[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Aniruddh Modi (202151022)

Winter 2023-2024
Lecture 14 15 (Week 9)

1 Diffie-Hellman Key Exchange Algorithm

Alice selects a from 0 to $n-1$, and Bob selects b from 0 to $n-1$.

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

where g is a cyclic group on which Bob and Alice agree to select. Alice sends g^a to Bob, and Bob sends g^b to Alice.

Alice has the following data:

- a
- g^a because we know what g is and we already have a
- g^b which is sent by Bob

Bob has the following data:

- b
- g^b because we know what g is and we already have b
- g^a which is sent by Alice

So, for Alice, the secret key is a, and the public key is g^a. For Bob, the secret key is b, and the public key is g^b.

Hence, the shared key between Alice and Bob will be g^{ab}. a is the secret key of Alice because Alice is sharing g^a, not a, which is why it remains secret throughout the communication. Based on the property of Group $G =< g >$, finding x from g^x will be computationally difficult. This is a hard problem known as the discrete log problem.

1.1 Man in the middle attack

Oscar has intercepted g^a from Alice and g^b from Bob, and replaced them with g^c. Thus, Oscar has g^{ac} and g^{bc}, allowing them to decrypt both messages and read the information. A real-world example is WhatsApp acting as Oscar, intercepting messages between Alice and Bob.

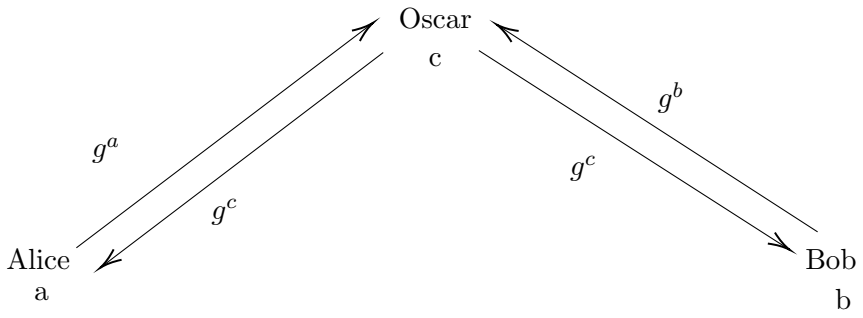

Figure 1: Illustration of a Man-in-the-Middle Attack

1.2 Finding C from x^C

We know we can write $C = \sum_{i=1}^{l-1} C_i 2^i$

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

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

We can use Square and Multiply algorithm for it

1.3 Square and Multiply

$Z \leftarrow 1$
for $i = l - 1$ **to** 0 **do**
  $Z \leftarrow Z^2$
  if $C_i == 1$ **then**
    $Z \leftarrow Z * x$
  end if
end for
return Z

2 RSA

The RSA algorithm is a popular public-key encryption method. To generate a key pair, follow these steps:

1. Choose two distinct prime numbers p and q.
2. Calculate $n = pq$.
3. Calculate $\phi(n) = (p-1)(q-1)$. It is the number of integers which are less than n and are also coprime to it.
4. Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
5. Calculate the unique integer d such that $de \equiv 1 \pmod{\phi(n)}$.

The public key is (n, e) and the private key is (n, d).

3 Euler's Theorem

If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$, where $\varphi(m)$ is Euler's totient function.

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

$S = \{s_1, s_2, s_3, \ldots, s_\varphi\}$

Multiplying by a:

$S_1 = \{as_1, as_2, as_3, \ldots, as_\varphi\}$

where $as_i \neq as_j$

But it is possible that we can have $s_i = as_j$

$\prod_{i=1}^{\varphi(m)} S_i \equiv \prod_{j=1}^{\varphi(m)} a \cdot S_j \pmod{m}$

$\prod_{i=1}^{\varphi(m)} S_i \equiv a^{\varphi(m)} \cdot \prod_{j=1}^{\varphi(m)} \cdot S_j \pmod{m}$

$\gcd(x, m) \equiv 1$

$a^{\varphi(m)} \equiv 1 mod(m)$

4 Fermat's Little Theorem

If p is a prime number and p*a then.

$a^{p-1} \equiv 1 mod(p)$

So we can also say that $a^p \equiv a \cdot mod(p)$. This is known as the "shortcut" or "repeated squaring" method for modular exponentiation. It is much faster than computing a^p directly and then reducing the result modulo p, since it involves only $O(\log p)$ modular multiplications rather than $O(p)$.

Fermat's Little Theorem is used in many cryptographic algorithms, including RSA, where it is used to compute the private key exponent d from the public key exponent e and the prime factors p and q of the modulus n. It is also used in primality testing algorithms to check whether a given integer is prime.

5 RSA Cryptosystem

- $n = pq$
- Plaintext space = Z_n
- Ciphertext space = Z_n
- Key Space = $\{K = (n, p, q, e, d) | (ed \equiv 1 (mod \varphi(m))\}$
- Encryption: Enc(x,k) = C
- c = Enc(x,k) = x^e(mod n)
- Decryption: Dec(c,k) = x
- x = Dec(c,k) = c^d(mod n)

  $ed \equiv 1 mod \varphi(n)$

  $ed - 1 = t \cdot \varphi(n)$

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

Decryption: 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 write x as a pair of residues (x_p, x_q) modulo p and q, respectively. That is, $x \equiv x_p \pmod{p}$ and $x \equiv x_q \pmod{q}$. By the Chinese Remainder Theorem, we can recover x from (x_p, x_q) using: $x \equiv x_p q q_p^{-1} + x_q p p_q^{-1} \pmod{n}$

5.1 Example

For Alice

- n= p.q
- $e \cdot d \equiv 1 (mod \varphi(n)$
- Public Key of alice = n,e
- Secret Key of Alice = p,q,d

For Bob:

- x
- y = $x^e (mod n)$

Encryption is done using public key component and decryption is done using secret key component hence we'll be having $y^d (mod n) = x$

5.2 Security in RSA

The security of RSA is based on the assumption that factoring the large composite number $n = pq$, where p and q are large primes, is computationally hard. If an attacker can factor n, then they can recover the private key and decrypt any ciphertext encrypted under that public key. However, factoring large integers is currently believed to be infeasible for sufficiently large primes p and q, making RSA a secure encryption scheme.

The RSA problem is the task of recovering the plaintext x from the public key (n, e) and the ciphertext $c = x^e$. This problem is equivalent to the integer factorization problem, since knowledge of the prime factors p and q of n allows us to compute the private key $d = e^{-1} \bmod \varphi(n)$ and decrypt the ciphertext as $x = c^d \bmod n$.

In practice, the security of RSA relies on using sufficiently large key sizes to make factoring computationally infeasible. The recommended key size for RSA is currently 2048 bits or larger, which makes factoring the modulus using current algorithms prohibitively expensive. However, advances in quantum computing and new factoring algorithms may pose a threat to the security of RSA in the future.

6 DSA - Digital Signature Algorithm

Digital Signature Algorithm (DSA) is a method for creating and verifying digital signatures, ensuring that a message has not been altered by anyone. Consider the following example:

- Alice wants to send a message (m) to Bob securely.

- Alice signs the message with her private key and sends it to Bob.
- Let $n = p \cdot q$ and $\phi(n) = (p-1) \cdot (q-1)$.
- Choose e and d such that $e \cdot d \equiv 1 \pmod{\phi(n)}$.
- Alice generates a public key y and a private key x using the formula: $y = g^x \mod p$.
- Alice signs the message (m) with her private key (x) and sends it to Bob.
- The signature is calculated as $s = m^d \mod n$.
- Bob verifies the signature using Alice's public key (y).

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