. Thus $L_h^{NTRU} \in L(M_h^{NTRU})$.

Combining (i),(ii) we see that $L_h^{NTRU} = L(M_h^{NTRU})$ □

Now, the public key $h(x) = F_q \star g(x)$ in R_q. Hence, $h \star F_q^{-1} = h \star f(x) = g(x)$ in R_q. Thus $(f, g) \in L_h^{NTRU} = L(M_h^{NTRU})$.Now suppose we take standard NTRUEncrypt parameters[4]:

$$p = 3, d \approx N/3, q \approx 6pd = 2pN$$

Then, using the Gaussian Heuristic, the shortest vector in the lattice $L(M_h^{NTRU})$ has the expected length:

$$||v_{shortest}|| \approx \sigma(M_h^{NTRU}) = \sqrt{\frac{2N}{2\pi e}} (detL)^{1/(2n)} = \sqrt{\frac{N}{\pi e}} (q)^{1/(2)} = \sqrt{\frac{Nq}{\pi e}}$$
$$= N\sqrt{2p/(\pi e)} \approx 0.838N$$

As M_h^{NTRU} is an upper triangular matrix, its determinant is the product of its diagonal entries $q^N \cdot 1^N = q^N$. As f and g has (approximately) d coordinates equal to 1 and d coordinates equal to -1, $||(f, g)|| \approx \sqrt{4d} \approx \sqrt{(4N)/3} = 1.155\sqrt{N}$. Hence $\frac{||(f,g)}{\sigma(M_h^{NTRU})} \approx \frac{1.38}{\sqrt{N}}$.So $||(f, g)||$ is factor of $O(1/\sqrt{N})$ shorter than predicted by

the Gaussian heuristic. Thus, the shortest non-zero vectors are very likely to be (f, g) or some permutation of it.

However, in practice, it is possible that solving the SVP problem for the lattice does not return exactly $||(f, g)$ but rather some permutation of it or even some other short vector. It is sufficient for a malicious actor, Eve, to solve the approximate SVP problem and find a short vector that will likely serve as a decryption key. Elaborating on the exact bound for solving $apprSVP$ or the probability that the short vector found is out of scope for this paper, but more details can be found in [1]. It is easy to see that a short vector is likely to be a decryption key.

Proposition: Let (a, b) be a short vector in $L(M_h^{NTRU})$. Then (a, b) is likely to be a decryption key.

Sketch of Proof: We have the ciphertext $e \equiv pr \star h + m$. Now compute:

$$\begin{aligned} a \star e &\equiv a \star (pr \star h + m) \text{ mod q} \\ &\equiv pb \star r + a \star m \text{ mod q. } (a, b) \in M_h^{NTRU} = L_h^{NTRU} \text{ implies } a \star h \equiv b \text{ mod q.} \rightarrow (2) \end{aligned}$$

Now (2) is analogous to the equation (1) obtained by Alice during the normal decryption process. As (a, b) is a short vector in M_h^{NTRU}, $||b|| \approx ||g||$. Thus, analogous to the normal decryption process, it is highly likely that the largest coefficient is less than $q/2$. Eve has further leeway due to the fact that it is unlikely that the coefficients line up exactly to create the maximal value. Hence, Eve can lift (2) to R and obtain: $\alpha(x) = pb \star r + a \star m$ From here, Eve can calculate $F_p(x) = a(x)^{-1}$ in R_p using the extended Euclidean algorithm (which is fast) and proceed with decryption as normal. This illustrates the need to choose a large enough value of N to make such an attack intractable.

4 Parameter Selection And Runtime

A key advantage of lattice-based cryptosystems is their high speed [3]. The public key is also relatively memory efficient in that it needs $Nlog_2q$ bits as an N length vector where each entry is less than q. The key computationally expensive step in encryption/decryption is calculating the convolution products: $r \star h, f \star e, F_p \star a$.In all cases, at least one polynomial is a ternary polynomial, so rather than performing multiplications, convolution products can be computed with just additions and subtraction. Since each ternary polynomial has approximately $(2/3)N$ non-zero coefficients, computing each product requires approximately $(2/3)N^2$ additions/subtractions. Thus, NTRUEncrypt encryption should take approximately $O(N^2)$ steps where each step is just an addition/subtraction and hence extremely fast. The figure below experimentally validates this using a Python implementation of NTRUEncrypt. The runtime scales roughly quadratically, keeping in mind constant costs.

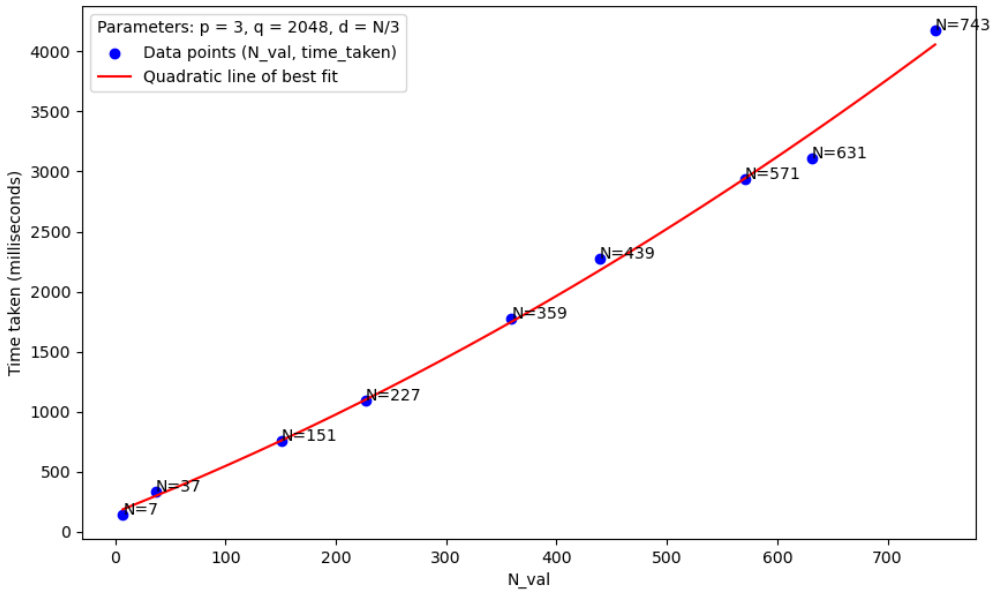

Figure 1: Runtime of 500 encryption/decryptions for different values of N

References

[1] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, 1998.

[2] Ieee standard specification for public key cryptographic techniques based on hard problems over lattices. *IEEE Std 1363.1-2008*, pages 1–81, March 2009.

[3] Jeffrey Hoffstein, Jill Pipher, and Silverman .H Joseph. *An Introduction to Mathematical Cryptography Second Edition*. Springer, 2014.

[4] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, pages 3–18, Cham, 2017. Springer International Publishing.

[5] Silverman.H Joseph. Ntru and lattice-based crypto: Past, present, and future, January 12–16, 2015.