**DES amlgoritham**

#include <stdio.h>

#include <stdint.h>

// Simplified Initial and Final Permutation Tables

int IP[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

};

int FP[64] = {

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

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

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

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

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

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

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

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

};

// Permutation function

uint64\_t permute(uint64\_t input, int\* table) {

uint64\_t output = 0;

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

output <<= 1;

output |= (input >> (64 - table[i])) & 1;

}

return output;

}

// Simplified Feistel function using XOR

uint32\_t feistel(uint32\_t half, uint64\_t key) {

return half ^ (uint32\_t)(key & 0xFFFFFFFF);

}

// One round of simplified DES

void des\_round(uint32\_t\* left, uint32\_t\* right, uint64\_t key) {

uint32\_t temp = \*right;

\*right = \*left ^ feistel(\*right, key);

\*left = temp;

}

// DES-like encrypt/decrypt process

uint64\_t des(uint64\_t block, uint64\_t key, int decrypt) {

block = permute(block, IP);

uint32\_t left = (block >> 32) & 0xFFFFFFFF;

uint32\_t right = block & 0xFFFFFFFF;

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

int round = decrypt ? 15 - i : i;

des\_round(&left, &right, key); // Using same key every round for simplicity

}

uint64\_t combined = ((uint64\_t)right << 32) | left;

return permute(combined, FP);

}

// Main function

int main() {

int choice;

uint64\_t message, key;

printf("DES Cipher (Simplified)\n");

printf("1. Encrypt\n2. Decrypt\nChoose (1 or 2): ");

scanf("%d", &choice);

printf("Enter 16-digit hex message (e.g., 0123456789ABCDEF): ");

scanf("%llx", &message);

printf("Enter 16-digit hex key (e.g., 133457799BBCDFF1): ");

scanf("%llx", &key);

uint64\_t result = des(message, key, choice == 2);

if (choice == 1)

printf("Encrypted: %016llX\n", result);

else

printf("Decrypted: %016llX\n", result);

return 0;

}

