[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Sanidhya Kumar (202151138)

Winter 2023-2024
Lecture (Week 9)

1 Diffie-Hellman Key Exchange

The Diffie-Hellman algorithm introduces the concept of public key cryptography, facilitating secure key exchange over insecure channels, ensuring data transmission security and integrity. Consider two users, Alice and Bob, who agree to communicate using a cyclic group $G = \langle g \rangle$, where $|G| = P$ (P is a large prime number). Alice chooses a secret key a, $0 \leq a \leq p-1$, and computes $K_a = g^a$. Similarly, Bob chooses b, $0 \leq b \leq p-1$, and computes $K_b = g^b$. They exchange K_a and K_b publicly.

Alice	Bob
$K_a = g^a$	$K_b = g^b$

Then, Alice computes $(K_b)^a = (g^b)^a = g^{ba}$, and Bob computes $(K_a)^b = (g^a)^b = g^{ab}$. And, the shared secret key is $g^{ab} = g^{ba}$. The difficulty of compuing x from g^x is known as the discrete logarithm problem. This algorithm was published in IEEE Transactions on Information Theory. Let us understand this in mathematical format:

1. If $a, b \in G$, then $a * b \in G$.
2. There exists an element $e \in G$ such that $a * e = a = e * a$, where e is the identity element of G.
3. $a * (b * c) = (a * b) * c$ (associative property).
4. For every element $a \in G$, there exists an inverse.

$$|G| = n$$

$$0 \leq a \leq n-1 \quad K_a = g^a \quad 0 \leq b \leq n-1$$

	$K_a = g^a$	$K_a = g^b$
a		b
	$K_b = g^b$	$K_b = g^a$

Compute:

$(K_b)^a = (g^b)^a$	$(K_a)^b = (g^a)^b$
$\Rightarrow g^{ba}$	$\Rightarrow g^{ab}$

Shared Secret Key: g^{ab}

Alice	**Bob**
Secret Key $= a$	Secret Key $= b$
Public Key $= g^a$	Public Key $= g^b$

Without knowing a and b (secret keys), we cannot compute g^{ab} (shared secret key). Given g^x (public) and x (secret), finding x from g^x is computationally difficult due to the properties of the group $< G >=< g >$.

Note: This hard problem is known as the discrete logarithm problem.

$$y = g^x$$

$$\log_g y = x$$

1. $|G|$ is very large.
2. The $*$ operation must be sufficiently secure.

$$(Z_p, +_p) =< g >$$

$$g_i = (g +_p g +_p \cdots +_p g) = ig$$

$$g^i = (g +_p g +_p \dots) = ig = y$$

$$i = gy^{-1} \bmod (p)$$

1.1 Man-in-the-Middle Attack

The Diffie-Hellman key exchange is vulnerable to a man-in-the-middle attack. In this attack, an opponent, Oscar, intercepts Alice's public value and sends her own public value to Bob. When Bob transmits his public value, Oscar substitutes it with her own and sends it to Alice. Thus, Oscar and Alice agree on one shared key, and Oscar and Bob agree on another shared key.

For example:

Alice has a and computes g^a, sending it towards Bob. Oscar intercepts this and has c, calculates and shares g^c with Bob. Bob, unaware that g^c is sent by Oscar, assumes it's from Alice. Similarly, Bob has b, calculates g^b, and shares it with Alice, intercepted by Oscar, who shares g^c with Alice as well.

Hence, Alice encrypts a message m as $C_1 = \text{Enc}(m, g^{ac})$. When intercepted by Oscar, who also has g^{ac}, Oscar can alter the message or read it. Oscar then encrypts again with $C_2 = \text{Enc}(m, g^{bc})$ and sends it to Bob, who decrypts with g^{bc}, thinking nothing is wrong.

$Alice$	$\lvert G\rvert = n - large$	Bob
$0 < a < n$	$Z_p{}^*, \dot{m}od(p)$	$0 < b < n$
g^a		g^b

$$=< g >\rightarrow Cyclic - Group$$

P: Prime Number
$P = 2^{255} - 19$

How to continue $g^a(\underbrace{g * g * g * g}_{a-Times})$

$t = 1$
for $(i = 2; i < a; i++)$
$t = (t * g)\%p$

2 Square and Multiply Algorithm

We wish to compute x^c

Convert c to binary: $C \to (C_{l-1} \dots C_0)$

$z \leftarrow 1$
for $i \leftarrow l-1$ downto 0
  do $z \leftarrow z^2 \mod n$
    if $c_i = 1$
      then $z \leftarrow (z \times x) \mod n$
return (z)

Complexity: $log(c)$

$$c = \sum_{i=0}^{l} c_i 2^i$$
$$x^c = x^{\sum_{i=0}^{l} c_i 2^i}$$
$$= \prod_{i=0}^{l} x^{c_i z^i}$$
$$= x^{c_0 z^0} . x^{c_1 z^1} \dots .$$

3 RSA (Rivest, Shamir, Adleman)

$\phi(m) =$ number of positive integers $< m$ which are co-prime to m.

$$g(x, m) = 1$$
$$\phi(8) = 4; \{1, 3, 5, 7\}$$
$$\gcd(a, m) = 1$$
$$S = \{x \mod (m)\}$$
$$= \{r_1, r_2, \dots, r_m\}$$
$$S_1 = \{ar_1, ar_2, \dots, ar_m\}$$
$$ar_i = ar_j \quad r_i \neq r_j$$
$$ar_i \equiv ar_j \mod (n) \quad r_i \neq r_j \mod (m)$$
$$\gcd(a, m) = 1$$
$$1 = ab + ms$$
$$\exists b \text{ such that } ab \equiv 1 \mod (m)$$
$$ar_i \equiv ar_j \mod (m)$$
$$bar_i = bar_j \mod (m)$$
$$\therefore ar_i \not\equiv ar_j \mod (m)$$

3.1 Euler's Theorem

If $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$. Let us assume that we have a set S, such that:

$$S = \{x \mid \gcd(x, m) = 1\}$$

$$S = \{s_1, s_2, s_3, s_4, \ldots, s_{\phi(m)}\}$$

Let us consider $\gcd(a, m) = 1$ and create another set S_1 such that

$$S_1 = \{as_1, as_2, as_3, \ldots, as_{\phi(m)}\}$$

We know from the discussion above that, if $as_i \equiv as_j \pmod{m}$, then $s_i \equiv s_j \pmod{m}$. Given that $\gcd(a, m) = 1$ and $b \cdot a \equiv 1 \pmod{m}$,

$$|S| = \phi(m)$$

$$|S_1| = \phi(m)$$

Since a is coprime with m and s_i is also coprime with m, there must be some correspondence between elements of S and S_1.

$$s_i \equiv as_j \pmod{m}$$

Let us now take product on both sides and simplify:

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

3.2 Fermat's Theorem

If p is a prime number and p does not divide a (meaning that p is coprime to a), then

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

Using Fermat's theorem, $\Rightarrow a^p \equiv a \pmod{p}$.

Note: Fermat's theorem will not hold when p does not divide a.

3.3 RSA Cryptosystem

Facts:

- If $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$.
- $a^{p-1} \equiv 1 \pmod{p}$ if p is prime and p does not divide a.

Now, let's understand the components of RSA:

1. $n = pq$, where p and q are primes.
2. Plaintext space: $\mathbb{Z}_n$
   Ciphertext space: $\mathbb{Z}_n$
3. Key space: $\{\ K = (n, p, q, e, d) \mid ed \equiv 1 \pmod{\phi(n)}\ \}$
4. Encryption:

$$E(x, K) = c$$

$$c = E(x, K) = x^e \pmod{n}$$

5. Decryption:

$$\text{Dec}(c, K) = x$$

$$c = \text{Dec}(c, K) = c^d \pmod{n}$$

We know that e and d are related as:

$$ed \equiv 1 \pmod{\phi(n)}$$

$$\Rightarrow ed - 1 = t \cdot \phi(n)$$

$$\Rightarrow 1 = ed + t_1 \cdot \phi(n)$$

$$1 = \gcd(e, \phi(n)) = ed + t_1 \cdot \phi(n)$$

Encryption:

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

Decryption:

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

$$c^d = (x^e)^d \pmod{n}$$

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

Now using $ed = 1 + t \cdot \phi(n)$ from above:

$$c^d = x^{1+t\cdot\phi(n)} \pmod{n}$$

$$c^d = x \cdot x^{t\cdot\phi(n)} \pmod{n}$$

Since p and q are primes and $n = pq$, then $\phi(n) = (p-1)(q-1)$:

$$c^d = x \cdot x^{t[(p-1)(q-1)]} \pmod{n}$$

Finally,

$$c^d = x \cdot x^{t[(p-1)(q-1)]} \pmod{pq}$$

Now, let us simplify the part $x^{t[(p-1)(q-1)]} \pmod{pq}$, where $x \in \mathbb{Z}$:

We check $x^{t[(p-1)(q-1)]} \pmod{p}$:

$$\equiv (x^{p-1})^{t(q-1)} \pmod{p}$$

$$\equiv 1 \pmod{p}$$

(As $x^{p-1} \equiv 1 \pmod{p}$)

Now we check $x^{t[(p-1)(q-1)]} \pmod{q}$:

$$\equiv (x^{q-1})^{t(p-1)} \pmod{q}$$

$$\equiv 1 \pmod{q}$$

(As $x^{q-1} \equiv 1 \pmod{q}$) We finally have:

$$x^{t[(p-1)(q-1)]} \equiv 1 \pmod{p}$$

$$x^{t[(p-1)(q-1)]} \equiv 1 \pmod{q}$$
$$\Rightarrow x^{t[(p-1)(q-1)]} \equiv 1 \pmod{pq}$$

Substituting the above result:

$$c^d = x \cdot x^{t[(p-1)(q-1)]} \pmod{pq}$$

$$c^d = x \cdot 1 \pmod{pq}$$
$$c^d = x \pmod{pq}$$

Hence, our decryption is successful.

Now let us consider a scenario where Alice is trying to communicate with Bob. Here, two keys play the main role: one is the public key and the other is the secret key. Bob encrypts the message and sends it to Alice, who has both keys. The public key is known to Bob, but the secret key is not known to Bob.

Alice and Bob communication using RSA	
Alice	**Bob**
$n = pq$ and p and q are large prime numbers.	Now Bob selects a message x from $\mathbb{Z}_n$.
$ed \equiv 1 \pmod{\phi(n)}$	x
She chooses e.	He knows n and e for Alice, so he can encrypt.
Public key of Alice: (n, e)	$y = x^e \pmod{n}$
She can generate d using the extended Euclidean algorithm.	
Secret key of Alice: (p, q, d)	

Now the message y is sent to Alice, who can decrypt it with her secret key as:

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

Note: If we are able to compute p and q from n, then we will be able to compute $\phi(n)$. And then we already have e, so we will be able to find d using the extended Euclidean algorithm and the security of RSA will be broken. So, RSA is based on the fact that the factorization problem is hard.

3.4 RSA Vulnerability

The RSA encryption scheme faces a critical vulnerability: if given the public key (n, e) and a ciphertext c, allowing the retrieval of the original plaintext x ($c = x^e$), the security of RSA is compromised.

While breaking RSA implies the capability to factorize n, the converse is not necessarily true. Therefore, RSA security relies on two fundamental assumptions: first, the difficulty of factorization, and second, the complexity of decryption. It has the following two problems.

3.4.1 The RSA Problem

When presented with the public key (n, e) and a ciphertext c generated by encrypting a plaintext x ($c = x^e$), the RSA problem involves deducing the corresponding private key d directly from these elements.

3.4.2 The Factorization Problem

On the other hand, the factorization problem revolves around determining the prime factors p and q of n. Once these factors are determined, computing d becomes feasible.

Note: Solving the factorization problem allows for the resolution of the RSA problem. However, the reverse is not necessarily true. RSA security relies on the assumption that both the RSA problem and the factorization problem pose significant computational challenges.

4 Digital Signature Algorithm (DSA)

The Digital Signature Algorithm (DSA) is a cryptographic algorithm utilized to generate digital signatures, authenticate message senders, and prevent message tampering. It is represented as $(P, S, K, \text{Sign}, V)$, where:

- P is the plaintext space,
- S is the signature space,
- K is the K-tuple,
- Sign is the signature algorithm, and
- V is the validation algorithm.

It operates with **two keys K:** a private key held by the sender and a public key distributed to recipients. The sign algorithm in DSA is represented as:

$$S = m^d \pmod{n}$$

Here, m is the message, d is the private key, and n is the modulus.

4.1 Differences between RSA and DSA

In RSA encryption:

$$c_1 = m_1^e \mod n$$
$$c_2 = m_2^e \mod n$$
$$c_1 \times c_2 = (m_1 \times m_2)^e \mod n$$

However, in DSA, hash functions are used:

$$s_1 = (h(m_1))^d \mod n$$
$$s_2 = (h(m_2))^d \mod n$$
$$s_1 \times s_2 \neq (h(m_1 \times m_2))^d \mod n$$

4.2 Signature Generation

1. Choose two distinct prime numbers p and q.
2. Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$.
3. Select an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$.
4. Compute the integer d such that $1 < d < \varphi(n)$ and $ed \equiv 1 \pmod{\varphi(n)}$.
5. The public key is (n, e) and the private key is (n, d).

4.3 Signature Validation

To encrypt a message m using the public key (n, e), compute the ciphertext S as:

$$S = m^d \mod n$$

To decrypt the ciphertext S using the private key (n, d), compute the message V as:

$$V = S^e \mod n$$

In DSA, the security verification involves checking the equality:

$$S_1 \times S_2 \neq (h(m_1 \times m_2))^d \mod n$$

where $h()$ represents a hash function. This algorithm ensures message integrity and authenticity in digital communications.