Experiment No - 01

Experiment Name - Write a program to implement encryption and decryption using Caesar cipher.

Theory -

The Caesar cipher is the simplest and oldest method of cryptography. The Caesar cipher method is based on a mono-alphabetic cipher and is also called a shift cipher or additive cipher. Julius Caesar used the shift cipher (additive cipher) technique to communicate with his officers. For this reason, the shift cipher technique is called the Caesar cipher. The Caesar cipher is a kind of replacement (substitution) cipher, where all letter of plain text is replaced by another letter.

Algorithm for Caesar Cipher:

Input:

1. Choose a shift value between 1 and 25.

2. Write down the alphabet in order from A to Z.

3. Create a new alphabet by shifting each letter of the original alphabet by the shift value. For example, if the shift value is 3, the new alphabet would be:

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

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

4. Replace each letter of the message with the corresponding letter from the new alphabet. For example, if the shift value is 3, the word “hello” would become “khoor”.

5.To decrypt the message, shift each letter back by the same amount. For example, if the shift value is 3, the encrypted message “khoor” would become “hello”.

Procedure:

Traverse the given text one character at a time.

For each character, transform the given character as per the rule, depending on whether we’re encrypting or decrypting the text.

Return the new string generated.

Decrypt:We can either write another function decrypt like encrypt, that’ll apply the given shift in the opposite direction to decrypt the original text. However, we can use the cyclic property of the cipher under modulo, hence we can simply observe

Cipher(n) = De-cipher(26-n)

Conclusion

The Caesar cipher, though a rudimentary encryption technique, offers a fundamental understanding of substitution ciphers. Its simplicity in both encryption and decryption makes it a useful educational tool. However, its vulnerability to brute force attacks due to its limited key space renders it unsuitable for securing sensitive information. As demonstrated, more robust encryption algorithms are necessary for ensuring the confidentiality and integrity of data in real-world applications.

Experiment No - 02

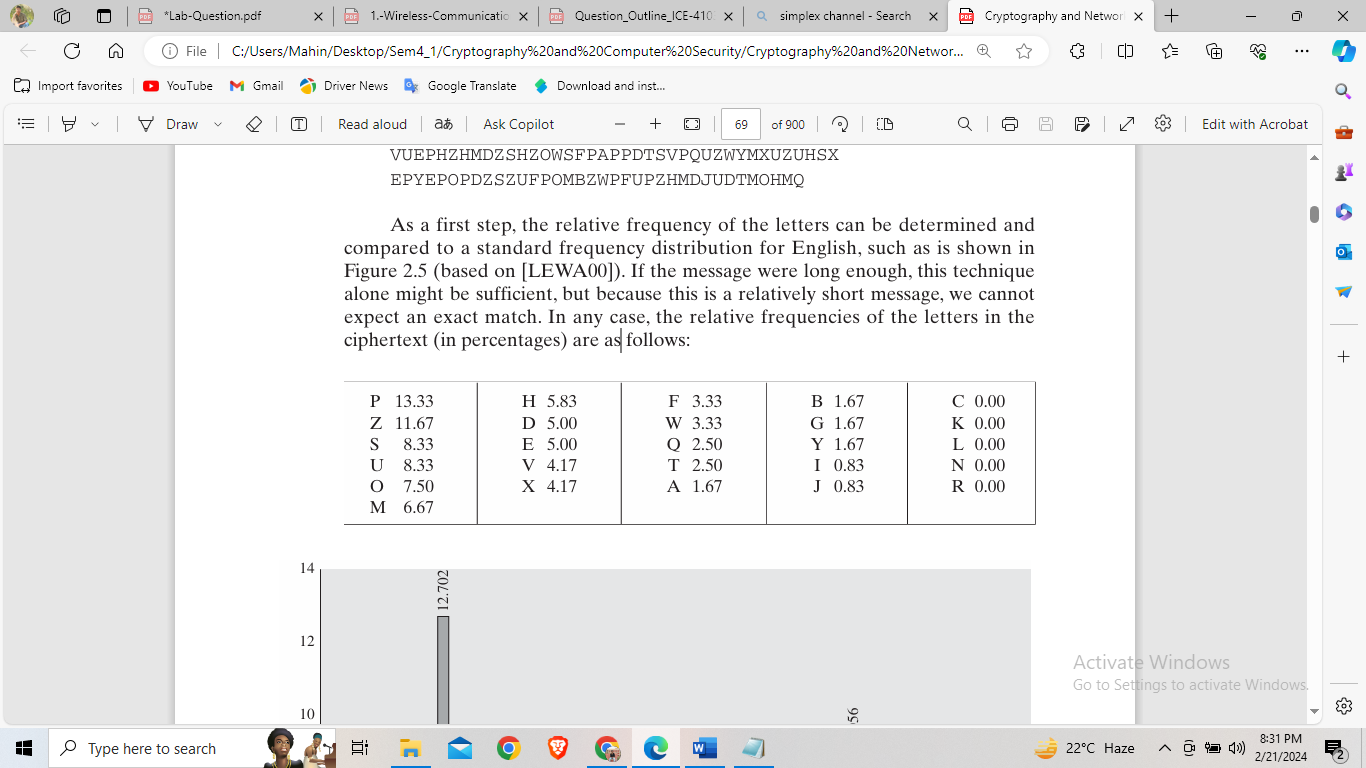
Experiment Name – 02 Write a program to implement encryption and decryption using Mono-Alphabetic cipher.

Theory:

Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. For example, if ‘A’ is encrypted as ‘D’, for any number of occurrences in that plaintext, ‘A’ will always get encrypted to ‘D’.

Here the relative frequency of the letters can be determined and

compared to a standard frequency distribution for English



**Fig – Chart of Relative Frequency**

We can find the cipher text according with the plaintext by using relative frequency. For example, certain words may be known to be in the text. Or we could look for

repeating sequences of cipher letters and try to deduce their plaintext equivalents.

A powerful tool is to look at the frequency of two-letter combinations, known

as digrams.

Monoalphabetic ciphers are easy to break because they reflect the frequency

data of the original alphabet.

Decrypt:

This is using the same process as encryption algorithm.

Conclusion

The Mono-Alphabetic cipher, though instructive in its simplicity, demonstrates the vulnerabilities inherent in basic substitution ciphers. Through our experimentation, we grasped the fundamental concept of mapping plaintext characters to ciphertext equivalents. However, the cipher's susceptibility to frequency analysis and pattern recognition underscores its inadequacy for secure communication. As such, while valuable for educational purposes, reliance on the Mono-Alphabetic cipher for real-world encryption is ill-advised. Modern encryption methods offer far greater security through their complexity and resilience against cryptographic attacks.

Experiment No - 03

Experiment Name - Write a program to implement encryption and decryption using Playfair cipher.

Theory:

Playfair cipher is an encryption algorithm to encrypt or encode a message. It is the same as a traditional cipher. The only difference is that it encrypts a **digraph** (a pair of two letters) instead of a single letter.

It initially creates a key-table of 5\*5 matrix. The matrix contains alphabets that act as the key for encryption of the plaintext. Note that any alphabet should not be repeated. Another point to note is that there are 26 alphabets and we have only 25 blocks to put a letter inside it. Therefore, one letter is excess so, a letter will be omitted (usually J) from the matrix. Nevertheless, the plaintext contains J, then **J** is replaced by **I**. It means treat I and J as the same letter, accordingly.

Since Playfair cipher encrypts the message **digraph by digraph**. Therefore, the Playfair cipher is an example of a **digraph substitution cipher**.

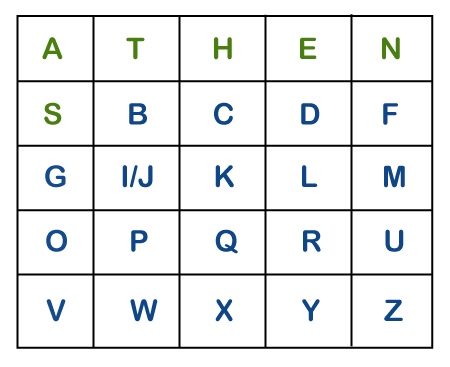
## Playfair Cipher Encryption Rules

1. First, split the plaintext into **digraphs** (pair of two letters). If the plaintext has the odd number of letters, append the letter **Z** at the end of the plaintext. It makes the plaintext of **even.** For example, the plaintext **MANGO** has five letters. So, it is not possible to make a digraph. Since, we will append a letter **Z** at the end of the plaintext, i.e. **MANGOZ.**

2. After that, break the plaintext into **digraphs** (pair of two letters). If any letter appears twice (side by side), put **X** at the place of the second occurrence. Suppose the plaintext is **COMMUNICATE** then its digraph becomes **CO MX MU NI CA TE**. Similarly, the digraph for the plaintext **JAZZ** will be **JA ZX ZX**, and for plaintext **GREET**, the digraph will be **GR EX ET.**

3. To determine the cipher (encryption) text, first, build a 5\*5 key-matrix or key-table and filled it with the letters of alphabets, as directed below:

* Fill the first row (left to right) with the letters of the given keyword (**ATHENS**). If the keyword has duplicate letters (if any) avoid them. It means a letter will be considered only once. After that, fill the remaining letters in alphabetical order. Let's create a 5\*5 key-matrix for the keyword **ATHENS**.

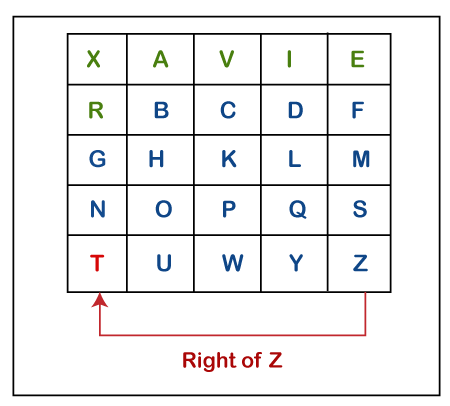


Note that in the above matrix any letter is not repeated. The letters in the first row (in green color) represent the keyword and the remaining letters sets in alphabetical order.

4. There may be the following three conditions:

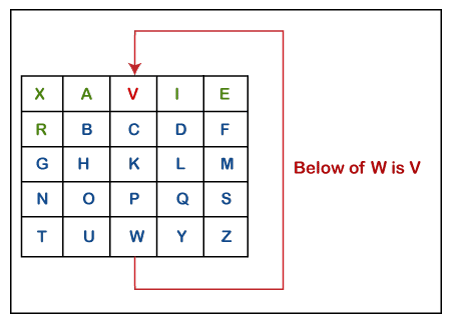
**i) If a pair of letters (digraph) appears in the same row**

In this case, replace each letter of the digraph with the letters immediately to their right. If there is no letter to the right, consider the first letter of the same row as the right letter. Suppose, **Z** is a letter whose right letter is required, in such case, **T** will be right to Z.



**ii) If a pair of letters (digraph) appears in the same column**

In this case, replace each letter of the digraph with the letters immediately below them. If there is no letter below, wrap around to the top of the same column. Suppose, **W** is a letter whose below letter is required, in such case, **V** will be below W.



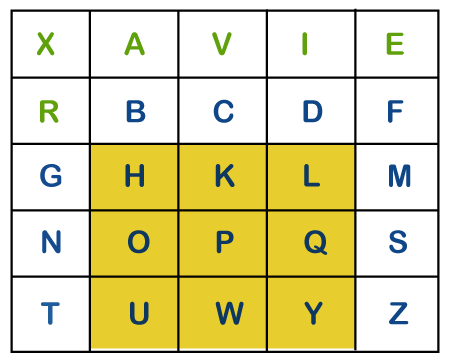
**iii) If a pair of letters (digraph) appears in a different row and different column**

In this case, select a 3\*3 matrix from a 5\*5 matrix such that pair of letters appear in the 3\*3 matrix. Since they occupy two opposite corners of a square within the matrix. The other corner will be a cipher for the given digraph.

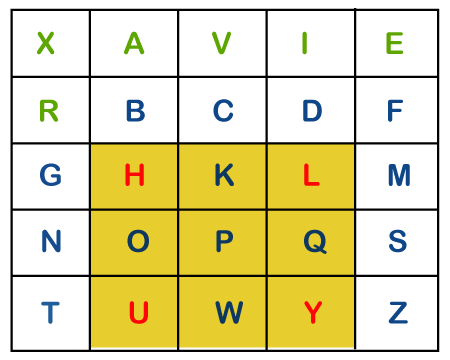
AD

In other words, we can also say that intersection of H and Y will be the cipher for the first letter and

Suppose, a digraph is **HY** and we have to find a cipher for it. We observe that both H and Y are placed in different rows and different columns. In such cases, we have to select a 3\*3 matrix in such a way that both H and Y appear in the 3\*3 matrix (highlighted with yellow color). Now, we will consider only the selected matrix to find the cipher.



Now to find the cipher for HY, we will consider the diagonal **opposite** to HY, i.e. **LU.** Therefore, the cipher for **H** will be **L,** and the cipher for **Y** will be **U**.



## Playfair Cipher Decryption

The decryption procedure is the same as encryption, but the steps are applied in **reverse** order. For decryption cipher is symmetric (move left along rows and up along columns). The receiver of the plain text has the same key and can create the same key-table that is used to decrypt the message.

Conclusion

In our exploration of the Playfair cipher, we uncovered a more intricate and resilient method of encryption compared to simpler substitution ciphers. By employing a 5x5 grid and specific rules for handling letter pairs, the Playfair cipher introduces complexity that enhances security by obscuring individual letter frequencies and breaking patterns. While it represents an advancement in encryption techniques, it's important to recognize that the Playfair cipher still has vulnerabilities, albeit less pronounced than those found in monoalphabetic ciphers. As such, while useful for educational purposes, its application in real-world scenarios may require additional layers of security to ensure robust protection of sensitive information

Experiment No: - 04

Experiment Name - Write a program to implement encryption and decryption using Hill cipher.

Theory:

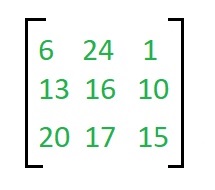
Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.  
The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).  
**Examples:**

Input : Plaintext: ACT  
 Key: GYBNQKURP  
Output : Ciphertext: POH

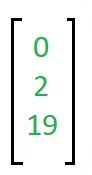
Input : Plaintext: GFG  
 Key: HILLMAGIC   
Output : Ciphertext: SWK

**Encryption**

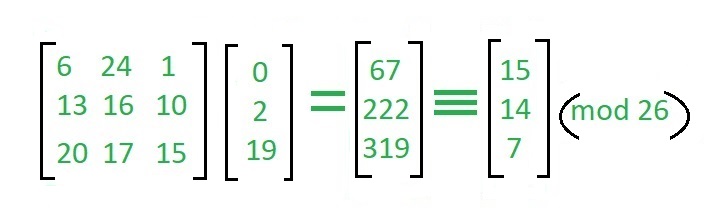
We have to encrypt the message ‘ACT’ (n=3).The key is ‘GYBNQKURP’ which can be written as the nxn matrix:



The message ‘ACT’ is written as vector:



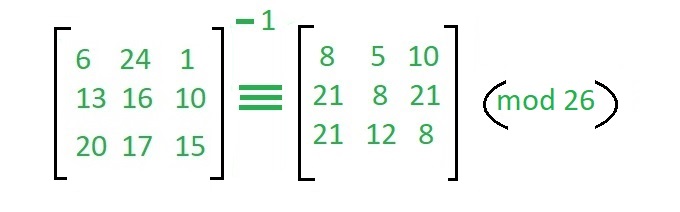
The enciphered vector is given as:



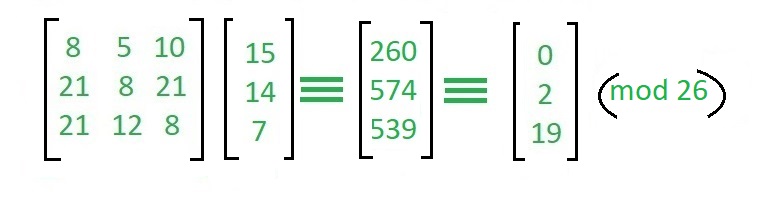
which corresponds to ciphertext of ‘POH’

**Decryption**

To decrypt the message, we turn the ciphertext back into a vector, then simply multiply by the inverse matrix of the key matrix (IFKVIVVMI in letters).The inverse of the matrix used in the previous example is:



For the previous Ciphertext ‘POH’:



which gives us back ‘ACT’.   
Assume that all the alphabets are in upper case.   
Below is the implementation of the above idea for n=3.

Conclusion

Our investigation into the Hill cipher has underscored its efficacy as a sophisticated encryption technique based on matrix operations. By transforming plaintext blocks into vectors and applying matrix multiplication modulo a chosen integer, the Hill cipher introduces a level of complexity that enhances security. Through experimentation, we observed how the choice of the encryption key significantly impacts the cipher's resistance to cryptanalysis. However, while the Hill cipher offers robust protection against traditional frequency analysis, its practical implementation may necessitate careful key management and computational considerations. Despite these challenges, our exploration reaffirms the importance of diversifying cryptographic methods to ensure the integrity and confidentiality of digital communication in real-world scenarios.

Experiment No – 5

# Experiment Name – Write a program to implement encryption and decryption using Poly-Alphabetic cipher.

Theory:

Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenère square or Vigenère table.

The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.

At different points in the encryption process, the cipher uses a different alphabet from one of the rows.

The alphabet used at each point depends on a repeating keyword.

Example:

Input:

Plaintext: GEEKSFORGEEKS  
 Keyword : AYUSH  
Output :

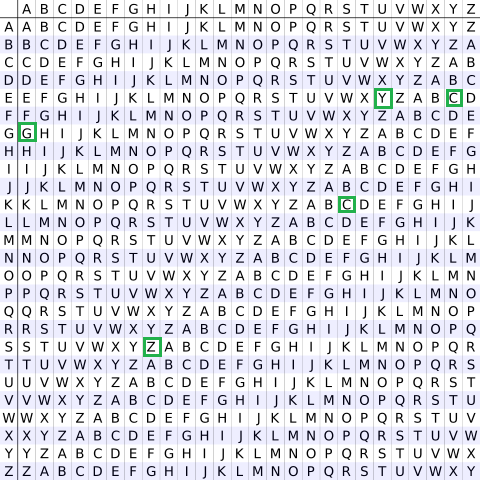
Ciphertext: GCYCZFMLYLEIM

For generating key, the given keyword is repeated in a circular manner until it matches the length of the plain text.  
The keyword "AYUSH" generates the key "AYUSHAYUSHAYU"  
The plain text is then encrypted using the process   
explained below.

Encryption:

The first letter of the plaintext, G is paired with A, the first letter of the key. So use row G and column A of the Vigenère square, namely G. Similarly, for the second letter of the plaintext, the second letter of the key is used, the letter at row E, and column Y is C. The rest of the plaintext is enciphered in a similar fashion.

Table to encrypt – Geeks



Decryption:  
Decryption is performed by going to the row in the table corresponding to the key, finding the position of the ciphertext letter in this row, and then using the column’s label as the plaintext. For example, in row A (from AYUSH), the ciphertext G appears in column G, which is the first plaintext letter. Next, we go to row Y (from AYUSH), locate the ciphertext C which is found in column E, thus E is the second plaintext letter.

A easier implementation could be to visualize Vigenère algebraically by converting [A-Z] into numbers [0–25].

Encryption  
The plaintext(P) and key(K) are added modulo 26.  
Ei = (Pi + Ki) mod 26  
  
Decryption  
Di = (Ei - Ki) mod 26

Note: Di denotes the offset of the i-th character of the plaintext. Like offset of A is 0 and of B is 1 and so on.

Conclusion

In our exploration of the Poly-Alphabetic cipher, we've encountered a fascinating encryption technique that introduces complexity and enhanced security compared to mono-alphabetic ciphers. By utilizing multiple alphabets based on a keyword or key phrase, the Poly-Alphabetic cipher scrambles plaintext characters in a manner that obscures patterns and makes frequency analysis significantly more challenging. Through our experimentation, we've observed how the choice of key and the length of the keyword directly influence the strength of the encryption. While the Poly-Alphabetic cipher offers greater resistance to cryptanalysis than simpler substitution ciphers, it is not impervious to attacks. Nonetheless, our investigation highlights the importance of leveraging diverse cryptographic methods to safeguard sensitive information in the digital age, emphasizing the need for continual research and innovation in the field of cryptography.

Experiment No – 06

Experiment Name - Write a program to implement encryption and decryption using Vernam cipher.

Theory:

Vernam Cipher is a method of encrypting alphabetic text. It is one of the Substitution techniques for converting plain text into cipher text. In this mechanism, we assign a number to each character of the Plain-Text, like (a = 0, b = 1, c = 2, … z = 25).   
Method to take key: In the Vernam cipher algorithm, we take a key to encrypt the plain text whose length should be equal to the length of the plain text.

Encryption Algorithm

Assign a number to each character of the plain text and the key according to alphabetical order.

Bitwise XOR both the number (Corresponding plain-text character number and Key character number).

Subtract the number from 26 if the resulting number is greater than or equal to 26, if it isn’t then leave it.

Example 1:

Plain-Text: O A K  
Key: S O N  
O ==> 14 = 0 1 1 1 0  
S ==> 18 = 1 0 0 1 0  
Bitwise XOR Result: 1 1 1 0 0 = 28

Since the resulting number is greater than 26, subtract 26 from it. Then convert the Cipher-Text character number to the Cipher-Text character.

28 - 26 = 2 ==> C  
CIPHER-TEXT: C

Similarly, do the same for the other corresponding characters,

PT: O A K  
NO: 14 00 10  
KEY: S O N  
NO: 18 14 13

New Cipher-Text is after getting the corresponding character from the resulting number.

CT-NO: 02 14 07  
CT: C O H

Example 2:

Plain-Text: RAMSWARUPK  
Key: RANCHOBABA

Now according to our encryption algorithm, we assign a number to each character of our plain text and key.

PT: R A M S W A R U P K  
NO: 17 0 12 18 22 0 17 20 15 10  
KEY: R A N C H O B A B A   
NO: 17 0 13 2 7 14 1 0 1 0

Now Bitwise XOR the number of Plain-Text and Key and after doing the XOR operation and subtraction operation (if required), we will get the corresponding Cipher-Text character number.

CT-NO: 0 0 1 16 17 14 16 20 14 10

Since there are no numbers that are greater than or equal to 26 we do not have to subtract 26 from any of them.

New Cipher-Text is after getting the corresponding character from the number.

CIPHER-TEXT: A A B Q R O Q U O K

Note: For the Decryption apply the just reverse process of encryption.

We can drive decryption algorithms by using a similar process of encryption algorithm.

Conclusion

Our exploration of the Vernam cipher has revealed a powerful encryption technique that offers unparalleled security through its use of a one-time pad. By combining the plaintext with a random key of equal length, the Vernam cipher ensures that each character in the ciphertext is essentially random and independent of any patterns in the plaintext. Through our experimentation, we observed how the security of the Vernam cipher relies heavily on the randomness and secrecy of the key. However, while the Vernam cipher provides theoretical perfect secrecy, practical challenges such as key distribution and management pose significant hurdles to its widespread use. Nonetheless, our investigation underscores the importance of exploring cryptographic techniques like the Vernam cipher, as they serve as valuable tools in certain specialized applications where security is paramount.

Experiment No – 07

Experiment No - Write a program to implement encryption and decryption using Brute force attack cipher.

Theory:

Encryption is the process of transforming plaintext into ciphertext to secure it from unauthorized access. Decryption, on the other hand, is the reverse process of converting ciphertext back into plaintext. In this experiment, we explore encryption and decryption using a simple substitution cipher and discuss the application of a brute force attack to crack the cipher.

Substitution Cipher

A substitution cipher is a method of encryption where each letter in the plaintext is replaced by another letter according to a predetermined key. The key defines the mapping between the original letters and their substitutes. For example, in a Caesar cipher, each letter is shifted a fixed number of positions down or up the alphabet.

Encryption Process

To encrypt plaintext using a substitution cipher, each letter is replaced with its corresponding mapped letter according to the key. This process ensures that the resulting ciphertext appears random and unintelligible without knowledge of the key.

Decryption Process

Decryption involves reversing the encryption process. With knowledge of the key, the ciphertext can be converted back into plaintext by substituting each letter with its original counterpart according to the key mapping.

Brute Force Attack

A brute force attack is a cryptanalysis technique that involves systematically trying all possible keys until the correct one is found. For a simple substitution cipher, the key space is relatively small, making it feasible to try all possible keys. Once the ciphertext is obtained, the attacker iterates through each possible key, decrypting the ciphertext with each key, and evaluating the resulting plaintext. The key that produces the most intelligible plaintext is likely the correct one.

Limitations

While a brute force attack is effective against simple substitution ciphers due to their small key space, it becomes impractical for more complex ciphers with larger key spaces. Additionally, the success of a brute force attack relies on the ability to recognize the correct plaintext among the many possible decryption results.

**Conclusion**

In conclusion, encryption and decryption using a substitution cipher provide a basic method of securing data. However, the vulnerability of such ciphers to brute force attacks highlights the importance of using more robust encryption techniques, especially for sensitive information.

Experiment No – 08

Experiment Name - Write a program to implement encryption and decryption using RSA algorithm.

Theory:  
RSA (Rivest-Shamir-Adleman) is a widely used asymmetric encryption algorithm named after its inventors. Unlike symmetric encryption algorithms, which use the same key for both encryption and decryption, RSA uses a pair of keys: a public key for encryption and a private key for decryption. In this experiment, we explore the theory behind RSA encryption and decryption.

Key Generation

The security of RSA relies on the difficulty of factoring large composite numbers into their prime factors. The key generation process involves the following steps:

Choose two distinct prime numbers, p and q.

Compute their product, n = p \* q. The modulus n is part of both the public and private keys.

Calculate Euler's totient function, φ(n) = (p - 1) \* (q - 1).

Choose an integer e such that 1 < e < φ(n) and e is coprime with φ(n). This is the public exponent.

Compute the modular multiplicative inverse of e modulo φ(n), denoted as d. This is the private exponent.

The public key consists of the modulus n and the public exponent e, while the private key consists of the modulus n and the private exponent d.

Encryption Process

To encrypt a message M using RSA, the sender obtains the recipient's public key (n, e) and performs the following computation:

C ≡ M^e (mod n)

Here, C is the ciphertext obtained after raising the plaintext message M to the power of the public exponent e modulo n.

Decryption Process

To decrypt the ciphertext C using RSA, the recipient uses their private key (n, d) and computes:

M ≡ C^d (mod n)

The plaintext message M is obtained by raising the ciphertext C to the power of the private exponent d modulo n.

Security

The security of RSA is based on the difficulty of factoring the modulus n into its prime factors. As long as the factors of n remain unknown, it is computationally infeasible to derive the private key d from the public key (n, e) alone. Therefore, RSA provides a secure method of communication over insecure channels.

Conclusion

In conclusion, RSA encryption and decryption provide a robust method of securing communication by leveraging the mathematical properties of large prime numbers. The use of asymmetric keys allows for secure transmission of information without the need for a shared secret key. However, it is essential to choose appropriate key sizes to withstand potential attacks as computational power increases.