**Experiment No: -05**

**Name:** Pravin Patil

**Roll No:**3138

**Div:** A

**SUB:** Network Security

**Code:**

#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};

// 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 generate subkeys

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

// Implement key generation here

}

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; }

**OUTPUT: -**

