**3.Write a Simple program for Encryption and Decryption.**

#include <stdio.h>

int main() {

int i, x = 0; // Initialize x

char str[100];

printf("\nPlease enter a string:\t");

fgets(str, sizeof(str), stdin); // Use fgets instead of gets

printf("\nPlease choose following options:\n");

printf("1 = Encrypt the string.\n");

printf("2 = Decrypt the string.\n");

scanf("%d", &x);

// using switch case statements

switch(x) { // Correctly place the opening curly brace

case 1:

for(i = 0; (i < 100 && str[i] != '\0'); i++)

str[i] = str[i] + 3; // the key for encryption is 3 that is added to ASCII value

printf("\nEncrypted string: %s\n", str);

break;

case 2:

for(i = 0; (i < 100 && str[i] != '\0'); i++)

str[i] = str[i] - 3; // the key for encryption is 3 that is subtracted to ASCII value

printf("\nDecrypted string: %s\n", str);

break;

default:

printf("\nError\n");

}

return 0;

}

**2.Write a program for security of username and password.**

**Explain Encapsulating Security Payload [ESP]**

#include<stdio.h>

int main() {

char password[9], username[10], ch; // Increased password array size to accommodate null character

int i;

printf("Enter username: ");

gets(username);

printf("Enter the password (any 8 characters): ");

for(i = 0; i < 8; i++) { // Removed semicolon after for loop

ch = getchar();

password[i] = ch;

ch = '\*';

}

password[i] = '\0'; // Placed null character at the correct index

printf("\nYour password is: ");

for(i = 0; i < 8; i++) { // Removed semicolon after for loop

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

}

return 0;

}

1. **Write a program for security of username.**

**Explain Authentication Header**

#include<stdio.h>

#include<ctype.h> // Include ctype.h library for character functions

int main() {

char username[20]; // Increased username array size

int i;

printf("Enter username (up to 20 characters, no spaces): ");

scanf("%s", username); // Use scanf instead of gets for security

// Check if username meets criteria (no spaces and length <= 20)

int isValid = 1;

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

if (isspace(username[i])) { // Check for spaces

isValid = 0;

break;

}

}

if (isValid && i <= 20) { // Check if username is valid

printf("Username is valid.\n");

} else {

printf("Invalid username. Please make sure it contains no spaces and is up to 20 characters long.\n");

}

return 0;

}

**4.Write a program to perform encryption and using Ceaser Cipher Substitution technique.**

#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;

}

**5.Write a program to perform decryption and using Ceaser Cipher Substitution technique.**

# include<stdio.h>

#include<ctype.h>

int main()

{

char text[500],ch;

int key;

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

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)%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;

}

**6.Write a program for Implementation of Asymmetric cryptography using DH algorithm.**

#include<stdio.h>

#include<math.h>

// Function to calculate power

long long int power(long long int a, long long int b, long long int mod) {

if (b == 0)

return 1;

long long int temp = power(a, b / 2, mod);

temp = (temp \* temp) % mod;

if (b % 2 == 1)

temp = (temp \* a) % mod;

return temp;

}

int main() {

long long int p, g, a, b, A, B, secretKeyA, secretKeyB;

// Choose a large prime number 'p'

printf("Enter a prime number (p): ");

scanf("%lld", &p);

// Choose a primitive root modulo 'p'

printf("Enter a primitive root modulo (g): ");

scanf("%lld", &g);

// Private key for user A

printf("Enter private key for user A (a): ");

scanf("%lld", &a);

// Private key for user B

printf("Enter private key for user B (b): ");

scanf("%lld", &b);

// Calculate public keys for A and B

A = power(g, a, p);

B = power(g, b, p);

// Calculate secret keys for A and B

secretKeyA = power(B, a, p);

secretKeyB = power(A, b, p);

// Display calculated keys

printf("\nPublic key for user A: %lld\n", A);

printf("Public key for user B: %lld\n", B);

printf("Secret key for user A: %lld\n", secretKeyA);

printf("Secret key for user B: %lld\n", secretKeyB);

return 0;

}

**7.Write a program for subkey generation using DES algorithm.**

#include <stdio.h>

#include <stdint.h>

// Initial Permutation Table

const int initial\_permutation\_table[64] = {58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7};

// Key Permutation Table

const int key\_permutation\_table[56] = {57, 49, 41, 33, 25, 17, 9, 1,

58, 50, 42, 34, 26, 18, 10, 2,

59, 51, 43, 35, 27, 19, 11, 3,

60, 52, 44, 36, 63, 55, 47, 39,

31, 23, 15, 7, 62, 54, 46, 38,

30, 22, 14, 6, 61, 53, 45, 37,

29, 21, 13, 5, 28, 20, 12, 4};

// Key Schedule Left Shifts

const int key\_shifts[16] = {1, 1, 2, 2, 2, 2, 2, 2,

1, 2, 2, 2, 2, 2, 2, 1};

// Function to perform initial permutation

uint64\_t initial\_permutation(uint64\_t data) {

uint64\_t result = 0;

int i;

for (i = 0; i < 64; i++) {

result |= ((data >> (64 - initial\_permutation\_table[i])) & 1) << (63 - i);

}

return result;

}

// Function to perform key permutation and generate subkeys

void generate\_subkeys(uint64\_t key, uint64\_t \*subkeys) {

uint64\_t permuted\_key = 0;

int i;

// Apply key permutation

for (i = 0; i < 56; i++) {

permuted\_key |= ((key >> (64 - key\_permutation\_table[i])) & 1) << (55 - i);

}

// Split the key into two halves

uint32\_t left\_half = (uint32\_t)(permuted\_key >> 28);

uint32\_t right\_half = (uint32\_t)(permuted\_key & 0x0FFFFFFF);

// Generate subkeys

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

// Perform left circular shift on both halves

left\_half = ((left\_half << key\_shifts[i]) | (left\_half >> (28 - key\_shifts[i]))) & 0x0FFFFFFF;

right\_half = ((right\_half << key\_shifts[i]) | (right\_half >> (28 - key\_shifts[i]))) & 0x0FFFFFFF;

// Merge halves and perform permutation

uint64\_t merged\_key = ((uint64\_t)left\_half << 28) | right\_half;

subkeys[i] = 0;

// Apply key permutation for subkey

for (int j = 0; j < 48; j++) {

subkeys[i] |= ((merged\_key >> (56 - key\_permutation\_table[j])) & 1) << (47 - j);

}

}

}

int main() {

// Test initial permutation

uint64\_t plaintext = 0x000056789ABCD11; // 64-bit plaintext

printf("Plaintext: %016llx\n", plaintext);

uint64\_t permuted\_plaintext = initial\_permutation(plaintext);

printf("Initial Permuted Plaintext: %016llx\n", permuted\_plaintext);

// Test key generation

uint64\_t key = 0x133457799BBCDFF1; // 64-bit key

printf("Key: %016llx\n", key);

uint64\_t subkeys[16];

generate\_subkeys(key, subkeys);

int i;

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

printf("Subkey %d: %016llx\n", i + 1, subkeys[i]);

}

return 0;

}

**8.Write a program for Implementation of shift row module using AES algorithm.**

#include <stdio.h>

#include <stdint.h>

#define AES\_BLOCK\_SIZE 16

// ShiftRows: cyclically shift the rows of the state matrix

void ShiftRows(uint8\_t \*state) {

uint8\_t temp;

// Second row shift

temp = state[1];

state[1] = state[5];

state[5] = state[9];

state[9] = state[13];

state[13] = temp;

// Third row shift

temp = state[2];

state[2] = state[10];

state[10] = temp;

temp = state[6];

state[6] = state[14];

state[14] = temp;

// Fourth row shift

temp = state[3];

state[3] = state[15];

state[15] = state[11];

state[11] = state[7];

state[7] = temp;

}

// Function to print state matrix

void printState(uint8\_t \*state) {

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

printf("%02x ", state[i + 4 \* j]);

}

printf("\n");

}

printf("\n");

}

int main() {

uint8\_t state1[AES\_BLOCK\_SIZE] = {0x32, 0x88, 0x31, 0xe0, 0x43, 0x5a, 0x31, 0x37, 0xf6, 0x30, 0x98, 0x07, 0xa8, 0x8d, 0xa2, 0x34};

uint8\_t state2[AES\_BLOCK\_SIZE] = {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0, 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77};

printf("Name:- Yadav Harshdeep Sanjay\n");

printf("Roll no.:- TE-B 3260\n");

printf("\nState Matrix 1 before ShiftRows:\n");

printState(state1);

// ShiftRows operation for state1

ShiftRows(state1);

printf("State Matrix 1 after ShiftRows:\n");

printState(state1);

printf("\nState Matrix 2 before ShiftRows:\n");

printState(state2);

// ShiftRows operation for state2

ShiftRows(state2);

printf("State Matrix 2 after ShiftRows:\n");

printState(state2);

return 0;

}