[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024

Scribed by: Sanidhya Kumar (202151138) — Lecture (Week 11)

1 The Discrete Logarithm Problem (DLP)

The challenge of the Discrete Logarithm Problem lies in finding an integer x within the range $0 \leq x \leq (n-1)$ such that $\alpha^x = \beta$, given a cyclic group G of order n, its generator α, and an element $\beta \in G$.

To compute a when given g and g^a, an exhaustive search method can be employed. By iterating a loop from $i = 1$ to $i = n$, the size of the group (n), we calculate g^i. If $g^i = g^a$, we have found the desired result. The complexity of this approach corresponds to the size of the group. Alternatively, the Baby-Step Giant-Step Algorithm offers a solution to the Discrete Logarithm Problem with a complexity of approximately $\sqrt{n}$.

1.1 Baby-Step Giant-Step Algorithm

Initially, we calculate $m = \sqrt{n}$. Since n represents the order of the group and α serves as the generator of the group, it follows that $\alpha^n = 1$. Now, assuming $\beta = \alpha^x$, we can express x using the Division Algorithm as follows:

$$x = i \cdot m + j, \text{ where } 0 \leq i < j$$
$$\text{Thus, } \alpha^x = \alpha^{i \cdot m} \cdot \alpha^j$$
$$\text{Implying } \beta = \alpha^{i \cdot m} \cdot \alpha^j$$

Shifting $\alpha^{i \cdot m}$ to the right side, we obtain:

$$\alpha^j = \beta (\alpha^{im})^{-1}$$
$$\alpha^j = \beta (\alpha^{-m})^i$$

Now, instead of determining x, the focus shifts to finding i and j, both of which range within $m = \sqrt{n}$. We aim to accomplish this without significantly escalating complexity.

Formalizing the algorithm, the input comprises the generator α of cyclic group G, the order of group G (n), and $\beta \in G$. The output is the discrete logarithm $x = \log_\alpha \beta$. The procedure is outlined below:

1. Assign $m \leftarrow \lceil \sqrt{n} \rceil$
2. Create a table T with entries j, α^j $0 \leq j < m$. Sort T based on α^j values.
3. Compute α^{-m} and set $\gamma \leftarrow \beta$
4. For $i = 0$ to $i = (m-1)$, perform:
    - Check if γ is the second component of any entry in T.

- If $\gamma = \alpha^j$, then compute $x = i \cdot m + j$
- Update γ as $\gamma = \gamma \cdot \alpha^{-m}$

The table can be generated offline, necessitating $O(\sqrt{n})$ space. The runtime of the algorithm involves $O(\sqrt{n})$ multiplications. Sorting the table requires $O(\sqrt{n} \cdot \log(\sqrt{n})) \implies O(\sqrt{n} \cdot \log(n))$ time.

Example

Let us consider an example to understand this algorithm more clearly.

Given -

$$\begin{aligned}
&G = \mathbb{Z}_{113} \quad (p = 113) \\
&\alpha = 3 \\
&|G| = 112 \quad \text{(where } |G| \text{ is the order of the group)} \\
&\beta = 57
\end{aligned}$$

To find x such that $3^x = 57$ and $0 \leq x \leq \sqrt{112}$:

Solution -

Step 1: Find m which is $\lceil \sqrt{n} \rceil = \lceil \sqrt{112} \rceil \approx 11$

Step 2: Find $(j, \alpha^j \mod p)$ such that $0 \leq j \leq 11$

j	0	1	2	3	4	5	6	7	8	9	10
$(3^j \mod 113)$	1	3	9	27	81	17	51	40	7	21	63

Step 3: Sort the table based on $3^j \mod 113$ value

j	0	1	8	2	5	9	3	7	6	10
4										
$3^j \mod 113$	1	3	7	9	17	21	27	40	51	63
81										

Step 4: Find α inverse: $3^{-11} = (3^{-1})^{11} = (38)^{11} \mod 113 = 58$ Note: Inverse is under modulo 113 and is calculated using the extended Euclidean algorithm.

Step 5: Calculate $r = \beta \cdot \alpha^{-mi} \mod 113$ for $i = 0, 1, 2, \ldots, 10$

i	0	1	2	3	4	5	6	7	8	9
10										
$r = 57 \cdot (58)^i$	57	29	100	37	112	55	26	39	2	3
no-need										

$$\beta \cdot \alpha^{-9m} = 3 = \alpha^1$$
$$\beta \cdot \alpha^{-9 \cdot 11} = \alpha^1$$
$$\text{Hence, } i = 9 \text{ and } j = 1$$
$$\text{Now, } x = im + j$$
$$x = 9 \cdot 11 + 1 = 100$$

Hence, $x = 100$**, which is the solution.**

2 ElGamal Public Key Cryptosystem

ElGamal Public Key Cryptosystem operates similarly to RSA but relies on the Discrete Log Problem for security. The following steps outline the encryption and decryption processes in detail:

1. Choose a prime number p.
2. Define the group $(Z_p^*, *_p)$.

   $$Z_p^* = \{1, 2, 3, \ldots, (p-1)\}$$
   $$x *_p y = x * y \pmod p$$

   If $x \in Z_p^*$, then $\gcd(x, p) = 1$, implying the existence of the multiplicative inverse of x modulo p.
3. Choose a primitive element $\alpha \in Z_p^*$, also referred to as the generator of Z_p^*.

   $$Z_p^* \text{ forms a cyclic group}$$
   $$Z_p^* = \langle \alpha \rangle$$
4. The plaintext space is Z_p^*, and the key space is $\{(p, \alpha, a, \beta), \beta = \alpha^a \pmod p\}$.
5. The public key consists of $\{P, \alpha, \beta\}$, and the secret key is $\{a\}$.
6. Select a random number $x \in Z_{p-1}$, keeping it secret.
7. **Encryption:** The encryption algorithm yields a ciphertext tuple.

   $$e_K(m, x) = (y_1, y_2)$$
   $$y_1 = \alpha^x \pmod p$$
   $$y_2 = m \cdot \beta^x \pmod p$$
8. **Decryption:** The decryption process is as follows,

   $$d_K(y_1, y_2) = y_2 \cdot (y_1^a)^{-1} \pmod p = m$$
   $$y_1^a = (\alpha^x)^a \pmod p = \beta^x \pmod p$$
   $$y_2 \cdot (y_1^a)^{-1} = m \cdot \beta^x \cdot \beta^{x-1} \pmod p = m \pmod p$$

   The presence of randomness in the ciphertext stems from the randomly chosen x.

The public key is $\{\beta, \alpha, p\}$. The challenge lies in finding a from β and α (the Discrete Log Problem). However, if we possess the ciphertext and

$$y_1 = \alpha^x,$$

computing α^{ax} from $\beta = \alpha^a$ and α^x would enable us to derive the message m without needing to determine a or x. This would break the ElGamal Encryption, yet it wouldn't guarantee solving the Discrete Log Problem. This scenario mirrors the Diffie-Hellman Problem. If one can compute g^{ab} from g^a and g^b, the Diffie-Hellman Key Exchange Algorithm is compromised, but the Discrete Log Problem remains unsolved.

Example

Let us consider an example of ElGamal to understand this algorithm more clearly.

Given common information **(Public Information)**:

$$P = 7, \quad \alpha = 4, \quad \beta = 4^3 \equiv 1 \pmod 7$$

Alice's Side:

$$\begin{aligned} \text{Message } M &= 5 \\ x &= 2 \\ y_1 &= \alpha^x = 4^2 \equiv 2 \pmod 7 \\ y_2 &= M \cdot \beta^x = 5 \pmod 7 \end{aligned}$$

Now, Alice sends (y_1, y_2) to Bob.

Bob's Side:

$$\begin{aligned} \text{Bob's secret key } a &= 3 \\ \text{Bob receives } (y_1, y_2) &\text{ from Alice} \\ \text{Bob calculates:} & \\ y_2 \cdot (y_1^a)^{-1} &= 5 \cdot (2^3)^{-1} \\ &= 5 \cdot (8)^{-1} \\ &= 5 \cdot (1)^{-1} \text{ (calculate inverse modulo } P \text{ using extended Euclidean algorithm)} \\ &= 5 \\ &= M \text{ (original message by Alice)} \end{aligned}$$

Note: Everything is under modulo P (7 here).

3 Kerberos (Version 4)

Kerberos is a network security protocol designed to authenticate service requests among trusted hosts over potentially untrusted networks, such as the internet. It relies on secret-key cryptography and a trusted third-party system to validate client-server interactions and authenticate user identities. It is important to note that this protocol uses Symmetric Key Cryptography at its core.

The Kerberos protocol involves three key entities:

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

The authentication process involves communication between a client C and various servers, namely the Authentication Server (AS), the Ticket Granting Server (TGS), and a Verifier (V). Below is a step-by-step description of the communication flow:

1. The client initiates communication by logging into the server and sending the following details to the Authentication Server:

$$\begin{aligned} C \rightarrow AS &: ID_c || ID_{TGS} || TS_1 \\ ID_c &\rightarrow \text{Identity of Client} \\ ID_{TGS} &\rightarrow \text{Identity of TGS} \\ TS_1 &\rightarrow \text{Timestamp} \end{aligned}$$

2. The AS will receive the information and send an encrypted message back to the client with the following information. AS performs symmetric key encryption using the key SK_c, which is shared between the AS and the client.

$$AS \rightarrow C : E(SK_c, [SK_{c,TGS} \mid\mid ID_{TGS} \mid\mid TS_2 \mid\mid Lifetime_2 \mid\mid Ticket_{TGS}])$$
$$SK_{c,TGS} \rightarrow \text{key used for communication between Client and TGS}$$
$$Lifetime_2 \rightarrow \text{for how long this ticket/data will be valid}$$

   ID_{TGS} and TS_2 are ID of TGS and timestamp as earlier. The ticket is generated by AS and contains the following information.

$$Ticket_{TGS} = E(SK_{TGS}, [SK_{c,TGS} \mid\mid ID_c \mid\mid AD_c \mid\mid ID_{TGS} \mid\mid Lifetime_2])$$
$$AD_c \rightarrow \text{Address of client}$$

   As it can be seen the $Ticket_{TGS}$ is encrypted using the key SK_{TGS}. It is the key used for communication between AS and TGS and it is not known to any other party.
   The client will receive the response from the AS and will be able to decrypt it as it was encrypted using SK_c, which is shared between client and AS. The client will recieve the following information:

$$[SK_{c,TGS},\ ID_{TGS},\ TS_2,\ Lifetime_2,\ Ticket_{TGS}]$$

   The client will receive the ticket but will not be able to decrypt it as client doesn't have SK_{TGS}. The client will also receive the key $SK_{c,TGS}$ which is the shared secret key between client and TGS. This secret key is also called as session key. When you establish a session in server, you get a session key for encrypted communication.

3. With the session key $SK_{c,TGS}$, the client initiates communication with the TGS:

$$C \rightarrow TGS : ID_v || Ticket_{TGS} || Authenticator_c$$
$$Authenticator_c = E(SK_{c,TGS}, [ID_c || AD_c || TS_3])$$

4. The TGS decrypts the ticket and authenticator to verify the client's identity and sends back encrypted information:

$$TGS \rightarrow C : E(SK_{c,TGS}, [SK_{c,v} || ID_v || TS_4 || Ticket_v])$$

   The ticket $Ticket_v$ includes information encrypted using the key SK_v, known only to TGS and the verifier.

5. The client sends the ticket and a fresh authenticator to the verifier:

$$C \rightarrow V : Ticket_v || Authenticator_c$$

6. The verifier decrypts the received data to authenticate the client:

$$V \rightarrow C : E(SK_{c,v}, [TS_5 + 1])$$

Note - Refer Cryptography and Network Security - Principles and Practice by William Stallings for further reading.