Apurba Koirala

22BCE3799

Cryptography and Network Security Lab Assessment 3

DES operations

3 a. To take input plaintext and perform initial permutation.

Code:

#include <iostream>

#include <bitset>

using namespace std;

int IP[] = {

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

};

bitset<64> initialPermutation(bitset<64> input) {

bitset<64> permuted;

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

permuted[63 - i] = input[64 - IP[i]];

}

return permuted;

}

int main() {

uint64\_t input;

cout << "Enter a 64-bit number (in hex): ";

cin >> hex >> input;

bitset<64> inputBits(input);

cout << "Original 64-bit input: " << inputBits << endl;

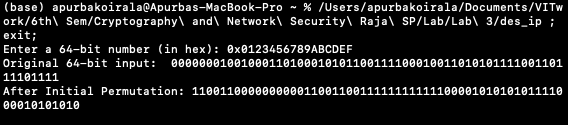
bitset<64> permutedBits = initialPermutation(inputBits);

cout << "After Initial Permutation: " << permutedBits << endl;

return 0;

}

Output:



3 b.

Key transformation from 64bits to 48bits including PC1, left circular shifts and PC2.

Code:

#include <iostream>

#include <bitset>

using namespace std;

int PC1[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

};

int PC2[48] = {

14, 17, 11, 24, 1, 5, 3, 28,

15, 6, 21, 10, 23, 19, 12, 4,

26, 8, 16, 7, 27, 20, 13, 2,

41, 52, 31, 37, 47, 55, 30, 40,

51, 45, 33, 48, 44, 49, 39, 56,

34, 53, 46, 42, 50, 36, 29, 32

};

int shiftSchedule[16] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

bitset<56> permutePC1(bitset<64> key) {

bitset<56> permutedKey;

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

permutedKey[55 - i] = key[64 - PC1[i]];

}

return permutedKey;

}

bitset<48> permutePC2(bitset<56> combinedCD) {

bitset<48> subKey;

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

subKey[47 - i] = combinedCD[56 - PC2[i]];

}

return subKey;

}

bitset<28> leftCircularShift(bitset<28> half, int shifts) {

return (half << shifts) | (half >> (28 - shifts));

}

int main() {

uint64\_t inputKey;

int roundNumber;

cout << "Enter 64-bit key (in hexadecimal): ";

cin >> hex >> inputKey;

cout << "Enter round number (1-16): ";

cin >> roundNumber;

if (roundNumber < 1 || roundNumber > 16) {

cout << "Invalid round number. Must be between 1 and 16." << endl;

return 1;

}

bitset<64> key(inputKey);

bitset<56> permutedKey = permutePC1(key);

bitset<28> C, D;

for (int i = 0; i < 28; i++) {

C[27 - i] = permutedKey[55 - i];

D[27 - i] = permutedKey[27 - i];

}

int shifts = shiftSchedule[roundNumber - 1];

C = leftCircularShift(C, shifts);

D = leftCircularShift(D, shifts);

bitset<56> combinedCD = (C.to\_ullong() << 28) | D.to\_ullong();

bitset<48> roundKey = permutePC2(combinedCD);

cout << "Round " << roundNumber << " Key: " << roundKey << endl;

return 0;

}

Output:

A computer screen with numbers and text

AI-generated content may be incorrect.

3 c.

Round 1 operation with 64bit text after initial permutation and 48bit key after PC2

#include <iostream>

#include <bitset>

using namespace std;

int E[48] = {

32, 1, 2, 3, 4, 5, 4, 5,

6, 7, 8, 9, 8, 9, 10, 11,

12, 13, 12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21, 20, 21,

22, 23, 24, 25, 24, 25, 26, 27,

28, 29, 28, 29, 30, 31, 32, 1

};

int SBox[8][4][16] = {

{{14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7},

{0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8},

{4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0},

{15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13}},

};

int P[32] = {

16, 7, 20, 21, 29, 12, 28, 17,

1, 15, 23, 26, 5, 18, 31, 10,

2, 8, 24, 14, 32, 27, 3, 9,

19, 13, 30, 6, 22, 11, 4, 25

};

bitset<48> expand(bitset<32> R) {

bitset<48> expanded;

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

expanded[47 - i] = R[32 - E[i]];

}

return expanded;

}

bitset<32> sBoxSubstitution(bitset<48> input) {

bitset<32> output;

for (int i = 0; i < 8; i++) {

int row = (input[47 - (i \* 6)] << 1) | input[47 - (i \* 6 + 5)];

int col = (input[47 - (i \* 6 + 1)] << 3) | (input[47 - (i \* 6 + 2)] << 2) |

(input[47 - (i \* 6 + 3)] << 1) | input[47 - (i \* 6 + 4)];

int value = SBox[i][row][col];

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

output[31 - (i \* 4 + j)] = (value >> (3 - j)) & 1;

}

}

return output;

}

bitset<32> permute(bitset<32> input) {

bitset<32> output;

for (int i = 0; i < 32; i++) {

output[31 - i] = input[32 - P[i]];

}

return output;

}

int main() {

bitset<64> input("0000000100100011010001010110011110001001101010111100110111101111");

bitset<48> roundKey("000110110000001011101111111111000111000001110010");

bitset<32> L, R;

for (int i = 0; i < 32; i++) {

L[i] = input[i + 32];

R[i] = input[i];

}

bitset<48> expandedR = expand(R);

bitset<48> xorResult = expandedR ^ roundKey;

bitset<32> sBoxOutput = sBoxSubstitution(xorResult);

bitset<32> permutedOutput = permute(sBoxOutput);

bitset<32> newR = L ^ permutedOutput;

bitset<32> newL = R;

bitset<64> finalOutput;

for (int i = 0; i < 32; i++) {

finalOutput[i + 32] = newL[i];

finalOutput[i] = newR[i];

}

cout << "L: " << L << endl;

cout << "R: " << R << endl;

cout << "Expanded R: " << expandedR << endl;

cout << "XOR Result: " << xorResult << endl;

cout << "S-Box Output: " << sBoxOutput << endl;

cout << "Permuted Output: " << permutedOutput << endl;

cout << "New R: " << newR << endl;

cout << "Final 64-bit Output: " << finalOutput << endl;

return 0;

}

Output:

