
[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Priyansh Vaishnav (202151120)

Winter 2023-2024
Lecture 14& 15 (Week 9)

LECTURE 14 Date:- (2nd April 2024)

.....

1 Diffie-Hellman Key Exchange Algorithm

Diffie-Hellman key exchange algorithm is a foundational concept in **public-key cryptography**. It was invented by Whitfield Diffie and Martin Hellman in 1976.

1. $G \rightarrow$ cyclic group $= \langle g \rangle$
 $(G, *)$
 - (a) If $a, b \in G$, then $a * b \in G$
 - (b) There exists an element $e \in G$ such that $a * e = a = e * a$
 e : Identity element of G .
 - (c) $a * (b * c) = (a * b) * c$
 $*$: Associative
 - (d) For every element $a \in G$, there exists $a^{-1} \in G$ such that $a * a^{-1} = e = a^{-1} * a$
Here, a^{-1} is known as the Inverse of a .

Alice

$$0 \leq a \leq n - 1$$

$$K_a = g^a$$

$$a$$

$$K_b = g^b$$

$$(K_b)^a = (g^b)^a$$

$$(K_b)^a = (g^{ba})$$

$$g^{ba} = g^{ab}$$

Secret key shared: g^{ab}

Secret key: a

Public key: g^a

$g^x \rightarrow$ Public

Bob

$$0 \leq a \leq n - 1$$

$$K_b = g^b$$

$$b$$

$$K_a = g^a$$

$$(K_a)^b = (g^a)^b$$

$$(K_a)^b = (g^{ab})$$

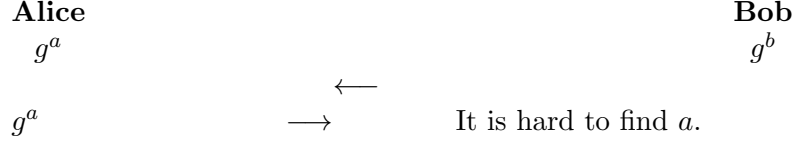
Secret key: b

Public key: g^b

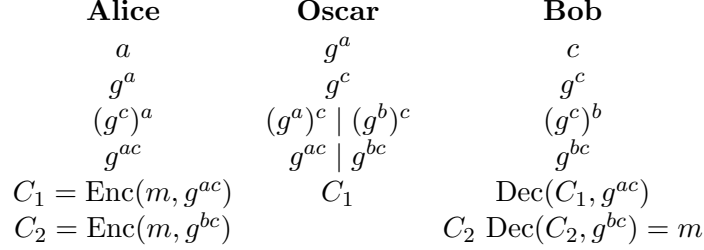
$x \rightarrow$ Secret

Based on the properties of the group $G = \langle g \rangle$, finding x from g^x will be computationally difficult. This problem is known as the **Discrete Log Problem**.

In cryptographic protocols, the computational difficulty of the discrete logarithm problem arises from finding x given g^x in a group G with a large order $|G|$, crucial for security.



Now, let's consider *OSCAR* as a man-in-the-middle attack.



1.1 Square and Multiply

To compute x^c , where c is a binary number. Let $c = \sum_{i=0}^{l-1} C_i 2^i$, where $C_i \in \{0, 1\}$. Convert c to binary representation (C_{l-1}, \dots, C_0) . Set $z = 1$.

$$x^c = \prod_{i=0}^{l-1} x^{C_i 2^i}$$

$$x^c = x^{C_{l-1} 2^{l-1}} \cdot x^{C_{l-2} 2^{l-2}} \cdot \dots \cdot x^{C_0 2^0}$$

For i from $l-1$ to 0: $z = z^2$ if $C_i = 1$: $z = z \times x$ Return z .

2 RSA: -

$\Phi(m)$: It is a function, which counts the number of positive integers less than m that are coprime to m also known as Euler's totient function .

$$\Phi(m) = 4 \longrightarrow \{1, 3, 5, 7\}$$

$$\Phi(p) = p - 1 \quad \text{where } p \text{ is prime}$$

$$\Phi(p^k) = p^k - p^{k-1}$$

$$\Phi(p^k) = p^k \left(1 - \frac{1}{p} \right)$$

Given $\gcd(a, m) = 1$, let $S = \{x \bmod m\}$. Then $S = \{r_1, r_2, \dots, r_m\}$, and $S = \{ar_1, ar_2, \dots, ar_m\}$.

If $ar_i \equiv ar_j \pmod{m}$, then $r_i \neq r_j$.

Since $\gcd(a, m) = 1$, there exists an integer b such that $ab \equiv 1 \pmod{m}$.

$$ar_i \equiv ar_j \pmod{m} \quad \text{where } r_i \neq r_j \pmod{m}$$

$$\implies b \cdot ar_i \equiv b \cdot ar_j \pmod{m}$$

$$\implies r_i \equiv r_j \pmod{m}$$

$$\implies ar_i \neq ar_j \pmod{m}$$

3 Euler's Theorem

Euler's theorem states that for any integer a coprime to m , where $\Phi(m)$ is the Euler's totient function, we have:

$$a^{\Phi(m)} \equiv 1 \pmod{m}$$

This theorem has wide applications in number theory and cryptography, particularly in the RSA encryption algorithm.

4 Fermat's Little Theorem

Fermat's Little Theorem states that if p is a prime number and a is an integer not divisible by p , then:

$$a^{p-1} \equiv 1 \pmod{p}$$

This theorem is a special case of Euler's theorem and has significant applications in primality testing and cryptography.

LECTURE 15 Date:- (5th April 2024)

RSA Cryptosystem

- $n = pq$
- Plaintext space: \mathbb{Z}_n
- Ciphertext space: \mathbb{Z}_n
- Key Space: $K = (n, p, q, e, d)$ where $(ed \equiv 1 \pmod{\varphi(n)})$

Encryption:

$$\begin{aligned} \text{Enc}(x, k) &= C \\ c &= \text{Enc}(x, k) = x^e \pmod{n} \end{aligned}$$

Decryption:

$$\begin{aligned} \text{Dec}(c, k) &= x \\ x &= \text{Dec}(c, k) = c^d \pmod{n} \end{aligned}$$

To decrypt the ciphertext c using the private key $k = (n, p, q, e, d)$, we compute:

$$x = c^d \pmod{n}$$

Recall that we chose d such that $ed \equiv 1 \pmod{\varphi(n)}$, which implies that $ed - 1 = t \cdot \varphi(n)$ for some integer t . Therefore, we can rewrite x as:

$$x \equiv c^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{1+t\varphi(n)} \pmod{n}$$

Since $n = pq$ and $x \in \mathbb{Z}_n$, we can also express x as a pair of residues (x_p, x_q) modulo p and q , respectively. That is, $x \equiv x_p \pmod{p}$ and $x \equiv x_q \pmod{q}$. Using the Chinese Remainder Theorem, we can recover x from (x_p, x_q) :

$$x \equiv x_p q^{q-1} + x_q p^{p-1} \pmod{n}$$

5 DSA - Digital Signature Algorithm

Digital Signature Algorithm (DSA) is a cryptographic method used for creating and verifying digital signatures to ensure the integrity and authenticity of messages.

Example Process:

1. **Message Transmission:** Alice wishes to securely transmit a message m to Bob.
2. **Signing:** Alice signs the message with her private key, ensuring the authenticity of the message.
3. **Key Generation:**
 - Let $n = p \cdot q$ and $\phi(n) = (p - 1) \cdot (q - 1)$, where p and q are large prime numbers.
 - Choose e and d such that $e \cdot d \equiv 1 \pmod{\phi(n)}$, ensuring the existence of the modular multiplicative inverse.
 - Alice generates a public key y and a private key x using the formula: $y = g^x \pmod{p}$, where g is a generator of the multiplicative group modulo p .
4. **Signing the Message:** Alice signs the message m with her private key x and sends it to Bob.
5. **Signature Calculation:** The signature s is calculated as $s = m^d \pmod{n}$.
6. **Signature Verification:** Bob verifies the signature using Alice's public key y .

This process ensures that Bob can verify the authenticity and integrity of the message sent by Alice.