**CRYPTO PRACTICALS:**

**1.DH key**

**Program:**

#include<stdio.h>

#include<cmath>

int main()

{

int q=11,g=2;

int xa,xb;

int ya,yb;

int ka,kb;

printf("enter the private key of sender\n");

scanf("%d",&xa);

printf("enter the private key of receiver\n");

scanf("%d",&xb);

int a=pow(g,xa);

ya=a % q;

int b=pow(g,xb);

yb=b % q;

printf("public key of sender is:%d\n",ya);

printf("public key of reciever is:%d\n",yb);

int c=pow(ya,xb);

int d=pow(yb,xa);

ka=c % q;

kb=d % q;

if(ka == kb)

{

printf("secret key of the sender and reciever is:%d\n",ka);

}

else

{

printf("secret key is not generated");

}

return 0;

}

**2.RSA:**

**Program:**

#include<stdio.h>

#include<cmath>

int main()

{

int p=3,q=11;

int n,x,y;

int e,d;

n=p\*q;

x=(p-1)\*(q-1);

for(int i=2;i<20;i++)

{

if(x % i!=0)

{

e=i;

break;

}

}

printf("public key is:%d %d\n",e,n);

for(int i=2;i<100;i++)

{

if((e\*i) % x ==1)

{

d=i;

break;

}

}

printf("private key is:%d %d\n",d,n);

int pt=7,ct;

int en,de;

int m=pow(pt,e);

en=m % n;

int l=pow(en,d);

de=l % n;

printf("encryption is:%d\n",en);

printf("decryption is:%d\n",de);

return 0;

}

**3.CAESAR CIPHER:**

**Program:**

#include <stdio.h>

#include <string.h>

void encryptCaesarCipher(char message[], int key)

{

int i;

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

{

char character = message[i];

if (character >= 'a' && character <= 'z')

{

character = 'a' + (character - 'a' + key) % 26;

} else if (character >= 'A' && character <= 'Z')

{

character = 'A' + (character - 'A' + key) % 26;

}

message[i] = character;

}

}

void decryptCaesarCipher(char message[], int key)

{

encryptCaesarCipher(message, 26 - key);

}

int main()

{

char message[100];

int key;

printf("Enter a message: ");

gets(message);

printf("Enter the key (shift): ");

scanf("%d", &key);

encryptCaesarCipher(message, key);

printf("Encrypted message: %s\n", message);

decryptCaesarCipher(message, key);

printf("Decrypted message: %s\n", message);

return 0;

}

**4.MONOALPHABETIC CIPHER:**

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define ALPHABET\_SIZE 26

char encrypt(char letter, char key[]) {

if (isalpha(letter)) {

char base = isupper(letter) ? 'A' : 'a';

return key[letter - base];

}

return letter;

}

void monoalphabeticEncrypt(char message[], char key[]) {

int i = 0;

while (message[i] != '\0') {

message[i] = encrypt(message[i], key);

i++;

}

}

int main() {

char message[1000];

char key[ALPHABET\_SIZE + 1] = "ZYXWVUTSRQPONMLKJIHGFEDCBA"; // Example key (reverse alphabet)

printf("Enter the message to encrypt: ");

fgets(message, sizeof(message), stdin);

printf("Original message: %s\n", message);

monoalphabeticEncrypt(message, key);

printf("Encrypted message: %s\n", message);

return 0;

}

5.**POSSIBLE KEYS(11)**

Program:

#include <stdio.h>

#include <math.h>

unsigned long long factorial(int n) {

unsigned long long fact = 1;

for(int i = 2; i <= n; i++) {

fact \*= i;

}

return fact;

}

int main() {

int n = 25;

unsigned long long fact = factorial(n);

double log2\_fact = log2(fact);

unsigned long long approx\_power\_of\_2 = round(log2\_fact);

printf("The number of possible keys expressed as an approximate power of 2, is 2^%llu.\n", approx\_power\_of\_2);

return 0;

}

**6.Recoverd key(13)**

**Program:**

#include <stdio.h>

#define MOD 26

void encryptVector(int key[][2], int plaintext[], int encrypted[])

{

encrypted[0] = (key[0][0] \* plaintext[0] + key[0][1] \* plaintext[1]) % MOD;

encrypted[1] = (key[1][0] \* plaintext[0] + key[1][1] \* plaintext[1]) % MOD;

if (encrypted[0] < 0) encrypted[0] += MOD;

if (encrypted[1] < 0) encrypted[1] += MOD;

}

int main()

{

int plaintext1[2] = {1, 3};

int ciphertext1[2] = {2, 3};

int plaintext2[2] = {1, 2};

int ciphertext2[2] = {1, 4};

int key[2][2];

key[0][0] = (ciphertext1[0] - ciphertext2[0] + MOD) % MOD;

key[0][1] = (ciphertext1[1] - ciphertext2[1] + MOD) % MOD;

key[1][0] = (plaintext1[0] \* key[0][0] + plaintext1[1] \* key[0][1]) % MOD;

key[1][1] = (plaintext2[0] \* key[0][0] + plaintext2[1] \* key[0][1]) % MOD;

if (key[1][0] < 0) key[1][0] += MOD;

if (key[1][1] < 0) key[1][1] += MOD;

printf("Recovered key:\n");

printf("%d %d\n", key[0][0], key[0][1]);

printf("%d %d\n", key[1][0], key[1][1]);

return 0;

}

7.**VIGENERE CIPHER:**

Program:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

void encrypt(char \*plaintext, const int \*key, int key\_length) {

int len = strlen(plaintext);

for (int i = 0; i < len; i++) {

if (isalpha(plaintext[i])) {

int shift = key[i % key\_length];

if (isupper(plaintext[i])) {

plaintext[i] = (plaintext[i] - 'A' + shift) % 26 + 'A';

} else {

plaintext[i] = (plaintext[i] - 'a' + shift) % 26 + 'a';

}

}

}

}

int main() {

char plaintext[] = "Meet me at the usual place at ten rather than eight o'clock";

int key[] = {3, 19, 5};

int key\_length = sizeof(key) / sizeof(key[0]);

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

printf("Key: ");

for (int i = 0; i < key\_length; i++) {

printf("%d ", key[i]);

}

printf("\n");

encrypt(plaintext, key, key\_length);

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

return 0;

}