Speaker's Notes: Public-key Crypto based on Discrete Log and LWE

Total Time: ~15 minutes

Introduction (0-0:30)

- Today: two foundational problems for public-key crypto
- First: Discrete Log and related problems (DH, DDH)
- Second: Learning With Errors (LWE) - post-quantum alternative
- Both enable key exchange and public-key encryption

Discrete Log Problem (0:30-2:30)

> **Given:** Group G, generator α, element $\beta \in G$
>
> **Find:** Integer a such that $\alpha^a = \beta$

Definition: Given group G, generator α, and element $\beta \in G$, find integer a, such that $\alpha^a = \beta$.

Key Points:

- "Reverse" of modular exponentiation
- Computing $\alpha^a \bmod p$ is **efficient** (polynomial time)
- Finding a from result is **computationally infeasible**
- Need large prime p (at least 2048 bits)
- Classic example: group $\mathbb{Z}_p^*$

Why it matters:

- Asymmetry creates one-way function
- Foundation for many cryptosystems
- Security relies on computational hardness

Intuition:

- Like mixing paint: easy to mix two colors, nearly impossible to separate them back
- Forward direction (exponentiation) wraps around the group many times
- Reverse direction would require trying exponentially many possibilities

[TRANSITION] "DL is hard, but can we use it directly? Not quite—need related problem that's easier to use..."

Diffie-Hellman (DH) Problem (2:30-4:00)

> **Given:** α, α^a, α^b (where a, b random from $\mathbb{Z}_t$)
>
> **Compute:** α^{ab}

Definition: Given group G, generator α, and α^a, α^b, where a, b randomly and independently chosen from $\mathbb{Z}_t$, compute α^{ab}.

Key distinction from DL:

- NOT asked to find a or b
- Only need shared secret α^{ab}
- If you solve DL → can solve DH (one exponentiation)
- Therefore: **DH no harder than DL**

Relationship:

- DL harder (or equal difficulty)
- DH is what we actually use in protocols

Intuition:

- Two parties each take "one step" (a and b), publish where they landed
- Goal: figure out where you'd be after taking both steps together
- Can't work backwards to find the steps, but maybe can jump directly to the combined result?

[TRANSITION] "But there's a verification problem with DH—how do you check a solution is correct without solving DL? That's where DDH comes in..."

Decisional Diffie-Hellman (DDH) Problem (4:00-5:30)

> **Given:** α^a, α^b, α^c (where a, b random from $\mathbb{Z}_t$)
>
> **Decide:** Is $c = ab \bmod t$, or is c uniformly random?

The problem:

- Can't verify DH solution without solving DL
- Need **decision problem** instead

Definition: Given group G, generator α, and $\alpha^a, \alpha^b, \alpha^c$, where a, b randomly chosen from $\mathbb{Z}_t$; and c chosen either as $c = ab$, or uniformly random from $\mathbb{Z}_t$. Guess which case.

Hierarchy of hardness:

- If solve DH → solve DDH (compute α^{ab} and compare)
- Therefore: **DDH no harder than DH**

- Chain: DL ≥ DH ≥ DDH

Intuition:

- Even easier than DH: don't need to compute the answer, just recognize it
- Is this triple "related" or just random noise?
- Like being given three numbers and asking: is the third one the product of first two, or just random?

[TRANSITION] "Now let's see how DH problem enables actual cryptographic protocols, starting with key exchange..."

Diffie-Hellman Key Exchange (DHE) (5:30-8:00)

> **Protocol:**
>
> - A: choose S_A, send $y_A = \alpha^{S_A}$
> - B: choose S_B, send $y_B = \alpha^{S_B}$
> - Both compute: $\alpha^{S_A S_B}$

Historic context:

- Diffie-Hellman 1977
- Revolutionary: shared secret without prior secret
- Public agreement on G and α

Protocol steps:

1. A chooses S_A random in $\mathbb{Z}_t$, B chooses S_B random in $\mathbb{Z}_t$
2. A sends $y_A = \alpha^{S_A}$ to B, B sends $y_B = \alpha^{S_B}$ to A
3. A computes $y_B^{S_A}$ and B computes $y_A^{S_B}$

Why it works:

- Both get same value: $y_B^{S_A} = y_A^{S_B} = \alpha^{S_A S_B}$
- Adversary sees α^{S_A} and α^{S_B}
- Must solve DH to get $\alpha^{S_A S_B}$

Modern relevance:

- Still used in TLS; mandatory for latest, TLS 1.3 handshakes
- (EC)DHE (elliptic curve) mandatory for (eg. AES) key agreement
- Provides forward secrecy

Intuition:

- Two people pick secret numbers, shout out transformed versions publicly
- Each uses their own secret + other's public value to compute same shared secret

- Eavesdropper hears both public values but can't derive the shared secret (DH problem)

[TRANSITION] "DHE is for key exchange. Can we build public-key encryption? Yes—El Gamal turns DH into full cryptosystem..."

El Gamal Cryptosystem (8:00-11:00)

> **Public Key:** $(G, \alpha, \beta = \alpha^a)$
>
> **Encrypt** m: $(\alpha^r, \beta^r m)$
>
> **Decrypt** (c, d): $c^{-a}d$

Core idea:

- A's first DHE message becomes public key
- B's message modified to encrypt

Key generation:

- Choose a random from $\mathbb{Z}_t$
- **Public key:** specification of G and $\beta = \alpha^a$
- **Secret key:** a
- Plaintext space: G, ciphertext space: $G \times G$

Encryption:

- To encrypt $m \in G$: choose r random from $\mathbb{Z}_t$
- Ciphertext: $(\alpha^r, \beta^r m)$

Decryption:

- To decrypt (c, d): compute $c^{-a}d$

> **Correctness:**
>
> $$c^{-a}d = (\alpha^r)^{-a}\beta^r m = \alpha^{-ra}(\alpha^a)^r m = m$$

Security properties:

- Decryption (without key) ≡ solving DH problem
- If DDH hard → El Gamal is **CPA secure**
- Randomized encryption (fresh r each time)

Intuition:

- Message m gets "masked" by multiplying with β^r (ephemeral shared secret)
- Sender includes α^r so receiver can "unmask" using secret key a
- Like locking message in a box using DH key exchange, but one-way

- Adversary sees masked message and α^r, but needs to solve DH to unmask

[TRANSITION] "El Gamal relies on DL hardness. But quantum computers can break DL efficiently using Shor's algorithm. Need quantum-resistant alternative—that's where LWE comes in..."

Learning With Errors (LWE) (11:00-15:00)

Problem

> **Secret:** $\vec{s} \in F_q^n$
>
> **Given:** $\{\vec{a_i},\ \vec{a_i} \cdot \vec{s} + e_i\}$ for random $\vec{a_i} \in F_q^n$
>
> **Find:** $\vec{s}$

Setup:

- Defined over finite field F_q where q is prime
- All computations modulo q
- Secret vector $\vec{s} \in F_q^n$

The Problem: Given $\{\vec{a_i} \in F_q^n\}_{i=1}^m$ and $\{\vec{a_i} \cdot \vec{s} + e_i\}_{i=1}^m$ where e_i's are random but numerically "small". Goal: find $\vec{s}$.

What "small" means:

- Much smaller than q
- Key property: sum of m errors < $q/4$ w.h.p.
- E.g., uniform in $[-\sqrt{q}..\sqrt{q}]$

Decision version (like DDH):

- Given random $\{\vec{a_i}\}_{i=1}^m$
- Either $\{\vec{a_i} \cdot \vec{s} + e_i\}_{i=1}^m$ or $\{u_i\}_{i=1}^m$ (uniform random)
- Decide which case

Intuition:

- Without errors: linear algebra, solve system in polynomial time
- Small errors: computationally hard, noise drowns out the structure
- Like trying to hear a whispered secret (linear relationship) in a noisy room
- Errors must be small enough that legitimate user can still decode, but large enough to hide structure from attacker

> **LWE Cryptosystem:**
>
> **Public Key:** $\{(\vec{a_i}, \vec{a_i} \cdot \vec{s} + e_i)\}_{i=1}^m$; **Secret key:** random $\vec{s} \in F_q^n$

> **Encrypt** bit w: $\sum_{i=1}^{m} b_i c_i + (\vec{0}, \lfloor q/2 \rfloor w)$
>
> **Decrypt** $(\vec{u}, v)$: Is $v - \vec{s} \cdot \vec{u} \approx 0$ or $\approx q/2$?

LWE-based Cryptosystem:

Key Generation:

- Secret key: random $\vec{s} \in F_q^n$
- Public key: $\{c_i\}_{i=1}^m$, where $c_i = (\vec{a_i}, \vec{a_i} \cdot \vec{s} + e_i)$
  - (Linear equation where c_i's solve to 0)

Encryption (one bit w):

- Choose random bits $b_1, ..., b_m$
- Ciphertext: $\sum_{i=1}^{m} b_i c_i + (0, \lfloor q/2 \rfloor w)$
- aka. randomly select some of the linear equations that are the public key
- send $(\vec{u}', v')$ where:
  - $\vec{u}' = \sum b_i \vec{a_i}$ Sum of selected $\vec{a_i}$
  - $v' = \sum b_i (\vec{a_i} \cdot \vec{s} + e_i) + \lfloor q/2 \rfloor w$ Sum of selected noisy equations + offset

Decryption:

- To decrypt $(\vec{u}, v)$: compute $v - \vec{s} \cdot \vec{u}$
- Output 0 if value closer to 0 than to $\lfloor q/2 \rfloor$
- Output 1 otherwise

Why decryption works:

$$v - \vec{u} \cdot \vec{s} = \sum_{i=1}^{m} b_i e_i + \left\lfloor \frac{q}{2} \right\rfloor w$$

- Since $\sum |e_i| < \frac{q}{4}$, can distinguish:
  - $w = 0$: result close to 0
  - $w = 1$: result close to $q/2$

Why LWE matters:

- Believed quantum-resistant
- Based on lattice problems
- Foundation for post-quantum crypto (NIST candidates)
- Similar structure to DL-based systems

Conclusion (15:00-15:30)

- DL problem: classical foundation (DH, DDH, El Gamal)
- Security hierarchy: DL ≥ DH ≥ DDH

- LWE: post-quantum alternative with similar structure
- Both enable key exchange and public-key encryption
- LWE likely future of cryptography in quantum era