Variants of NTRU Cryptosystem

CS681 – Extra Talk

Shaurya Johari
shauryaj23@iitk.ac.in

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

April 2025

Outline

Overview

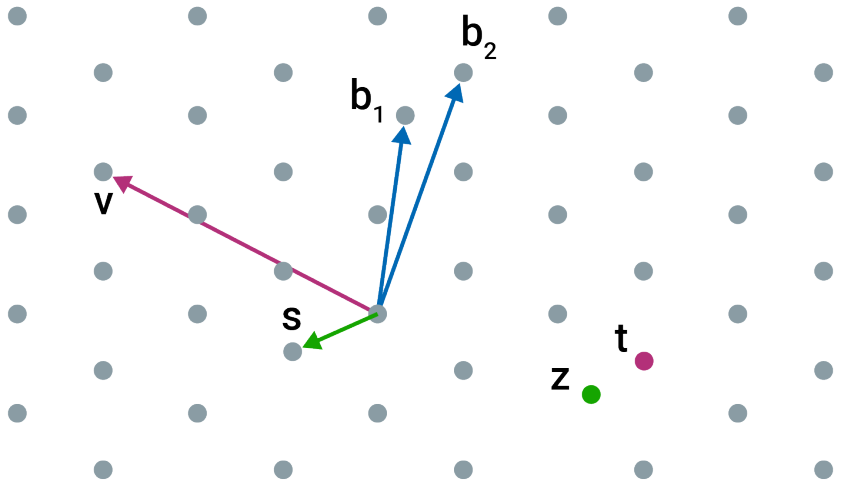
- **NTRU:** Lattice-based public key cryptosystem known for efficiency and security.
- **Applications:** Post-quantum cryptography, especially suited for constrained devices.
- **This Talk:** Explore different variants of NTRU and their design motivations.

Need of Strong Cryptography Schemes

- Quantum computers pose a major threat to current encryption methods, capable of breaking keys, certificates, and data protections within minutes.
- Quantum cyberattacks could rapidly compromise large networks and critical infrastructure, enabling cybercriminals to access sensitive data at unprecedented speed.
- Thus, to secure all sorts of data and make such systems capable of withstanding such breaches from quantum computers, there's a dire need to develop new cryptography protocols that don't break so easily.

What is a Lattice?

- A **lattice** is a discrete set of points in $\mathbb{R}^n$ generated by integer linear combinations of basis vectors.
- Mathematically: $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} a_i \mathbf{b}_i \mid a_i \in \mathbb{Z}\}$
- Lattices are central in modern cryptography, especially post-quantum schemes.



2D lattice formed by basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$

What is NTRU?

- **NTRU** stands for *Nth degree truncated polynomial ring units*.
- It is a lattice-based public key cryptosystem.
- Proposed in 1996 by Hoffstein, Pipher, and Silverman.
- Based on operations in a polynomial ring $\mathbb{Z}_q[x]/(x^N - 1)$.
- Efficient in both encryption and decryption, suitable for constrained devices.

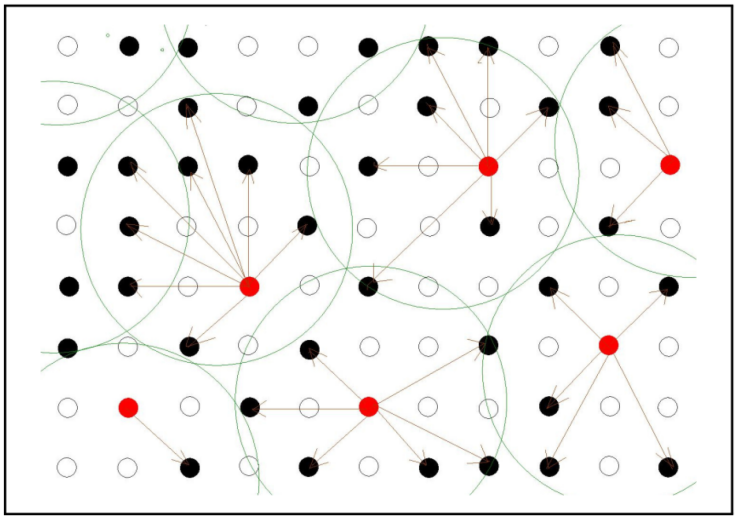
Why Study NTRU?

- **Post-Quantum Secure:** Resistant to attacks from quantum computers. Unlike RSA and ECC which can be decoded in short time by quantum algorithms, no such algorithms exist for NTRU yet.
- **Fast:** Faster than RSA and ECC in many implementations.
- **Small Key Sizes:** Compact public/private keys compared to other lattice-based schemes.

NTRU has found many useful applications in real life. For example, OpenSSH by default uses NTRU combined with the X25519 ECDH key exchange since August 2022

The Closest Vector Problem (CVP)

- **Problem:** Given a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target vector $\vec{t}$, find the closest lattice point $\vec{v} \in \mathcal{L}$ to $\vec{t}$.
- **Why it's hard:** CVP's NP hard ad there's no known polynomial-time algorithm to find the shortest vector in a lattice, and any polynomial-time algorithm for CVP could potentially be used to solve any problem in NP
- **Cryptographic relevance:** Many lattice-based cryptosystems, including NTRU, rely on the hardness of CVP or its variants.



NTRU and Closest Vector Problem (CVP)

Key Insight:

- In NTRU, the public key defines a lattice.
- Recovering the private key requires finding a short vector in this lattice.
- This is equivalent to solving a variant of the **Closest Vector Problem (CVP)**.

Why study CVP?:

- Understanding and solving CVP, or approximating it efficiently, is important for both theoretical reasons, such as exploring computational complexity, and practical reasons, such as designing and implementing the secure cryptographic scheme

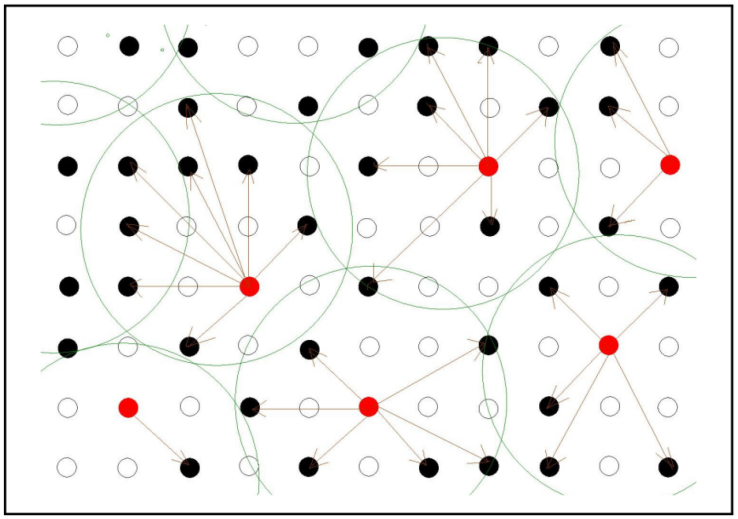


Illustration of CVP in 2D lattice

NTRU Scheme

- Based on polynomial rings over finite fields.
- Involves three main algorithms:
  - Key Generation
  - Encryption
  - Decryption
- We shall now go over these phases in the next few slides.

Notation and Setup

- Parameters: (N, p, q) with $\gcd(p, q) = 1$, $q \gg p$.
- Ring: $R = \mathbb{Z}[X]/(X^N - 1)$
- Polynomials are vectors:

$$F = \sum_{i=0}^{N-1} F_i x^i = [F_0, F_1, \ldots, F_{N-1}]$$

- Multiplication $(*)$ is cyclic convolution:

$$(F * G)_k = \sum_{i=0}^{N-1} F_i G_{(k-i) \bmod N}$$

- Multiplication mod q: reduce coefficients modulo q.

Key Generation

- Choose random polynomials $f, g \in R$ such that f is invertible mod p and q.
- Compute:

$$F_q * f \equiv 1 \mod q, \quad F_p * f \equiv 1 \mod p$$

- Public key: $h = F_q * g \mod q$
- Private key: f (and optionally F_p for decryption)

Encryption

- Shravan chooses message $m \in R$, random $r \in R$
- Ciphertext:

$$e = p \cdot r * h + m \mod q$$

- Sends e to Devansh

Decryption

- Devansh computes:

$$a = f * e \mod q, \quad \text{coefficients centered in } [-q/2, q/2]$$

- Then recovers message:

$$m = F_p * a \mod p$$

- Works since:

$$a = f * (p \cdot r * h + m) = p \cdot r * g + f * m$$

- If coefficients of a stay within range, decryption succeeds.

Remarks on Decryption

- Decryption may fail if coefficients overflow range.
- Adding check bits can help detect failure.
- Recovery possible by re-centering coefficients if near-boundary.
- Well-chosen parameters ensure high probability of success.

NTRU Parameters: Why p and q Matter

- NTRU encryption reduces modulo q, then decryption reduces modulo p.
- If q is too small, coefficients may wrap around after encryption.
- This leads to **information loss** during mod p — incorrect message recovery.

Constraints:

- p: small, prime (e.g., $p = 3$), invertible mod q.
- q: large prime or power of 2 (e.g., $q = 2048$).
- Ensure:

$$q > p \cdot (2d + 1) \cdot B$$

where:

  - d: non-zero the coefficients in private key ($\approx N/3$),
  - B: bound on coefficient growth from convolution.

Note on faulty parameter selection

Dangers of Poor Parameter Choice:

- Coefficients may exceed $\pm q/2$ post-encryption.
- Modulo q wraps incorrectly.
- Modulo p gives garbage output — message lost!

Good Practices:

- Use small p (typically 3), large q (2048+).
- Ensure all coefficients after decryption lie within:

$$\left(-\frac{q}{2}, \frac{q}{2}\right)$$

  before reducing mod p.
- Run tests on coefficient growth to validate bounds.

Summary: Always size q to handle noise from polynomial operations to **preserve recoverability** of the message.

Brute Force Attacks on NTRU

- Brute force aims to recover the private key by trying all possible short polynomial pairs (f, g).
- The number of such candidate pairs grows combinatorially:

$$\text{Number of tries} \sim \binom{N}{d_f} \cdot \binom{N}{d_g}$$

  where d_f and d_g are the numbers of ± 1 coefficients in f and g.
- For recommended parameter sets (e.g., $N = 167$), brute force is infeasible due to exponential growth.

Lattice Attacks on NTRU: Overview

- Goal: Recover private key (f, g) by solving a shortest vector problem (SVP).
- Construct a lattice $\mathcal{L}$ where (f, g) lies as a particularly short vector.
- Use lattice reduction (e.g., LLL, BKZ) to attempt recovery.
- Security relies on gap between true short vector and others in lattice.

Constructing the NTRU Lattice

- Define a $2N \times 2N$ basis matrix B_h using public key h:

$$B_h = \begin{bmatrix} I_N & H \\ O & qI_N \end{bmatrix}$$

  where H is the circulant matrix of h and I_N is the identity.

- (f, g) satisfies:

$$f * h \equiv g \mod q$$

  There exist a lattice (f, u) such that

$$(f, u)B_h = (f, g) \Rightarrow (f, g) \in \mathcal{L}(B_h)$$

Lattice Attack on e_i

- From the encryption equation:

$$e_i \equiv pr_i * h_j + f_i \pmod{q}$$

  we can express in vector form:

$$[0, e_i] \equiv [r_i, r_i * (ph_j)] + [-r_i, f_i] \pmod{q}$$

- The vector $[r_i, r_i * (ph_j)]$ lies in the lattice $\mathcal{L}_{ph_j}$.
- Thus, $[0, e_i]$ is close to a lattice point — it is off by the short vector $[-r_i, f_i]$.
- **Attack Idea:** Use CVP to find the closest vector in $\mathcal{L}_{ph_j}$ to $[0, e_i]$, recovering the short offset $[-r_i, f_i]$.

Time Complexity of LLL-Based Lattice Attacks

- **LLL Reduction:** Polynomial-time lattice basis reduction algorithm.
- **Runtime:**

$$\mathcal{O}(n^4 \log B) \quad \text{where } n = 2N,\ B = \text{entry bound}$$

- **Approximation Factor:**

$$\|v\| \leq 2^{(n-1)/2} \cdot \lambda_1$$

LLL finds a vector exponentially longer than the true shortest vector.

Why Lattice Attacks Fail with Good Parameters

- The private vector (f, g) is very short and not easily reached by reduction algorithms.
- Many moderately short vectors exist in the lattice, making it hard to find the true secret key.
- Proper parameter selection (e.g., large enough q) ensures a large "short vector gap".

Why Variants Exist

- Improve security against new attack vectors used for decoding
- Some NTRU variants aim to optimize the algorithm for speed and efficiency, making it more practical for real-world applications. Security: (e.g., smaller key sizes, faster computations).
- Adaptation of NTRU for different platforms done through Variants (IoT, servers, etc.).
- Some popular NTRU Variants: NTRUEncrypt, NTRUSign, NTRU Prime etc.

NTRUSign

- A digital signature scheme based on the same lattice principles as NTRUEncrypt.
- This uses trapdoors to generate short preimages in a lattice.
- Signature: a short vector s such that $H(s) = m$ modulo lattice structure.
- Original versions had issues with key leakage due to statistical patterns.
- Modern improvements mitigate leakage with better sampling techniques.

Key Generation

Input: Parameters $N, q, d_f, d_g, B \geq 0$; string $t \in \{\text{"standard"}, \text{"transpose"}\}$

1. Sample polynomials f, g with:
   - $f \leftarrow \text{Poly}(N, d_f, d_f)$
   - $g \leftarrow \text{Poly}(N, d_g, d_g)$
   - That is, f and g have d_f and d_g coefficients of ± 1, rest zero
2. Compute:
$$h = g \cdot f^{-1} \mod q$$
   where f^{-1} is computed in the NTRU ring modulo q.
3. **Output:**
   - Public Key: h
   - Secret Key: f

Signature Generation

Input: A digital document $D \in \mathcal{D}$ and the private key $\{f_i, f_i', h_i\}$ for $i = 0 \ldots B$

1. Set $r = 0$
2. Set $s = 0$. Let $i = B$. Encode r as a bit string. Set $m_0 = H(D \| r)$, where $\|$ denotes concatenation. Set $m = m_0$
3. **Perturb the point using the private lattices:** While $i \geq 1$:
   (a) Set $x = \lfloor (1/q) m * f_i \rfloor$, $y = \lfloor (1/q) m * f_i' \rfloor$
   (b) $s_i = x * f_i + y * f_i'$
   (c) $m = s_i + (h_i - h_{i-1}) \mod q$
   (d) $s = s + s_i$, set $i = i - 1$
4. **Sign the perturbed point using the public lattice:**

$$x = \lfloor (1/q) m * f_0 \rfloor, \quad y = \lfloor (1/q) m * f_0' \rfloor$$

$$s_0 = x * f_0 + y * f_0', \quad s = s + s_0$$

5. **Check the signature:**
   (a) Set $m' = \lfloor s * h \mod q \rfloor$
   (b) If $\|s\| \geq \mathcal{N}$, set $r = r + 1$ and go to step 3
6. **Output:** The triplet (D, r, s)

Signature Verification

Input: A signed document (D, r, s) and the public key h

1. Encode r as a bit string. Set $m = H(D \| r)$
2. Set $m' = \lfloor s * h \mod q \rfloor$
3. **Output:** Valid if $\|s\| < \mathcal{N}$, invalid otherwise

Note: Verification uses the same hash function H, norm function $\| \cdot \|$, and the norm bound $\mathcal{N} \in \mathbb{R}$

Security Considerations

- Based on hardness of lattice problems (e.g., SIS, SVP).
- Early versions vulnerable to signature leakage due to deterministic behavior.
- Later variants add perturbation/randomization to improve security.
- Still active research area with connections to post-quantum cryptography.

What is NTRU Prime?

- A lattice-based post-quantum cryptosystem
- A redesign of the original NTRUEncrypt, avoiding cyclotomic rings
- Introduced by Bernstein, Chuengsatiansup, Lange, and van Vredendaal in 2016
- Secure against quantum attacks; efficient in software

Need for NTRU Prime

- Classical NTRU used cyclotomic rings, leading to certain algebraic attacks
- NTRU Prime uses non-cyclotomic ring $\mathbb{Z}[x]/(x^p - x - 1)$
- Better security reductions and simpler implementations
- Resistance to quantum attacks and improved cryptographic robustness

Use Cases

- Secure key exchange in post-quantum TLS
- Lightweight cryptography for embedded systems
- Hybrid encryption schemes in quantum-safe communication
- Candidate in NIST Post-Quantum Cryptography Standardization

Key Generation in NTRU Prime

Inputs: Parameters (p, q, t)
Output: Public key h and private keys f, $1/g$

1. Choose random small $g \in R$ such that g is invertible in $R/3$
2. Choose t-small $f \in R$ (nonzero, hence invertible in R/q)
3. Compute $h = g/(3f)$ in R/q
4. Encode h as $\overline{h}$. This is the public key.
5. Save $f \in R$ and $1/g \in R/3$

Rings:

$$R = \mathbb{Z}[x]/(x^p - x - 1), \quad R/q = (\mathbb{Z}/q)[x]/(x^p - x - 1)$$

Streamlined NTRU Prime is actually a "key encapsulation mechanism" (KEM). This means that the sender takes a public key as input and produces a ciphertext and session key as output.

Encapsulation

Goal: Given public key $\overline{h}$, produce ciphertext $C\|c$ and session key K

1. From $\overline{h}$ Decode $h \in R/q$
2. Generate t-small $r \in R$
3. Compute $hr \in R/q$ to coefficients lying in $(\frac{-(q-1)}{2}, \frac{-(q-1)}{2}$, round coefficients to nearest multiple of $3 \Rightarrow c \in R$
4. Encode c as a string $\overline{c}$
5. Hash r using SHA-512 to get a left half C ("key confirmation") and a right half K. (C, K)
6. Ciphertext is $C\|c$ and session key is K

We refer to an element of R as small if all of its coefficients are in $\{1, 0, 1\}$. We refer to a small element as t-small if exactly $2t$ of its coefficients are nonzero, i.e., its Hamming weight is $2t$.

Decapsulation

Goal: Given $C \| c$ and private key $(f, 1/g)$, recover session key K

1. From $\overline{c}$ Decode $c \in R/q$
2. Compute $3fc$ in R/q to coefficients lying in $(\frac{-(q-1)}{2}, \frac{-(q-1)}{2}$, reduce coefficients modulo 3 to get $e = gr \in R/3$
3. Multiply e by $1/g$ in $R/3$ and lift it in R to get r'
4. Recompute c_0, C_0, K_0 from r' as in encapsulation.
5. If valid (t-small, $c_0 = c$, $C_0 = C$), output K_0, else output `False`

Conclusion

- Thus we have explored NTRU and few variants.

References

- Hoffstein, J., Pipher, J., Silverman, J.H. (1998). NTRU: A ring-based public key cryptosystem.
- Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C. (2017). NTRU Prime.
- Hoffstein, J., Pipher, J., Silverman, Whyte, W., (2002) NTRUSign: Digital Signatures using NTRU Lattices
- NIST PQC Project: `https://csrc.nist.gov/projects/post-quantum-cryptography`

Acknowledgements

- I would like to express my sincere gratitude to my professor Dr.Nitin Saxena for his invaluable guidance throughout this course, and Teaching Assistant Tufan Sir for helping me.
- I would also like to thank my like-minded peers Shravan Agrawal and Devansh Bhardwaj for engaging with me on many levels, sharpening my mind.
- I'm also Grateful to my peers, friends, and seniors for insightful discussions on other topics of Computational Number Theory and Algebra.

Thank You!

Questions are welcome.