**Week - 1 Stream Ciphers**

* 1. **Write a C Program to implement Shift Cipher.**

**AIM:** To write a C Program to implement Shift Cipher.

**DESCRIPTION:**

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.A Caesar cipher is a weak method of cryptography. It can be easily hacked. It means the message encrypted by this method can be easily decrypted.

**Plaintext:** It is a simple message written by the user.

**Ciphertext:** It is an encrypted message after applying some technique.

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down.

The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

(Encryption Phase with shift n) En(x) = (x+n)mod 26

(Decryption Phase with shift n) Dn(x) = (x-n)mod 26

If any case (Dn) value becomes negative (-ve), in this case, we will add 26 in the negative value.

# Where,

E denotes the encryption

D denotes the decryption

x denotes the letters value

n denotes the key value (shift value)

# PROGRAM:

# #include<ctype.h>

# #include<stdio.h>

# int main()

# {

# char text[500], ch;

# int key,i;

# printf("Enter a message to decrypt: ");

# scanf("%s", text);

# printf("Enter the key: ");

# scanf("%d", & key);

# for(i = 0; text[i] != '\0'; i++)

# {

# ch = text[i];

# if (isalnum(ch))

# {

# if (islower(ch))

# {

# ch = (ch - 'a' - key + 26) % 26 + 'a';

# }

# if (isupper(ch))

# {

# ch = (ch - 'A' - key + 26) % 26 + 'A';

# }

# if (isdigit(ch))

# {

# ch = (ch - '0' - key + 10) % 10 + '0';

# }

# }

# else

# {

# printf("Invalid Message");

# }

# text[i] = ch;

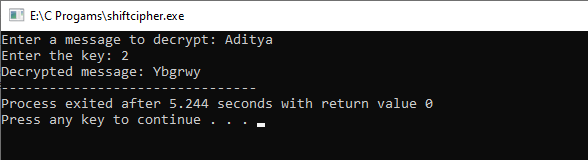
# }

# printf("Decrypted message: %s", text);

# return 0;

# }

**Output:**



**1.2) Write a C Program to implement Mono-Alphabetic Substitution Cipher.**

**AIM:** To write a C Program to implement Mono-Alphabetic Substitution Cipher. **DESCRIPTION:**

The substitution cipher is the oldest forms of encryption algorithms according to creates each character of a plaintext message and require a substitution process to restore it with a new character in the ciphertext. This substitution method is deterministic and reversible, enabling the intended message recipients to reverse-substitute ciphertext characters to retrieve the plaintext.

The specific form of substitution cipher is the Monoalphabetic Substitution Cipher, is known as “Simple Substitution Cipher”. Monoalphabetic Substitution Ciphers based on an individual key mapping function K, which consistently replaces a specific character α with a character from the mapping K (α).

A monoalphabetic cipher is any cipher in which the letters of the plain text are mapped to cipher text letters based on a single alphabetic key. Examples of monoalphabetic ciphers would include the Caesar- shift cipher, where each letter is shifted based on a numeric key, and the atbash cipher, where each letter is mapped to the letter symmetric to it about the center of the alphabet.

Monoalphabetic cipher is one where each symbol in plain text is mapped to a fixed symbol in cipher text. The relationship between a character in the plain text and the characters in the cipher text is one-to-one. Each alphabetic character of plain text is mapped onto a unique alphabetic character of a cipher text.

A stream cipher is a monoalphabetic cipher if the value of key does not depend on the position of the plain text character in the plain text stream. It includes additive, multiplicative, affine and monoalphabetic substitution cipher. Monoalphabetic Cipher is described as a substitution cipher in which the same fixed mappings from plain text to cipher letters across the entire text are used. Monoalphabetic ciphers are not that strong as compared to polyalphabetic cipher.

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 occurrence in that plaintext, ‘A’ will always get encrypted to ‘D’.

**Program:**

#include<stdio.h>

char monocipher\_encr(char);

char alpha[26][2] = { { 'a', 'f' }, { 'b', 'a' }, { 'c', 'g' }, { 'd', 'u' }, {'e', 'n' }, { 'f', 'i' }, { 'g', 'j' }, { 'h', 'k' }, { 'i', 'l' },{'j', 'm' }, { 'k', 'o' }, { 'l', 'p' }, { 'm', 'q' }, { 'n', 'r' }, {'o', 's' }, { 'p', 't' }, { 'q', 'v' }, { 'r', 'w' }, { 's', 'x' }, {'t', 'y' }, {'v', 'b' }, { 'u', 'z' }, { 'w', 'c' }, { 'x', 'd' }, {'y', 'e' }, { 'z', 'h' }};

char str[20];

int main() {

char str[20], str2[20];

int i;

printf("\n Enter String:");

gets(str);

for (i = 0; str[i]; i++) {

str2[i] = monocipher\_encr(str[i]);

}

str2[i] = '\0';

printf("\n Before Decryption:%s", str);

printf("\n After Decryption:%s\n", str2);

}

char monocipher\_encr(char a)

{

int i;

for (i = 0; i< 26; i++) {

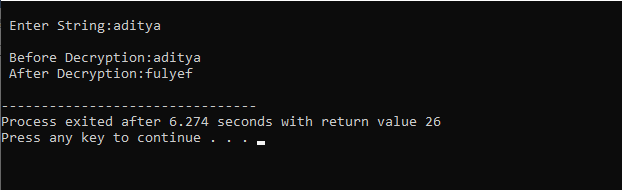
if (a == alpha[i][0])

return alpha[i][1];

}

}

**Output:**



* 1. **Write a C Program to implement vernam cipher.**

**AIM:** To write a C Program to implement vernam cipher.

**DESCRIPTION:**

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.

# Encrption 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.
* E (Pi , Ki) = Pi (XOR) Ki

# EXAMPLE:

**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

**Decryption Process**

The process of decrypting the ciphertext to convert it back into plain text is performed in the same way as the encryption process. Therefore, the formula for decryption of the text under Vernam cipher is as follows,

D (Ci , Ki) = Ci (XOR) Ki

**Program:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void encrypt(char \*plaintext, char \*key,char \*ciphertext)

{

int i;

for(i=0; i<strlen(plaintext);i++)

{

ciphertext[i] = plaintext[i] ^ key[i];

}

}

void decrypt(char \*ciphertext , char \*key , char \*plaintext)

{

int i;

for(i=0; i<strlen(ciphertext);i++)

{

plaintext[i] = ciphertext[i] ^ key[i];

}

}

int main(int argc, char \*argv[])

{

char plaintext[100];

char key[100];

char ciphertext[100];

printf("Enter Plaintext ");

scanf("%s",plaintext);

printf("Enter Key ");

scanf("%s",key);

encrypt(plaintext,key,ciphertext);

printf("Ciphertext : %s\n",ciphertext);

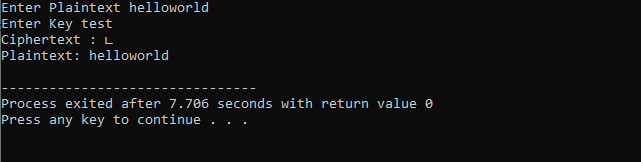
decrypt(ciphertext,key,plaintext);

printf("Plaintext: %s\n",plaintext);

return 0;

}

**Output:**

****

* 1. **Write a C Program to implement RSA algorithm.**

**AIM:** To write a C Program to implement RSA algorithm.

**DESCRIPTION:**

RSA encryption algorithm is a type of public-key encryption algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

# Public key encryption algorithm:

RSA is the common public-key algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**Public Key encryption algorithm is also called the Asymmetric algorithm. Asymmetric algorithms are those algorithms in which sender and receiver use different keys for encryption and decryption. Each sender is assigned a pair of keys:

# Public key 2.Private key

The **Public key** is used for encryption, and the **Private Key** is used for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using user's public key. But only the user can decrypt the message using his private key.

The RSA algorithm holds the following features −

* RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
* The integers used by this method are sufficiently large making it difficult to solve.
* There are two sets of keys in this algorithm: private key and public key.

**Encryption Formula**

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

Ciphertext = Pe mod n

**Decryption Formula**

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as –

Plaintext = Cd mod n

**Program:**

#include<stdio.h>

#include<math.h>

//to find gcd

int gcd(int a, int h)

{

int temp;

while(1)

{

temp = a%h;

if(temp==0)

return h;

a = h;

h = temp;

}

}

int main()

{

//2 random prime numbers

double p = 3;

double q = 7;

double n=p\*q;

double count;

double totient = (p-1)\*(q-1);

//public key

//e stands for encrypt

double e=2;

//for checking co-prime which satisfies e>1

while(e<totient){

count = gcd(e,totient);

if(count==1)

break;

else

e++;

}

//private key

//d stands for decrypt

double d;

//k can be any arbitrary value

double k = 2;

//choosing d such that it satisfies d\*e = 1 + k \* totient

d = (1 + (k\*totient))/e;

double msg = 12;

double c = pow(msg,e);

double m = pow(c,d);

c=fmod(c,n);

m=fmod(m,n);

printf("Message data = %lf",msg);

printf("\np = %lf",p);

printf("\nq = %lf",q);

printf("\nn = pq = %lf",n);

printf("\ntotient = %lf",totient);

printf("\ne = %lf",e);

printf("\nd = %lf",d);

printf("\nEncrypted data = %lf",c);

printf("\nOriginal Message Sent = %lf",m);

return 0;

}

**Output:**

