FIT5124 Advanced Topics in Security

Lecture 4: Lattice-Based Crypto. IV

Ron Steinfeld
Clayton School of IT
Monash University

March 2016

Plan for this lecture

- **How to construct lattice-based encryption schemes? (continued)**
  - Security of LWE: How to choose parameters for a given security level?
  - Efficiency Considerations: How to make lattice-based crypto. practical
    - **Multibit encryption**: Reducing ciphertext expansion
    - **Structured Lattices (Ring-LWE): Reducing key length and computation time**

Security of Learning with Errors (LWE) Problem

Why do we believe LWE is hard?

Theoretical Reason: Analogue of Ajtai's average-case to worst-case connection Theorem for SIS can also be established for LWE (Regev 2005 [Reg05]):

Theorem

If there is an algorithm A that solves Decision-LWE$_{q(n),m(n),n,\alpha(n)}$ *in poly-time, with* **non-negligible distinguishing advantage**, *for* $\alpha(n) \cdot q(n) > 2\sqrt{n}$

Then there is a quantum algorithm B that solves $\gamma(n)$*-GapSVP in polynomial time for* **all** *input lattices L of dimension n with:*

$$\gamma = \widetilde{O}(n/\alpha).$$

- $\gamma(n)$-GapSVP is a decision variant of $\gamma(n)$-SVP that asks, given a basis B for an n-dim. lattice L and an integer d, to decide whether $\lambda_1(L) \leq d$, or $\lambda_1(L) > \gamma(n) \cdot d$.
- More recent improvements to this result allow B to be a classical algorithm if either $q > 2^{n/2}$ [Pei09], or the dimension of the lattice input to B is $\sqrt{n}$ [B13].
- We won't study this proof, but it gives us a theoretical foundation for security of LWE.

Learning with Errors (LWE) Problem - Practical Security

Why do we believe LWE is hard?

Practical Reason: In most cases, essentially best known attack on Decision LWE is a reduction of LWE to SIS.

Given an LWE instance $(A \in \mathbb{Z}_q^{m \times n}, \vec{y} \in \mathbb{Z}_q^m)$:

- Find a short non-zero vector $\vec{v}$ in SIS lattice $L_q^{\perp}(A^T)$ with $\|\vec{v}\| \leq \beta$ (i.e. solve β-SIS for A^T).
  - Note that $A^T \cdot \vec{v} = \vec{0} \bmod q$, or $\vec{v}^T \cdot A = \vec{0}^T \bmod q$.
- Compute $e' = \vec{v}^T \cdot \vec{y} \bmod q$.
  - In 'Real LWE Scenario' ($\vec{y} = A \cdot \vec{s} + \vec{e}$): $e' = \vec{v}^T \cdot \vec{e} \bmod q$. Since $\vec{e}$ and $\vec{v}$ are both 'small', so is e': for fixed $\vec{v}$, e' is normally distributed with std. dev. $\|\vec{v}\| \cdot \alpha q$, so is 'small' if $\vec{v} \cdot \alpha q << q$, or
  $$\beta = \|\vec{v}\| << 1/\alpha.$$
  - In 'Random LWE Scenario' ($\vec{y}$ uniform in $\mathbb{Z}_q^m$): $e' = \vec{v}^T \cdot \vec{e} \bmod q$ is uniformly random in $\mathbb{Z}_q$, not likely to be 'small' compared to q
- If $|e'| < q/4$, Return 'REAL LWE', else return 'Random LWE'.

Conclusion: Solving Decision $\text{LWE}_{q,m,n,\alpha}$ reduces to solving $\text{SIS}_{q,m,n,\beta \approx 1/\alpha}$. Choose parameters so that $\text{SIS}_{q,m,n,\beta \approx 1/\alpha}$ is hard!

Learning with Errors (LWE) Problem - Practical Security

The condition $\alpha q > 2\sqrt{n}$ from Regev's security reduction is important to security (in general)!

- LWE insecure when $\alpha q \approx 1$ and m is sufficiently large ($\geq m^2$)!!
- Idea: Algebraic attacks!

Efficiency Considerations in Lattice-Based Crypto.

Recall Regev's public-key encryption scheme [Reg05]:

- Public key $pk = (A \hookleftarrow U(\mathbb{Z}_q^{m \times n}, \vec{p} = A \cdot \vec{s} + \vec{e} \bmod q)$ with $\vec{e} \hookleftarrow \chi_{\alpha q}^m$.
  - **Length(pk):** $= m \cdot (n+1) \log q \geq n^2 \log q$ bits — at least quadratic in sec. par λ: $O(\lambda^2)$!!
- Secret key $\vec{s} \in \mathbb{Z}_q^n$.
- Encryption – Enc ($m \in \mathbb{Z}_t$): Return ciphertext $C = (\vec{a}^T = \vec{r}^T \cdot A \bmod q, c = \vec{r}^T \cdot \vec{p} + \lceil q/t \rfloor \cdot m \bmod q)$.
  - **Ciphertext expansion ratio:** $= \frac{\text{Length}(C)}{\text{Length}(m)} = \frac{(n+1) \cdot \log q}{\log t}$ – at least linear in sec. par. λ: $n + 1 = O(\lambda)$!!.
  - **Encryption time:** $O(mn \log q)$ bit ops. – at least quadratic in λ: $O(\lambda^2)$!!
- Decryption – Dec ($C = (\vec{a}^T, c)$): Compute $c' = c - \vec{a}^T \cdot \vec{s} \bmod q$, round to nearest multiple of $\lceil q/t \rfloor$ mod q to get c''. Return plaintext $m = \frac{\vec{c''}}{\lceil q/t \rfloor}$.

Efficiency Considerations: Ciphertext Expansion

Reducing ciphertext expansion ratio in Regev encryption

Observe: The $\vec{a}^T$ component of ciphertext encodes only enc. randomness, not message bits.

Idea ([PVW08]): 'Reuse' this randomness with new secrets $\vec{s}_i$:

Modified Regev Scheme (ℓ = number of secret key vectors):

- Public key $pk = (A \hookleftarrow U(\mathbb{Z}_q^{m \times n}, P = (\vec{p}_1, \ldots, \vec{p}_\ell)$ where $\vec{p}_i = A \cdot \vec{s}_i + \vec{e}_i \bmod q)$ with $\vec{e}_i \hookleftarrow \chi_{\alpha q}^m$.
  - **Length(pk):** $= m \cdot (n + \ell) \cdot \log q - \approx (1 + \ell/n)$-times larger than orig. scheme ($\ell = 1$).
- Secret key $S = (\vec{s}_1, \ldots, \vec{s}_\ell) \in \mathbb{Z}_q^{n \times \ell}$ – ℓ times longer but not in practical storage!
- Encryption – Enc ($\vec{m} \in \mathbb{Z}_t^\ell$): Return ciphertext

  $C = (\vec{a}^T = \vec{r}^T \cdot A \bmod q, \vec{c}^T = \vec{r}^T \cdot P + \lceil q/t \rfloor \cdot \vec{m} \bmod q)$.
  - **Ciphertext expansion ratio:** $\frac{\text{Length}(C)}{\text{Length}(\vec{m})} = \frac{(n+\ell)\cdot \log q}{\ell \log t} = (1 + \frac{n}{\ell}) \cdot \frac{\log q}{\log t}$
    - If $q = t^{O(1)}$, expansion ratio $= O(1)$ for $\ell \geq n$!
  - **Encryption time:** $O(m(n+\ell) \log q)$ bit ops – $\approx (1 + \ell/n)$-times larger than orig. scheme ($\ell = 1$).
- Decryption – Dec ($C = (\vec{a}^T, \vec{c}^T)$): Compute $(\vec{c}')^T = \vec{c}^T - \vec{a}^T \cdot S \bmod q$, round to nearest multiple of $\lceil q/t \rfloor \bmod q$ to get c''. Return plaintext $\vec{m} = \frac{c''}{\lceil q/t \rfloor}$.

Efficiency Considerations: Ciphertext Expansion

Q: But, why is reusing $\vec{a}^T$ still as secure as LWE?

A: Security reduction from LWE – example of 'hybrid argument'. Suppose there was an efficient IND-CPA attack algorithm B, breaking 2^λ security of Regev's encryption scheme:

- B runs in time T_B and wins IND-CPA game with prob. $1/2 + \varepsilon_B$ (with $T_B < 2^\lambda$ and non-neg. $\varepsilon_B > 1/2^\lambda$).

Then, we construct ℓ Dec-LWE algorithms, $\mathsf{D}_1, \ldots, \mathsf{D}_\ell$ such that at least one D_i advantage $\geq \frac{\varepsilon_B - 1/2^{\lambda+1}}{\ell} \geq 1/2^{\lambda+1+\log \ell}$.
Given Dec-LWE instance $(q, n, A, \vec{y})$, D_i does following:

- D_i runs attacker B on input public key $(A, P = (\vec{p}_1, \ldots, \vec{p}_\ell))$, where
  - For $j = 1, \ldots, i-1$, D_i sets $\vec{p}_j = A \cdot \vec{s}_j + \vec{e}_j \bmod q$, where $\vec{s}_j \hookleftarrow U(\mathbb{Z}_q^n)$ and $\vec{e}_j \hookleftarrow \chi_{\alpha q}^m$ are sampled independently by D_i.
  - For $j = i$, D_i sets $\vec{p}_i = \vec{y}$.
  - For $j = i+1, \ldots, \ell$, D_i samples independent $\vec{p}_i \hookleftarrow U(\mathbb{Z}_q^m)$.
- When B makes its challenge query $(\vec{m}_0, \vec{m}_1)$, D_i behaves like the real challenger: chooses a random bit b, picks coefficient vector $\vec{r} \hookleftarrow U(\{-B_r, \ldots, B_r\}^m)$ and computes:

$$\vec{a}^T = \vec{r}^T \cdot A, \vec{c}^T = \vec{r}^T \cdot P + \lceil q/t \rfloor \cdot \vec{m}_b \bmod q.$$

  D_i returns challenge ciphertext $(\vec{a}^T, c)$.
- When B returns a guess b' for b, D returns 'Real' if $b' = b$, and 'Rand' if $b' \neq b$.

Efficiency Considerations: Ciphertext Expansion

'Reusing' $\vec{a}^T$ security reduction (cont.): Consider two LWE scenarios for $\vec{y}$ as input to D_i:

- 'Real' LWE scenario, $\vec{p}_i = \vec{y} = A \cdot \vec{s} + \vec{e} \bmod q$ – first i vectors $\vec{p}_1, \ldots, \vec{p}_i$ in public key are computed exactly as in the real IND-CPA game, remaining $\ell - i$ vectors $\vec{p}_{i+1}, \ldots, \vec{p}_\ell$ are random.
  - Call this distribution of P (first i $\vec{p}_j$'s 'real', last $\ell - i$ $\vec{p}_j$'s 'random') the ith 'hybrid' distribution.
  - Define the winning probability of B for ith 'hybrid' distribution of P as $p_i = 1/2 + \varepsilon_i$, hence D_i returns 'Real' with prob. $1/2 + \varepsilon_i$.
  - Note two extreme values of p_j are known:
    - $p_0 \leq 1/2 + 1/2^{\lambda+1}$ (all $\vec{p}_j$'s uniformly random) by LHL argument (as before), except the LHL condition becomes $(2B_r + 1)^m >> q^{n+\ell}$.
    - $p_\ell = 1/2 + \varepsilon_B$ (all $\vec{p}_j$'s as in real IND-CPA game) by assumption on B.
- 'Random' LWE scenario, $\vec{p}_i = \vec{y} \hookleftarrow U(\mathbb{Z}_q^m)$ – first $i - 1$ vectors $\vec{p}_1, \ldots, \vec{p}_{i-1}$ in public key are computed exactly as in the real IND-CPA game, remaining $\ell - i + 1$ vectors $\vec{p}_i, \ldots, \vec{p}_\ell$ are random.
  - This is the $(i - 1)$'th hybrid distribution of P.

So: Distinguishing advantage of D_i is $\Delta_i = |p_i - p_{i-1}|$.

- Since $p_\ell - p_0 \geq \varepsilon_B - 1/2^\lambda$, one of ℓ Δ_i's (say $i = i^*$) must be $\geq (\varepsilon_B - 1/2^\lambda)/\ell \geq 1/(\ell \cdot 2^{\lambda+1})$.

Conclusion: D_{i^*} contradicts the $2^{\lambda+1+\log \ell}$-security of LWE!

Reducing Storage and Computation: Structured Lattices

How to reduce quadratic stored key length of matrix A?
Recall A is a random $m \times n$ matrix with $m \geq n$ – number of elements $m \cdot n \geq n^2$:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

Idea: Reuse some $a_{i,j}$'s in matrix, only store them once!

- Structured matrices / lattices!

But, how to do it securely?

Reducing Storage and Computation: Structured Lattices

Idea: [HPS96,M02] Replace $m \times n$ random matrix A (entropy $\widetilde{O}(n^2)$) with m/n blocks of $n \times n$ negacyclic square matrices (entropy $\widetilde{O}(n)$): Use $n \times n$ negacyclic 'rot' matrices. For an n-dim. vector $\vec{a} \in \mathbb{Z}^n$, define

$$\mathrm{rot}(\vec{a}) = \begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix}$$

to build

$$A = \begin{bmatrix} \mathrm{rot}(\vec{a}_1) \\ \mathrm{rot}(\vec{a}_2) \\ \vdots \\ \mathrm{rot}(\vec{a}_{m/n}) \end{bmatrix}.$$

Correspondence with Polynomial Ring $R_q = Z_q[x]/(x^n + 1)$

rot matrix-vector product $\equiv$ Polynomial Mult. mod $x^n + 1$: A polynomial $a(x) = a_0 + a_1 \cdot x \ldots + a_{n-1} \cdot x^{n-1}$ with $a_i \in \mathbb{Z}_q$ can be represented by its coefficient vector $\overrightarrow{a(x)}$:

$$\overrightarrow{a(x)}^T = [a_0, a_1, \ldots, a_{n-1}] \in \mathbb{Z}_q^n.$$

- For two polynomials $a(x), s(x) \in \mathbb{Z}_q[x]$ of deg. $< n - 1$, let $c(x) = a(x) \cdot s(x) \bmod x^n + 1$.
  - $c(x) = \sum_{i<n} s_i x^i a(x) \bmod x^n + 1$.
  - $x \cdot (a_0 + a_1 x + a_2 x^2 \cdots a_{n-1} x^{n-1}) \bmod x^n + 1 = -a_{n-1} + a_0 x + a_1 x^2 + \cdots + a_{n-2} x^{n-1}$.
- Hence can write $c(x)$ as vector-matrix product $\overrightarrow{c} = \mathrm{rot}(\overrightarrow{a(x)}) \cdot \overrightarrow{s(x)} \bmod q$:

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix}.$$

Correspondence with Polynomial Ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$

Set of polynomials $\{a(x) = a_0 + a_1 \cdot x \ldots + a_{n-1} \cdot x^{n-1} : a_i \in \mathbb{Z}_q\}$ of degree $< n$ with $\mathbb{Z}_q$ coefficients forms a polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$ under the operations

- polynomial addition modulo x^n+1 – corresponds to addition of coefficient vectors:

$$\overrightarrow{a(x) + b(x) \bmod x^n + 1} = \overrightarrow{a(x)} + \overrightarrow{b(x)}.$$

- polynomial multiplication modulo x^n+1 – corresponds to (rot-matrix) times (coefficient vector) product:

$$\overrightarrow{a(x) \cdot b(x) \bmod x^n + 1} = (\mathrm{rot}\overrightarrow{a(x)}) \cdot \overrightarrow{b(x)}.$$

with the operations on the coefficients performed in $\mathbb{Z}_q$ (i.e. modulo q). (When working in R_q, we won't write $\bmod x^n+1$ (understood)).

Sometimes, also refer to ring $R = \mathbb{Z}[x]/(x^n+1)$: same as R_q except coefficients arithmetic is in $\mathbb{Z}$ (not mod q).

Reducing Computation: FFT

Q: How does correspondence to polynomial multiplication help?
A: Use fast polynomial multiplication algorithms to speed up $\mathrm{rot}(\vec{a}) \cdot \vec{s}$ computation!

- Use $O(m/n \cdot n \log n)$ add./mult. ops. over $\mathbb{Z}_q$ instead of $O(m/n \cdot n^2)$!

Idea: Reduce to (Number Theory) Fast Fourier Transform (FFT) computations For $a(x), s(x) \in \mathbb{Z}_q[x]$, to compute $c(x) = a(x) \cdot s(x) \bmod x^n + 1$, (deg. of $a(x), s(x) < n$):

- Choose q such that $2n$ divides $q - 1$.
  - Then $x^n + 1$ has n zeros in $\mathbb{Z}_q$ of the form ζ^{2i+1} for $i = 0, \ldots, n-1$, where $\zeta \in \mathbb{Z}_q$ is a primitive $2n$th root of 1 in $\mathbb{Z}_q$.
- Evaluate $a(x)$ and $b(x)$ at the n points ζ^{2i+1} in $\mathbb{Z}_q$ to compute the evaluation vectors: $(a(\zeta), \ldots, a(\zeta^{2n-1}))$ and $(s(\zeta), \ldots, s(\zeta^{2n-1}))$. Corresponds to multiplication by an FFT-like matrix. (takes $O(n \log n)$ mult./add. over $\mathbb{Z}_q$).
- Multiply the evaluations at each point: $c(\zeta^{2i+1}) = a(\zeta^{2i+1}) \cdot s(\zeta^{2i+1})$ for $i = 0, \ldots, n-1$.
- Interpolate to reconstruct $(a(\zeta), \ldots, a(\zeta^{2n-1}))$ to reconstruct $c(x)$. Corresponds to multiplication by an FFT-like matrix. (takes $O(n \log n)$ mult./add. over $\mathbb{Z}_q$).

Efficiency Considerations in Lattice-Based Crypto.

Ring Variant of Regev's public-key encryption scheme over ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ ($m' = m/n$ for 'orig.' m, $\ell = n$):

- Public key $pk = (A \hookleftarrow U(R_q^{m' \times 1}), \vec{p} = A \cdot s + \vec{e} \bmod q)$ with $\vec{e} = [e_1, \ldots, e_{m'}]^T$ and coefficients of e_i sampled independently from $\chi_{\alpha q}$.
  - **Length(pk):** $= m' \cdot 2n \log q = O(n \log^2 q) = O(\lambda \log^2 \lambda)$ bits — 'quasi-linear' in sec. par λ!
- Secret key $s \in R_q$.
- Encryption – Enc ($m \in \mathbb{R}_t$): Return ciphertext $C = (a = \vec{r}^T \cdot A \in R_q, c = \vec{r}^T \cdot \vec{p} + \lceil q/t \rfloor \cdot m \bmod q \in R_q)$.
  - **Ciphertext expansion ratio:** $= \frac{\text{Length}(C)}{\text{Length}(m)} = \frac{2n \cdot \log q}{n \log t} = \frac{2 \cdot \log q}{\log t} = O(\log \lambda)$!
  - **Encryption time:** With FFT, $O(m' n \log n \cdot \log^2 q) = O(\lambda \log^3 \lambda)$ bit ops. – 'quasi-linear'!
- Decryption – Dec ($C = (a, c)$): Compute $c' = c - a \cdot s \in R_q$, round to nearest multiple of $\lceil q/t \rfloor \bmod q$ to get $c'' \in R_q$. Return plaintext $m = \frac{\vec{c''}}{\lceil q/t \rfloor} \in R_t$.

Efficiency Considerations in Lattice-Based Crypto.

Poly. ring naturally can improve other lattice crypto. schemes:

Definition

Ring variant of Ajtai's Hash Function $g_{q,m',n,d,A}$: Pick $A = (a_1, \ldots, a_{m'})$ uniformly random $1 \times m'$ matrix over R_q (A = function 'public key'). Given input $\vec{x} \in R^{m'}$ having 'small' coordinates ($\|\vec{x}\|_\infty \leq d$), hash function output is defined as

$$g_{q,m,n,d,A}(\vec{x}) = A \cdot \vec{x} = a_1 \cdot x_1 + \cdots a_{m'} \cdot x_{m'} \in R_q.$$

Security: Ring-SIS problem (see next slides). Efficiency: $O(n \log n)$ key (A), $O(n \log^2 n)$ multiplications mod q.

Example implementation: SWIFFT hash function [LMPR08,ADLMPR08]

- Parameters: $n = 64, m = 16, q = 257$, compression function input (binary): 1024-bit, output: $\approx$ 512-bit
- Key length: $\approx$ 8 kbit
- Eval. Speed (optimized FFT, SIMD): $\approx$ 60 cycles/byte ($\approx$ 40

Security of Ring Learning with Errors (Ring-LWE) Problem

Problem

Decision Ring Learning with Errors (Decision-RLWE) Problem – *Decision* $-$ *RLWE*$_{q,m,n,\alpha}$*:*
Given $q, m, n, \alpha, A \hookleftarrow U(R_q^{m' \times n}), \vec{y}$, *distinguish between the following two scenarios:*

- *'Real' Scenario:* $\vec{y} = A \cdot \vec{s} + \vec{e} \bmod q$ *(with* $\vec{e} \hookleftarrow \chi_{\alpha q}^{m'}$ *and* $\vec{s} \hookleftarrow U(\mathbb{Z}_q^n)$*) (exactly as in search LWE).*
- *'Random' Scenario:* $\vec{y} \hookleftarrow U(\mathbb{Z}_q^m)$.

Note $chi_{\alpha q}$ is a rounded Gaussian distribution as in LWE definition.
Why do we believe Ring-LWE/Ring-SIS are hard? Similar situation to SIS/LWE, but less certain...
Theoretical Reason: Analogue of Regev's average-case to worst-case reduction for LWE can also be established for Ring-SIS/Ring-LWE (Lyubashevsky Peikert Regev 2010 [LPR10]):

Ring Learning with Errors (RLWE) Problem - Practical Security

Analogously to LWE, we also have:

Practical Reason: In most cases, essentially best known attack on Decision RLWE is a reduction of RLWE to RSIS. Hardness of RSIS for same R is assessed similarly to SIS!

Problem

Ring Small Integer Solution (RSIS) Problem – $RSIS_{q,m',n,\beta}$:
Given n and a matrix A sampled uniformly in $R_q^{1\times m'}$,
find $\vec{z} \in R^{m'} \setminus \{\vec{0}\}$ such that $A\vec{v} = \vec{0} \mod q$ and $\|\vec{v}\| \leq \beta$.

- Worst-case to average case connection for RSIS for ring R is known, analogously to Ajtai's theorem.
- The choice of ring R is important for security and efficiency (usually our usual choice of R suffices).

Efficiency Considerations in Lattice-Based Crypto.

Ring-Regev encryption scheme has:

- $2m'$ ring elements in public key: $(A, \vec{p}) \in R_q^{m' \times 2}$.
- 2 ring elements in ciphertext $(a, c) \in R_q^2$.

How to reduce public key and/or ciphertext to just 2 or even 1 ring elements?

Two schemes:

- 'Diffie-Hellman/ElGamal' analogue of Ring-Regev [LPR10]
  - Public key and ciphertext: 2 elements of R_q each
  - Security: as hard as Ring-LWE [LPR10]
- NTRUEncrypt [HPS96]
  - Public key and ciphertext: 1 element of R_q each
  - Security:
    - 'NTRU key-cracking' + Ring-LWE – original variant [HPS96], or
    - Ring-LWE, longer n, q – Modified variant [SS11]

Efficiency Considerations in Lattice-Based Crypto.

Ring-based 'Diffie-Hellman/ElGamal' analogue Encryption Scheme [LPR10,LP11]:

Recall Diffie-Hellman/ElGamal encryption scheme in a group G of order q with generator g:

Public key: $(g, p_b = g^b) \in G^2$, **Secret key:** $b \hookleftarrow U(R_q)$.

Encryption($m \in G$**;** $a \hookleftarrow U(R_q)$**):**

$(p_a = g^a \in G, c = p_b^a \cdot m = g^{a \cdot b} \cdot m \in G)$.

Decryption($(p_a, c) \in G^2$**):** $c/p_a^b = c/g^{a \cdot b} = m$.

Ring-based Diffie-Hellman analogue in R_q:

Public key: $(g \hookleftarrow U(R_q), p_b = g \cdot b + e_b \in G,\ b, e_b \hookleftarrow \chi_{\alpha q})$,

Secret key: $b \in R_q$.

Encryption($m \in R_t$**;** $a, e_a, e_c \hookleftarrow \chi_{\alpha q}$**):**

$(p_a = g \cdot a + e_a \in R_q, c = p_b \cdot a + e_c + \lceil q/t \rfloor \cdot m \in R_q)$.

- Note: $c = g \cdot b \cdot a + e_c \cdot a + \lceil q/t \rfloor \cdot m \in R_q)$.

Decryption($(p_a, c) \in G^2$**):**

$c - p_a \cdot b = c - (g \cdot a \cdot b + e_a \cdot b) = \lceil q/t \rfloor \cdot m + e_c \cdot a + e_a \cdot b \approx \lceil q/t \rfloor \cdot m$.

Efficiency Considerations in Lattice-Based Crypto.

Security of Ring-based Diffie-Hellman analogue scheme based on variant of Ring-LWE with small secret.

Ring-LWE with secret sampled from error distribution (SSRing-LWE): Same as Ring-LWE, but secret $s \hookleftarrow \chi_{\alpha q}$ instead of $s \hookleftarrow U(R_q)$.

Lemma. Ring-LWE with parameters m', n, α, q and secret sampled from the error distribution (i.e. SSRing-LWE) is as hard as standard Ring-LWE with parameters $m' + 1, n, \alpha, q$. (next week's tute!).

Simple security reduction for Diffie-Hellman encryption scheme from SSRing-LWE can be given. (tutorial).

Lemma. The Diffie-Hellman encryption scheme is as secure as Ring-LWE with parameters $m' = 2, n, \alpha, q$.

NTRU Cryptosystem (original variant [HPS96]): Key Generation

Ring Parameters: n prime, $q \approx n$ a power of 2, p small, ring $R^- = \mathbb{Z}[x]/(x^n - 1)$.
(e.g. $(n, q, p) = (503, 256, 3)$).

- Secret key sk: $f, g \in R^-$ sampled indep. from distrib. χ_σ with:
  - f is invertible mod q and mod p
  - The coeffs of f and g are small
    - $\mathrm{Supp}(\chi_\sigma) = \{-1, 0, 1\}^n$.
- Public key pk: $h = g/f \bmod q$.

NTRU key cracking Security intuition

Given $h \in R_q^-$, finding $g, f \in R^-$ small s.t. $h = g/f \ [q]$ is hard.

NTRU Cryptosystem (original variant [HPS96]): Key Generation

Ring Parameters: n prime, $q \approx n$ a power of 2, p small, ring $R^- = \mathbb{Z}[x]/(x^n - 1)$.
(e.g. $(n, q, p) = (503, 256, 3)$).

- Secret key sk: $f, g \in R^-$ sampled indep. from distrib. χ_σ with:
  - f is invertible mod q and mod p
  - The coeffs of f and g are small
    - $\mathrm{Supp}(\chi_\sigma) = \{-1, 0, 1\}^n$.
- Public key pk: $h = g/f \bmod q$.

NTRU key cracking Security intuition

Given $h \in R_q^-$, finding $g, f \in R^-$ small s.t. $h = g/f$ $[q]$ is hard.

NTRU Cryptosystem (original variant [HPS96]): Key Generation

Ring Parameters: n prime, $q \approx n$ a power of 2, p small, ring $R^- = \mathbb{Z}[x]/(x^n - 1)$.
(e.g. $(n, q, p) = (503, 256, 3)$).

- Secret key sk: $f, g \in R^-$ sampled indep. from distrib. χ_σ with:
  - f is invertible mod q and mod p
  - The coeffs of f and g are small
    - $\mathrm{Supp}(\chi_\sigma) = \{-1, 0, 1\}^n$.
- Public key pk: $h = g/f \bmod q$.

NTRU key cracking Security intuition

Given $h \in R_q^-$, finding $g, f \in R^-$ small s.t. $h = g/f \ [q]$ is hard.

NTRU Cryptosystem (original variant [HPS96]): Key Generation

Ring Parameters: n prime, $q \approx n$ a power of 2, p small, ring $R^- = \mathbb{Z}[x]/(x^n - 1)$.

(e.g. $(n, q, p) = (503, 256, 3)$).

- Secret key sk: $f, g \in R^-$ sampled indep. from distrib. χ_σ with:
  - f is invertible mod q and mod p
  - The coeffs of f and g are small
    - $\mathrm{Supp}(\chi_\sigma) = \{-1, 0, 1\}^n$.
- Public key pk: $h = g/f \bmod q$.

NTRU key cracking Security intuition

Given $h \in R_q^-$, finding $g, f \in R^-$ small s.t. $h = g/f$ $[q]$ is hard.

NTRU Cryptosystem (original variant [HPS96]): Encryption/Decryption

- *sk*: $f, g \in R^-$ small with f invertible mod q and mod p
- *pk*: $h = g/f \bmod q$

Encryption of $M \in R$ with coeffs in $\{0, \ldots, p-1\}$:

- Sample $s \in R_q^-$ from distrib. χ_ρ resp. with small coeffs – $\mathrm{Supp}(\chi_\rho) = \{-1, 0, 1\}^n$.
- Send $C := phs + M \bmod q$

Decryption of $C \in R_q^-$:

- $f \times C = p(gs) + fM \bmod q$
- Smallness $\Rightarrow$ equality holds over R^-
- $(f \times C \bmod q) \bmod p = fM \bmod p$
- Multiply by the inverse of f mod p

Security intuition

The mask *phs* hides the plaintext M in the ciphertext C.

Security of NTRU: Computational/Statistical Problems

Essentially two ways to break the IND-CPA security of NTRU:

- Crack the **public key**:

NTRU Decision Key Cracking Problem $\mathtt{DNKC}_{n,q,\phi,\chi_\sigma}$

Given (n, q, ϕ) and h, distinguish

- NTRU key distribution $D_0 = \{h = g/f \in R_q : f, g \hookleftarrow \chi_\sigma\}$.
- Uniform key distribution $D_1 = \{h \hookleftarrow U(R_q^*)\}$.

- Crack the **ciphertext** for a uniform key:

NTRU Decision Ciphertext Cracking Problem $\mathtt{DNCC}_{n,q,\phi,\chi_\rho,\chi_\beta}$

Given (n, q, ϕ), h sampled from $U(R_q^*)$, and c, distinguish

- NTRU zero-message ciphertext distribution $D_0 = \{c = phs : s \hookleftarrow \chi_\rho, e \hookleftarrow \chi_\beta\}$.
- Uniform distribution $D_1 = \{c \hookleftarrow U(R_q)\}$.

Security of NTRU: Computational/Statistical Problems

Essentially two ways to break the IND-CPA security of NTRU:

- Crack the **public key**:

NTRU Decision Key Cracking Problem $\mathtt{DNKC}_{n,q,\phi,\chi_\sigma}$

Given (n, q, ϕ) and h, distinguish

- NTRU key distribution $D_0 = \{h = g/f \in R_q : f, g \hookleftarrow \chi_\sigma\}$.
- Uniform key distribution $D_1 = \{h \hookleftarrow U(R_q^*)\}$.

- Crack the **ciphertext** for a uniform key:

NTRU Decision Ciphertext Cracking Problem $\mathtt{DNCC}_{n,q,\phi,\chi_\rho,\chi_\beta}$

Given (n, q, ϕ), h sampled from $U(R_q^*)$, and c, distinguish

- NTRU zero-message ciphertext distribution $D_0 = \{c = phs : s \hookleftarrow \chi_\rho, e \hookleftarrow \chi_\beta\}$.
- Uniform distribution $D_1 = \{c \hookleftarrow U(R_q)\}$.

NTRU Cryptosystem: Security of original variant [HPS96]

Security aspects of original variant:

- NTRU Decision Key Cracking problem:
  - Non-uniform distribution of h in R_q^- due to very small coefficients of f, g
  - No known attacks, but also not related to well-known lattice problems...
- NTRU Decision Ciphertext Cracking problem:
  - Trivial distinguishing attack (no noise): Given h, c, can easily distinguish if $c = phs$ or c uniform in R_q^-!

Modified variant of NTRU given in [SS11] 'fixes' these two issues.

NTRU Cryptosystem Original [HPS96] variant

Parameters: n, q a power of 2, $R = R^-$.

Key generation:

- sk: $f, g \in R$ with:
  - f invertible mod q and p.
  - Coeffs of f and g in $\{-1, 0, 1\}$
- pk: $h = g/f \bmod q$.

Encryption of $M \in R$ with coeffs in $\{0, 1\}$:

- $C := phs + M \bmod q$, with coeffs of s in $\{-1, 0, 1\}$.

Decryption of $C \in R_q$:

- $f \times C \bmod q \;=\; pgs + fM \qquad$ (over R)
- $(f \times C \bmod q) \bmod p \;=\; fM \bmod p$.
- Multiply by the inverse of f mod p.

NTRU Cryptosystem (Modified variant [SS11])

Parameters: n a power of 2, q prime, $R = R^+$.

Key generation:

- sk: $f, g \in R$ with:
  - f invertible mod q and p.
  - Coeffs of f and g of magnitude $\approx \sqrt{q}$
- pk: $h = g/f \bmod q$.

Encryption of $M \in R$ with coeffs in $\{0, 1\}$:

- $C := p(hs + e) + M \bmod q$, with coeffs of s, e of magnitude $\approx \beta$.

Decryption of $C \in R_q$:

- $f \times C \bmod q \;=\; p(gs + fe) + fM \qquad$ (over R)
- $(f \times C \bmod q) \bmod p \;=\; fM \bmod p$.
- Multiply by the inverse of $f \bmod p$.

Security of NTRU: Computational/Statistical Problems

Essentially two ways to break the IND security of NTRU:

- Crack the **public key**:

NTRU Decision Key Cracking Problem $\mathtt{DNKC}_{n,q,\phi,\chi_\sigma}$

Given (n, q, ϕ) and h, distinguish

- NTRU key distribution $D_0 = \{h = g/f \in R_q : f, g \hookleftarrow \chi_\sigma\}$.
- Uniform key distribution $D_1 = \{h \hookleftarrow U(R_q^*)\}$.

- Crack the **ciphertext** for a uniform key:

NTRU Decision Ciphertext Cracking Problem $\mathtt{DNCC}_{n,q,\phi,\chi_\rho,\chi_\beta}$

Given (n, q, ϕ), h sampled from $U(R_q^*)$, and c, distinguish

- NTRU ciphertext distribution $D_0 = \{c = phs + e : s \hookleftarrow \chi_\rho, e \hookleftarrow \chi_\beta\}$.
- Uniform distribution $D_1 = \{c \hookleftarrow U(R_q)\}$.

Security of NTRU: Computational/Statistical Problems

Essentially two ways to break the IND security of NTRU:

- Crack the **public key**:

NTRU Decision Key Cracking Problem $\mathtt{DNKC}_{n,q,\phi,\chi_\sigma}$

Given (n, q, ϕ) and h, distinguish

- NTRU key distribution $D_0 = \{h = g/f \in R_q : f, g \hookleftarrow \chi_\sigma\}$.
- Uniform key distribution $D_1 = \{h \hookleftarrow U(R_q^*)\}$.

- Crack the **ciphertext** for a uniform key:

NTRU Decision Ciphertext Cracking Problem $\mathtt{DNCC}_{n,q,\phi,\chi_\rho,\chi_\beta}$

Given (n, q, ϕ), h sampled from $U(R_q^*)$, and c, distinguish

- NTRU ciphertext distribution $D_0 = \{c = phs + e : s \hookleftarrow \chi_\rho, e \hookleftarrow \chi_\beta\}$.
- Uniform distribution $D_1 = \{c \hookleftarrow U(R_q)\}$.

IND Security of NTRU: Sufficient Condition

Proposition (Adapted from [SS11])

If DNKC and DNCC are both hard, then NTRUcryptosystem achieves semantic (IND) security.

Proof by contradiction – three 'games' with adversary A:

- IND_b – pk: $h = g/f$, ciph: $c_b = p \cdot (hs + e) + m_b$, $p_b = \Pr_{\text{IND}_b}[A(h, c_b) = 1]$.
- IND'$_b$ – pk: $h \hookleftarrow U(R_q^*)$, ciph: $c_b = p \cdot (hs + e) + m_b$, $p'_b = \Pr_{\text{IND'}_b}[A(h, c_b) = 1]$.
  - $|p'_b - p_b| = \text{non-neg}(n) \rightarrow A$ breaks DNKC.
- IND''$_b$ – pk: $h \hookleftarrow U(R_q^*)$, ciph: $c_b = p \cdot U(R_q) + m_b$, $p'' = \Pr_{\text{IND''}_b}[A(h, c_b) = 1]$.
  - $|p''_b - p'_b| = \text{non-neg}(n) \rightarrow A$ breaks DNCC.

Else, A can distinguish IND''$_0$ from IND''$_1$: contradiction – $p \cdot U(R_q)$ term perfectly hides m_b!

How to make both DNKCand DNCCproblems provably hard?

[SS11] strategy to prove hardness of DNKC and DNCC problems:

- Choose χ_σ for f, g to make DNKC statistically hard.
  - $f, g \hookleftarrow \chi_\sigma \rightarrow h = g/f$ almost uniformly distributed on R_q^*.
  - Must work in statistical region: $|\mathrm{Supp}(\chi_\sigma)| > |R_q^*| \rightarrow \sigma > \sqrt{q}$. (tradeoff: larger parameters than original scheme to avoid additional 'NTRU key cracking assumption').
  - Use a (modified) discrete Gaussian distribution χ_σ.
  - Tradeoff: Larger size of q, n versus original variant.
- Choose $\chi_\rho = \chi_\beta$ for s, e to make DNCC computationally hard.
  - Change rings: $R_q^- = \mathbb{Z}_q[x]/(x^n - 1) \rightarrow R_q^+ = \mathbb{Z}_q[x]/(x^n + 1)$, $n = 2^k$.
  - $h \hookleftarrow U(R_q^*), s, e \hookleftarrow \chi_\beta \rightarrow (h, c = hs + e)$ computationaly indist. from $U(R_q^* \times R_q)$, if SSRing-LWE problem hard.
  - Use a rounded Gaussian distribution χ_β.
  - Addition of error term: low-cost fix for IND-CPA security (avoid known attack!).

Estimated Parameters / Performance of NTRU (Orig. variant)

Sample parameters / implementation figures for NTRU (orig. variant) [HHPW09]:

k	N	d	d_{m_0}	q	size	RSA size	ECC size	enc/s	dec/s	ECC mult/s	Enc ECC ratio	Dec ECC ratio
112	401	113	113	2048	4411	2048	224	2640	1466	1075	4.91	1.36
128	449	134	134	2048	4939	3072	256	2001	1154	661	6.05	1.75
160	547	175	175	2048	6017	4096	320	1268	718	n/a	n/a	n/a
192	677	157	157	2048	7447	7680	384	1188	674	196	12.12	3.44
256	1087	120	120	2048	11957	15360	512	1087	598	115	18.9	5.2

Table 1.1 Size-optimized NTRUEncrypt parameter sets with trinary polynomials.

k	N	d	d_{m_0}	q	size	RSA size	ECC size	enc/s	dec/s	ECC mult/s	Enc ECC ratio	Dec ECC ratio
112	659	38	38	2048	7249	2048	224	4778	2654	1075	8.89	2.47
128	761	42	42	2048	8371	3072	256	3767	2173	661	11.4	3.29
160	991	49	49	2048	10901	4096	320	2501	1416	n/a	n/a	n/a
192	1087	63	63	2048	11957	7680	384	1844	1047	196	18.82	5.34
256	1499	79	79	2048	16489	15360	512	1197	658	115	20.82	5.72

Table 1.2 Speed-optimized NTRUEncrypt parameter sets with trinary polynomials.

Estimated Parameters / Performance of Ring Diffie-Hellman analogue

Sample parameters / implementation figures for Diffie-Hellman analogue scheme [LP11]:

n	q	s	Per-User Key ($\mathbf{P}$)	Full Key ($\mathbf{P}$ & $\bar{\mathbf{A}}$)	Ciphertext ($\mathbf{c}$)	Msg Expansion
128	2053	6.77	1.8×10^5	3.6×10^5	2.8×10^3	22.0
192	4093	8.87	2.9×10^5	7.4×10^5	3.8×10^3	30.0
256	4093	8.35	4.0×10^5	11.2×10^5	4.6×10^3	36.0
320	4093	8.00	4.9×10^5	17.2×10^5	5.4×10^3	42.0
136	2003	13.01	2.8×10^6	5.8×10^6	2.9×10^3	22.6
214	16381	7.37	2.4×10^6	6.4×10^6	4.8×10^3	18.7

Figure 5: Sizes (in bits) of public keys and ciphertexts for the cryptosystem described in Section 3; for comparison, the last two rows are for parameters given in [MR09]. In each case, the message size is $\ell = 128$ bits. The "message expansion" factor is the ratio of ciphertext size to plaintext size. Recall that in the ring-based system, the public key sizes are about a factor of n smaller.

References referred to in the Slides

Reg05 O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography,Journal of the ACM 56(6), 2009.

Pei09 C. Peikert. Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem, In Proceedings of STOC 2009.

B13 Z. Brakerski, A. Langlois, C. Peikert, O. Regev, D. Stehl. Classical Hardness of Learning with Errors, In Proceedings of STOC 2013.

PVW08 C. Peikert, V. Vaikuntanathan, B. Waters. A framework for efficient and composable oblivious transfer, In Proceedings of CRYPTO 2008.

HPS96 J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public key cryptosystem. In Proceedings of ANTS-III, 1998.

References referred to in the Slides (cont.)

M02 D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. Computational Complexity, 16(4):365411, 2007.

LPR10 V. Lyubashevsky, C. Peikert, O. Regev. On Ideal Lattices and Learning with Errors Over Rings. Journal of the ACM, 60(6):43:143:35, 2013.

SS11 D. Stehl and R. Steinfeld. Making NTRU as Secure as Worst-Case Problems over Ideal Lattices. In Proceedings of EUROCRYPT 2011.

LP11 R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In Proceedings of CT-RSA 2011.

HHPW09 J. Hoffstein, N. Howgrave-Graham, J. Pipher and W. Whyte. Practical lattice-based cryptography: NTRUEncrypt and NTRUSign, 2009. Book Chapter in P. Q. Nguyen and B. Valle (editors), The LLL Algorithm: Survey and Applications, Information Security and Cryptography, Springer, 2009.