**Cryptography and Network Security**

**Assignment 2**

**Submitted by: Himanshu Verma**

**201210023**

1. **Find the key for decryption using Hill cipher if the key for encryption is "DIMENSION."**

* **Sol:** To find the key for decryption using Hill cipher, the inverse matrix of the key matrix for encryption must be found.
* Firstly, the letters in the key "DIMENSION" must be converted to numbers using a simple substitution where A=0, B=1, C=2, and so on.

D = 3  
I = 8  
M = 12  
E = 4  
N = 13  
S = 18  
I = 8  
O = 14  
N = 13

* These numbers can then be arranged into a 3x3 matrix.

| 3 8 12 |  
| 4 13 18 |  
| 8 14 13 |

* The determinant of this matrix must then be calculated.

det = 3(13)(13) + 8(18)(8) + 12(4)(14) - 12(13)(8) - 3(18)(14) - 8(4)(13)

det = 507

* The multiplicative inverse of the determinant must then be found modulo 26.

507^-1 mod 26 = 19

* If the determinant is not relatively prime to 26, then no multiplicative inverse exists and the Hill cipher cannot be used.
* The inverse matrix can then be found by taking the transpose of the cofactor matrix and multiplying by the multiplicative inverse of the determinant.

| 229 122 23 |  
| 158 120 140 |  
| 93 199 54 |

* Then, this matrix must be converted back into letters using the inverse of the substitution.

23 -> X  
54 -> O  
93 -> H  
120 -> E  
122 -> G  
140 -> U  
158 -> S  
199 -> Z  
229 -> R

* Therefore, the key for decryption using Hill cipher is "XGH EUS ZRO."

1. **Describe Playfair cipher encryption rules through examples also implemented in Python.**

**Sol**: The Playfair cipher is a substitution cipher that encrypts pairs of letters at a time. It uses a 5x5 grid of letters, called a Playfair square or cipher key, which contains a keyword or phrase (excluding repeated letters) followed by the remaining letters of the alphabet. The key is used to create the initial Playfair square.

Here are the encryption rules for the Playfair cipher:

* Preparing the Playfair Square:
  + First, create a 5x5 grid using the keyword. Fill the grid row by row with the letters of the keyword, avoiding repeated letters.
  + Then, fill the remaining spaces in the grid with the letters of the alphabet, omitting the letters already used in the keyword.
* Handling Repeated Letters:
  + If there are repeated letters in the plaintext, insert a filler letter (often 'X') between the repeated letters and proceed with encryption.
  + If the plaintext length is odd after inserting filler letters, append another filler letter ('X') at the end.
* Encryption:
  + Take the plaintext in pairs of letters.
  + For each pair, find the letters' positions in the Playfair square.
  + If the letters are in the same row, replace them with the letters to their right (wrapping around to the beginning of the row if necessary).
  + If the letters are in the same column, replace them with the letters below them (wrapping around to the top of the column if necessary).
  + If the letters form a rectangle, replace them with the letters at the opposite corners of the rectangle.
* Output:
  + The ciphertext consists of the encrypted pairs of letters.

Example:

def prepare\_playfair\_square(keyword):

keyword = keyword.replace("J", "I") # Replace 'J' with 'I' to have 25 letters

keyword = "".join(dict.fromkeys(keyword)) # Remove duplicate letters

alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ" # Exclude 'J'

key = keyword + alphabet

playfair\_square = [list(key[i:i+5]) for i in range(0, 25, 5)]

return playfair\_square

def find\_position(letter, square):

for row in range(5):

for col in range(5):

if square[row][col] == letter:

return row, col

def encrypt\_pair(pair, square):

a, b = pair[0], pair[1]

row\_a, col\_a = find\_position(a, square)

row\_b, col\_b = find\_position(b, square)

if row\_a == row\_b:

return square[row\_a][(col\_a + 1) % 5] + square[row\_b][(col\_b + 1) % 5]

elif col\_a == col\_b:

return square[(row\_a + 1) % 5][col\_a] + square[(row\_b + 1) % 5][col\_b]

else:

return square[row\_a][col\_b] + square[row\_b][col\_a]

def playfair\_encrypt(plaintext, keyword):

playfair\_square = prepare\_playfair\_square(keyword.upper())

plaintext = plaintext.upper().replace("J", "I") # Replace 'J' with 'I'

plaintext\_pairs = [plaintext[i:i+2] for i in range(0, len(plaintext), 2)]

ciphertext = ""

for pair in plaintext\_pairs:

if len(pair) == 1: # Odd number of characters, append 'X' as filler

pair += "X"

ciphertext += encrypt\_pair(pair, playfair\_square)

return ciphertext

# Example usage:

keyword = "KEYWORD"

plaintext = "HELLO"

ciphertext = playfair\_encrypt(plaintext, keyword)

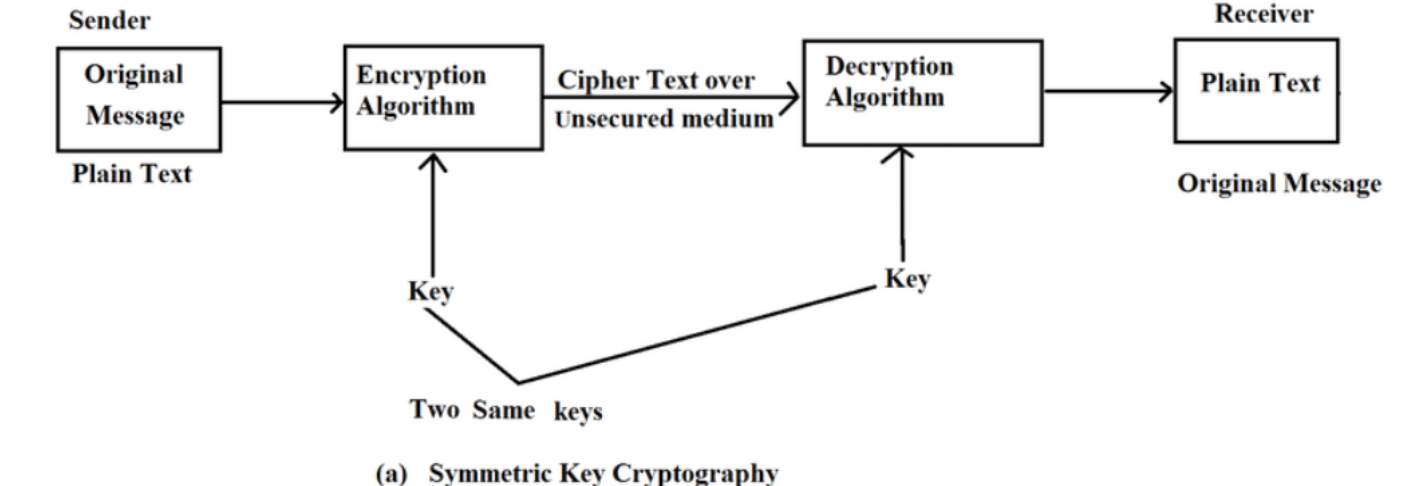
print("Plaintext:", plaintext)

print("Ciphertext:", ciphertext)

1. **Distinguish between cryptography and steganography.**
2. **Sol**: **Cryptography:**
   * Cryptography is the practice and study of techniques for secure communication in the presence of adversaries.
   * It involves converting plaintext into ciphertext using encryption algorithms and keys to ensure confidentiality, integrity, authentication, and non-repudiation of the data.
   * The main goal of cryptography is to make the content of a message unreadable to anyone except those who have the decryption key.
   * Cryptographic techniques include symmetric encryption (e.g., AES), asymmetric encryption (e.g., RSA), hashing algorithms (e.g., SHA-256), digital signatures, and cryptographic protocols (e.g., SSL/TLS).
3. **Steganography:**
   * Steganography is the practice of concealing information within other non-secret data, such as images, audio files, or text, in a way that hides the existence of the secret data.
   * It involves embedding the secret message into the carrier medium in such a way that it is not apparent to casual observers.
   * The primary goal of steganography is to maintain secrecy by hiding the existence of the message rather than its content.
   * Steganographic techniques include hiding information within the least significant bits of digital images or audio files, using invisible ink on physical documents, or employing specific patterns or techniques to hide messages in plain sight.
4. Differentiate between asymmetric and symmetric keys with block diagrams.

**Sol**: **Symmetric Key Encryption:**

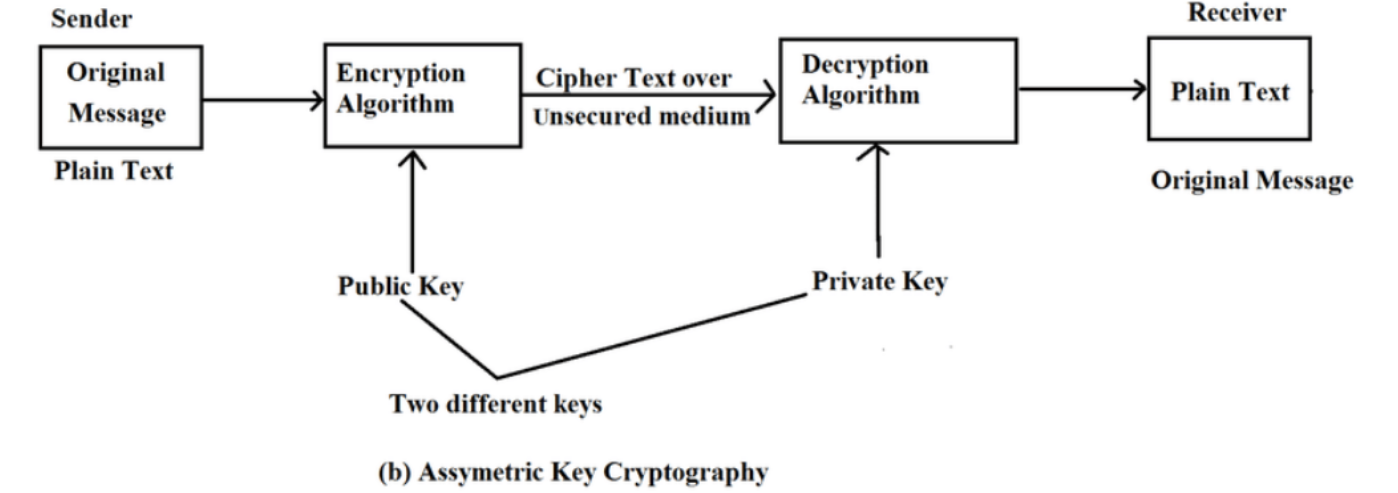
* In symmetric key encryption, also known as secret key encryption, the same key is used for both encryption and decryption. This means that both the sender and the receiver must possess the same secret key.
* Block Diagram of Symmetric Key Encryption:



* The plaintext message is encrypted using the secret key, resulting in ciphertext.
* The ciphertext is transmitted to the receiver.
* The receiver decrypts the ciphertext using the same secret key, recovering the original plaintext.

**Asymmetric Key Encryption:**

* In asymmetric key encryption, also known as public-key encryption, a pair of keys is used: a public key for encryption and a private key for decryption. The keys are mathematically related, but it is computationally infeasible to derive the private key from the public key.
* Block Diagram of Asymmetric Key Encryption:



* The sender obtains the recipient's public key.
* The plaintext message is encrypted using the recipient's public key, resulting in ciphertext.
* The ciphertext is transmitted to the receiver.
* The receiver decrypts the ciphertext using their private key, recovering the original plaintext.

1. **Transposition Ciphers**
   1. **Single Columnar Transposition**
      1. **Plain text - MEET ME AT SEVEN PM IN GATES**
      2. **Key - COMPSCI**
   2. **Double Columnar transposition**
      1. **Plain text - (Note- consider previous cipher text which you got to solve a section.)**
      2. **Key - take a random key**

**Sol:** (a)

* Write the message in rows underneath the keyword, filling in any empty spaces with placeholders (such as X).

C O M P S C I  
M E E T M E A  
T S E V E N P  
M I N G A T E  
S X X X X X X

* Read the columns in the order specified by the keyword to create the ciphertext.
* The keyword "COMPSI" tells us to read the columns in the order 3-1-6-2-4-5-7.
* So, the ciphertext for "MEET ME AT SEVEN PM IN GATES" using the key "COMPSCI" would be:

TSNEAXMVGMEMEIIXTPEX

(b)

* Double Columnar Transposition is a type of transposition cipher where a message is encrypted using two different columnar transposition ciphers. Here's an example of how to use Double Columnar Transposition to encrypt the previously encrypted message "TSNEAXMVGMEMEIIXTPEX" using a random key:

1. Break the ciphertext into two equal-length rows and arrange them horizontally.

TSNEAXMVGMEMEIIXTP  
EX

1. Choose two different keys, each specifying a different order for the columns. Let's say the first key is "3142" and the second key is "2413", which means that we'll read the columns in the order 3-1-4-2 for the first row and in the order 2-4-1-3 for the second row.
2. Rearrange the columns of the two rows according to their respective keys.

TSNEAXMVGMEMEIIXTP -> NEAXTGXIMIEMPSVEMG  
EX -> XTIXEEGMEPMMEASPNV

1. Break the concatenated ciphertext into two separate rows and present the final ciphertext.

NEAXTGXIMIEMPSVEMGXTIXEEGMEPMMEASPNV

So, the final ciphertext for the message "MEET ME AT SEVEN PM IN GATES" encrypted with a Double Columnar Transposition cipher using the keys "3142" and "2413" would be:  
NEAXTGXIMIEMPSVEMGXTIXEEGMEPMMEASPNV