

CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2025 Lecture 18

Announcements

1. HW 8 is due in one week!

Last time:

- Lattice-based cryptanalysis

This time:

- Post-quantum cryptography

Quantum computers might be coming

- The modern computing era is based off of digital computers: logical operations on binary bits.
- A quantum computer is a different structure of computer that takes advantage of quantum phenomena: superposition, interference, entanglement
- People have been saying quantum computers are 20–30 years away for the past 30 years
- Progress is being made on quantum computers (Google quantum supremacy experiment last year, Chinese quantum supremacy experiment this week)
- Big technical barrier: Noisy qubits. Doing error correction increases number of qubits required by a lot.

Quantum computers' impacts on cryptography

Quantum computers are not arbitrarily powerful: at a theoretical level they only seem to provide an exponential speedup for some problems.

- Grover's algorithm: Given a black-box function f and a value y , find x s.t. $f(x) = y$.
Classical search: $O(N)$ Quantum: $O(\sqrt{N})$
- Shor's algorithm: Use quantum Fourier transform to factor, compute finite field discrete logs, and elliptic curve discrete logs in polynomial time.
Exponential speedup over classical algorithms.

Quantum computers' impacts on symmetric crypto

At a theoretical level, Grover search gives square root speedup over classical.

⇒ Probably need to double (or more) symmetric key lengths.

Computation times are larger if quantum error correction is necessary.

- Breaking AES-128 is 2^{64} in theory, but 2^{101} with optimistic error rates.

<https://globalriskinstitute.org/publications/>

[resource-estimation-framework-quantum-attacks-cryptographic-functions/](https://globalriskinstitute.org/publications/resource-estimation-framework-quantum-attacks-cryptographic-functions/)

Quantum computers' impacts on symmetric crypto

At a theoretical level, Grover search gives square root speedup over classical.

⇒ Probably need to double (or more) symmetric key lengths.

Computation times are larger if quantum error correction is necessary.

- Breaking AES-128 is 2^{64} in theory, but 2^{101} with optimistic error rates.

<https://globalriskinstitute.org/publications/>

[resource-estimation-framework-quantum-attacks-cryptographic-functions/](https://globalriskinstitute.org/publications/resource-estimation-framework-quantum-attacks-cryptographic-functions/)

- For hash functions, depends on application:
 - Digital signatures: Easier to attack public-key signature scheme instead.
 - Password hashing: Grover search could give square root speedup for dictionary searching.
 - Cryptocurrency mining: Hash-based proof of work unlikely to be faster to break with quantum computers than classical.

Quantum computers' impacts on asymmetric crypto

Summary: Bad.

- RSA-2048: 4338 logical qubits, 6.2×10^6 physical qubits, 29 hours.
- NIST P-256: 2330 logical qubits, 3.21×10^6 physical qubits, 11 hours.

<https://globalriskinstitute.org/publications/>

[resource-estimation-framework-quantum-attacks-cryptographic-functions-part-2-rsa-ecc/](https://globalriskinstitute.org/publications/resource-estimation-framework-quantum-attacks-cryptographic-functions-part-2-rsa-ecc/)

Classical part of Shor's algorithm for factoring

Input: Integer N to be factored.

Intuition: Want to find s s.t. $s^2 = 1 \pmod N$ and $s \neq -1, 1$.

Then $s^2 - 1 = (s + 1)(s - 1) \equiv 0 \pmod N$

and we hope that either $s + 1$ or $s - 1$ is a nontrivial factor.

Classical part of Shor's algorithm for factoring

Input: Integer N to be factored.

Intuition: Want to find s s.t. $s^2 \equiv 1 \pmod{N}$ and $s \neq -1, 1$.

Then $s^2 - 1 = (s + 1)(s - 1) \equiv 0 \pmod{N}$

and we hope that either $s + 1$ or $s - 1$ is a nontrivial factor.

Quantum Fourier transform allows one to compute a multiplicative group order: Given x find r s.t. $x^r \equiv 1 \pmod{N}$.

To factor, we use QFT to construct a square root.

1. Choose a random x .
2. Compute the order r of x .
3. If r is even, let $s = x^{r/2}$ and check if $\gcd(s - 1, N)$ factors N .

Details: $1/2$ of elements of $\mathbb{Z}/N\mathbb{Z}^*$ have even order; bounded pr. that $x^{r/2} \equiv -1 \pmod{N}$.

Classical part of Shor's algorithm for discrete logs

Input: Target x , generator g , modulus p , order q of g .

1. Define $f(a, b) = x^a g^{-b} \bmod p$.
2. Use Quantum order-finding algorithm to find a, b s.t.
 $f(a, b) = 1$.
- 3.

$$x^a g^{-b} \equiv 1 \pmod{p}$$

$$a \log_g x - b \equiv 0 \pmod{q}$$

$$\log_g x \equiv ba^{-1} \pmod{q}$$

Cryptographic solutions for quantum computers

- Quantum Key Distribution
 - Use quantum behavior to transmit key material; eavesdropper can be detected because measurement collapses state.
 - Requires line of sight or dedicated fiber-optic cables between every node and authenticated classical channel.
 - Not thought to be a realistic solution by computer scientists.
- Post-Quantum Cryptography
 - Develop encryption schemes that work on classical computers but are secure against quantum computers.
 - Allows continued use of existing communications infrastructure.
 - NIST and crypto community are hard at work doing this right now.

Candidate post-quantum algorithms

- Key exchange/key encapsulation/public-key encryption
 - Lattice-based schemes: NTRU, LWE, Ring-LWE
 - Code-based schemes: McEliece
 - Multivariate Quadratic Schemes
 - Supersingular Isogenies
- Digital Signatures
 - Lattice-based schemes
 - Multivariate quadratic schemes
 - Hash-based signatures

NTRU Cryptosystem

Parameters and setup

- Polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$, relatively prime integers p , q
- **Private key** $f, g \in R$ with coefficients in $\{0, \pm 1\}$
- f is invertible mod p and mod q
- **Public key** $h = gf^{-1} \bmod q$

NTRUEncrypt

1. Encode message as binary or ternary polynomial m .
2. Generate random polynomial r with coefficients in $\{0, \pm 1\}$.
3. $c = pr \cdot h + m \bmod q$

Decryption

1. Compute $a = c \cdot f = pr \cdot h \cdot f + m \cdot f = pr \cdot g + m \cdot f$
2. Reduce $a \bmod p$ to get $m \cdot f$
3. Multiply by $f^{-1} \bmod p$ to get $m \bmod p$.

The NTRU Problem

Given the public key h recover the secret key f, g .

Coppersmith–Shamir attack [CS97]

Construct the lattice:

$$A = \left(\begin{array}{cccc|cccc} & \text{f-part} & & & \text{g-part} & & & \\ & \text{---} & & & \text{---} & & & \\ 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & 1 & \dots & 0 & -h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & -h_1 & -h_2 & \vdots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \vdots & q \end{array} \right) = \begin{bmatrix} I & M_h \\ 0 & qI \end{bmatrix}_{2n \times 2n}$$

$L(A)$ contains the vector $(f, g) = (f_0, \dots, f_{n-1}, g_0, \dots, g_{n-1})$.

Vector is short: coefficients of f, g are $\pm 1, 0$. Hope LLL will find it.

Countermeasure: Choose parameters so that LLL cannot find the target.

Learning With Errors (LWE)

Learning With Errors

Fix $s \in \mathbb{Z}_q^n$.

Input: m samples $(a_i, b_i = \langle a_i, s \rangle + e_i \bmod q)$

for randomly chosen $a_i \in \mathbb{Z}_q^n$ and error $e_i \in \mathbb{Z}_q$.

Goal: Find s .

$$\begin{bmatrix} \text{---} & a_1 & \text{---} \\ & \vdots & \\ \text{---} & a_m & \text{---} \end{bmatrix} \begin{bmatrix} s \\ \vdots \\ s \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

- Generalization of Learning Parity with Noise (LPN) problem from machine learning.

LWE as a lattice problem

Learning With Errors

Fix s .

Input: m samples $(a_i, b_i = \langle a_i, s \rangle + e_i \bmod q)$
for randomly chosen a_i and error e_i .

Goal: Find s .

$$\begin{bmatrix} \text{---} & a_1 & \text{---} \\ & \vdots & \\ \text{---} & a_m & \text{---} \end{bmatrix} \begin{bmatrix} s \\ \vdots \\ s \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

- Define $L(A) = \{y \in \mathbb{Z}^m : y = As \bmod q \text{ for some } s \in \mathbb{Z}^n\}$.
- For e_i small, LWE asks to solve a closest vector problem on the lattice L .

Decisional LWE

Decisional LWE Problem

Distinguish samples from LWE distribution from samples from uniform distribution.

Random self-reduction

Given a sample $(a, b = \langle a, s \rangle + e)$, then

$$(a, b + \langle t, a \rangle) = (a, \langle s + t, a \rangle + e)$$

is a valid instance.

Decisional LWE

Decisional LWE Problem

Distinguish samples from LWE distribution from samples from uniform distribution.

Reduction from search to decision

Assume we can distinguish LWE distribution from uniform.

For all k up to q do:

1. Sample instances (a, b) from LWE.
2. Map (a, b) to $(a + (r, 0, \dots, 0), b + rk)$.

Try to distinguish from uniform. If k is correct, then will look like proper LWE sample. If k is incorrect, then it will look uniform.

Public key encryption based on LWE

Private key: s

Public key: m samples $\{(a_i, b_i = \langle a_i, s \rangle + e_i)\}$ from LWE dist.

Encrypt: Chose random subset S of LWE samples. Compute

$$A_S = \sum_S a_i, B_S = \sum_S b_i.$$

$$\text{Enc}(\text{bit}) = \begin{cases} (A_S, B_S) & \text{if } 0 \\ (A_S, B_S + q/2) & \text{if } 1 \end{cases}$$

Decrypt:

$$\text{Dec}(A_S, B_S) = \begin{cases} 0 & \text{if } B_S - \langle A_S, s \rangle \approx 0 \\ 1 & \text{if } B_S - \langle A_S, s \rangle \approx q/2 \end{cases}$$

Public key encryption based on LWE

Private key: s

Public key: m samples $\{(a_i, b_i = \langle a_i, s \rangle + e_i)\}$ from LWE dist.

Encrypt: Chose random subset S of LWE samples. Compute

$$A_S = \sum_S a_i, B_S = \sum_S b_i.$$

$$\text{Enc}(\text{bit}) = \begin{cases} (A_S, B_S) & \text{if } 0 \\ (A_S, B_S + q/2) & \text{if } 1 \end{cases}$$

Decrypt:

$$\text{Dec}(A_S, B_S) = \begin{cases} 0 & \text{if } B_S - \langle A_S, s \rangle \approx 0 \\ 1 & \text{if } B_S - \langle A_S, s \rangle \approx q/2 \end{cases}$$

Proof of security

- Chosen plaintext attacks: Have algorithm to guess encrypted bit for non-negligible fraction of secrets.
- Then can distinguish encryptions with LWE distribution from encryptions with random vectors, and thus distinguish LWE distribution from random. (Decisional LWE)

Downside of LWE-based encryption:

Large public keys: An $n \times m$ matrix large enough that lattice algorithms are inefficient.

Ring-LWE

Luybashevsky, Peikert, Regev

Ring-LWE

Let $R = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$. Fix $s \in R$ uniformly at random.

Input: m samples $(a_i, b_i = a_i \cdot s + e_i)$, random $a_i, e_i \in R$.

Goal: Find s .

Ring-LWE

Luybashevsky, Peikert, Regev

Ring-LWE

Let $R = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$. Fix $s \in R$ uniformly at random.

Input: m samples $(a_i, b_i = a_i \cdot s + e_i)$, random $a_i, e_i \in R$.

Goal: Find s .

Public-key cryptography based on Ring-LWE

Secret Key $s \in R$.

Public Key $(a, b = a \cdot s + e)$, $a, e \in R$.

Encryption Choose random small $r, e_1, e_2 \in R$. Output

$$u = a \cdot r + e_1 \bmod q \quad v = b \cdot r + e_2 + \lfloor q/2 \rfloor \cdot m \bmod q$$

Decryption $v - u \cdot s = (r \cdot e - s \cdot e_1 + e_2) + \lfloor q/2 \rfloor \cdot m \bmod q$

Recover m by rounding.

Ring-LWE allows much smaller public keys because you only need to store one ring element rather than a large matrix.

Downside: Potential for attacks exploiting algebraic structure of ring lattice.

Number fields for ideal lattices

Definition

A **number field** is a finite-degree extension $K = \mathbb{Q}(\alpha)$, α a root of irreducible polynomial $f \in \mathbb{Z}[x]$.

Papers in the area use cyclotomic fields $K = \mathbb{Q}(\zeta)$ ζ a root of unity.

Fact

If $\deg f = n$, elements of K are polynomials in α of degree $\leq n - 1$:

$$\mathbb{Q}(\alpha) = \{a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} : a_0, \dots, a_{n-1} \in \mathbb{Q}\}$$

n is **degree** of K (the dimension of K as a \mathbb{Q} -vector space).

Definition

The **ring of integers** $\mathcal{O}_K \subseteq K$ consists of all elements of K that are roots of monic polynomials in $\mathbb{Z}[x]$.

Ideal lattices

Definition

There are n **embeddings** of K into \mathbb{C} $\sigma_i : \alpha \rightarrow \alpha_i$ for α_i roots of f . Each embedding gives rise to an absolute value $|\gamma|_i = |\sigma_i(\gamma)|$.

Consider an ideal $I \subseteq \mathcal{O}_K$.

Fact

An ideal $I \subseteq \mathcal{O}_K$ has an **integral basis** b_1, \dots, b_n .

Definition

An **ideal lattice** is the *canonical embedding* of an ideal $I \subseteq \mathcal{O}_K$:

$$\begin{bmatrix} \sigma_1(b_1) & \sigma_2(b_1) & \dots & \sigma_n(b_1) \\ \sigma_1(b_2) & \sigma_2(b_2) & & \sigma_n(b_2) \\ \vdots & & & \vdots \\ \sigma_1(b_n) & \sigma_2(b_n) & \dots & \sigma_n(b_n) \end{bmatrix}$$

Reduction from ideal SVP to Ring-LWE

Theorem (Lyubashevsky, Peikert, Regev)

There is a polynomial-time quantum reduction from $O(\sqrt{n} \log n / \alpha)$ -approximate SVP for cyclotomic number field ideal lattices to Ring-LWE with $\text{poly}(n)$ samples when $\alpha q > \omega(\sqrt{n})$.

Quantum security: Current best attacks try to apply Grover speedup to lattice algorithms. (No proof of hardness.)

NIST Encryption Recommendation

- ML-KEM (Crystals-Kyber): Module-LWE

Supersingular isogeny Diffie-Hellman key exchange (SIDH/SIKE)
taken out in spectacular fashion July 2022.

Hash-based signatures

Intuition: One-bit Lamport signatures.

- Secret key: Random values (x_0, x_1)
- Public key: $(H(x_0), H(x_1)) = (y_0, y_1)$

- $\text{Sign}(m) = x_m$ for $m \in \{0, 1\}$.
- $\text{Verify}(\sigma, m)$: Check that $H(\sigma) = y_m$.

256-bit Lamport Signatures

- Secret key: 2×256 random values $\begin{pmatrix} x_{0,0} & x_{0,1} & \dots & x_{0,255} \\ x_{1,0} & x_{1,1} & \dots & x_{1,255} \end{pmatrix}$
- Public key: $\begin{pmatrix} y_{0,0} & y_{0,1} & \dots & y_{0,255} \\ y_{1,0} & y_{1,1} & \dots & y_{1,255} \end{pmatrix}$, $y_{i,j} = H(x_{i,j})$
- $\text{Sign}(m = m_0 m_1 \dots m_{255}) = (x_{m_0,0}, x_{m_1,1}, \dots, x_{m_{255},255})$ for $m \in \{0, 1\}^{256}$.
- $\text{Verify}(\sigma = (\sigma_0, \dots, \sigma_{255}), m)$: Check $H(\sigma_i) = y_{m_i,i}$.

Making hash-based signatures practical

- Need a many-time signature algorithm
- Some variants require signer to keep state: usability issue
- Want to compress public and secret keys

Quantum security: 2^{128} for a 256-bit hash function via Grover search

Downside: Relatively large signatures

NIST Digital Signature Recommendations

- ML-DSA (Crystals-Dilithium): Module-LWE
- FN-DSA (Falcon): Variant of NTRU
- SLH-DSA (SPHINCS+): Stateless hash-based signatures
- Ongoing: Additional Digital Signature Schemes (maybe multivariate?)

Multivariate quadratic scheme Rainbow taken out in spectacular fashion January 2022.

