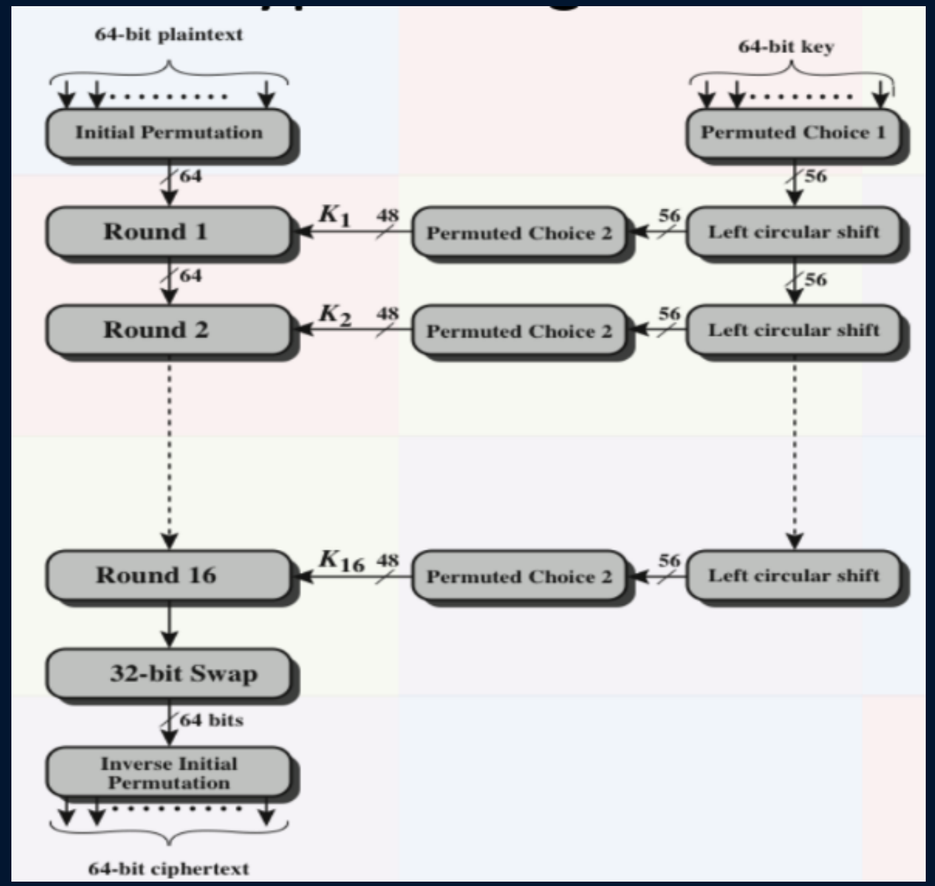
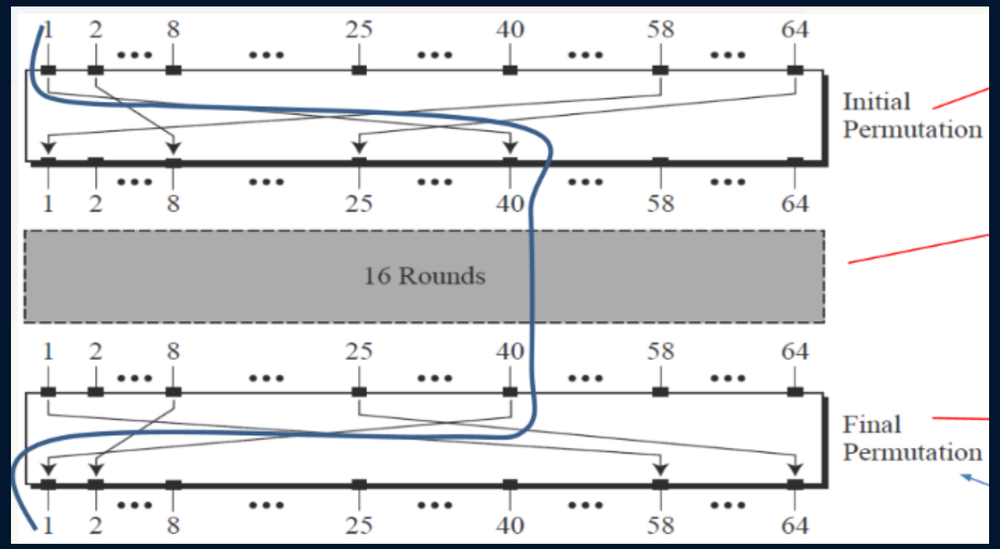
ASSAIGNMENT2

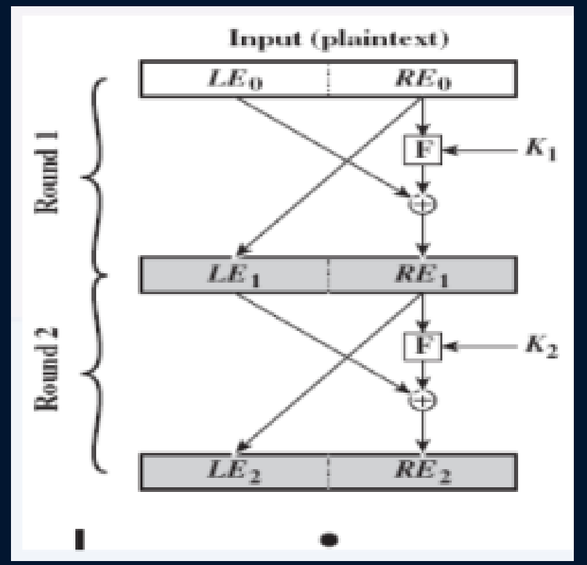
DES

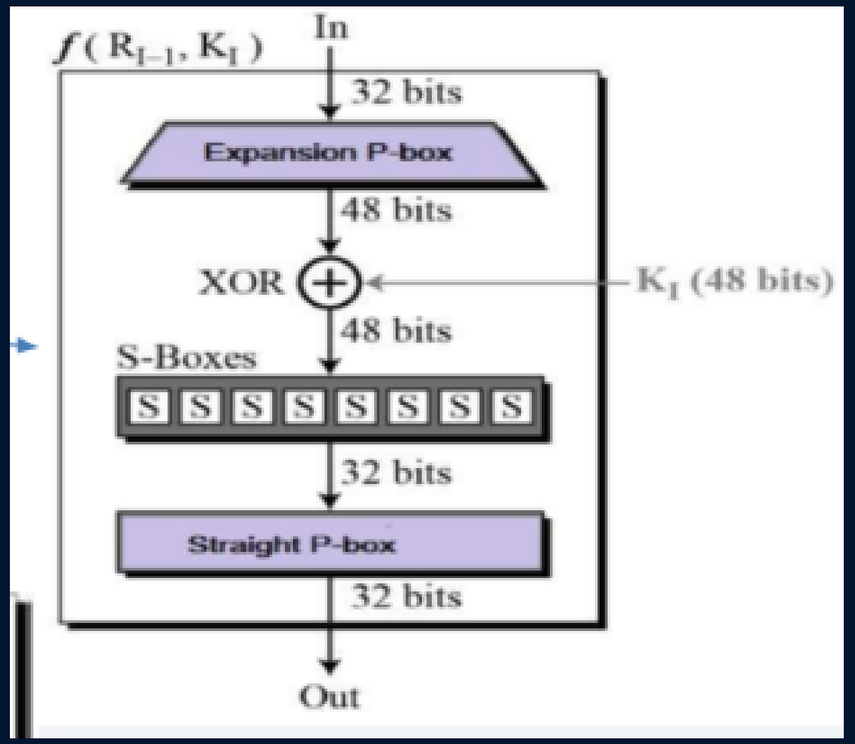
Cheti reddy Sreekar reddy 2021318
Satwik rampelli 2021276

DES encryption



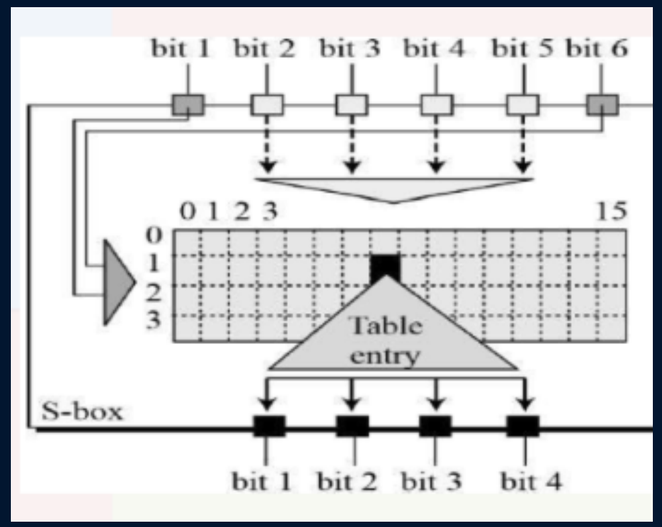






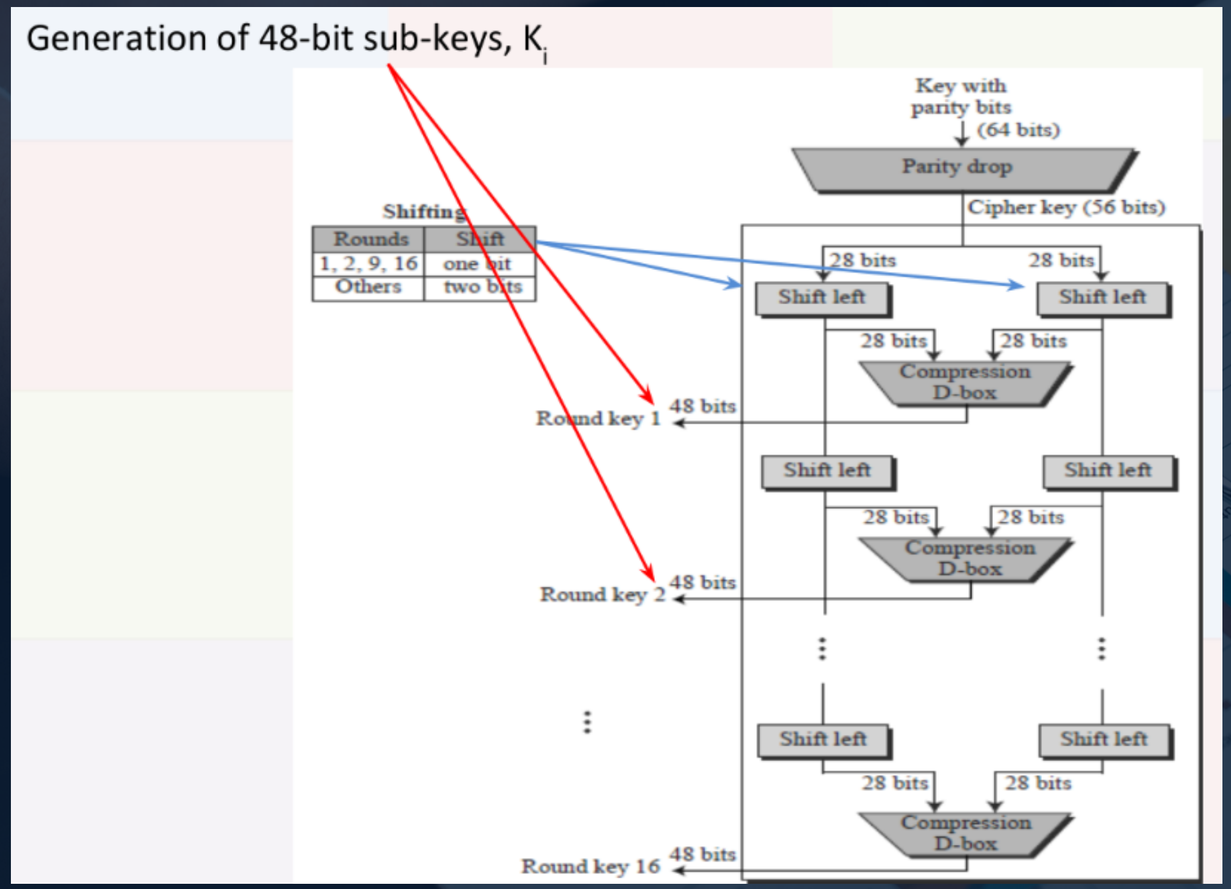
F box

```
// Expansion P-box: Expanding the 32 bits data into 48 bits
string right_expanded = permute(right, exp_p, 48);

// XOR RoundKey[i] and right_expanded
string xor_x = xor_strings(right_expanded, rkb[i]);

// S-boxes: substituting the value from s-box table by calculating row and column
string sbox_out = "";
for (int j = 0; j < 8; j++) {
    string r,c;
    r+= xor_x[j * 6];
    r+= xor_x[j * 6 + 5];
    int row = bin2dec(r);
    c+= xor_x[j * 6 + 1];
    c+= xor_x[j * 6 + 2];
    c+= xor_x[j * 6 + 3];
    c+= xor_x[j * 6 + 4];
    int col = bin2dec( c);
    int val = sbox[j][row][col];
    sbox_out += dec2bin(val);
}
// Straight P-box: After substituting rearranging the bits
string sp_box = permute(sbox_out, per, 32);

// XOR left and sp_box
string result = xor_strings(left, sp_box);
left = result;

string s = left; //swap
```



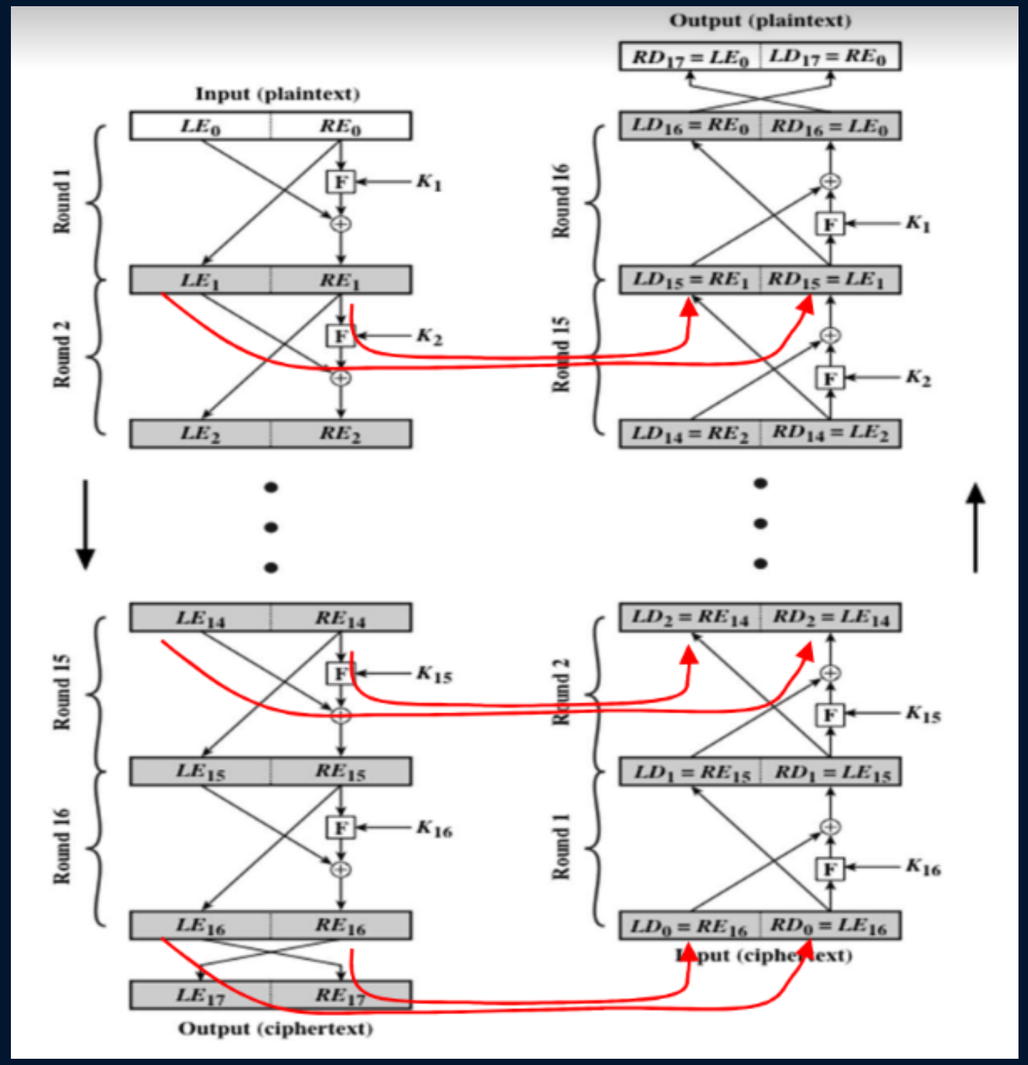
S-box

Key generation




```cpp
// Splitting
string left = key.substr(0, 28);
string right = key.substr(28, 56);

vector<string> rkb; // rkb for RoundKeys in binary
for (int i = 0; i < 16; i++) {
    // Shifting the bits by nth shifts by checking from shift table
    left = shift_left(left, shift_table[i]);
    right = shift_left(right, shift_table[i]);

    string combine_str = left + right;

    // Compression of key from 56 to 48 bits
    string round_key = permute(combine_str, key_comp, 48);

    rkb.push_back(round_key);

}

vector<string> ct(3);
ct = encrypt(pt[i], rkb, "e"); //encryption

reverse(rkb.begin(), rkb.end());


vector<string> text(3);
text = encrypt(ct[0], rkb, "d");  //decryption
```

Same round keys are used for both encryption and decryption



Decryption

- It is exactly same as decryption, only thing to do is swap left and right parts of the output of round 16 of encryption before round 1 of decryption.

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Output

```
Enter plaintext 1 (16 hexadecimal characters): 02468ACEECA86420
Enter cipher key 1 (16 hexadecimal characters): 0F1571C947D9E859
Enter plaintext 2 (16 hexadecimal characters): 23D58A1002FBC891
Enter cipher key 2 (16 hexadecimal characters): 4B72A09DE0A1BFC3
Enter plaintext 3 (16 hexadecimal characters): 5CDF9000358A0B27
Enter cipher key 3 (16 hexadecimal characters): 2EA11460DBF77269
pair - 1
matched ciphertext: DA02CE3A89ECAC3B generated plaintext: 02468ACEECA86420
matched 1st encrytption: 3CF03C0FBAD22845- 15th decryption: 3CF03C0FBAD22845
matched 14th encrytption: C6A62C4E56B0BD75 2nd decryption: C6A62C4E56B0BD75
pair - 2
matched ciphertext: 082259A168BF3341 generated plaintext: 23D58A1002FBC891
matched 1st encrytption: E621643532BEA25F- 15th decryption: E621643532BEA25F
matched 14th encrytption: CAFABB34FF473CEE 2nd decryption: CAFABB34FF473CEE
pair - 3
matched ciphertext: 790FCACAB7D8C4A8 generated plaintext: 5CDF9000358A0B27
matched 1st encrytption: 269063E2763F9277- 15th decryption: 269063E2763F9277
matched 14th encrytption: F27AEE0923B988D4 2nd decryption: F27AEE0923B988D4
```

Output of encryption round “i” is equal to swap of output of decryption round “16-i”

```
Round 4: 0BAE3B9E 42415649
Round 5: 42415649 18B3FA41
Round 6: 18B3FA41 9616FE23
Round 7: 9616FE23 67117CF2
Round 8: 67117CF2 C11BFC09
Round 9: C11BFC09 887FBC6C
Round 10: 887FBC6C 600F7E8B
Round 11: 600F7E8B F596506E
Round 12: F596506E 738538B8
Round 13: 738538B8 C6A62C4E
Round 14: C6A62C4E 56B0BD75
Round 15: 56B0BD75 75E8FD8F
Round 16: 75E8FD8F 25896490
decryption
Round 1: 75E8FD8F 56B0BD75
Round 2: 56B0BD75 C6A62C4E
Round 3: C6A62C4E 738538B8
Round 4: 738538B8 F596506E
Round 5: F596506E 600F7E8B
Round 6: 600F7E8B 887FBC6C
Round 7: 887FBC6C C11BFC09
Round 8: C11BFC09 67117CF2
Round 9: 67117CF2 9616FE23
Round 10: 9616FE23 18B3FA41
Round 11: 18B3FA41 42415649
Round 12: 42415649 0BAE3B9E
```


THANK YOU