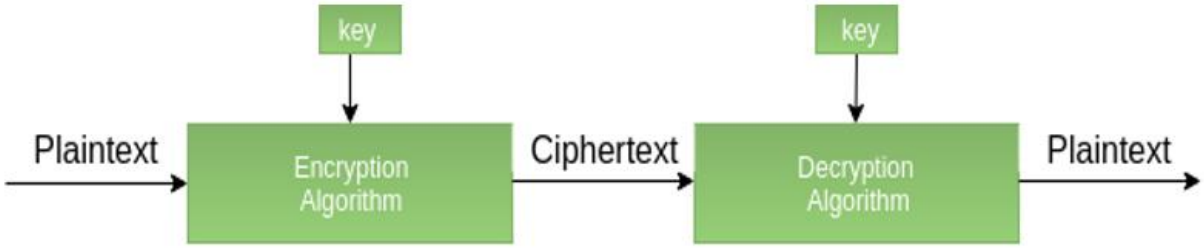
Module 1: Introduction to Cryptography

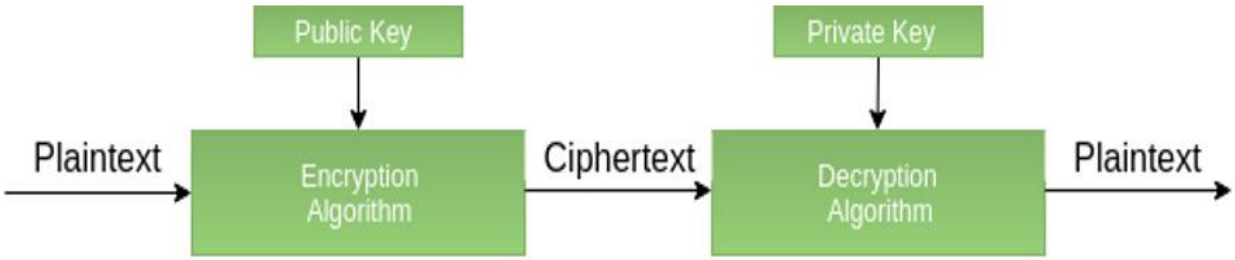
'Crypto' means secret or hidden. Cryptography is the science of secret writing with the intention of keeping the data secret. Cryptanalysis, on the other hand, is the science or sometimes the art of breaking cryptosystems. Both terms are a subset of what is called Cryptology. Cryptography is an important aspect when we deal with network security.

Cryptography is classified into **symmetric** cryptography and **asymmetric** cryptography. In some instances, Cryptography is classified into classical and modern cryptography. Below is the description of these types.

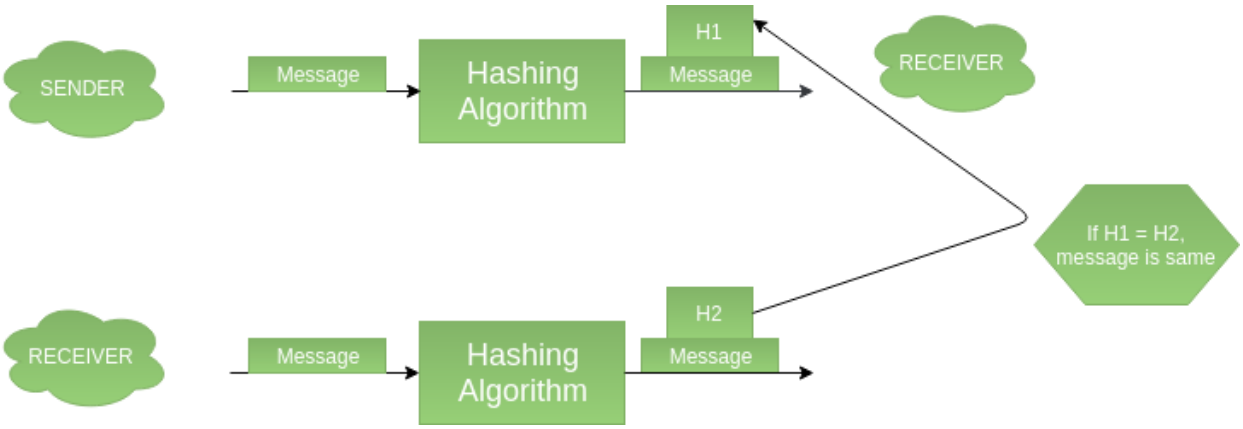
Symmetric key cryptography - It involves the usage of one secret key along with encryption and decryption algorithms which help in securing the contents of the message. The strength of symmetric key cryptography depends upon the number of key bits. It is relatively faster than asymmetric key cryptography. There arises a key distribution problem as the key has to be transferred from the sender to the receiver through a secure channel.



Asymmetric key cryptography: It is also known as public-key cryptography because it involves the usage of a public key along with the secret key. It solves the problem of key distribution as both parties use different keys for encryption/decryption. It is not feasible to use for decrypting bulk messages as it is very slow compared to symmetric key cryptography.

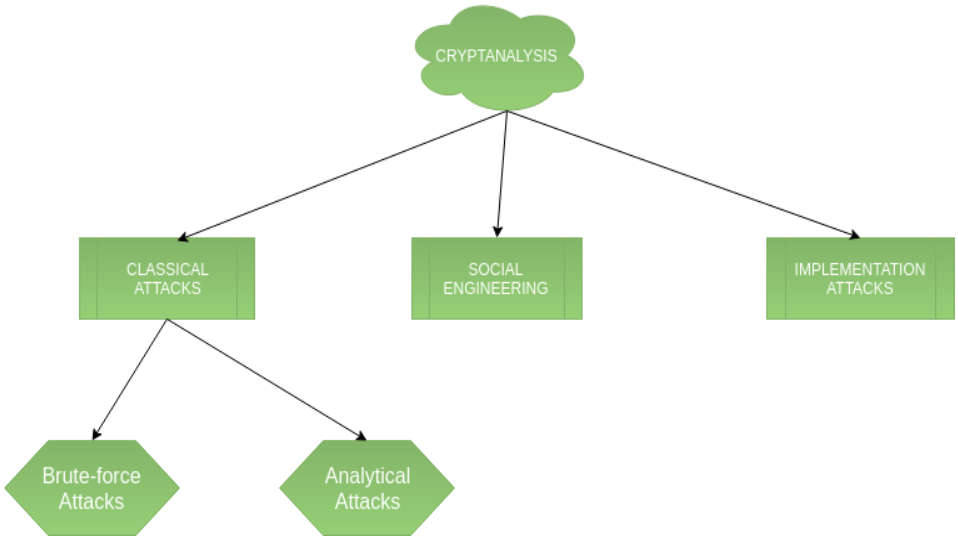


Hashing: It involves taking the plain text and converting it to a hash value of fixed size by a hash function. This process ensures the integrity of the message as the hash value on both, the sender's and receiver's sides should match if the message is unaltered.



Cryptanalysis

Cryptanalysis refers to the process of analyzing information systems in order to understand hidden aspects of the systems. Cryptanalysis is used to breach cryptographic security systems and gain access to the contents of encrypted messages, even if the cryptographic key is unknown.



Cryptographic attacks can be divided into:

1. **Classical attacks**

    a) Mathematical analysis: It's a type of attack that takes advantage of structural flaws in a specific algorithm.

    b) Brute-force attacks: The attacker uses a Brute Force Attack (BFA) to try all potential keys in order to figure out the key. If the key is long, the attack will take a long time to execute. Brute-force attacks run the encryption algorithm for all possible cases of the keys until a match is found. The encryption algorithm is treated as a black box. Analytical attacks are those attacks that focus on breaking the cryptosystem by analyzing the internal structure of the encryption algorithm.

2. **Social Engineering attack**: It is something that is dependent on the human factor. Tricking someone to reveal their passwords to the attacker or allowing access to the restricted area comes under this attack. People should be cautious when revealing their passwords to any third party which is not trusted.

Terms used in Cryptography

Plain Text

The original message that the person wants to connect with the other is represented as Plain Text. In cryptography the actual message that has to be send to the other end is provided as a specific name as Plain Text.

Cipher Text

The message that cannot be learned by anyone or meaningless message is what it can call as Cipher Text. In Cryptography the original message is changed into non-readable message before the communication of actual message.

Encryption

A process of transforming Plain Text into Cipher Text is known as Encryption. Cryptography needs the encryption approach to send confidential messages through an insecure channel.

Decryption

A reverse process of encryption is known as Decryption. It is a procedure of transforming Cipher Text into Plain Text. Cryptography needs the decryption approach at the receiver side to acquire the original message from non-readable message (Cipher Text).

Key

A Key is a numeric or alpha numeric text or can be a unique symbol. The Key can be used at the time of encryption takes place on the Plain Text and at the time of decryption create place on the Cipher Text.

Hashing

Hash algorithms are frequently used to support a digital fingerprint of a file's contents used to support that the file has not been converted by an intruder or virus. Hash functions are also generally employed by some operating framework to encrypt passwords. Hash functions supports a measure of the integrity of a record.

Module 2: Historical Ciphers

In this part we discuss some historical ciphers. However, understanding the construction of historical ciphers and how they were broken enables one to get a view of how modern cryptosystems came to be designed as they are. For example, modern block ciphers are built out of two key primitives, substitution and permutation, both of which occur in the construction of historical ciphers.

Encryption of most data today is accomplished using fast block and stream ciphers. These are examples of symmetric encryption algorithms. In addition, all historical, i.e. pre-1960, ciphers are symmetric in nature and share some design principles with modern ciphers. The main drawback of symmetric ciphers is that they give rise to the problem of how to distribute the secret keys.

An encryption algorithm, or cipher, is a means of transforming plaintext into ciphertext under the control of a secret key. This process is called encryption or encipherment. We write

$$c = e_k(m),$$

where

- m is the plaintext,
- e is the cipher function,
- k is the secret key,
- c is the ciphertext.

The reverse process is called decryption or decipherment, and we write

$$m = d_k(c).$$

Note that the encryption and decryption algorithms e, d are public: the secrecy of m given c depends totally on the secrecy of k.

The above process requires that each party needs access to the secret key. The key needs to be known to both sides, but needs to be kept secret. Encryption algorithms which have this property are called symmetric cryptosystems or secret key cryptosystems.

Shift Cipher

We first present one of the earliest ciphers, called the shift cipher. Encryption is performed by replacing each letter by the letter located a certain number of places further on in the alphabet. So for example if the key was three, then the plaintext A would be replaced by the ciphertext D, the letter B would be replaced by E and so on. The plaintext word **HELLO** would be encrypted as the ciphertext **KHOOR**.

When this cipher is used with the key three, it is often called the **Caesar cipher**, although in many books the name Caesar cipher is sometimes given to the shift cipher with any key. Strictly this is not correct since we only have evidence that Julius Caesar used the cipher with the key three.

How the algorithm works

First, we need to identify each letter of the alphabet with a number. It is usual to identify the letter A with the number 0, the letter B with number 1, the letter C with the number 2 and so on until we identify the letter Z with the number 25. After we convert our plaintext message into a sequence of numbers, the ciphertext in the shift cipher is obtained by adding to each number the secret key k modulo 26, where the key is a number in the range 0 to 25. In this way we can interpret the shift cipher as a stream cipher, with keystream given by the repeating sequence *k, k, k, k, k, k,.....* This keystream is not very random, which results in it being easy to break the shift cipher. A naïve way of breaking the shift cipher is to simply try each of the possible keys in turn, until the correct one is found. There are only 26 possible keys so the time for this exhaustive key search is very small, particularly if it is easy to recognize the underlying plaintext when it is decrypted.

Practical Questions

1. Encrypt the message "HELLO" using a shift cipher with key = 3.
2. Encrypt the message "SECURITY" using a Caesar cipher with a shift of 7.
3. Encrypt "ATTACK AT DAWN" with a shift of 4.
4. Decrypt the ciphertext "KHOOR" using a Caesar cipher with key = 3
5. Given the ciphertext "ZEBBWX", decrypt it using a shift of 5.
6. Decrypt "GWTYMJ NX ST QNRJ" with a Caesar cipher key = 5.
7. The ciphertext "FRZDUG" is obtained using a Caesar cipher. Find the plaintext and the key using brute-force attack.

Substitution Cipher

The main problem with the shift cipher is that the number of keys is too small; we only have 26 possible keys. To increase the number of keys the substitution cipher was invented. To write down a key for the substitution cipher we first write down the alphabet, and then a permutation of the alphabet directly below it. This mapping gives the substitution we make between the plaintext and the ciphertext.

Plaintext alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext alphabet GOYDSIPELUAVCRJWXZNHBQFTMK

Encryption involves replacing each letter in the top row by its value in the bottom row. Decryption involves first looking for the letter in the bottom row and then seeing which letter in the top row maps to it. Hence, the plaintext word HELLO would encrypt to the ciphertext ESVVJ if we used the substitution given above.

Practical Questions

1. Encrypt the message "HELLO" using the substitution key:
   Plain: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
   Cipher: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

2. Use the following substitution mapping to encrypt "ATTACK AT DAWN"
   A → M, T → Q, C → Z, K → S, D → H, W → V, N → J

3. Create your own substitution cipher key and encrypt the phrase "SECURE SYSTEMS".

4. Decrypt the ciphertext "ITSSG" using this substitution key:
   Cipher: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M
   Plain: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

5. Decrypt the ciphertext "ZQZZTQ" given the following partial key:
   Z → A, Q → T, T → K

6. You intercepted this ciphertext:
   "ZIT JXOEA WKGVF YGB PXDHL VJUQ NKMRS".
Assuming it's encrypted with a monoalphabetic substitution cipher, describe how you'd start cracking it using frequency analysis.

7. The ciphertext "WKH HDJOH KDV ODQGHG" seems suspicious.
   Try decrypting it by guessing the mapping and explain your steps.

Vigen`ere Cipher

The problem with the shift cipher and the substitution cipher was that each plaintext letter always encrypted to the same ciphertext letter. Hence underlying statistics of the language could be used to break the cipher. For example, it was easy to determine which ciphertext letter corresponded to the plaintext letter E. From the early 1800s onwards, cipher designers tried to break this link between the plaintext and ciphertext.

The substitution cipher we used above was a mono-alphabetic substitution cipher, in that only one alphabet substitution was used to encrypt the whole alphabet. One way to solve our problem is to take a number of substitution alphabets and then encrypt each letter with a different alphabet. Such a system is called a polyalphabetic substitution cipher. For example, we could take

Plaintext alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext alphabet one	TMKGOYDSIPELUAVCRJWXZNHBQF
Ciphertext alphabet two	DCBAHGFEMLKJIZYXWVUTSRQPON

Then we encrypt the plaintext letters in odd-numbered positions encrypt using the first ciphertext alphabet, whilst we encrypt the plaintext letters in even-numbered positions using the second alphabet. For example, the plaintext word HELLO would encrypt to SHLJV, using the above two alphabets. Notice that the two occurrences of L in the plaintext encrypt to two different ciphertext characters. Thus, we have made it harder to use the underlying statistics of the language. If one now does a naive frequency analysis one no longer obtains a common ciphertext letter corresponding to the plaintext letter E.

Practical Questions

1. Given the plaintext message ATTACKATDAWN and the keyword LEMON.
   Encrypt the message using the Vigenère cipher.

2. You receive the Vigenère ciphertext RIJVSUYVJN and the keyword KEY.
   Decrypt the message to retrieve the original plaintext.

3. You intercept a ciphertext that you suspect is encrypted with the Vigenère Cipher LXFOPVEFRNHR. You also know the plaintext starts with “ATTACK”. Use the known-plaintext attack approach to determine the key used for encryption.

Module 3: Essential Number Theory for Public-Key Algorithms

We will now study a few techniques from number theory which are essential for public-key cryptography. We introduce the Euclidean algorithm.

We start with the problem of computing the greatest common divisor (GCD), also known as the Highest Common Factor (HCF) or Greatest Common Factor (GCF), is the largest number that divides two or more numbers without leaving a remainder. Several methods can be used to calculate the GCD, including the Euclidean algorithm, prime factorization, and the LCM method.

The gcd of two positive integers r_0 and r_1 is denoted by

$$\gcd(r_0,r_1)$$

and is the largest positive number that divides both r_0 and r_1. For instance gcd(21,9) = 3. For small numbers, the gcd is easy to calculate by factoring both numbers and finding the highest common factor.

Example

Let $r_0 = 84$ and $r_1 = 30$. Factoring yields

$$r_0 = 84 = 2 \cdot 2 \cdot 3 \cdot 7$$
$$r_1 = 30 = 2 \cdot 3 \cdot 5$$

The gcd is the product of all common prime factors:

$$2 \cdot 3 = 6 = \gcd(30,84)$$

For the large numbers which occur in public-key schemes, however, factoring often is not possible, and a more efficient algorithm is used for gcd computations, the Euclidean algorithm. The algorithm, which is also referred to as Euclid's algorithm, is based on the simple observation that $\gcd(r_0,r_1) = \gcd(r_0 - r_1,r_1)$,

where we assume that $r_0 > r_1$, and that both numbers are positive integers. This property can easily be proven: Let $\gcd(r_0,r_1) = g$. Since g divides both r_0 and r_1, we can write $r_0 = g \cdot x$ and $r_1 = g \cdot y$, where $x > y$, and x and y are coprime integers, i.e., they do not have common factors. Moreover, it is easy to show that $(x-y)$ and y are also coprime. It follows from here that: $\gcd(r_0 - r_1,r_1) = \gcd(g \cdot (x-y),g \cdot y) = g$.

Let's verify this property with the numbers from the previous example:

Example. Again, let $r_0 = 84$ and $r_1 = 30$. We now look at the gcd of $(r_0 - r_1)$ and r_1:

$$r_0 - r_1 = 54 = 2 \cdot 3 \cdot 3 \cdot 3$$
$$r_1 = 30 = 2 \cdot 3 \cdot 5$$

The largest common factor still is $2 \cdot 3 = 6 = \gcd(30,54) = \gcd(30,84)$.

It also follows immediately that we can apply the process iteratively:

$$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1) = \gcd(r_0 - 2r_1, r_1) = \cdots = \gcd(r_0 - m\, r_1, r_1)$$

as long as $(r_0 - m\, r_1) > 0$. The algorithm uses the fewest number of steps if we choose the maximum value for m. This is the case if we compute:

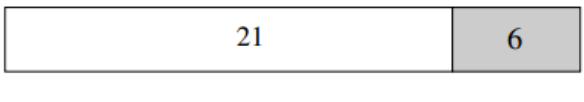
$$\gcd(r_0, r_1) = \gcd(r_0 \bmod r_1, r_1).$$

Since the first term $(r_0 \bmod r_1)$ is smaller than the second term r_1, we usually swap them:

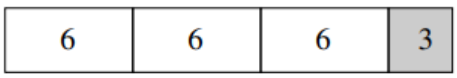
$$\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1).$$

The core observation from this process is that we can reduce the problem of finding the gcd of two given numbers to that of the gcd of two smaller numbers. This process can be applied recursively until we obtain finally $\gcd(r_l,0) = r_l$. Since each iteration preserves the gcd of the previous iteration step, it turns out that this final gcd is the gcd of the original problem, i.e.,

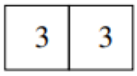
Example. Let $r_0 = 27$ and $r_1 = 21$.



gcd(27, 21) = gcd(1 · 21+6, 21) = gcd(21, 6)



gcd(21, 6) = gcd(3 · 6+3, 6) = gcd(6, 3)



gcd(6, 3) = gcd(2 · 3+0, 3) = gcd(3, 0) = 3

gcd(27, 21) = gcd(21, 6) = gcd(6, 3) = gcd(3, 0) = 3

The solution above gives us some feeling for the algorithm by showing how the lengths of the parameters shrink in every iteration. The shaded parts in the iteration

are the new remainders $r_2 = 6$ (first iteration), and $r_3 = 3$ (second iteration) which form the input terms for the next iterations.

Example. Let $r_0 = 973$ and $r_1 = 301$. The gcd is then computed as

$973 = 3 \cdot 301 + 70$	$\gcd(973, 301) = \gcd(301, 70)$
$301 = 4 \cdot 70 + 21$	$\gcd(301, 70) = \gcd(70, 21)$
$70 = 3 \cdot 21 + 7$	$\gcd(70, 21) = \gcd(21, 7)$
$21 = 3 \cdot 7 + 0$	$\gcd(21, 7) = \gcd(7, 0) = 7$

By now we should have an idea of Euclid's algorithm, and we can give a more formal description of the algorithm.

Euclidean Algorithm
Input: positive integers r_0 and r_1 with $r_0 > r_1$
Output: $\gcd(r_0, r_1)$
Initialization: $i = 1$
Algorithm:

1 DO
1.1 $i = i + 1$
1.2 $r_i = r_{i-2} \bmod r_{i-1}$
WHILE $r_i \neq 0$
2 RETURN
$\gcd(r_0, r_1) = r_{i-1}$

Note that the algorithm terminates if a remainder with the value $r_i = 0$ is computed. The remainder computed in the previous iteration, denoted by $r_l - 1$, is the gcd of the original problem.

Problems

Using the basic form of Euclid's algorithm, compute the GCD of the following

1. **48, 18**
2. **56, 42**
3. **120, 45**
4. **210. 84**
5. **101, 23**
6. **756, 294**
7. **1234, 567**
8. **9876, 5432**
9. **7469, 2464**
10. **2689, 4001**
11. **198, 243**
12. **1819, 3587**

Primitive Root of a Number

A primitive root of a number 'n' (usually a prime number) is an integer 'g' such that when you raise it to successive powers modulo 'n', you get all the integers from 1 to n-1. In simpler terms, it's a number that "generates" all the other numbers relatively prime to 'n' through modular exponentiation.

Mathematically, if α is said to be a primitive root of a prime number p, if α^1 mod p, α^2 mod p, α^3 mod p, …….. α^{p-1} mod p are distinct.

Example
If n is a prime number, like 5, the numbers coprime to 5 are 1, 2, 3, and 4. If 2 is a primitive root modulo 5, then $2^1 \equiv 2 \pmod 5$, $2^2 \equiv 4 \pmod 5$, $2^3 \equiv 8 \equiv 3 \pmod 5$, and $2^4 \equiv 16 \equiv 1 \pmod 5$. You get all the numbers 1, 2, 3, and 4. Therefore, 2 is a primitive root of 5.

Primitive roots are important in number theory, cryptography (like the Diffie-Hellman key exchange), and other areas. In essence, a primitive root provides a way to "cycle through" all the numbers relatively prime to a given number using powers and remainders.

Problems

1. Is 2 a primitive of a prime number 5?
2. Is 3 a primitive of a prime number 7?
3. Is 2 a primitive of a prime number 11?

Module 4: Diffie-Hellman Key Agreement Protocol

The Diffie-Hellman key exchange was first published in 1976 by Whitfield Diffie and Martin Hellman in their paper "New Directions in Cryptography". Key agreement allows two parties to securely agree on a symmetric key via a public channel, such as the Internet, with no prior key exchange. An attacker who is able to sniff the entire conversation is unable to derive the exchanged key. Whitfield Diffie and Martin Hellman created the Diffie-Hellman Key Agreement Protocol (also called the Diffie-Hellman Key Exchange) in 1976. Diffie-Hellman uses discrete logarithms to provide security.

Why Diffie-Hellman Key Exchange Algorithm Needed?

A key exchange algorithm is needed in communication and cryptography for several reasons:

1. It enables two or more parties to agree upon a secret key without exposing it to potential eavesdroppers, the key is then used for encryption and decryption, which is vital for maintaining confidentiality in communication.
2. Preserving the data integrity was also a major challenge in digital communication where data is always vulnerable to tempering while transmission. A key exchange algorithm helps in preserving the integrity of the transmitted data, it prevents unauthorized alteration or tampering of data during transmission.
3. A key exchange algorithm facilitates authentication of the communicating parties, verifies who they claim to be, thus escalating the risk of man-in-the-middle (impersonation) attack.

Thus, along with encrypting the data for maintaining the confidentiality of the communication, a key exchange algorithm was also needed to maintain the integrity and authorized access of the information.

Key Generation & Exchange Process

Diffie-Hellman key exchange algorithm is based on the principles of modular exponentiation and discrete logarithms to allow two parties to securely establish a shared secret key over an insecure communication channel. Here is an operational overview of the process in context to Alice and Bob:

1. Parameters Setup

Alice and Bob must agree upon two number:

- A large prime number **p**,
- A generator **g** of p, which is the primitive root of p

These two numbers are shared and are not kept secret.

2. Key Generation

- Alice and Bob randomly chose a private key, say X_a and X_b, where X_a is the private key of Alice and X_b is the private key of Bob.
- These private keys are kept secret and not being shared.

3. Public Key Exchange

- Both Alice and Bob perform a calculation to generate their corresponding public keys.

  $Y_a = g^a \pmod p$

  $Y_b = g^b \pmod p$,

  where Y_a is the public key of Alice and Y_b is the public key of Bob

- The public keys are then shared with each other, Y_a is shared with Bob and Y_b is shared with Alice.

4. Shared Secret Key Calculation

- Alice then calculates the shared secret using the Y_b received from Bob and her private key as:

  $k = (Y_b)^{Xa} \pmod p$

- Bob also calculates the shared secret using the Y_a received from Alice and his private key X_b as:

  $k = (Y_a)^{Xb} \pmod p$

5. Resulting Secret

Alice and Bob will end upon the same shared secret key, which can be used for encryption and decryption of information using symmetric key algorithms.

Example

Suppose Alice and Bob agreed on p as 7 and g as 5. Find the value of secret keys?

Solution:
Let us suppose Xa be 3, then Ya can be given as:
$Ya = 53 \bmod 7 = 6$

Also let us assume Xb as 4, then Yb can be calculated as:
$Yb = 54 \bmod 7 = 2$

The value of k which is the secret key, can be calculated and verified as:
$k = 23 \bmod 7 = 64 \bmod 7 = 1$
k = 1, for both the calculations, which is the shared secret.

Strength of Diffie-Hellman Key Exchange Algorithm

The Diffie-Hellman key exchange is secure because of the difficulty of calculation discrete logarithms. An eavesdropper listening to the communication channel for the exchanged value of Ya and Yb would find it extremely difficult to determine shared secret without knowing the value of Xa or Xb which are private keys and a limited to the one party.
Thus, it allows two parties (say Alice and Bob) to securely establish a shared secret key over an insecure channel without the need to transmit the key itself, establishing a secure means for encrypted communication eliminating the vulnerabilities associated in direct transmission of keys.

Perfect Forward Secrecy (PFS)
Perfect Forward Secrecy is the property in the cryptography that prevents the exposure of long-term secret keys from compromising the past or future communication. In context to Diffie-Hellman, prefect forward secrecy means that even if an attacker were somehow gain / compute the private keys used during a session, he would not be able to decrypt past communications or use those keys to decrypt any of the future communication. It's an important property of a systems where the long-term security of data is crucial, it helps to prevent the accumulation of data over time, making it more complex for attackers to decrypt large amounts of data even if they obtain private keys or have the ability to eavesdrop on communications.

Key Aspects of Perfect Forward Secrecy

- **Use of Session Keys:** Systems that implements Perfect Forward Secrecy generates a unique session key for every session, so even if an attacker manages to know the current session key, it cannot use it to decrypt past or future communications, as each session keys becomes invalid after session is over.
- **Temporary Keys:** The use of temporary keys generated by Perfect Forward Secrecy system for each session, which is not use for other sessions, ensuring that if one key is compromised, it doesn't compromise other communications.
- **Zero Dependence on Long-term Keys:** The long-term keys used for authentication or key exchange; in case they are compromised other communication remains secured. As they are used for the purpose of establishing the session keys.
- **Enhanced Security:** Perfect Forward Secrecy add a layer to the security, in scenarios where long-term keys might be at risk due to various factors such as complex cyber-attacks, compromised servers, or future cryptographic development.

Problems

1. Prime modulus p=23, primitive root g=5.
   - Alice chooses secret key a=6.
   - Bob chooses secret key b=15.

   Compute the shared secret key.

2. Prime modulus p=17, base g=3.
   - Alice’s secret key is a=7.
   - Bob’s secret key is b=11.

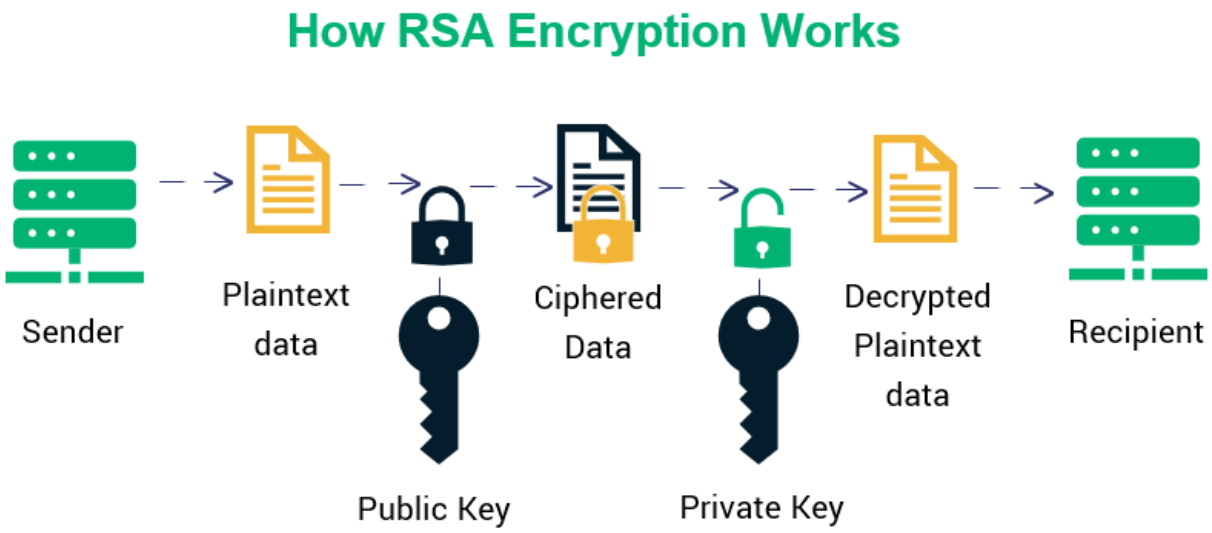
   Find the public keys exchanged and the final shared secret.

3. If Alice and Bob agree on p=19, g=2:
   - Alice chooses a=5.
   - Bob chooses b=13.

   Show all steps to find the shared secret.

Module 5: RSA Cryptosystem

The RSA(Rivest-Shamir-Adleman) Algorithm is an asymmetric or public-key cryptography algorithm which means it works on two different keys: Public Key and Private Key. The Public Key is used for encryption and is known to everyone, while the Private Key is used for decryption and must be kept secret by the receiver. RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.



Example of Asymmetric Cryptography:
If Person **A** wants to send a message securely to Person **B**:

- Person A **encrypts** the message using Person B's **Public** Key.
- Person B **decrypts** the message using their **Private** Key.

RSA Algorithm
*RSA Algorithm is based on **factorization** of large number and **modular arithmetic** for encrypting and decrypting data. It consists of three main stages:*

1. ***Key Generation:** Creating Public and Private Keys*
2. ***Encryption:** Sender encrypts the data using Public Key to get **cipher text**.*
3. ***Decryption:** Decrypting the **cipher text** using Private Key to get the original data.*

Standard RSA Algorithm

Phase 1: Key Generation

1: Initialize two random large primes as p, q

2: Evaluate modulus n as,

$n = p*q$

3: Find Euler totient function as,

$\varphi(n) = (p-1)*(q-1)$

4: Now, compute public key exponent e, such that,

$1 < e < \varphi(n)$, and GCD of $(e, \varphi(n)) = 1$

5: And compute private key exponent d, such that,

$e*d = 1 \bmod \varphi(n)$

6: Generated Public key is (n, e), and Private key is (n, d)

Phase 2: Encryption

$c = m^e \bmod n$,

that uses the public key components as (n, e),

and Plaint text message as m

Phase 3: Decryption

$m = c^d \bmod n$,

that uses the private key components as (n, d),

where c is the encoded cipher string and m is the original string.

Example 1:
Given Two [prime numbers **p=3 & q=5**

Key Generation

n=pxq

n=3x5 = 15

$\Phi(n) = (p-1).(q-1) = 8$

Chosen e such that gcd(e, $\Phi(n)$)=1 and 1<e< $\Phi(n)$

Let e=3

Compute private key d such that

e.d = 1* mod $\Phi(n)$

= 3.d = 1 mod 8

Lets d = 3

3.3 = 1 mod 8

9 mod 8 = 1

Therefore, d=3

Public key = {e,n} ={3,15}
Private key = {d,n} = {3,15}

Encryption

$C = P^e \bmod n$

Using h as plaintext

$C = 8^3 \bmod 15$

$C = 2$

Decryption

$P = C^d \bmod n$

Using 2 as cyphertext

$P = 2^3 \bmod 15$

$P = 8$

Questions

1. Given p=17 and q=11, compute:
    - n
    - φ(n)
    - Choose e=7, compute the private key d.
2. Using the RSA keys from Question 1, encrypt the message M=88.
3. With the same keys, decrypt the ciphertext C=11.
4. Suppose p=61 and q=53, compute the RSA public and private keys if e=17.
5. Using the keys from Question 4, encrypt the message M=65 and show the ciphertext.
6. Show that decryption reverses encryption in RSA by proving $M^{ed} \equiv M (\mod n)$.
7. A user chooses p=13, q=19, and e=5. Compute d and show how a message M=10 is encrypted and decrypted.
8. Explain what would happen if e and φ(n) are not coprime.
9. A message is encrypted with RSA and produces ciphertext C=21 using public key (e=5, n=77). Find the plaintext message M.
10. Demonstrate how RSA can be used to generate a digital signature for message M=25, given p=7,q=11,e=17.