Define discrete logarithms. How key generation, encryption, and decryption are done in RSA. In an RSA cryptosystem, given p=13 and q=7, determine the private key, and public key and perform encryption and decryption for the text M=" hi" using 0 to 25 for letters from a to z.

Discrete Logarithm:

- It is a one-way function and in mathematics, the logarithm of a number x concerning a base b is the exponent to which the base must be raised to produce the number x. In symbols, $\log_b (x) = y$ means $b^y = x$

Discrete Logarithms

The discrete logarithm is a concept from number theory, used extensively in cryptography. Given a finite cyclic group G, a generator g of G, and an element h in G, the discrete logarithm problem (DLP) asks for an integer x such that:

$$g^x \equiv h \pmod{p}$$

where p is a prime number. In other words, the discrete logarithm x is the power to which the generator g must be raised to obtain h.

RSA Cryptosystem

RSA (Rivest-Shamir-Adleman) is a widely used public-key cryptosystem for secure data transmission. The RSA algorithm involves three main steps: key generation, encryption, and decryption.

Key Generation

1. **Choose two distinct prime numbers** p and q.
2. **Compute** n: $n = p \times q$. This n is used as the modulus for both the public and private keys.
3. **Compute Euler's totient function** $\phi(n)$: $\phi(n) = (p-1)(q-1)$.
4. **Choose an integer** e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$. The integer e is the public key exponent.
5. **Determine** d: d is the modular multiplicative inverse of e modulo $\phi(n)$. This means d satisfies the equation $d \times e \equiv 1 \pmod{\phi(n)}$.

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

Encryption

To encrypt a message M:

1. Convert the message M into an integer m such that $0 \leq m < n$.
2. Compute the ciphertext c using the public key (e, n):

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

Decryption

To decrypt the ciphertext c:

1. Compute the original message m using the private key (d, n):

2. Compute the ciphertext c using the public key (e, n):

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

Decryption

To decrypt the ciphertext c:

1. Compute the original message m using the private key (d, n):

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

Example: RSA with $p = 13$ and $q = 7$

1. **Compute** n:

   $$n = 13 \times 7 = 91$$

2. **Compute** $\phi(n)$:

   $$\phi(n) = (13 - 1)(7 - 1) = 12 \times 6 = 72$$

3. **Choose** e: Let's choose $e = 5$, which is coprime to 72.

4. **Determine** d: d is the modular inverse of 5 modulo 72. We find d such that:

   $$5d \equiv 1 \pmod{72}$$

   Using the Extended Euclidean Algorithm, we get $d = 29$.

   (Check: $5 \times 29 = 145 \equiv 1 \pmod{72}$)

The public key is $(e, n) = (5, 91)$ and the private key is $(d, n) = (29, 91)$.

Encryption and Decryption Example

Let's encrypt and decrypt the message "hi". Using the 0-25 encoding for letters (a=0, b=1, ..., z=25), the message "hi" is converted to numbers as follows:

- 'h' -> 7
- 'i' -> 8

We will encrypt and decrypt each character separately.

Encryption

Encryption

1. **Encrypt 'h' (7)**:

   $$c = 7^5 \pmod{91}$$

   Calculating 7^5 modulo 91:

   $$7^2 = 49$$
   $$7^4 = 49^2 = 2401 \equiv 28 \pmod{91}$$
   $$7^5 = 7 \times 28 = 196 \equiv 14 \pmod{91}$$

   So, $c = 14$.

2. **Encrypt 'i' (8)**:

   $$c = 8^5 \pmod{91}$$

   Calculating 8^5 modulo 91:

   $$8^2 = 64$$
   $$8^4 = 64^2 = 4096 \equiv 1 \pmod{91}$$
   $$8^5 = 8 \times 1 = 8$$

   So, $c = 8$.

The encrypted message is $(14, 8)$.

Decryption

1. **Decrypt 14**:

   $$m = 14^{29} \pmod{91}$$

   Calculating 14^{29} modulo 91 using repeated squaring and modulo reduction:

   $$14^2 = 196 \equiv 14 \pmod{91}$$
   $$14^4 = 14^2 \times 14^2 \equiv 14 \times 14 = 196 \equiv 14 \pmod{91}$$
   $$14^8 = (14^4)^2 \equiv 14 \times 14 = 196 \equiv 14 \pmod{91}$$
   $$14^{16} = (14^8)^2 \equiv 14 \times 14 = 196 \equiv 14 \pmod{91}$$
   $$14^{29} = 14^{16} \times 14^8 \times 14^4 \times 14 \equiv 14 \times 14 \times 14 \times 14 \equiv 14 \pmod{91}$$

   So, $m = 14$.

$8^4 = 64^2 = 4096 \equiv 1 \pmod{91}$
$8^5 = 8 \times 1 = 8$

So, $c = 8$.

The encrypted message is $(14, 8)$.

Decryption

1. **Decrypt 14**:

   $m = 14^{29} \pmod{91}$

   Calculating 14^{29} modulo 91 using repeated squaring and modulo reduction:

   $14^2 = 196 \equiv 14 \pmod{91}$
   $14^4 = 14^2 \times 14^2 \equiv 14 \times 14 = 196 \equiv 14 \pmod{91}$
   $14^8 = (14^4)^2 \equiv 14 \times 14 = 196 \equiv 14 \pmod{91}$
   $14^{16} = (14^8)^2 \equiv 14 \times 14 = 196 \equiv 14 \pmod{91}$
   $14^{29} = 14^{16} \times 14^8 \times 14^4 \times 14 \equiv 14 \times 14 \times 14 \times 14 \equiv 14 \pmod{91}$

   So, $m = 14$.

2. **Decrypt 8**:

   $m = 8^{29} \pmod{91}$

   Calculating 8^{29} modulo 91 using repeated squaring and modulo reduction:

   $8^2 = 64$
   $8^4 = 64^2 = 4096 \equiv 1 \pmod{91}$
   $8^8 = 1^2 = 1 \pmod{91}$
   $8^{16} = 1^2 = 1 \pmod{91}$
   $8^{29} = 8^{16} \times 8^8 \times 8^4 \times 8 \equiv 1 \times 1 \times 1 \times 8 = 8 \pmod{91}$

   So, $m = 8$.

The decrypted message is $(14, 8)$, which corresponds to "hi" in the 0-25 letter encoding.

Thus, the encryption and decryption processes work correctly in this example of the RSA cryptosystem.

Write down the encryption and decryption process at 2-DES and 3-

Encrypt: c = m^e (mod n).
Decrypt: m = c^d (mod n)

RSA Algorithm

- RSA(River-Shamir-Adleman) is an algorithm used to encrypt & decrypt messages.
- It is an asymmetric cryptographic algorithm

Encryption -

$C = P^e \bmod n$

Decryption -

$P = C^d \bmod n$

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

Encryption -

$C = P^e \bmod n$

Decryption -

$P = C^d \bmod n$

public key = $\{e, n\}$

private key = $\{d, n\}$

Key Generation

1) Consider two large prime numbers q, p
2) Calculate $n = p \times q$
3) $\phi(n) = (p-1)(q-1)$ $\phi(n)$ Euler's Totient function
4) Choose a small number e, co-prime to $\phi(n)$ with $GCD(\phi(n), e) = 1$ and $1 < e < \phi(n)$
5) Find d, such that $d \times e \bmod \phi(n) = 1$

Example - Key Generation

1) Two prime numbers $p = 3$, $q = 5$
2) $n = p \times q = 3 \times 5 = 15$

$\boxed{n = 15}$

3) $\phi(n) = (p-1)(q-1)$
   $\phi(n) = (3-1)(5-1)$
   $\phi(n) = 8$
4) Assume e such that $\gcd(e, \phi(n)) = 1$ & $1 < e < \phi(n)$

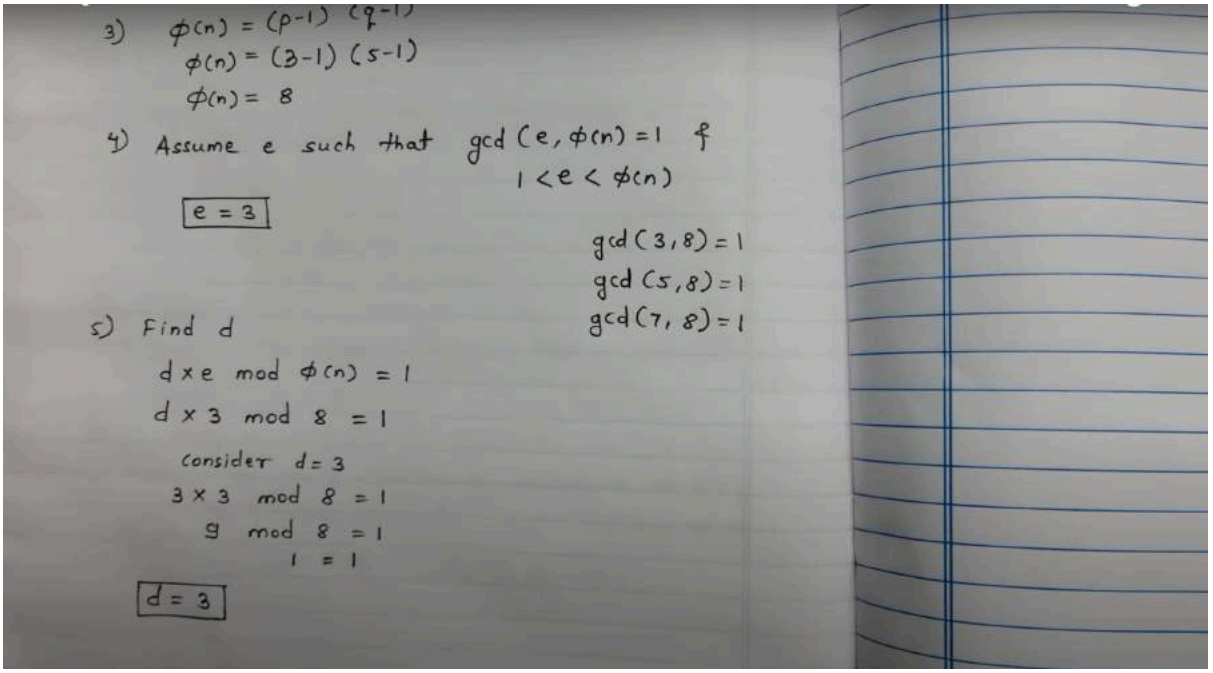
$\boxed{e = 3}$

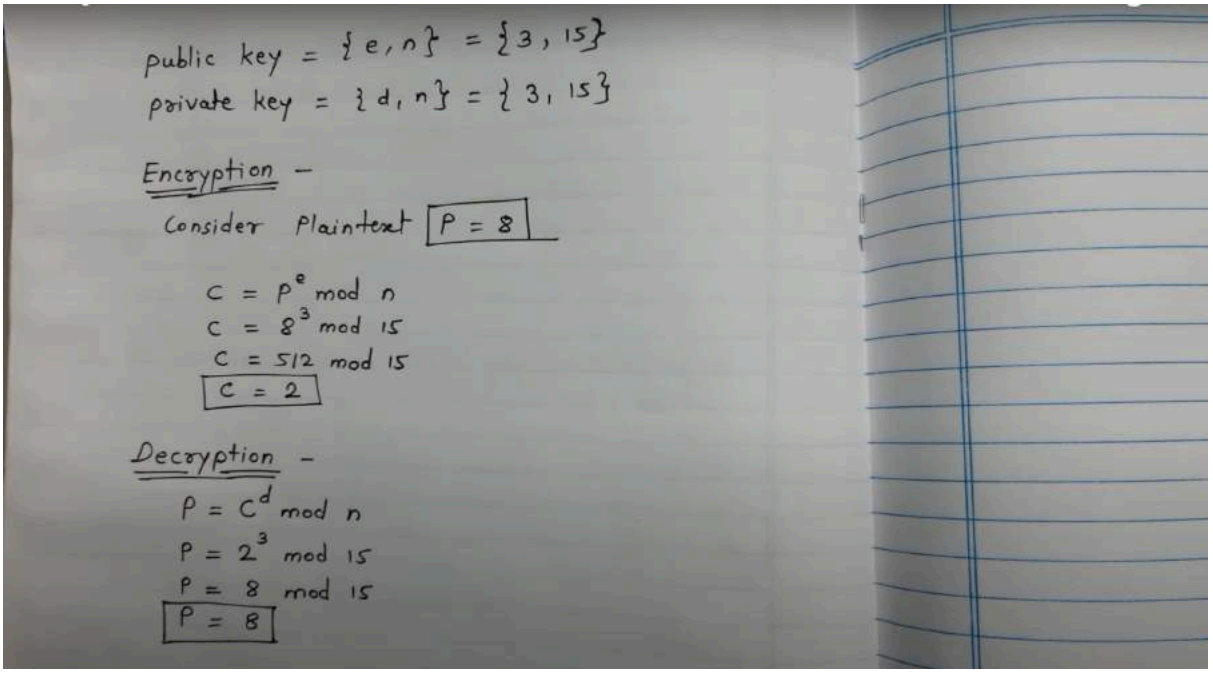
$\gcd(3, 8) = 1$
$\gcd(5, 8) = 1$
$\gcd(7, 8) = 1$

5) Find d

$d \times e \bmod \phi(n) = 1$

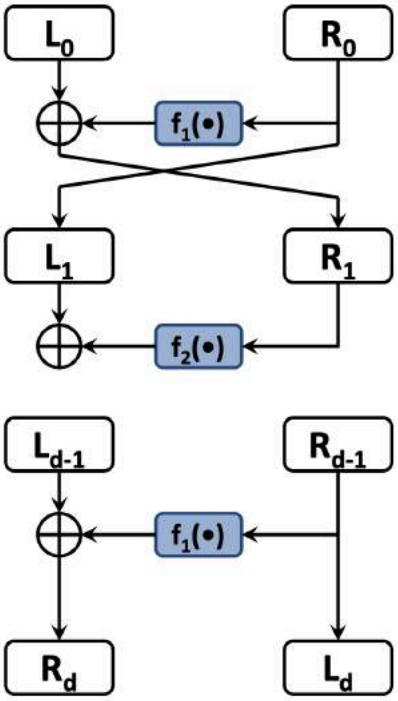
$d \times 3 \bmod 8 = 1$





Write down the encryption and decryption process at 2-DES and 3-DES. Explain the Fiestal cipher structure. Divide 5x2 + 4x + 6 by 2x + 1 over GF(7).

Feistel Network



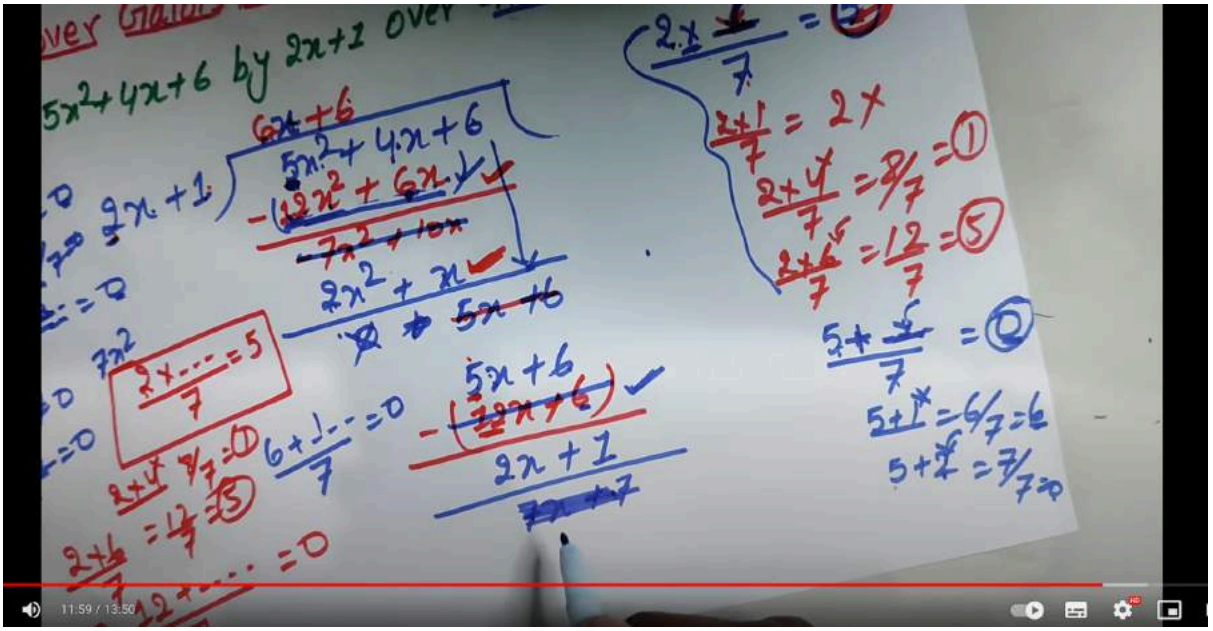
- **Encryption**:
  - $L_1 = R_0 \quad R_1 = L_0 \oplus f_1(R_0)$
  - $L_2 = R_1 \quad R_2 = L_1 \oplus f_2(R_1)$
  - ...
  - $L_d = R_{d-1} \quad R_d = L_{d-1} \oplus f_d(R_{d-1})$
- **Decryption**:
  - $R_{d-1} = L_d \quad L_{d-1} = R_d \oplus f_d(L_d)$
  - ...
  - $R_0 = L_1; \quad L_0 = R_1 \oplus f_1(L_1)$

5

- Several block ciphers are based on the structure proposed by *Feistel* in 1973
- A *Feistel Network* is fully specified given
  - the *block size*: n = 2w
  - *number of rounds*: d
  - d *round functions* $f_1, \ldots, f_d$: $\{0,1\}^w \rightarrow \{0,1\}^w$
- Used in DES, IDEA, RC5 (Rivest's Cipher n. 5), and many other block ciphers.
- Not used in AES

4

Divide 5x2 + 4x + 6 by 2x + 1 over GF(7).


11:59 / 13:50

Double DES

$$p \rightarrow E(k_1, \quad p) \rightarrow E(k_2, \quad E(k_1, \quad p)) = C$$

Double DES has a 112-bit key and enciphers blocks of 64 bits.

DES is not a group; i.e., $E(k_2, \quad E(k_1, \quad p))$ is not equivalent to DES encryption using a single key. Recall that, for example, the Caesar cipher is a group. If a message were encrypted with the Caesar cipher with a key of 3 and then re-encrypted with the Caesar cipher with a key of 5, the result is equivalent to encrypting the message with the Caesar cipher with a key of 8. For the Caesar cipher, double encryption does not increase security. DES is not a group; double encryption is not equivalent to single encryption. Security does increase by double encryption, but it does not increase much.

The security of DES depends on its having a large key space; so large that (at least when it first began being used in the 1970's a brute force attack was not practical [that has now changed]). Recall that DES has a 56-bit key (the key is actually 64 bits, but every 8^{th} bit is a parity check; so, only 56 or the 64 bits are meaningful); therefore, the size of the key space is $2^{56} = 72,057,594,037,927,936$. Recall that the algorithm that was originally proposed had a 128-bit key, but the size of the key space was reduced by the NSA (for some reason).

Intuitively, double encryption should double the size of the key space. But, that is not the case with DES.

Meet-in-the–middle attack on double encryption

This attack requires knowing some plaintext/ciphertext pairs. Let's assume that we have a plaintext/ciphertext pair; i.e., we know the plaintext p and the corresponding (double DES enciphered) ciphertext C. Attacks on DES have typically been brute force attacks (see "Breaking DES"); so, we will use brute force here.

Here is the double encryption:

Here is the double encryption:

$$p \rightarrow E(k_1, \ p) \rightarrow E(k_2, \ E(k_1, \ p)) = C$$

Encrypt p using all 2^{56} possible keys, and store the results. (Storage could be a problem.) The stored results will include all possible encryptions $p \rightarrow E(k_1, \ p)$.

Then decrypt C using all 2^{56} possible keys.

$$D(k_2, \ C) = D(k_2, \ E(k_2, \ E(k_1, \ p))) \rightarrow E(k_1, \ p)$$

After decrypting with each key, check for a match with the stored outputs of the 2^{56} possible encryptions. When we have a match, we have located a possibly correct pair of keys. Now, perhaps more than one pair of keys will result in a match, but the number of pairs of keys that return matches should be small. We could try each possible pair of keys. If more than one plaintext/ciphertext correspondence is known (for the key pair), then other correspondences could be used to check which of the keys is correct.

So, it only takes twice as long to break double DES using brute force. Because DES has 56-bit security, double DES has $2 \times 2^{56} = 2^{57}$ security.

Triple DES – 3DES

3DES was developed in 1999 by IBM – by a team led by Walter Tuchman. 3DES prevents a meet-in-the-middle attack. 3DES has a 168-bit key and enciphers blocks of 64 bits. 3DES effectively has 112-bit security.

After decrypting with each key, check for a match with the stored outputs of the 2^{56} possible encryptions. When we have a match, we have located a possibly correct pair of keys. Now, perhaps more than one pair of keys will result in a match, but the number of pairs of keys that return matches should be small. We could try each possible pair of keys. If more than one plaintext/ciphertext correspondence is known (for the key pair), then other correspondences could be used to check which of the keys is correct.

So, it only takes twice as long to break double DES using brute force. Because DES has 56-bit security, double DES has $2 \times 2^{56} = 2^{57}$ security.

Triple DES – 3DES

3DES was developed in 1999 by IBM – by a team led by Walter Tuchman. 3DES prevents a meet-in-the-middle attack. 3DES has a 168-bit key and enciphers blocks of 64 bits. 3DES effectively has 112-bit security.

3DES can be done with 2 or 3 keys.

3-key encryption

$$E\left(k_3,\ D\left(k_2,\ E\left(k_1,\ p\right)\right)\right)$$

2-key encryption

$$E\left(k_1,\ D\left(k_2,\ E\left(k_1,\ p\right)\right)\right)$$

Why would we want to do decryption as the second step? One reason might be that by taking $k_2 = k_1$; 2-key, 3DES becomes single DES with key k_1. 3DES can communicate with single DES.

2-DES (Double DES)

Encryption Process:

Given plaintext P, and keys $K1$ and $K2$:

1. **First Encryption:**

$$C1 = E_{K1}(P)$$

2. **Second Encryption:**

$$C2 = E_{K2}(C1) = E_{K2}(E_{K1}(P))$$

Final ciphertext $C2$.

Decryption Process:

Given ciphertext $C2$, and keys $K1$ and $K2$:

1. **First Decryption:**

$$C1 = D_{K2}(C2)$$

2. **Second Decryption:**

$$P = D_{K1}(C1) = D_{K1}(D_{K2}(C2))$$

Recovered plaintext P.

3-DES (Triple DES)

Encryption Process:

Given plaintext P, and keys $K1$, $K2$, and $K3$:

1. **First Encryption:**

$$C1 = E_{K1}(P)$$

2. **Decryption:**

$$C2 = D_{K2}(C1)$$

3. **Second Encryption:**

$$C3 = E_{K3}(C2) = E_{K3}(D_{K2}(E_{K1}(P)))$$

2. **Second Decryption**:

$$P = D_{K1}(C1) = D_{K1}(D_{K2}(C2))$$

Recovered plaintext P.

3-DES (Triple DES)

Encryption Process:

Given plaintext P, and keys $K1$, $K2$, and $K3$:

1. **First Encryption**:

$$C1 = E_{K1}(P)$$

2. **Decryption**:

$$C2 = D_{K2}(C1)$$

3. **Second Encryption**:

$$C3 = E_{K3}(C2) = E_{K3}(D_{K2}(E_{K1}(P)))$$

Final ciphertext $C3$.

Decryption Process:

Given ciphertext $C3$, and keys $K1$, $K2$, and $K3$:

1. **First Decryption**:

$$C2 = D_{K3}(C3)$$

2. **Encryption**:

$$C1 = E_{K2}(C2)$$

3. **Second Decryption**:

$$P = D_{K1}(C1) = D_{K1}(E_{K2}(D_{K3}(C3)))$$

Recovered plaintext P.

What are the applications of hash functions? Discuss how the SHA-1 algorithm generates a hash value from a given message.

Hash Function

It takes arbitrary input and produces a **fixed size of 160-bit output a** string of characters, called a hash value or hash. It's like creating a unique fingerprint for the data.

Key Properties:

- **Fixed Size Output:** No matter how big the original data is, the hash function always outputs a value with a predetermined length. This is useful for comparisons and storage.
- **One-Way Function:** You can't easily reverse the process to get the original data from the hash value. This is crucial for security applications.
- **Collision Resistant:** It's very unlikely for two different pieces of data to produce the same hash value. This helps ensure the uniqueness of the fingerprint.

Common Hash Functions:

- **SHA-256 (Secure Hash Algorithm 256):** A widely used and secure hash function with a 256-bit output.
- **MD5 (Message-Digest algorithm 5):** An older hash function, no longer considered secure for most applications due to potential vulnerabilities.

A hash function maps a variable-length message into a fixed-length hash value or message digest.

◆ *Virtually all cryptographic hash functions involve the iterative use of a compression function.*

◆ The compression function used in secure hash algorithms falls into one of two categories: *a function specifically designed for the hash function* or *an algorithm based on a symmetric block cipher.* SHA and Whirlpool are examples of these two approaches, respectively.

A **hash function** H accepts a variable-length block of data *M* as input and produces a fixed-size hash value ***h* = H(*M*)** .

A “good” hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are ***evenly distributed and random***.

A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either

(a) a data object that maps to a pre-specified hash result (the one-way property) or

(b) two data objects that map to the same hash result (the collision-free property).

Because of these characteristics, hash functions are often used to determine whether or not data has changed.

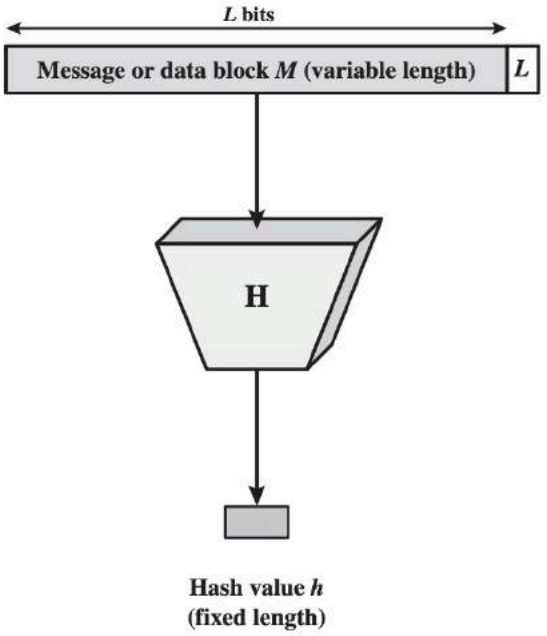


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

This chapter begins with a discussion of the wide variety of applications for cryptographic hash functions. Next, we look at the security requirements for such functions. Then we look at the use of cipher block chaining to implement a cryptographic hash function. The remainder of the chapter is devoted to the most important and widely used family of cryptographic hash functions, the Secure Hash Algorithm (SHA) family.

Application of Hash Functions

1. Message authentication
2. Digital Signature
3. One-way password file
4. Intrusion and virus detection
5. Password Hashing
6. Data Validation

illustrate the concept of security policy and mechanism with an example. Differentiate between block cipher and stream cipher. Explain the process of key expansion in AES.

A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.

◆ Many block ciphers have a Feistel structure. Such a structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one-half of the data being processed, followed by a permutation that interchanges the two halves. The original key is expanded so that a different key is used for each round.

◆ The Data Encryption Standard (DES) has been the most widely used encryption algorithm until recently. It exhibits the classic Feistel structure. DES uses a 64-bit block and a 56-bit key.

◆ Two important methods of cryptanalysis are differential cryptanalysis and linear cryptanalysis. DES has been shown to be highly resistant to these two types of attacks.

Difference Between Block Cipher and Stream Cipher

Block Cipher:

- **Definition**: Encrypts data in fixed-size blocks (e.g., 64 bits, 128 bits).
- **Operation**: Each block is encrypted independently, using the same key.
- **Modes of Operation**: ECB (Electronic Codebook), CBC (Cipher Block Chaining), CFB (Cipher Feedback), OFB (Output Feedback), and CTR (Counter Mode).
- **Example**: AES (Advanced Encryption Standard), DES (Data Encryption Standard).

Stream Cipher:

- **Definition**: Encrypts data as a stream of bits or bytes.
- **Operation**: Generates a keystream (a sequence of bits), which is XORed with the plaintext to produce ciphertext.
- **Application**: Suitable for real-time applications where data size is unknown or continuous.
- **Example**: RC4, Salsa20.

Feature	Block Cipher	Stream Cipher
Data Unit	Fixed-size blocks	Individual bits
Encryption Process	Independent blocks	Continuous stream
Error Propagation	Isolated errors	Propagates
Mode of Operation	Modes (CBC, CTR)	Modes (OFB, CFB)
Complexity	Simpler design	More complex
Speed	Slower	Faster
Memory Usage	More memory	Less memory
Key Size	Longer keys	Shorter keys (with keystream)
Suitability	Large files	Real-time streams
Key Expansion	Key expansion (AES)	No key expansion

Hash Function

In computer science, "hashing" is an overloaded term. In cryptography, hashing has a very precise meaning, so for the time being, it would be best to forget about any other concepts of hashing that may be clouding your mind.

A cryptographic hash function h(x) must provide all of the following.

- **Compression** — For any size input x, the output length of y = h(x) is small. In practice, the output is a fixed size (e.g., 160 bits), regardless of the length of the input.

- **Efficiency** — It must be easy to compute h(x) for any input x. The computational effort required to compute h{x) will, of course, grow with the length of x, but it cannot grow too fast.

- **One-way** — Given any value y, it's computationally infeasible to find a value a; such that h{x) = y. Another way to say this is that there is no feasible way to invert the hash.

- **Weak collision resistance** — Given x and h(x), it's infeasible to find any y, with y φ x, such that h(y) = h(x). Another way to state this requirement is that it is not feasible to modify a message without changing its hash value.

- **Strong collision resistance** — It's infeasible to find any x and y, such that x φ y and h(x) = h(y). That is, we cannot find any two inputs that hash to the same output.

  Many collisions must exist since the input space is much larger than the output space. For example, suppose a particular hash function generates a 128-bit output. If we consider, say, all possible 150-bit input values then, on average, 222 (that is, more than 4,000,000) of these input values hash to each possible output value. The collision resistance properties says that all of these collisions are computationally hard to find. This is asking a lot, and it might seem that, as a practical matter, no such function could possibly exist. Remarkably, practical cryptographic hash functions do indeed exist.

Section B

Attempt any eight questions.

Show encryption and decryption of "csit" using Hill cipher having key, k =

3 2

5 7

Hill Cipher

- Multi-letter cipher
- Encrypts a group of letters: digraph, trigraph or polygraph

Review from linear algebra concepts

- Determinant
- Matrix arithmetic modulo 26
- Square matrix
- Multiplicative inverse

The Hill Algorithm

This can be expressed as

$C = E(K,P) = P \times K \bmod 26$

$P = D(K,C) = C\ K^{-1} \bmod 26 = P \times K \times K^{-1} \bmod 26$

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

Encryption

$$C_1 = (P_1K_{11} + P_2K_{21} + P_3K_{31}) \bmod 26$$
$$C_2 = (P_1K_{12} + P_2K_{22} + P_3K_{32}) \bmod 26$$
$$C_3 = (P_1K_{13} + P_2K_{23} + P_3K_{33}) \bmod 26$$

Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Solution:

p	a	y	m	o	r	e	m	o	n	e	y
15	0	24	12	14	17	4	12	14	13	4	24

Key = 3 x 3 matrix.

PT = pay mor emo ney

Hill Cipher Example

Encrypting: pay

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

$$(C_1\ C_2\ C_3) = (15\ 0\ 24) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (15\text{x}17+0\text{x}21+24\text{x}2 \quad 15\text{x}17+0\text{x}18+24\text{x}2 \quad 15\text{x}5+0\text{x}21+24\text{x}19) \bmod 26$$

$$= (303\ 303\ 531) \bmod 26$$

$$= (17\ 17\ 11)$$

$$= (R\ R\ L)$$

Hill Cipher Example

Encrypting: mor

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

$$(C_1\ C_2\ C_3) = (12\ 14\ 17) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (12\text{x}17+14\text{x}21+17\text{x}2 \quad 12\text{x}17+14\text{x}18+17\text{x}2 \quad 12\text{x}5+14\text{x}21+17\text{x}19) \bmod 26$$

$$= (532\ 490\ 677) \bmod 26$$

$$= (12\ 22\ 1)$$

$$= (M\ W\ B)$$

Hill Cipher Example

Encrypting: emo

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

$$(C_1\ C_2\ C_3) = (4\ 12\ 14) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (4\text{x}17+12\text{x}21+14\text{x}2 \quad 4\text{x}17+12\text{x}18+14\text{x}2 \quad 4\text{x}5+12\text{x}21+14\text{x}19) \bmod 26$$

$$= (348\ 312\ 538) \bmod 26$$

$$= (10\ 0\ 18)$$

$$= (K\ A\ S)$$

Hill Cipher Example

Encrypting: ney

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

$$(C_1\ C_2\ C_3) = (13\ 4\ 24) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= (13\times17+4\times21+24\times2 \quad 13\times17+4\times18+24\times2 \quad 13\times5+4\times21+24\times19) \bmod 26$$

$$= (348\ 312\ 538) \bmod 26$$

$$= (15\ 3\ 7)$$

$$= (P\ D\ H)$$

Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Plaintext : pay more money

Ciphertext : RRLMWBKASPDH

DECRYPT THE HILL CIPHER

The Hill Algorithm

Decryption requires K^{-1}, the inverse matrix K.

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

The Hill Algorithm

To find the determinant of K: $\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$

$$\text{Det} \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$= 17(18 \times 19 - 2 \times 21) - 17(19 \times 21 - 2 \times 21) + 5(2 \times 21 - 2 \times 18) \bmod 26$$
$$= 17(342 - 42) - 17(399 - 42) + 5(42 - 36) \bmod 26$$
$$= 17(300) - 17(357) + 5(6) \bmod 26$$
$$= 5100 - 6069 + 30 \bmod 26$$
$$= -939 \bmod 26$$
$$= -3 \bmod 26$$
$$= 23$$

The Hill Algorithm

Decryption requires K^{-1}, the inverse matrix K.

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

To find Det K, Adj K

The Hill Algorithm

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adjoint } K$$

To find Adjoint K

$$\text{Adj } K = \begin{vmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{vmatrix}$$

$$\text{Adj } K = \begin{vmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{vmatrix}$$

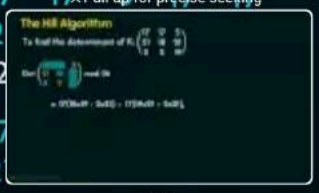
$$\text{Adj } K = \begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \\ 2 & 2 & 19 & 2 & 2 \end{matrix}$$

The Hill Algorithm

$$\text{Adj K} = \begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \\ 2 & 2 & 19 & 2 & 2 \end{matrix}$$

$$\text{Adj K} = \begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \\ 2 & 2 & 19 & 2 & 2 \end{matrix}$$

$$\text{Adj K} = \begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 2 & & & & \\ 17 & & & & \\ 2 & & & & \end{matrix}$$



The Hill Algorithm

$$\text{Adj K} = \begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \\ 2 & 2 & 19 & 2 & 2 \end{matrix}$$

$$\begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \end{matrix}$$

$$\text{Adj K} = \begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \\ 2 & 2 & 19 & 2 & 2 \end{matrix}$$

$$\begin{matrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \end{matrix}$$

The Hill Algorithm

$$= \begin{matrix} 18 & 21 & 21 & 18 \\ 2 & 19 & 2 & 2 \\ 17 & 5 & 17 & 17 \\ 18 & 21 & 21 & 18 \end{matrix}$$

Performing the operation - Column wise
Entering the matrix - Row wise

$$= \begin{matrix} 18\times19-2\times21 & 2\times5-17\times19 & 17\times21-18\times5 \\ 21\times2-19\times21 & 19\times17-5\times2 & 5\times21-21\times17 \\ 21\times2-2\times18 & 2\times17-17\times2 & 17\times18-21\times17 \end{matrix}$$

$$= \begin{matrix} 14 & -1 & 7 \\ -19 & 1 & -18 \\ 6 & 0 & -25 \end{matrix} \bmod 26$$

The Hill Algorithm

$$= \begin{matrix} 18 & 21 & 21 & 18 \\ 2 & 19 & 2 & 2 \\ 17 & 5 & 17 & 17 \\ 18 & 21 & 21 & 18 \end{matrix}$$

Performing the operation - Column wise
Entering the matrix - Row wise

$$= \begin{matrix} 18\times19-2\times21 & 2\times5-17\times19 & 17\times21-18\times5 \\ 21\times2-19\times21 & 19\times17-5\times2 & 5\times21-21\times17 \\ 21\times2-2\times18 & 2\times17-17\times2 & 17\times18-21\times17 \end{matrix}$$

$$= \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

The Hill Algorithm

Decryption requires K^{-1}, the inverse matrix K.

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

$$K^{-1} = \frac{1}{23} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

$$K^{-1} = 23^{-1} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

The Hill Algorithm

Decryption requires K^{-1}, the inverse matrix K.

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

$$K^{-1} = \frac{1}{23} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

$$K^{-1} = \boxed{17} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

The Hill Algorithm

Decryption requires K^{-1}, the inverse matrix K.

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

$$K^{-1} = \frac{1}{23} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

$$K^{-1} = \boxed{23^{-1}} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

$23^{-1} \times 23 = 1 \bmod 26$	$9 \times 23 = 25 \bmod 26$
$1 \times 23 = 23 \bmod 26$	$10 \times 23 = 22 \bmod 26$
$2 \times 23 = 20 \bmod 26$	$11 \times 23 = 19 \bmod 26$
$3 \times 23 = 17 \bmod 26$	$12 \times 23 = 16 \bmod 26$
$4 \times 23 = 14 \bmod 26$	$13 \times 23 = 13 \bmod 26$
$5 \times 23 = 11 \bmod 26$	$14 \times 23 = 10 \bmod 26$
$6 \times 23 = 8 \bmod 26$	$15 \times 23 = 7 \bmod 26$
$7 \times 23 = 5 \bmod 26$	$16 \times 23 = 4 \bmod 26$
$8 \times 23 = 2 \bmod 26$	$17 \times 23 = 1 \bmod 26$

NESO ACADEMY

The Hill Algorithm

$$K^{-1} = 17 \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

$$K^{-1} = \begin{pmatrix} 238 & 425 & 119 \\ 119 & 17 & 136 \\ 102 & 0 & 17 \end{pmatrix} \bmod 26$$

$$K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

Pull up for precise seeking

15:32

The Hill Algorithm

$$K \times K^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \quad K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Plaintext : pay more money

Ciphertext : RRLMWBKASPDH

Hill Cipher Example

Question: Decrypt "RRLMWBKASPDH" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Solution:

$P = C\ K^{-1} \bmod 26$

R	R	L	M	W	B	K	A	S	P	D	H
17	17	11	12	22	1	10	0	18	15	3	7

Hill Cipher Example

Decrypting: RRL

$$(P_1\ P_2\ P_3) = (R\ R\ L) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$ ← Decryption

$$(C_1\ C_2\ C_3) = (17\ 17\ 14) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$

$= (17x4+17x15+11x24 \quad 17x9+17x17+11x0 \quad 17x15+17x6+11x17) \bmod 26$

$= (587\ 442\ 544) \bmod 26$

$= (15\ 0\ 24)$

$= (P\ A\ Y)$

The Hill Algorithm

$$K^{-1} = 17 \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \bmod 26$$

$$K^{-1} = \begin{pmatrix} 238 & 425 & 119 \\ 119 & 17 & 136 \\ 102 & 0 & 17 \end{pmatrix} \bmod 26$$

$$K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

Pull up for precise seeking

15:32

The Hill Algorithm

$$K \times K^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \quad K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Plaintext : pay more money

Ciphertext : RRLMWBKASPDH

Hill Cipher Example

Question: Decrypt "RRLMWBKASPDH" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Solution:

$P = C\ K^{-1} \bmod 26$

R	R	L	M	W	B	K	A	S	P	D	H
17	17	11	12	22	1	10	0	18	15	3	7

Hill Cipher Example

Decrypting: RRL

$$(P_1\ P_2\ P_3) = (R\ R\ L) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$ ← Decryption

$$(C_1\ C_2\ C_3) = (17\ 17\ 14) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$

$= (17x4+17x15+11x24 \quad 17x9+17x17+11x0 \quad 17x15+17x6+11x17) \bmod 26$

$= (587\ 442\ 544) \bmod 26$

$= (15\ 0\ 24)$

$= (P\ A\ Y)$

Hill Cipher Example

CT	R	R	L	M	W	B	K	A	S	P	D	H
PT	p	a	y									

Hill Cipher Example

Decrypting: MWB

$$(P_1\ P_2\ P_3) = (M\ W\ B) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$

$$(C_1\ C_2\ C_3) = (12\ 22\ 1) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$

$$= (12x4+22x15+1x24 \quad 12x9+22x17+1x0 \quad 12x15+22x6+1x17) \bmod 26$$

$$= (402\ 482\ 329) \bmod 26$$

$$= (12\ 14\ 17)$$

$$= (M\ O\ R)$$

Hill Cipher Example

Decrypting: KAS

$$(P_1\ P_2\ P_3) = (K\ A\ S) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$

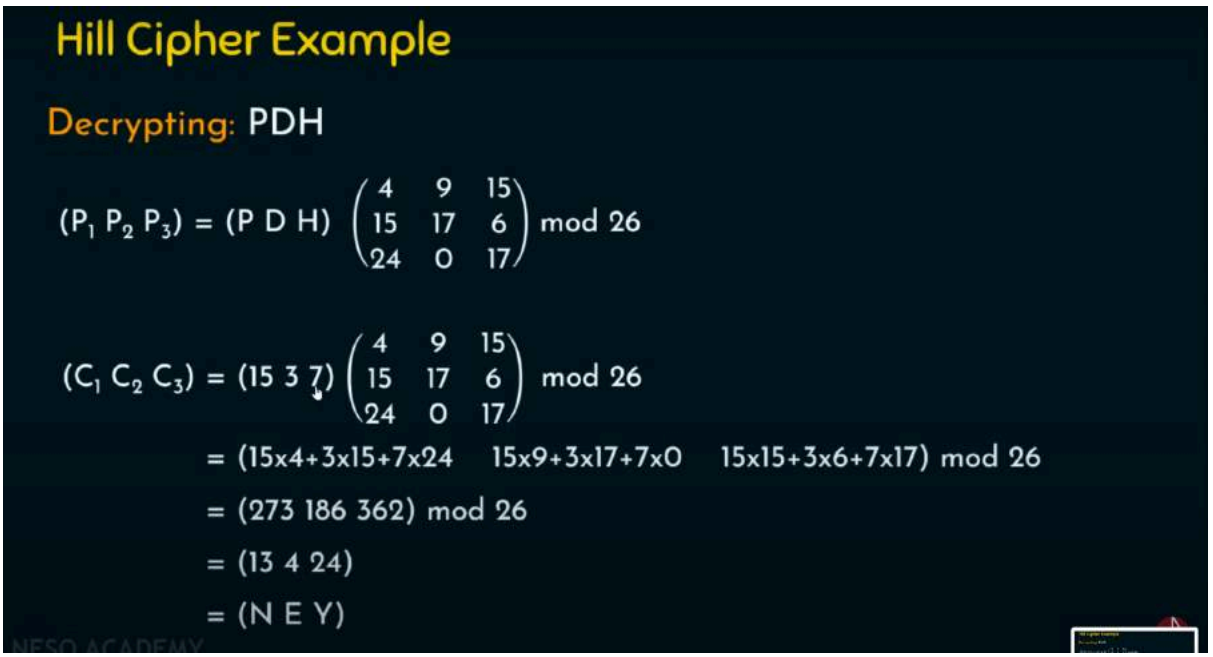
$$(C_1\ C_2\ C_3) = (10\ 0\ 18) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \bmod 26$$

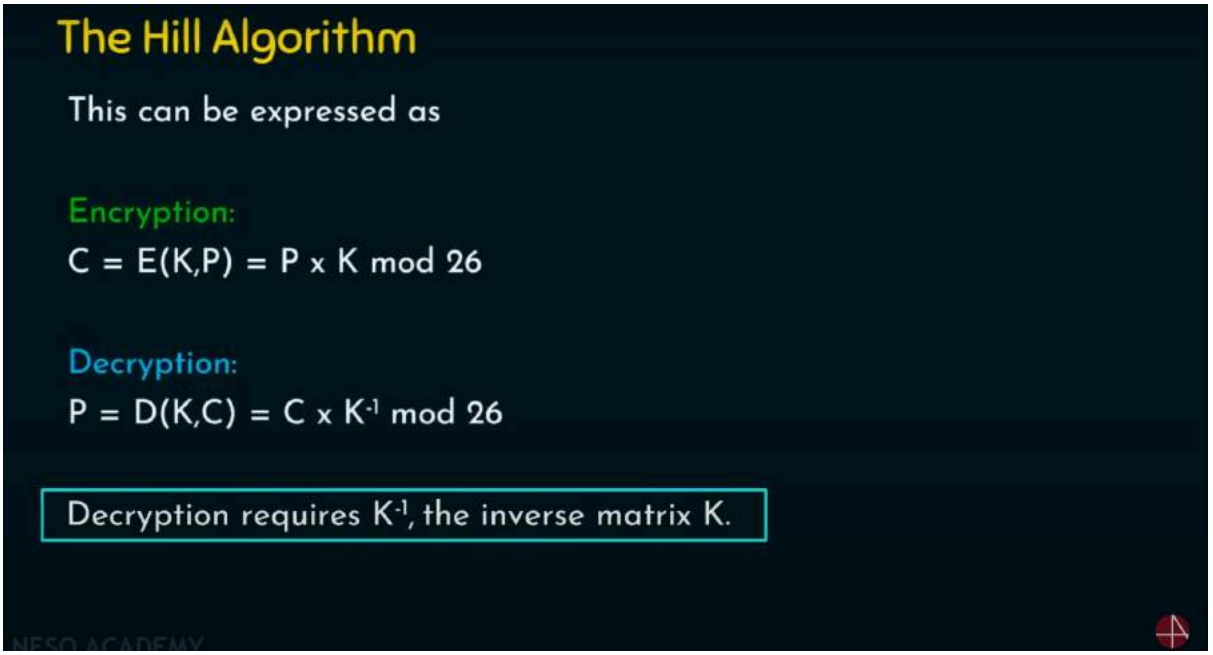
$$= (10\text{x}4+0\text{x}15+18\text{x}24 \quad 10\text{x}9+0\text{x}17+18\text{x}0 \quad 10\text{x}15+0\text{x}6+18\text{x}17) \bmod 26$$

$$= (472\ 90\ 456) \bmod 26$$

$$= (4\ 12\ 14)$$

$$= (E\ M\ O)$$





What is mutual authentication? How does mutual authentication work? Describe one-way and mutual authentication systems.

Mutual authentication is when two sides of a communications channel verify each other's identity, instead of only one side verifying the other. Mutual authentication is also known as "two-way authentication" because the process goes in both directions.

When someone uses a rideshare app, they usually check the license plate or the description of the vehicle to make sure they are getting into the right car. Once they get in, the driver asks the passenger for their name to confirm they are picking up the right person. The

passenger and driver each check that they are interacting with the intended person — so that the driver is providing the correct service, the passenger is in a car going to their destination, and both can confirm they are with someone who has been verified by the rideshare app.

Similarly, mutual authentication verifies both parties in a digital communications channel. For example, a client and a server using mutual authentication take steps to independently verify each other's identity, instead of only the client authenticating the server. Device-to-device connections, like those between Internet of Things (IoT) devices, often use mutual authentication as well.

Mutual authentication is most commonly associated with the Transport Layer Security (TLS) protocol, but it can be used by other protocols and in other contexts too. Learn about mutual TLS.

How does mutual authentication work?

There are three main methods for mutually authenticating the ends of a communications channel:

1. **Public key authentication:** This method relies on public key cryptography. A key is a string of data that can be used to encrypt or digitally sign data. Public key cryptography uses two keys — a public key and a private key. Data encrypted with the public key is decrypted with the private key.

In public key mutual authentication, both sides of the communication advertise a public key, and both have to prove they possess the private key that accompanies their public key — like someone showing a government-issued ID card to verify their name. Each side sends a digital signature to the other side. If the signature can be verified with the public key, then the correct private key was used, and the party that sent the signature is legitimate.

2. **Certificate authentication:** This approach is similar to public key authentication, except instead of just a public key, both parties have a public key certificate. The certificate contains additional information that helps verify the parties' identities — including who issued the certificate and public key, whom the certificate applies to, when the certificate expires, and so on. TLS certificates can be used for this type of mutual authentication if both sides have one.

3. **Username and password:** Despite the name, this method of mutual authentication still uses a certificate on the server side. The server presents a certificate to the client, which verifies the certificate. On the client side, it is just like typical username/password authentication: the client sends its username and password combination to the server, which verifies the credentials.

One-way and mutual authentication systems are essential concepts in security protocols, particularly in the context of secure communications.

One-way Authentication

In a one-way authentication system, only one party is authenticated by the other. This means that one entity verifies the identity of another without requiring the second party to prove its identity back.

Characteristics:

- Single Direction: Typically involves a client authenticating to a server. For example, when a user logs into a website, the server verifies the user's credentials (like a username and password) but does not require the user to verify the server's identity.
- Use Cases: Commonly used in scenarios where the client needs to access a service without needing to confirm the server's identity, such as in simple web transactions or when accessing public resources.
- Security Risks: Since the server's identity is not verified, users may be vulnerable to man-in-the-middle attacks, where an attacker impersonates the server.

Mutual Authentication

Mutual authentication, also known as two-way authentication, involves both parties verifying each other's identities. This ensures that both the client and the server are who they claim to be.

Characteristics:

- Bidirectional: Both parties authenticate each other. For instance, in a secure SSL/TLS connection, the client verifies the server's certificate while the server also verifies the client's identity.
- Use Cases: This is crucial in sensitive transactions, such as online banking or secure corporate communications, where both parties need assurance of each other's identities.
- Enhanced Security: By requiring both sides to authenticate, mutual authentication significantly reduces the risk of impersonation and man-in-the-middle attacks.

In summary, one-way authentication is simpler and often sufficient for less sensitive applications, while mutual authentication provides a higher level of security suitable for critical transactions.

One-Way Authentication

Definition: One-way authentication is a security process where only one party (typically the server) verifies the identity of the other party (typically the client). This ensures that the client is communicating with a legitimate server.

Process:

1. **Client Request**: The client initiates a connection to the server.
2. **Server Identity Verification**: The server presents its credentials, usually in the form of a digital certificate.
3. **Client Validation**: The client verifies the server's credentials using a trusted certificate authority (CA). If the verification is successful, the client proceeds with the communication.

Example:

- **HTTPS**: When you access a website, your browser verifies the website’s SSL/TLS certificate to ensure it’s legitimate before establishing a secure connection.

Pros:

- Simplifies the authentication process.
- Efficient for scenarios where only the server needs to be verified.

Cons:

- Does not verify the identity of the client, which could allow unauthorized clients to access the server if no additional security measures are in place.

Mutual Authentication

Definition: Mutual authentication, also known as two-way authentication, is a security process where both parties (client and server) verify each other's identity. This ensures that both the client and the server are legitimate and trusted.

Process:

1. **Client Request**: The client initiates a connection to the server.
2. **Server Identity Verification**: The server presents its credentials, usually in the form of a digital certificate.
3. **Client Validation**: The client verifies the server's credentials using a trusted certificate authority (CA).
4. **Client Identity Verification**: The server requests the client's credentials.
5. **Server Validation**: The server verifies the client's credentials. If both verifications are successful, the communication proceeds.

Example:

- **Mutual TLS (mTLS)**: Both the client and the server present and verify each other's certificates to establish a secure and authenticated connection.

Pros:

- Provides higher security by ensuring both parties are authenticated.
- Reduces the risk of man-in-the-middle attacks and unauthorized access.

Cons:

- More complex and requires additional configuration and management of client certificates.
- Can be more resource-intensive and time-consuming compared to one-way authentication.

Use Cases

- **One-Way Authentication**: Commonly used in web browsing, online shopping, and scenarios where the server needs to be trusted, but the client is not explicitly verified.
- **Mutual Authentication**: Used in sensitive communications, such as financial transactions, corporate networks, and secure communications between microservices within a distributed system.

Both authentication methods are crucial for ensuring secure communications, but the choice between them depends on the specific security requirements and context of the application.

One-Way and Mutual Authentication Systems

One-Way Authentication

In a one-way authentication system, only one party verifies the identity of the other. This is the most common type of authentication we encounter daily.

Example:

- Logging into a website: You (the user) prove your identity to the website by providing a username and password. The website doesn't need to prove its identity to you.

Key characteristics:

- Asymmetric relationship: One party verifies the other, but not vice versa.
- Lower security compared to mutual authentication.
- Simpler to implement.

Mutual Authentication

Mutual authentication involves both parties verifying each other's identity before proceeding with a transaction or communication. This provides a higher level of security.

Example:

- Secure communication between two servers: Both servers must prove their identities to each other before exchanging sensitive data.

- Online banking: The bank verifies the user, and the user verifies the bank's identity using a digital certificate.

Key characteristics:

- Symmetric relationship: Both parties verify each other.
- Higher security as it prevents man-in-the-middle attacks.
- More complex to implement.

Comparison Table:

Feature	One-Way Authentication	Mutual Authentication
Parties involved	One verifies the other	Both parties verify each other
Security level	Lower	Higher
Complexity	Simpler	More complex
Common use cases	Website login, ATM transactions	Secure communication, online banking

Export to Sheets

In essence:

- One-way authentication is like showing your ID to enter a building.
- Mutual authentication is like two people showing their IDs to each other before entering a secure room.

By understanding the differences between one-way and mutual authentication, you can better appreciate the security implications of various systems and choose the appropriate level of authentication for your specific needs.

List the stage of the certificate life cycle. What are the types of firewalls?

The stages of a certificate lifecycle can vary slightly depending on the source, but here are some common stages:

1. Generation (or Enrollment):

- In this stage, a certificate request is initiated. This typically involves proving control over a domain name or server by a user or administrator.

2. Issuance:

- A trusted Certificate Authority (CA) validates the request and, if successful, issues the digital certificate.

3. Distribution and Installation:

- The issued certificate needs to be securely delivered to the server or device where it will be used. This may involve manual installation or automated deployment tools.

4. Management and Monitoring:

- Throughout its validity period, the certificate needs to be monitored for potential issues like expiration or compromise. This stage may also involve tasks like certificate rotation or backup.

5. Renewal or Revocation:

- Certificates have a limited lifespan and need to be renewed before expiration to maintain security. Alternatively, a certificate can be revoked if it's compromised or no longer needed.

6. (Optional) Destruction:

- After a certificate expires or is revoked, it's important to securely destroy any copies to prevent misuse.

Firewalls can be categorized in various ways, depending on the aspect being considered. Here's a breakdown of two common classification methods:

By Deployment and Form Factor:

- **Network Firewall:** Installed on a dedicated hardware device that sits between a network and the internet, filtering traffic flow. (Often found in broadband routers)
- **Host-based Firewall:** Software program installed directly on a computer, monitoring and controlling incoming and outgoing traffic for that specific device.

By Inspection Method:

- **Packet-Filtering Firewall:** Basic type that inspects individual data packets based on source and destination IP addresses, port numbers, and protocols. It allows or blocks packets based on predefined rules.
- **Stateful Inspection Firewall:** More sophisticated, examining ongoing connections and data packets. It analyzes factors like packet sequence and content to identify and block suspicious activity.
- **Application-Level Gateway (Proxy Firewall):** Acts as an intermediary between a device and the internet. It inspects traffic at the application layer, understanding the functionality of specific applications and filtering based on security policies.
- **Next-Generation Firewall (NGFW):** Combines traditional firewall functionalities with advanced features like deep packet inspection, intrusion detection/prevention, and application control. It offers a more comprehensive security solution.

There are other firewall types as well, including:

- **Circuit-Level Gateway:** Manages connections between devices, focusing on TCP and UDP protocols.
- **Cloud Firewall:** Software-based firewall deployed within a cloud environment to secure cloud resources.

The choice of firewall type depends on your specific security needs and network architecture.

What is malicious logic? How are zombies different from Trojan horses?

Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.

◆ A virus is a piece of software that can “infect” other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

◆ A worm is a program that can replicate itself and send copies from a computer to a computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. In addition to propagation, the worm usually performs some unwanted function.

◆ A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service.

◆ A distributed denial of service attack is launched from multiple coordinated sources.

Name	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Logic bomb	A program inserted into software by an intruder. A logic bomb lies dormant until a predefined condition is met; the program then triggers an unauthorized act.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality.
Mobile code	Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Kit (virus generator)	Set of tools for generating new viruses automatically.
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

How is the Miller-Rabin test used for primality testing? Show whether the number 561 passes the test.

The Miller-Rabin test is a probabilistic algorithm used to determine whether a given number is prime. It is based on properties of modular arithmetic and can efficiently identify composite numbers. Here's a step-by-step outline of how the Miller-Rabin test works and its application to the number 561:

Miller-Rabin Test Procedure

1. **Representation**: Express $n-1$ as $2^s \cdot d$ where d is odd.
   - $n-1 = 2^s \cdot d$
   - Find s and d such that $n-1$ can be written in this form.
2. **Witness Loop**: Randomly select a base a where $2 \leq a \leq n-2$ and perform the following:
   - Compute $x = a^d \mod n$.
   - If $x = 1$ or $x = n-1$, then a is a witness for the primality of n and the test passes for this a.
   - Otherwise, repeat the squaring step $s-1$ times:
     - Compute $x = x^2 \mod n$.
     - If $x = n-1$, then a is a witness and the test passes for this a.
     - If $x = 1$, then n is composite.
     - If none of the above conditions are met, n is composite.

Example: Testing 561

1. **Representation**:
   - $n = 561$
   - $561 - 1 = 560$
   - Write 560 as $2^s \cdot d$:
     - $560 = 2^4 \cdot 35$
     - Thus, $s = 4$ and $d = 35$.
2. **Witness Loop**:
   - Select a base a. Let's try $a = 2$.

Example: Testing 561

1. **Representation**:
   - $n = 561$
   - $561 - 1 = 560$
   - Write 560 as $2^s \cdot d$:
     - $560 = 2^4 \cdot 35$
     - Thus, $s = 4$ and $d = 35$.
2. **Witness Loop**:
   - Select a base a. Let's try $a = 2$.
     - Compute $x = 2^{35} \mod 561$:
       - $x = 263$
     - Since $x \neq 1$ and $x \neq 560$, proceed with the squaring step:
       - Compute $x = x^2 \mod 561 = 263^2 \mod 561$:
         - $x = 166$
       - Compute $x = x^2 \mod 561 = 166^2 \mod 561$:
         - $x = 67$
       - Compute $x = x^2 \mod 561 = 67^2 \mod 561$:
         - $x = 1$
   - Since we reached $x = 1$ without previously reaching $x = 560$, 561 is composite.

Thus, 561 does not pass the Miller-Rabin test for $a = 2$ and is identified as a composite number.

In fact, 561 is known as a Carmichael number, which means it can fool Fermat's primality test for some bases, but the Miller-Rabin test correctly identifies it as composite.

Answer

The Miller-Rabin test is a probabilistic algorithm used for primality testing, particularly effective for large numbers. It builds on Fermat's Little Theorem and is designed to reduce the likelihood of incorrectly identifying composite numbers as primes.

Miller-Rabin Test Overview

Algorithm Steps

1. **Input**: A number n (to be tested for primality) and a parameter k (the number of iterations).
2. **Base Cases**:
   - If $n \leq 1$, return false.
   - If $n \leq 3$, return true (2 and 3 are prime).
   - If n is even, return false.
3. **Express $n-1$**: Write $n-1$ as $d \times 2^r$, where d is odd and $r \geq 1$.
4. **Witness Loop**: For k iterations:
   - Choose a random integer a in the range $[2, n-2]$.
   - Compute $x = a^d \mod n$.
   - If x is 1 or $n-1$, continue to the next iteration.
   - Square x up to $r-1$ times:
     - If x becomes 1, return false (composite).
     - If x becomes $n-1$, continue to the next iteration.
   - If none of the conditions are satisfied, return false (composite).
5. **Return**: If all iterations pass, return true (probably prime).

The accuracy of the test increases with more iterations. Each iteration reduces the probability that a composite number is falsely identified as prime.

Testing the Number 561

To determine whether 561 is prime using the Miller-Rabin test, we follow the algorithm:

1. **Initial Check**:
   - 561 is odd and greater than 3, so we proceed.
2. **Express $561-1$**:
   - $561 - 1 = 560$.

Testing the Number 561

To determine whether 561 is prime using the Miller-Rabin test, we follow the algorithm:

1. **Initial Check**:
   - 561 is odd and greater than 3, so we proceed.
2. **Express $561 - 1$**:
   - $561 - 1 = 560$.
   - Factor 560 as $560 = 2^4 \times 35$; thus, $d = 35$ and $r = 4$.
3. **Perform the Test**:
   - For simplicity, we will use $k = 3$ iterations and choose a values randomly.

   Iteration 1:
   - Let $a = 2$:
     - Compute $x = 2^{35} \mod 561$.
     - Calculate x:
       - $2^{35} \mod 561$ yields $x \neq 1$ and $x \neq 560$.
     - Square x multiple times:
       - If x becomes 1 or 560, continue; otherwise, return false.

   Iteration 2:
   - Let $a = 3$:
     - Compute $x = 3^{35} \mod 561$.
     - Again, check the conditions.

   Iteration 3:
   - Let $a = 5$:
     - Compute $x = 5^{35} \mod 561$.
     - Check conditions.

After performing these iterations, it is known that 561 is a Carmichael number, meaning it will pass the Miller-Rabin test for many bases a despite being composite. Specifically, it is a Fermat pseudoprime to several bases, including 2, 3, and 5, indicating it will likely pass the test.

In conclusion, 561 does not pass the Miller-Rabin test as a prime number; it is confirmed to be composite despite potentially passing some iterations of the test due to its properties as a Carmichael number [2] [4].

Show drafts

The Miller-Rabin Primality Test

The Miller-Rabin test is a probabilistic primality test, meaning it doesn't definitively prove a number is prime, but it can provide a very high probability of primality. It's based on the properties of prime numbers and their relationship to modular arithmetic.

Steps involved:

1. **Express n-1 as 2^s * d:**
   - Where n is the number to be tested, s is a non-negative integer, and d is an odd integer.
2. **Choose a random base a:**
   - This is a number between 2 and n-2.
3. **Compute x = a^d mod n:**
4. **If x is 1 or n-1, the test is inconclusive.**
5. **Repeat the following s-1 times:**
   - Compute x = x^2 mod n
   - If x is 1, the number is composite.
   - If x is n-1, the test is inconclusive.
6. **If we reach this point without finding x to be 1 or n-1, the number is composite.**

composite.

If the test passes for multiple randomly chosen bases, the probability of the number being composite becomes very small.

Testing 561

Let's test if 561 is prime using the Miller-Rabin test.

1. **Express n-1 as 2^s * d:**
   - 561 - 1 = 560 = 2^4 * 35
   - So, s = 4 and d = 35
2. **Choose a random base a:**
   - Let's choose a = 2
3. **Compute x = a^d mod n:**
   - x = 2^35 mod 561 = 263
4. **Iterate s-1 times:**
   - x = x^2 mod 561 = 263^2 mod 561 = 1

Since x became 1 before the loop ended, 561 is **composite**.

Note: Even though we only tested one base, we've already proven that 561 is composite. The Miller-Rabin test becomes more reliable as the number of bases tested increases.

Show that the set of integers is a Ring under addition and multiplication.

Rings

A ring R denoted by {R, +, *}, is a set of elements with two binary operations, called addition and multiplication, such that for all a, b, c ∈ R the following axioms are obeyed:

- Group (A1-A4), Abelian Group(A5).
- Closure under multiplication (M1): If a, b ∈ R then ab ∈ R
- Associativity of multiplication (M2): a (bc) = (ab) c for all a, b, c ∈ R
- Distributive laws (M3) :

  a (b + c) = ab + ac for all a, b, c ∈ R

  (a + b) c = ac + bc for all a, b, c ∈ R

Note:

Subtraction [a - b = a + (-b)]

Let's illustrate that $\mathbb{Z}$ (the set of integers) is a ring under addition and multiplication using specific numbers as examples.

1. $\mathbb{Z}$ under addition is an abelian group

- **Closure:**
  - Example: Let $a = 3$ and $b = 4$. Then $a + b = 3 + 4 = 7$, which is also an integer.
- **Associativity:**
  - Example: Let $a = 1$, $b = 2$, and $c = 3$. Then $(a + b) + c = (1 + 2) + 3 = 3 + 3 = 6$ and $a + (b + c) = 1 + (2 + 3) = 1 + 5 = 6$. Thus, $(a + b) + c = a + (b + c)$.
- **Identity element:**
  - Example: Let $a = -5$. Then $a + 0 = -5 + 0 = -5$ and $0 + a = 0 + (-5) = -5$. Thus, 0 is the additive identity.
- **Inverse element:**
  - Example: Let $a = 6$. The additive inverse of a is $-a = -6$. Then $a + (-a) = 6 + (-6) = 0$.
- **Commutativity:**
  - Example: Let $a = 2$ and $b = -3$. Then $a + b = 2 + (-3) = -1$ and $b + a = -3 + 2 = -1$. Thus, $a + b = b + a$.

2. $\mathbb{Z}$ under multiplication is a monoid

- **Closure:**
  - Example: Let $a = -2$ and $b = 5$. Then $a \cdot b = -2 \cdot 5 = -10$, which is also an integer.
- **Associativity:**
  - Example: Let $a = 2$, $b = 3$, and $c = 4$. Then $(a \cdot b) \cdot c = (2 \cdot 3) \cdot 4 = 6 \cdot 4 = 24$ and $a \cdot (b \cdot c) = 2 \cdot (3 \cdot 4) = 2 \cdot 12 = 24$. Thus, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Identity element:**

$2 = -1$. Thus, $a + b = b + a$.

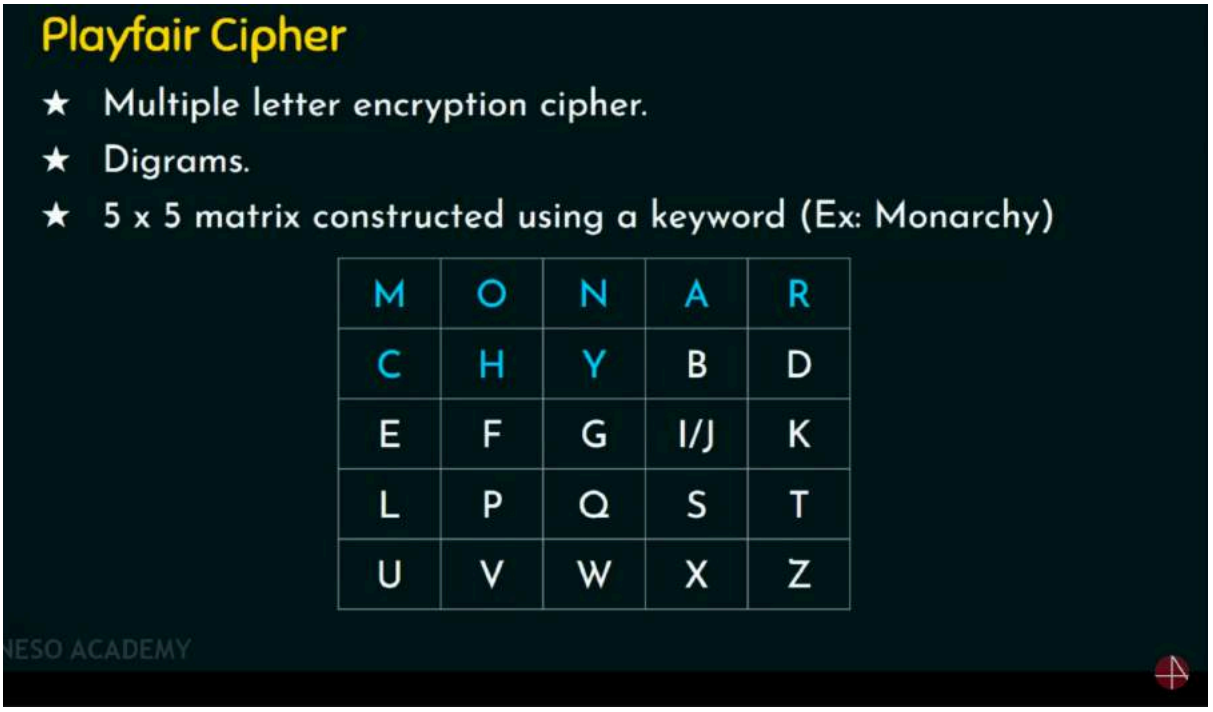
2. $\mathbb{Z}$ under multiplication is a monoid

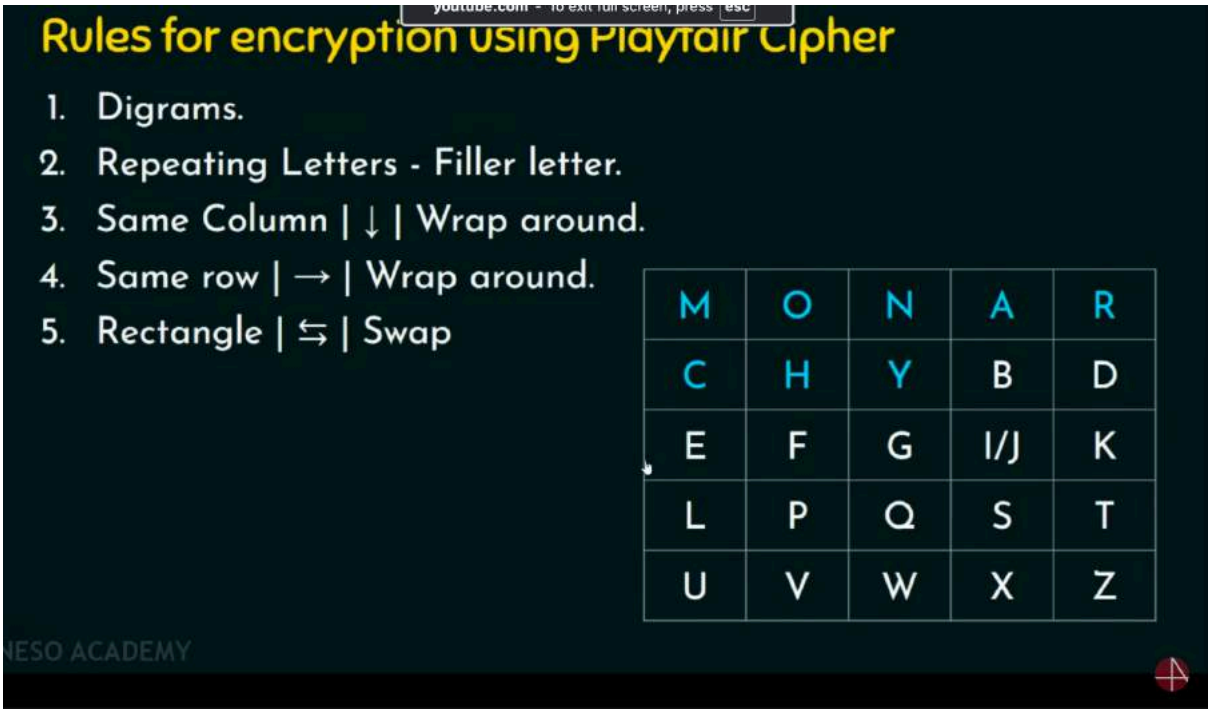
- **Closure:**
  - Example: Let $a = -2$ and $b = 5$. Then $a \cdot b = -2 \cdot 5 = -10$, which is also an integer.
- **Associativity:**
  - Example: Let $a = 2$, $b = 3$, and $c = 4$. Then $(a \cdot b) \cdot c = (2 \cdot 3) \cdot 4 = 6 \cdot 4 = 24$ and $a \cdot (b \cdot c) = 2 \cdot (3 \cdot 4) = 2 \cdot 12 = 24$. Thus, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Identity element:**
  - Example: Let $a = 7$. Then $a \cdot 1 = 7 \cdot 1 = 7$ and $1 \cdot a = 1 \cdot 7 = 7$. Thus, 1 is the multiplicative identity.

3. Distributive laws

- **Left distributive:**
  - Example: Let $a = 2$, $b = 3$, and $c = 4$. Then $a \cdot (b + c) = 2 \cdot (3 + 4) = 2 \cdot 7 = 14$ and $(a \cdot b) + (a \cdot c) = (2 \cdot 3) + (2 \cdot 4) = 6 + 8 = 14$. Thus, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.
- **Right distributive:**
  - Example: Let $a = 3$, $b = 2$, and $c = 5$. Then $(a + b) \cdot c = (3 + 2) \cdot 5 = 5 \cdot 5 = 25$ and $(a \cdot c) + (b \cdot c) = (3 \cdot 5) + (2 \cdot 5) = 15 + 10 = 25$. Thus, $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

2079





For an example if we take the repeating words Hello and we see H and E are not in the same col or row so we have to take it as rectangular matrix form
C H
E F

IF TWO LETTERS COMES TOGETHER THEN FILL WITH THE FILLER CHARACTER

Example

Plaintext: attack

Digrams: at ta ck

Plaintext: neso academy

Digrams: ne so ac ad em yx

Plaintext: balloon

Digrams: ba ll oo n

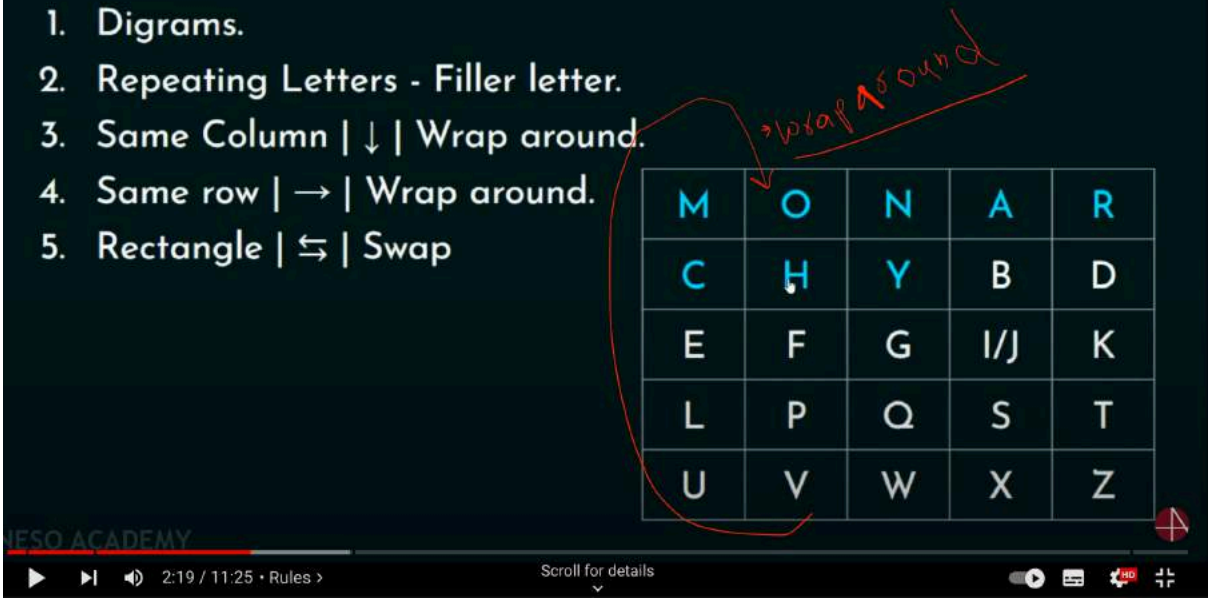
Digrams: ba lx lo on

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

NESO ACADEMY

Rules for encryption using Playfair Cipher

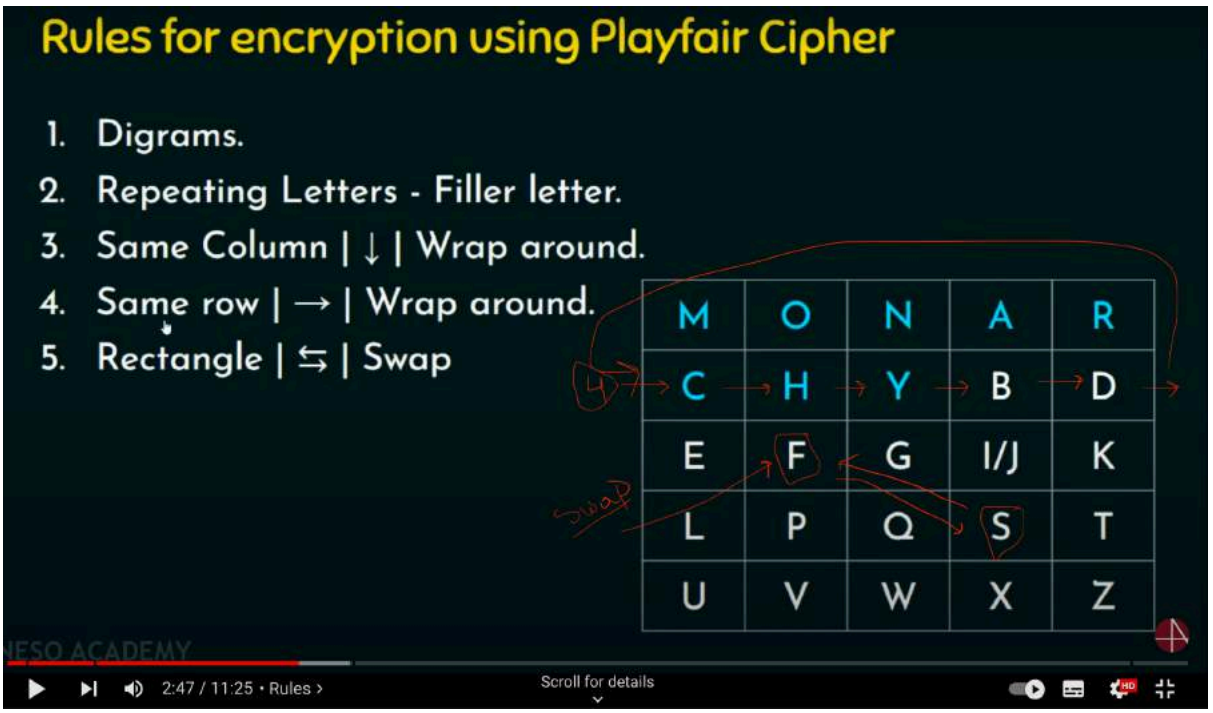
1. Digrams.
2. Repeating Letters - Filler letter.
3. Same Column | ↓ | Wrap around.
4. Same row | → | Wrap around.
5. Rectangle | ⇆ | Swap



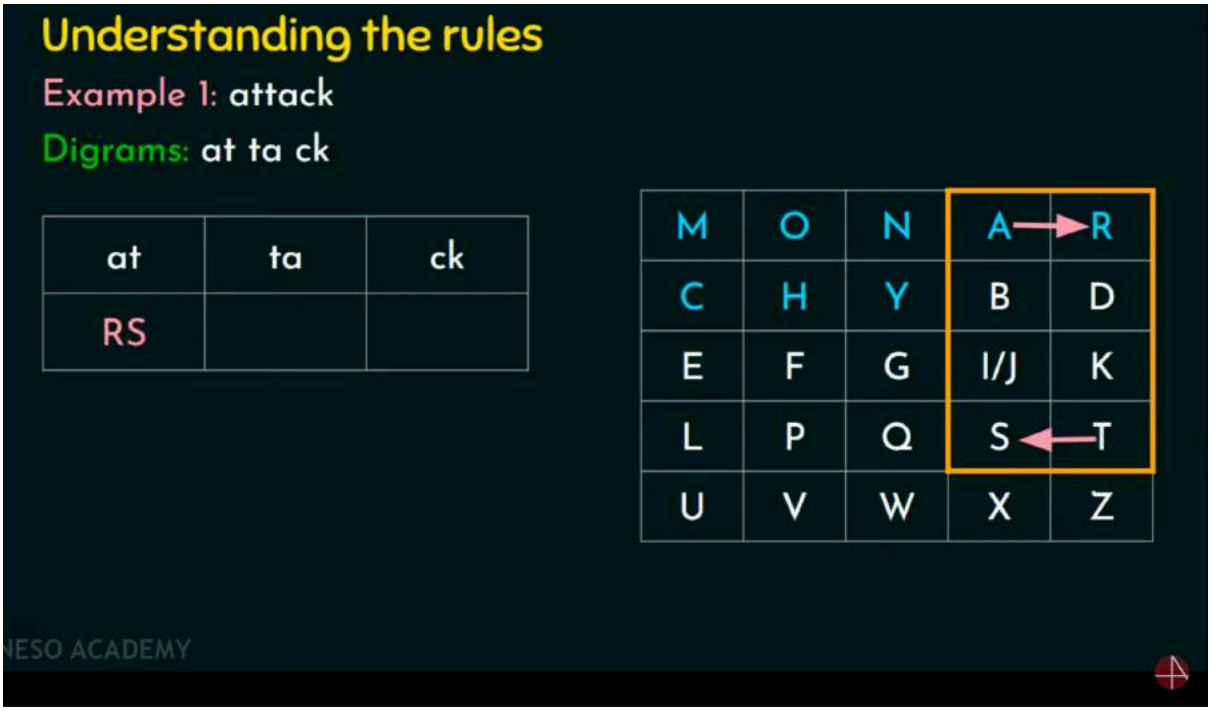
NESO ACADEMY

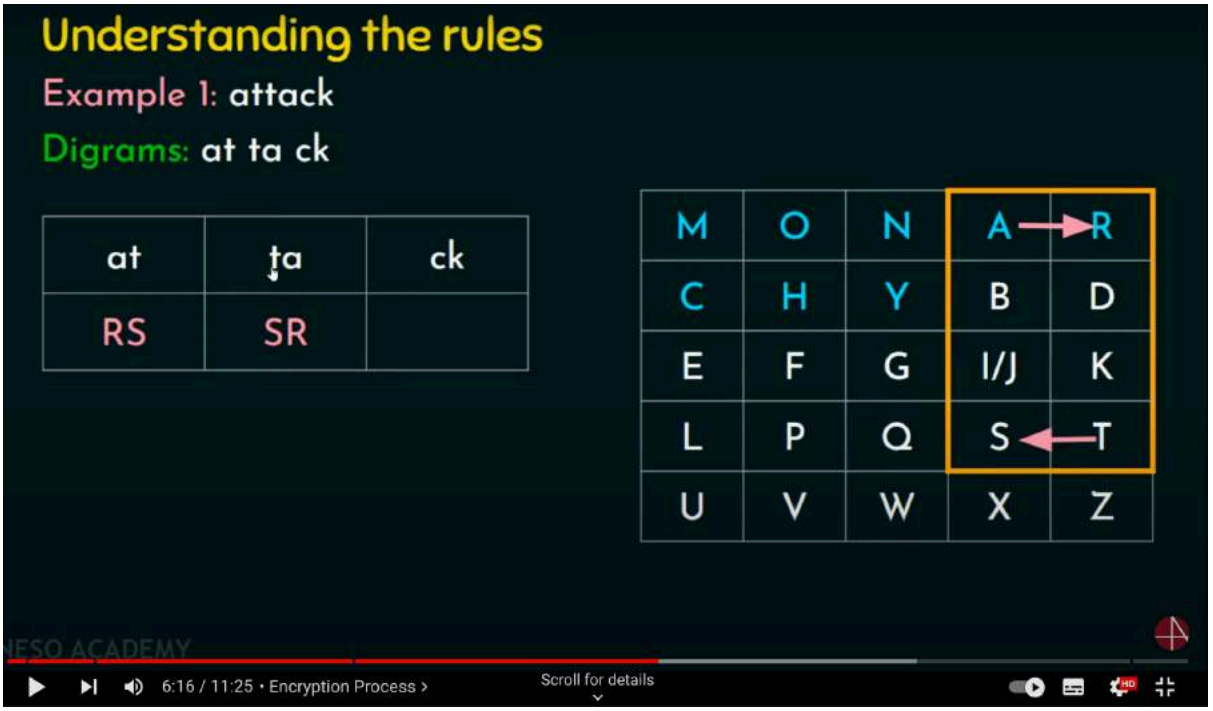
2:19 / 11:25 • Rules ›

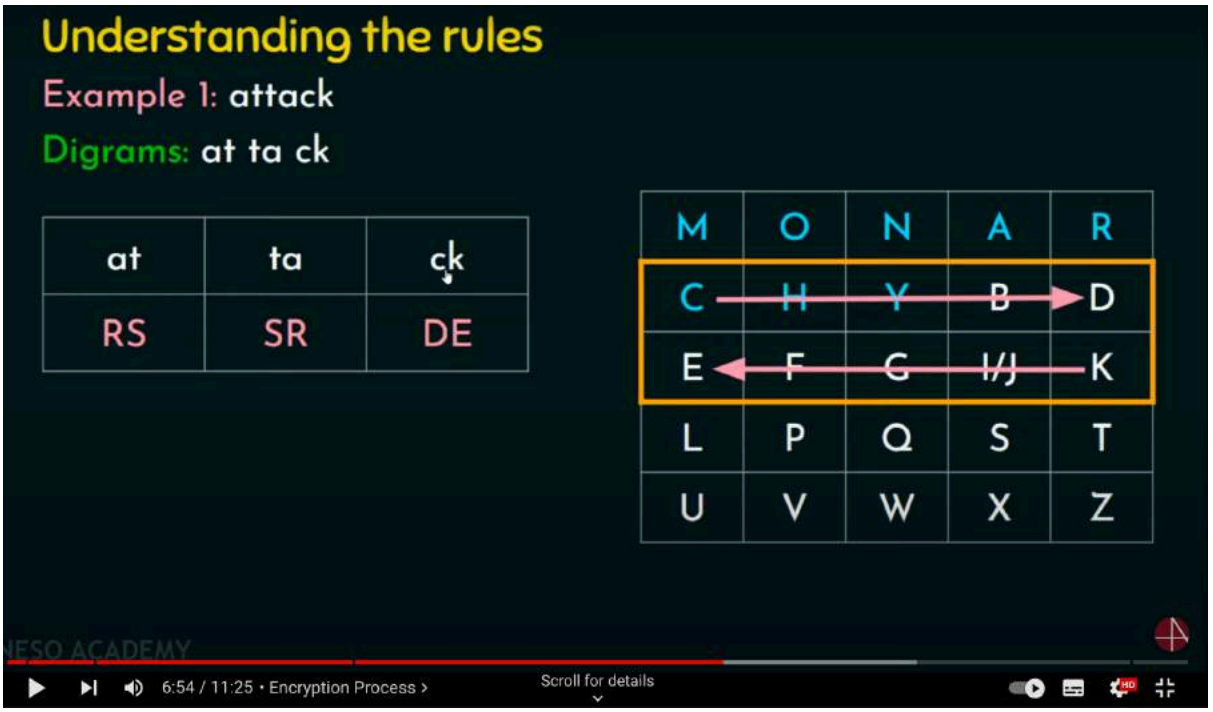
Scroll for details

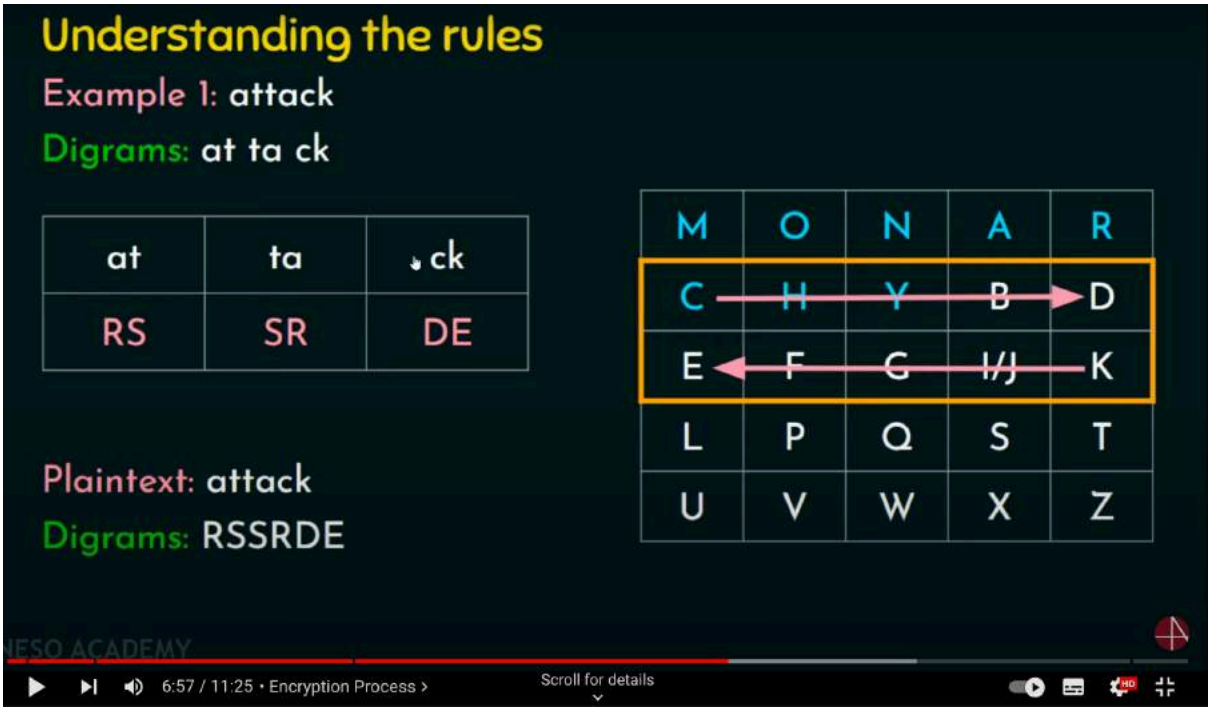

Rules for encryption using Playfair Cipher
1. Digrams.
2. Repeating Letters - Filler letter.
3. Same Column | ↓ | Wrap around.
4. Same row | → | Wrap around.
5. Rectangle | ⇆ | Swap
M O N A R
C H Y B D
E F G I/J K
L P Q S T
U V W X Z
Swap
NESO ACADEMY
2:47 / 11:25 • Rules ›
Scroll for details

ENCRYPT


Understanding the rules
Example 1: attack
Digrams: at ta ck
at ta ck
RS
M O N A R
C H Y B D
E F G I/J K
L P Q S T
U V W X Z
NESO ACADEMY

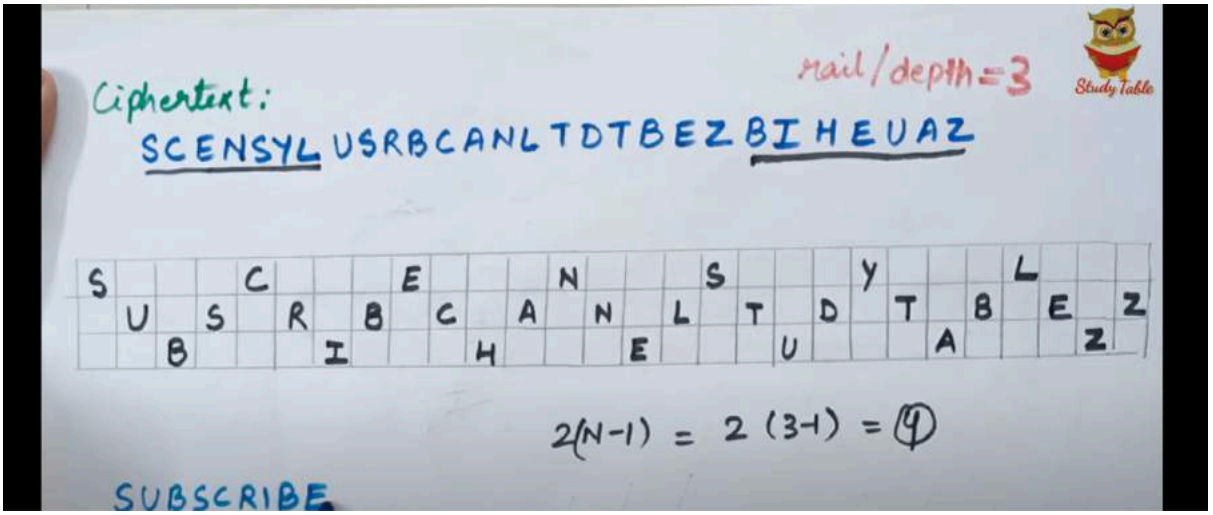

Understanding the rules
Example 1: attack
Digrams: at ta ck
at
ta
ck
RS
SR
M O N A R
C H Y B D
E F G I/J K
L P Q S T
U V W X Z
NESO ACADEMY
6:16 / 11:25 • Encryption Process >
Scroll for details

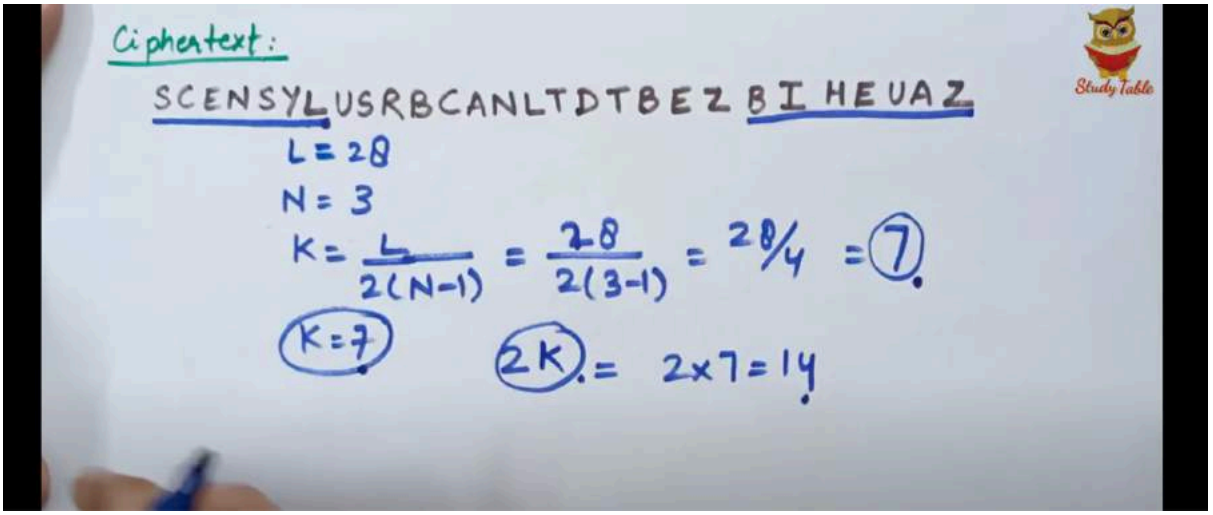

Understanding the rules
Example 1: attack
Digrams: at ta ck
at
ta
ck
RS
SR
DE
M O N A R
C H Y B D
E F G I/J K
L P Q S T
U V W X Z
NESO ACADEMY
6:54 / 11:25 • Encryption Process >
Scroll for details

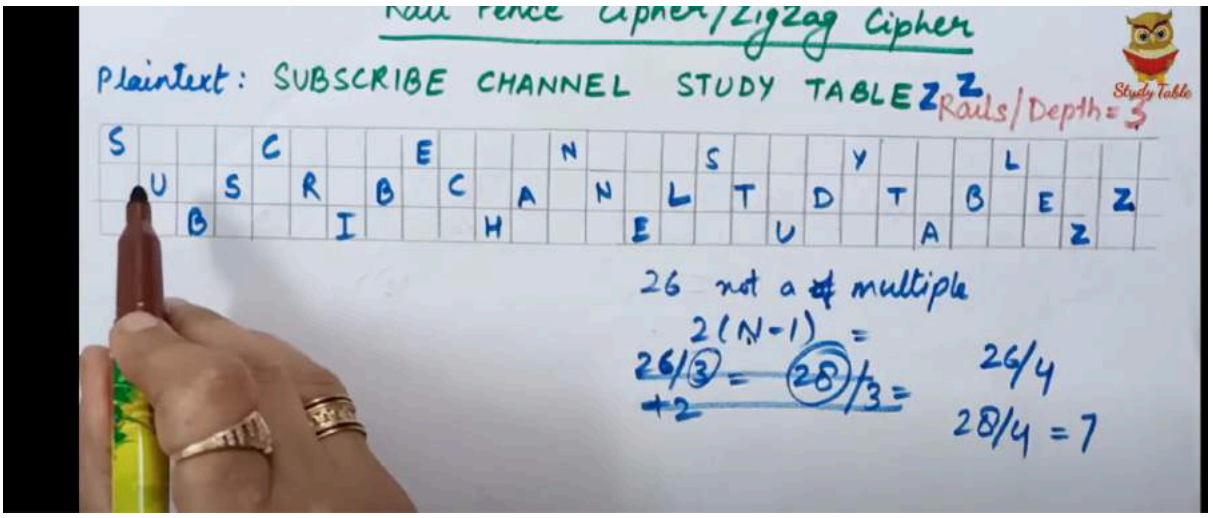


2. How are substitution ciphers different from transposition ciphers? Given a message M=" CSIT PROGRAM IS A HOT CAKE", encrypt M using Rail Fence cipher with rail size 3.

→







Describe about IPSec. List the five services of PGP.

What is IPSec?

IPSec is a set of communication rules or protocols for setting up secure connections over a network. Internet Protocol (IP) is the common standard that determines how data travels over the internet. IPSec adds encryption and authentication to make the protocol more secure. For example, it scrambles the data at its source and unscrambles it at its destination. It also authenticates the source of the data.

Why is IPSec important?

The Internet Engineering Task Force developed IPSec in the 1990s to ensure data confidentiality, integrity, and authenticity when accessing public networks. For example, users connect to the internet with an IPSec virtual private network (VPN) to access company files remotely. The IPSec protocol encrypts sensitive information to prevent unwanted monitoring. The server can also verify that the received data packets are authorized.

What are the uses of IPSec?

IPsec can be used to do the following:

- Provide router security when sending data across the public internet.
- Encrypt application data.
- Authenticate data quickly if the data originates from a known sender.
- Protect network data by setting up encrypted circuits, called IPsec tunnels, that encrypt all data sent between two endpoints.

Organizations use IPSec to protect against replay attacks. A replay attack, or man-in-the-middle attack, is an act of intercepting and altering ongoing transmission by routing data to an intermediary computer. IPSec protocol assigns a sequential number to each data packet and performs checks to detect signs of duplicate packets.

PGP - Securing Your Data Beyond Networks

Pretty Good Privacy (PGP) is a cryptographic software program used to encrypt and decrypt data for secure communication. Unlike IPSec, which focuses on network traffic, PGP secures individual files, emails, and disk partitions. Here are the five key services offered by PGP:

1. **Encryption and Decryption:** PGP uses public-key cryptography with a key pair (public and private key) to encrypt and decrypt data. The public key encrypts data, and only the corresponding private key can decrypt it. This ensures secure communication even without prior key exchange.
2. **Authentication:** PGP allows users to digitally sign messages using their private key. The recipient can verify the sender's identity using the sender's public key. This ensures authenticity and prevents message forgery.

3. **Non-Repudiation:** A digitally signed message with PGP provides non-repudiation. The sender cannot later deny sending the message as the signature proves their involvement.
4. **Data Compression (Optional):** PGP can optionally compress data before encryption, reducing file size and transmission time. This can be helpful for large files.
5. **Email Compatibility:** PGP is designed to work seamlessly with email communication. Emails can be encrypted and signed using PGP for secure and trustworthy communication.