**CRYPTOGRAPHY LAB**

1. CAESAR CIPHER

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int main()

{

char plain[10],cipher[10];

int i,key,length;

printf("enter the plain text\n");

scanf("%s",plain);

printf("enter the key vaue\n");

scanf("%d",&key);

printf("\n\n\t PLAIN TEXT:%s",plain);

printf("\n\n\t ENCRYPTION: ");

for(i=0,length=strlen(plain);i<length;i++)

{

cipher[i]=plain[i]+key;

if(isupper(plain[i])&&(cipher[i]>'Z'))

cipher[i]=cipher[i]+26;

if(islower(plain[i])&&(cipher[i]>'z'))

cipher[i]=cipher[i]+26;

printf("%c",cipher[i]);

}

printf("\n\n\n DECRYPTION:");

for(i=0;i<length;i++)

{

plain[i]=cipher[i]-key;

if(isupper(cipher[i])&&(plain[i]<'A'))

plain[i]=plain[i]-26;

if(islower(cipher[i])&&(cipher[i]<'a'))

plain[i]=plain[i]-26;

printf("%c",plain[i]);

}

return 0;

}

OUTPUT:

/tmp/A5YY7VGIv5.o

enter the plain text

sushma

enter the key vaue

3

PLAIN TEXT:sushma

ENCRYPTION: vxvkpd

DECRYPTION:sushma

1. PLAYFAIR CIPHER

#include <stdio.h>

#include <string.h>

#define SIZE 5

void prepareKeyMatrix(char key[], char keyMatrix[SIZE][SIZE]) {

char alphabet[26] = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

int i, j, k, len, flag = 0;

len = strlen(key);

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

for (j = 0; j < SIZE; j++) {

for (k = 0; k < SIZE; k++) {

if (keyMatrix[j][k] == key[i] || (key[i] == 'I' && keyMatrix[j][k] == 'J') || (key[i] == 'J' && keyMatrix[j][k] == 'I')) {

flag = 1;

break;

}

}

if (flag == 1) {

flag = 0;

continue;

}

keyMatrix[i / SIZE][i % SIZE] = key[i];

}

}

for (i = len; i < SIZE \* SIZE; i++) {

for (j = 0; j < SIZE; j++) {

for (k = 0; k < SIZE; k++) {

if (keyMatrix[j][k] == alphabet[i - len] || (alphabet[i - len] == 'I' && keyMatrix[j][k] == 'J') || (alphabet[i - len] == 'J' && keyMatrix[j][k] == 'I')) {

flag = 1;

break;

}

}

if (flag == 1) {

flag = 0;

continue;

}

keyMatrix[i / SIZE][i % SIZE] = alphabet[i - len];

}

}

}

void encrypt(char keyMatrix[SIZE][SIZE], char plaintext[]) {

int i, j, row1, col1, row2, col2;

char ciphertext[strlen(plaintext)];

for (i = 0; i < strlen(plaintext); i += 2) {

for (j = 0; j < SIZE; j++) {

for (int k = 0; k < SIZE; k++) {

if (keyMatrix[j][k] == plaintext[i] || (plaintext[i] == 'I' && keyMatrix[j][k] == 'J') || (plaintext[i] == 'J' && keyMatrix[j][k] == 'I')) {

row1 = j;

col1 = k;

}

if (keyMatrix[j][k] == plaintext[i + 1] || (plaintext[i + 1] == 'I' && keyMatrix[j][k] == 'J') || (plaintext[i + 1] == 'J' && keyMatrix[j][k] == 'I')) {

row2 = j;

col2 = k;

}

}

}

if (row1 == row2) {

ciphertext[i] = keyMatrix[row1][(col1 + 1) % SIZE];

ciphertext[i + 1] = keyMatrix[row2][(col2 + 1) % SIZE];

} else if (col1 == col2) {

ciphertext[i] = keyMatrix[(row1 + 1) % SIZE][col1];

ciphertext[i + 1] = keyMatrix[(row2 + 1) % SIZE][col2];

} else {

ciphertext[i] = keyMatrix[row1][col2];

ciphertext[i + 1] = keyMatrix[row2][col1];

}

}

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

}

int main() {

char key[25], plaintext[100];

char keyMatrix[SIZE][SIZE];

printf("Enter the key (no spaces, all uppercase): ");

scanf("%s", key);

printf("Enter the plaintext (no spaces, all uppercase): ");

scanf("%s", plaintext);

prepareKeyMatrix(key, keyMatrix);

encrypt(keyMatrix, plaintext);

return 0;

}

OUTPUT:

/tmp/xHf7xRTeCw.o

Enter the key (no spaces, all uppercase): SUSHMA

Enter the plaintext (no spaces, all uppercase): CRYPTOGRAPHY

Encrypted Text: BSASMTMSAQES

3.HILL CIPHER

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main(){

unsigned int a[3][3]={{6,24,1},{13,16,10},{20,17,15}};

unsigned int b[3][3]={{8,5,10},{21,8,21},{21,12,8}};

int i,j, t=0;

unsigned int c[20],d[20];

char msg[20];

printf("Enter plain text\n ");

scanf("%s",msg);

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

{

c[i]=msg[i]-65;

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

}

for(i=0;i<3;i++)

{ t=0;

for(j=0;j<3;j++)

{

t=t+(a[i][j]\*c[j]);

}

d[i]=t%26;

}

printf("\nEncrypted Cipher Text :");

for(i=0;i<3;i++)

printf(" %c",d[i]+65);

for(i=0;i<3;i++)

{

t=0;

for(j=0;j<3;j++)

{

t=t+(b[i][j]\*d[j]);

}

c[i]=t%26;

}

printf("\nDecrypted Cipher Text :");

for(i=0;i<3;i++)

printf(" %c",c[i]+65);

return 0;

}

OUTPUT:

Enter plain text

SUSHMA

18 20 18 7 12 0

Encrypted Cipher Text : I G I

Decrypted Cipher Text : S U S

--------------------------------

Process exited after 4.972 seconds with return value 0

4.VIGNERE CIPHER#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

void vigenereEncrypt(char \*plaintext, char \*key) {

int i, j;

int plaintextLen = strlen(plaintext);

int keyLen = strlen(key);

for (i = 0, j = 0; i < plaintextLen; ++i, ++j) {

if (j == keyLen) {

j = 0;

}

if (isupper(plaintext[i])) {

plaintext[i] = ((plaintext[i] + key[j] - 2 \* 'A') % 26) + 'A';

}

else if (islower(plaintext[i])) {

plaintext[i] = ((plaintext[i] + key[j] - 2 \* 'a') % 26) + 'a';

}

}

}

void vigenereDecrypt(char \*ciphertext, char \*key) {

int i, j;

int ciphertextLen = strlen(ciphertext);

int keyLen = strlen(key);

for (i = 0, j = 0; i < ciphertextLen; ++i, ++j) {

if (j == keyLen) {

j = 0;

}

if (isupper(ciphertext[i])) {

ciphertext[i] = ((ciphertext[i] - key[j] + 26) % 26) + 'A';

}

else if (islower(ciphertext[i])) {

ciphertext[i] = ((ciphertext[i] - key[j] + 26) % 26) + 'a';

}

}

}

int main() {

char input[100];

char key[100];

printf("Enter text: ");

scanf("%99[^\n]", input);

getchar();

printf("Enter key: ");

scanf("\n", key);

getchar();

vigenereEncrypt(input, key);

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

vigenereDecrypt(input, key);

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

return 0;

}

OUTPUT:

/tmp/q3YUh9rdoo.o

Enter text: sushma

Enter key: sai

Encrypted text: RKRXLQ

Decrypted text: SUSHMA

5.RAIL FENCE TECHNIQU#include<stdio.h>

#include<string.h>

void main()

{

int i,j,k,l;

char a[20],c[20],d[20];

printf("\n\t\t RAIL FENCE TECHNIQUE");

printf("\n\nEnter the input string : ");

gets(a);

l=strlen(a);

for(i=0,j=0;i<l;i++)

{

if(i%2==0)

c[j++]=a[i];

}

for(i=0;i<l;i++)

{

if(i%2==1)

c[j++]=a[i];

}

c[j]='\0';

printf("\nCipher text after applying rail fence :");

printf("\n%s",c);

if(l%2==0)

k=l/2;

else

k=(l/2)+1;

for(i=0,j=0;i<k;i++)

{

d[j]=c[i];

j=j+2;

}

for(i=k,j=1;i<l;i++)

{

d[j]=c[i];

j=j+2;

}

d[l]='\0';

printf("\nText after decryption : ");

printf("%s",d);

}

OUTPUT:

/tmp/xxfNDe0LiX.o

RAIL FENCE TECHNIQUE

Enter the input string : sushma

Cipher text after applying rail fence :

ssmuha

Text after decryption : sushma

6.MONOALPHABETIC CIPHER

#include<stdio.h>

int main(){

char alpha[100]="abcdefghijklmnopqrstuvwxyz",key[100]="zyxwvutsrqponmlkjihgfedcba",plain[100],cipher[100];

int m=0,index[100];

printf("Enter plain text :");

scanf("%s",&plain);

for(int i=0;i<strlen(plain);i++){

for(int j=0;j<strlen(alpha);j++){

if(plain[i]==alpha[j]){

index[m]=j;

m++;

}

}

}

printf("Cipher text: ");

for(int i=0;i<strlen(plain);i++){

cipher[i]=key[index[i]];

printf("%c",cipher[i]);

}

printf("\n Plain text : ");

for(int i=0;i<strlen(plain);i++){

plain[i]=alpha[index[i]];

printf("%c",plain[i]);

}

}

OUTPUT:

/tmp/c8yI20P1O6.o

Enter plain text :sushma

Cipher text: hfhsnz

Plain text : sushma

7.RSA ALGORITHM

#include<stdio.h>

#include<math.h>

int main(){

int c,p,q,n,n1,i,j,m=5,result=0,d[1000],result2=0,temp;

printf("Enter a value : ");

scanf("%d",&p);

printf("Enter another value : ");

scanf("%d",&q);

n=p\*q;

printf("Value of n = %d\n",n);

n1=(p-1)\*(q-1);

printf("Value of n1 = %d\n",n1);

int e[10]={3,5,7,11,13,17};

for(i=0;i<e[i];i++){

if(n1%e[i]==0){

result=e[i];

break;

}

}

printf("The value of e is %d\n",result);

for(i=0;i<e[i] && result2!=1;i++){

for(j=1;j<1000;j++){

result2=(j\*e[i])%n1;

if(result2==1){

break;

}

}

printf("The value of d is %d\n",j);

}

temp=(int)pow(m,result);

c=temp%n;

printf("Encrypted value : %d",c);

}

OUTPUT:

/tmp/s7EWyY7qSt.o

Enter a value : 7

Enter another value : 11

Value of n = 77

Value of n1 = 60

The value of e is 3

The value of d is 1000

The value of d is 1000

The value of d is 43

Encrypted value : 48

8.DIFFIE-HELLIEMAN ALGORITHM

#include<stdio.h>

#include<math.h>

int main(){

int q,b,Xa,Xb,Ya,Yb,K1,K2,temp1,temp2,temp3,temp4;

printf("Enter the value of q : ");

scanf("%d",&q);

printf("Enter the value of alpha : ");

scanf("%d",&b);

printf("Enter the value of Xa : ");

scanf("%d",&Xa);

printf("Enter the value of Xb : ");

scanf("%d",&Xb);

temp1=(pow(b,Xa));

Ya=temp1%q;

printf("Ya = %d\n",Ya);

temp2=(pow(b,Xb));

Yb=temp2%q;

printf("Yb = %d\n",Yb);

temp3=(pow(Yb,Xa));

K1=temp3%q;

temp4=(pow(Ya,Xb));

K2=temp4%q;

if(K1==K2){

printf("The value of K = %d",K1);

}

return 0;

}

OUTPUT:

/tmp/s7EWyY7qSt.o

Enter the value of q : 17

Enter the value of alpha : 5

Enter the value of Xa : 3

Enter the value of Xb : 4

Ya = 6

Yb = 13

The value of K = 4

9.SHA ALGORITHM

#include <stdio.h>

#include <stdint.h>

#include <string.h>

// MD5 constants

#define MD5\_BLOCK\_SIZE 64

// MD5 functions

#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))

#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))

#define H(x, y, z) ((x) ^ (y) ^ (z))

#define I(x, y, z) ((y) ^ ((x) | (~z)))

// Left-rotate operation

#define LEFT\_ROTATE(x, n) (((x) << (n)) | ((x) >> (32 - (n))))

// MD5 state

typedef struct {

uint32\_t A, B, C, D,E;

} MD5\_STATE;

void md5\_transform(uint32\_t state[4], const uint8\_t block[64]) {

uint32\_t a = state[0];

uint32\_t b = state[1];

uint32\_t c = state[2];

uint32\_t d = state[3];

uint32\_t e = state[4];

uint32\_t x[20];

int i;

for ( i = 0; i < 20; i++)

x[i] = \* (uint32\_t\*)(block + i \* 5);

// Round 1

for ( i = 0; i < 20; i++) {

uint32\_t temp = F(b, c, d) + x[i] + 0x5A827999 + a;

a = d;

d = c;

c = b;

d = a;

a = a + LEFT\_ROTATE(temp, 5);

}

// Round 2

for ( i = 0; i < 20; i++) {

uint32\_t temp = G(b, c, d) + x[(5 \* i + 1) % 16] + 0x6ED9EBA1 + a;

a = d;

d = c;

c = b;

d = a;

a = a + LEFT\_ROTATE(temp, 5);

}

// Round 3

for ( i = 0; i < 20; i++) {

uint32\_t temp = H(b, c, d) + x[(3 \* i + 5) % 16] + 0x8F1BBCDC + a;

a = d;

d = c;

c = b;

a = d;

a = a + LEFT\_ROTATE(temp, 5);

}

// Round 4

for ( i = 0; i < 20; i++) {

uint32\_t temp = I(b, c, d) + x[(7 \* i) % 16] + 0xCA62C1D6 + a;

a = d;

d = c;

c = b;

a = d;

a = a + LEFT\_ROTATE(temp, 5);

}

state[0] += a;

state[1] += b;

state[2] += c;

state[3] += d;

state[4] += e;

}

void md5\_hash(const uint8\_t \*data, size\_t length, uint8\_t hash[16]) {

MD5\_STATE state;

state.A = 0x67452301;

state.B = 0xEFCDAB89;

state.C = 0x98BADCFE;

state.D = 0x10325476;

state.E = 0x765431AB;

size\_t block\_count = length / MD5\_BLOCK\_SIZE;

size\_t i;

for (i = 0; i < block\_count; i++) {

md5\_transform((uint32\_t\*)&state, data + i \* MD5\_BLOCK\_SIZE);

}

memcpy(hash, &state, 20);

}

int main() {

const char \*input = "sushma";

uint8\_t hash[20];

int i;

md5\_hash((uint8\_t\*)input, strlen(input), hash);

printf("Input: %s\n", input);

printf("SHA Hash: ");

for (i = 0; i < 20; i++) {

printf("%02x", hash[i]);

}

printf("\n");

}

OUTPUT:

/tmp/9AOijgNuIz.o

Input: sushma

SHA Hash: 0123456789abcdeffedcba9876543210ab315476

11.COLUMUNAR TRANSPOSITION CIPHER

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

void gridStr(char \*line, char \*key);

char encrypt(char \*key);

int main() {

char key[50];

char line[256];

printf("Enter your string: ");

if (fgets(line, sizeof line, stdin) == NULL) {

fprintf(stderr, "No line read\n");

exit(EXIT\_FAILURE);

}

printf("Enter your key: ");

if (fgets(key, sizeof key, stdin) == NULL) {

fprintf(stderr, "No line read\n");

exit(EXIT\_FAILURE);

}

int len = strlen(line);

if (len && line[len - 1] == '\n')

line[--len] = '\0';

int len1 = strlen(key);

if (len1 && key[len1 - 1] == '\n')

key[--len]= '\0';

gridStr(line, key);

encrypt(key);

}

void gridStr(char \*line, char \*key)

{

char mat[10][10] = {0};

int columns = strlen(key)-1;

int rows = 0;

int i=0,j = 0;

while (line[i]) {

if (line[i] == ' ') {

putchar('\_');

} else {

putchar(line[i]);

}

mat[rows][i % columns] = line[i];

i++;

if (i > 0 && i % columns == 0) {

putchar('\n');

rows++;

}

}

if (i % columns != 0) putchar('\n');

rows++; // from current row to number of rows

printf("\nMatrix:\n");

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

for (j = 0; j < columns; j++) {

if (mat[i][j] == ' ') {

putchar('\_');

} else {

putchar(mat[i][j]);

}

}

printf("\n");

}

}

char encrypt(char \*key){

char temp;

int i,j;

int n = strlen(key);

for (i = 0; i < n-1; i++) {

for (j = i+1; j < n; j++) {

if (key[i] > key[j]) {

temp = key[i];

key[i] = key[j];

key[j] = temp;

}

}

}

printf("encrypted text : %s", key);

printf("decrypted text :%s");

return 0;

}