1. DIFFE HELLMAN

#include<stdio.h>

#include<string.h>

#include<math.h>

int main(){

int a,q,xa,xb,ya,yb,x,y,ka,kb,m,n;

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

scanf("%d",&a);

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

scanf("%d",&q);

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

scanf("%d",&xa);

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

scanf("%d",&xb);

x=pow(a,xa);

ya=x%q;

y=pow(a,xb);

yb=y%q;

m=pow(yb,xa);

ka=m%q;

n=pow(ya,xb);

kb=n%q;

printf("Secret key of user A : %d ", ka);

printf("\nSecret key of user B : %d ", kb);

}

2.HILL CIPHER

#include<stdio.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;

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

char msg[20];

int determinant = 0, t = 0;

printf("Enter plain text: ");

scanf("%s", msg);

for (i = 0; i < 3; 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;

}

3.MD5

#include <stdio.h>

#include <stdint.h>

#include <string.h>

#define MD5\_BLOCK\_SIZE 64

#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)))

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

typedef struct {

uint32\_t A, B, C, D;

} 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 x[16];

int i;

for ( i = 0; i < 16; i++)

x[i] = \* (uint32\_t\*)(block + i \* 4);

// Round 1

for ( i = 0; i < 16; i++) {

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

a = d;

d = c;

c = b;

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

}

// Round 2

for ( i = 0; i < 16; i++) {

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

a = d;

d = c;

c = b;

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

}

// Round 3

for ( i = 0; i < 16; i++) {

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

a = d;

d = c;

c = b;

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

}

// Round 4

for ( i = 0; i < 16; i++) {

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

a = d;

d = c;

c = b;

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

}

state[0] += a;

state[1] += b;

state[2] += c;

state[3] += d;

}

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;

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, 16);

}

int main() {

const char \*input = "Hello, MD5!";

uint8\_t hash[16];

int i;

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

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

printf("MD5 Hash: ");

for (i = 0; i < 16; i++) {

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

}

printf("\n");

return 0;

}

4.PLAYCIPHER

#include<stdio.h>

int check(char table[5][5], char k) {

int i, j;

for (i = 0; i < 5; ++i)

for (j = 0; j < 5; ++j) {

if (table[i][j] == k)

return 0;

}

return 1;

}

int main() {

int i, j, key\_len;

char table[5][5];

for (i = 0; i < 5; ++i)

for (j = 0; j < 5; ++j)

table[i][j] = '0';

printf("\*\*\*Playfair Cipher\*\*\*\*\*\n\n");

printf("Enter the length of the Key. ");

scanf("%d", &key\_len);

char key[key\_len];

printf("Enter the Key. ");

for (i = -1; i < key\_len; ++i) {

scanf("%c", &key[i]);

if (key[i] == 'j')

key[i] = 'i';

}

int flag;

int count = 0;

for (i = 0; i < 5; ++i) {

for (j = 0; j < 5; ++j) {

flag = 0;

while (flag != 1) {

if (count > key\_len)

goto l1;

flag = check(table, key[count]);

++count;

}

table[i][j] = key[(count - 1)];

}

}

l1: printf("\n");

int val = 97;

for (i = 0; i < 5; ++i) {

for (j = 0; j < 5; ++j) {

if (table[i][j] >= 97 && table[i][j] <= 123) {

} else {

flag = 0;

while (flag != 1) {

if ('j' == (char) val)

++val;

flag = check(table, (char) val);

++val;

}

table[i][j] = (char) (val - 1);

}

}

}

printf("The table is as follows:\n");

for (i = 0; i < 5; ++i) {

for (j = 0; j < 5; ++j) {

printf("%c ", table[i][j]);

}

printf("\n");

}

int l = 0;

printf("\nEnter the length length of plain text.(without spaces) ");

scanf("%d", &l);

printf("\nEnter the Plain text. ");

char p[l];

for (i = -1; i < l; ++i) {

scanf("%c", &p[i]);

}

for (i = -1; i < l; ++i) {

if (p[i] == 'j')

p[i] = 'i';

}

printf("\nThe replaced text(j with i)");

for (i = -1; i < l; ++i)

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

count = 0;

for (i = -1; i < l; ++i) {

if (p[i] == p[i + 1])

count = count + 1;

}

printf("\nThe cipher has to enter %d bogus char.It is either 'x' or 'z'\n",

count);

int length = 0;

if ((l + count) % 2 != 0)

length = (l + count + 1);

else

length = (l + count);

printf("\nValue of length is %d.\n", length);

char p1[length];

char temp1;

int count1 = 0;

for (i = -1; i < l; ++i) {

p1[count1] = p[i];

if (p[i] == p[i + 1]) {

count1 = count1 + 1;

if (p[i] == 'x')

p1[count1] = 'z';

else

p1[count1] = 'x';

}

count1 = count1 + 1;

}

char bogus;

if ((l + count) % 2 != 0) {

if (p1[length - 1] == 'x')

p1[length] = 'z';

else

p1[length] = 'x';

}

printf("The final text is:");

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

printf("%c ", p1[i]);

char cipher\_text[length];

int r1, r2, c1, c2;

int k1;

for (k1 = 1; k1 <= length; ++k1) {

for (i = 0; i < 5; ++i) {

for (j = 0; j < 5; ++j) {

if (table[i][j] == p1[k1]) {

r1 = i;

c1 = j;

} else if (table[i][j] == p1[k1 + 1]) {

r2 = i;

c2 = j;

}

}

}

if (r1 == r2) {

cipher\_text[k1] = table[r1][(c1 + 1) % 5];

cipher\_text[k1 + 1] = table[r1][(c2 + 1) % 5];

}

else if (c1 == c2) {

cipher\_text[k1] = table[(r1 + 1) % 5][c1];

cipher\_text[k1 + 1] = table[(r2 + 1) % 5][c1];

} else {

cipher\_text[k1] = table[r1][c2];

cipher\_text[k1 + 1] = table[r2][c1];

}

k1 = k1 + 1;

}

printf("\n\nThe Cipher text is:\n ");

for (i = 1; i <= length; ++i)

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

return 0;

}

6.RSA

#include<stdio.h>

#include<string.h>

#include<math.h>

int main(){

int p,q,m,n,dn,e,c,de,x,y;

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

scanf("%d",&p);

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

scanf("%d",&q);

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

scanf("%d",&m);

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

scanf("%d",&e);

n=p\*q;

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

int d;

for(int i=1;i<dn;i++){

if(((e%dn)\*(i%dn))%dn==1){

d=i;

break;

}

}

x=pow(m,e);

c=x%n;

y=pow(c,d);

de=y%n;

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

printf("\nDecrypted text : %d ", m);

}

7. Railfence Transposition

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

char \*plainTextToCipherText(char plainText[],int n)

{

int i,j,counter,limit,index=0,len;

char \*cipherText;

len=strlen(plainText);

cipherText=(char\*)malloc(sizeof(char)\*(len+1));

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

{

counter=0;

for(j=i;j<len;j+=limit)

{

cipherText[index++]=plainText[j];

if(i==0 || i==n-1)

limit=2\*n-2;

else if(counter%2==0)

limit=2\*(n-i-1);

else

limit=2\*i;

if(limit<=0)

break;

counter++;

}

}

cipherText[index]='\0';

return cipherText;

}

int main()

{

int n;

char plainText[100];

printf("Enter the plain text : ");

scanf("%s",plainText);

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

scanf("%d",&n);

printf("%s\n",plainTextToCipherText(plainText,n));

return 0;

}

8.SHA

#include <stdio.h>

#include <stdint.h>

#include <string.h>

void compute\_sha1(const char \*input, size\_t input\_len, uint32\_t hash[5])

{

}

int main()

{

char message[1024];

printf("Enter the message: ");

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

message[strcspn(message, "\n")] = '\0';

uint32\_t hash[5];

size\_t message\_len = strlen(message);

compute\_sha1(message, message\_len, hash);

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

printf("SHA-1 Hash: ");

for (int i = 0; i < 5; i++)

{

printf("%08x", hash[i]);

}

printf("\n");

return 0;

}

9. [monoalphabetic cipher](https://github.com/KSaiDharaneeswaraReddy/CSA5103-Cryptography-and-Network-Security/commit/cbf307e028c9ef144c82bd559d9fc74621d0e6a8)

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int main() {

char message[500], key[27];

printf("Enter the substitution key (26 unique lowercase letters):\n ");

printf("enter the plain text value:");

fgets(key, sizeof(key), stdin);

key[strlen(key) - 1] = '\0';

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

message[strlen(message) - 1] = '\0';

for (int i = 0; message[i]; ++i) {

if (isalpha(message[i])) {

if (isupper(message[i]))

message[i] = toupper(key[message[i] - 'A']);

else

message[i] = key[message[i] - 'a'];}}

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

}

10.POLYalphabetic cipher

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int main()

{

char m[500], k[100];

printf("enter the plaintext:");

fgets(m,500,stdin);

printf("enter the key:");

fgets(k,100,stdin);

for(int i=0,j=0;m[i];++i,j=(j+1)%strlen(k))

if(isalpha(m[i]))

m[i]=(m[i]-'a'+tolower(k[j])-'a')%26 +'a';

printf("Encrypted message: %s",m);

return 0;

}

1.CAESER CIPHER

#include<stdio.h>

#include<ctype.h>

int main() {

char text[500], ch;

int key;

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

scanf("%s", text);

printf("Enter the key: ");

scanf("%d", & key);

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

ch = text[i];

if (isalnum(ch)) {

if (islower(ch)) {

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

}

if (isupper(ch)) {

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

}

if (isdigit(ch)) {

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

}

}.

else {

printf("Invalid Message");

}

text[i] = ch;

}

printf("Encrypted message: %s", text);

return 0;

}

2.DES

#include <stdio.h>

#include <stdint.h>

void des\_encrypt(uint64\_t plainText, uint64\_t key, uint64\_t \*cipherText);

void print\_binary(uint64\_t num);

int main() {

uint64\_t plainText, key, cipherText;

printf("Enter the 64-bit plaintext: ");

scanf("%llx", &plainText);

printf("Enter the 64-bit encryption key:");

scanf("%llx", &key);

des\_encrypt(plainText, key, &cipherText);

printf("\nPlaintext: ");

print\_binary(plainText);

printf("\nKey: ");

print\_binary(key);

printf("\nCiphertext: ");

print\_binary(cipherText);

return 0;

}

void des\_encrypt(uint64\_t plainText, uint64\_t key, uint64\_t \*cipherText) {

\*cipherText = plainText;

}

void print\_binary (uint64\_t num) {

for (int i = 63; i >= 0; i--) {

uint64\_t bit = (num >> i) & 1;

printf("%llu", bit);

if (i % 8 == 0)

printf(" ");

}

printf("\n");

}

3.DSS

#include <stdio.h>

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int mod\_inverse(int a, int m) {

int m0 = m, t, q;

int x0 = 0, x1 = 1;

if (m == 1) return 0;

while (a > 1) {

q = a / m;

t = m;

m = a % m;

a = t;

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

if (x1 < 0) x1 += m0;

return x1;

}

int main() {

int p = 61;

int q = 31;

int alpha = 6;

int private\_key;

printf("Enter your private key: ");

scanf("%d", &private\_key);

int public\_key = 1;

int i;

for (i = 0; i < private\_key; i++) {

public\_key = (public\_key \* alpha) % p;

}

printf("Public Key: %d\n", public\_key);

int message;

printf("Enter the message to be signed: ");

scanf("%d", &message);

int k;

printf("Enter a random value k: ");

scanf("%d", &k);

int r = 1;

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

r = (r \* alpha) % p;

}

r = r % q;

int s = (mod\_inverse(k, q) \* (message + private\_key \* r)) % q;

printf("Signature (r, s): (%d, %d)\n", r, s);

return 0;

}