Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024
Scribed by: Priyansh Vaishnav (202151120) — Lecture 16 (Week 11)

LECTURE 16 Date:- (16th April 2024)

# Discrete Logarithm Problem

Given a cyclic group $G$ of order $n$, with generator $\alpha$, and an element $\beta \in G$, find $x$ such that $\alpha^x = \beta$, where $0 \leq x \leq (n-1)$.

To compute $x$ from $g$ and $g^x$, exhaustive search runs a loop from $i = 1$ to $n$, with complexity proportional to $n$.

The Baby-Step Giant-Step Algorithm solves the Discrete Log Problem in $\sqrt{n}$ complexity.

## Baby-Step Giant-Step Algorithm

First, compute $m = \lceil \sqrt{n} \rceil$, where $n$ is the order of the cyclic group $G$ with generator $\alpha$. Since $\alpha^n = 1$, for $\beta = \alpha^x$, we can express $x$ using the Division Algorithm as:

$$x = i \cdot m + j, \quad 0 \leq i < j$$

Hence, $\alpha^x = \alpha^{i \cdot m} \cdot \alpha^j = \beta$. Taking $\alpha^{i \cdot m}$ to the right side:

$$\alpha^j = \beta(\alpha^{-m})^i$$

Now, instead of finding $x$, we need to find $i$ and $j$. The complexity of finding $i$ and $j$ should not increase.

The algorithm input is $\alpha$, $n$, and $\beta \in G$, with output $x = \log_\alpha \beta$. Here are the steps:

1. Set $m \leftarrow \lceil \sqrt{n} \rceil$.
2. Prepare a table $T$ with entries $j, \alpha^j$, $0 \leq j < m$. Sort $T$ by $\alpha^j$ values.
3. Compute $\alpha^{-m}$ and set $\gamma \leftarrow \beta$.
4. For $i = 0$ to $i = (m-1)$:
   - Check if $\gamma$ is the second component of some entry in $T$.
   - If $\gamma = \alpha^j$, compute $x = i \cdot m + j$.
   - Set $\gamma \leftarrow \gamma \cdot \alpha^{-m}$.

The table can be prepared offline, requiring $O(\sqrt{n})$ space. During runtime, the algorithm performs $O(\sqrt{n})$ multiplications. Sorting the table takes $O(\sqrt{n} \cdot \log n)$ time.

## ElGamal Public Key Cryptosystem

ElGamal encryption, unlike RSA, relies on the Discrete Log Problem. Here's how it works:

1. Choose a prime $p$.
2. Define the group $(\mathbb{Z}_p^*, *_p)$:

$$\mathbb{Z}_p^* = \{1, 2, 3, \ldots, (p-1)\}$$
$$x *_p y = x \cdot y \mod p$$

   Ensure $\gcd(x, p) = 1$ for $x \in \mathbb{Z}_p^*$.

3. Select a primitive element $\alpha \in \mathbb{Z}_p^*$.
4. Define plaintext and key spaces: $\{(p, \alpha, a, \beta), \beta = \alpha^a \mod p\}$.
5. Public key: $\{P, \alpha, \beta\}$; Secret key: $\{a\}$.
6. Choose a secret random number $x \in \mathbb{Z}_{p-1}$.
7. **Encryption:**

$$e_K(m, x) = (\alpha^x \mod p, m \cdot \beta^x \mod p)$$

8. **Decryption:**

$$d_K(y_1, y_2) = y_2 \cdot (y_1^a)^{-1} \mod p = m$$

   The randomness in the ciphertext arises from the secret $x$.

Given the public key $\{\beta, \alpha, p\}$, finding $a$ from $\beta$ and $\alpha$ (the discrete log problem) is difficult. While breaking ElGamal encryption yields $m$ from the ciphertext and $y_1 = \alpha^x$, it doesn't solve the Discrete Log Problem. This parallels the Diffie-Hellman Problem, where computing $g^{ab}$ from $g^a$ and $g^b$ breaks the Diffie-Hellman Key Exchange Algorithm but doesn't solve the Discrete Log Problem.

## Kerberos (Version 4)

Kerberos is a protocol for securely authenticating service requests between trusted hosts over untrusted networks like the internet. It relies on three key entities:

- Ticket Generating Server (TGS)
- Authentication Server (AS)
- Verifier (V)

Here's how the authentication process unfolds:

1. When a client logs into a server, it sends its identity ($ID_c$), the TGS identity ($ID_{TGS}$), and a timestamp ($TS_1$) to the Authentication Server.

2. The AS responds by encrypting a message with the client-TGS session key ($SK_{c,TGS}$), the TGS identity, a timestamp, and ticket validity information.

3. The client receives and decrypts the message, obtaining the session key ($SK_{c,TGS}$) and a ticket for accessing the TGS.

4. Using the session key, the client communicates with the TGS, providing its identity, the TGS ticket, and a freshly generated authenticator.

5. The TGS verifies the client's identity and authenticity, then responds with a session key for communicating with the verifier and a ticket for accessing the verifier.

6. The client forwards the ticket and a new authenticator to the verifier.

7. The verifier decrypts the received data, verifies the client's authenticity, and responds with a timestamp incremented by 1.

This process ensures secure authentication through encryption and decryption using shared keys, thus facilitating trusted communication between network entities.