Data Encryption Standard

- Setya Wibawa 17-28
- Bagas Yanuar Sudrajad 17-74
- Komang Yogananda Mahaputra Wisna 17-114
- Fadhil Musaad Al Giffary 17-116

Bagaimana DES Bekerja Secara Umum

DES bekerja menggunakan 64 bits plain text dan 64 bits key (8 parity bits dan 56 key sesungguhnya), serta menghasilkan 64 bits cipher text.

64 bit plain text -> Initial Permutation -> Split menjadi 2, Left dan Right plain text. Lalu bersama dengan key, dienkripsi menggunakan suatu prosedur yang diulang 16 ronde -> Final Permutation -> 64 bit cipher text.

56 bit key sendiri akan diolah terlebih dahulu untuk menjadi 16 buah 48 bit key untuk masing-masing ronde.

Untuk lebih detailnya akan dijelaskan bersama dengan penjelasan program.

Penjelasan Program

Struct Table

Didefinisikan sebuah struct yang beranggotakan tabel-tabel yang akan dibutuhkan dalam program.

Untuk seterusnya struct ini dapat direferensi dengan `table`

```
typedef struct T_table {

} table
```

Berikut adalah tabel-tabel yang dibutuhkan pada program ini, yang mana semuanya ada pada struct table:

Tabel yang digunakan untuk initial permutation

```
int table_initial_permutation[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
    };
```

Tabel yang digunakan untuk mengekspansi dari `plain_text_right` yang berukuran 32 bits menjadi 48 bits.

```
int table_expansion_d_box[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
    };
```

Tabel yang digunakan untuk Final Permutation.

```
int table_final_permutation[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
    };
```

Tabel yang digunakan untuk permutasi 64 bit key menjadi 56 bit key tanpa parity bit

```
int table_56_key_permutation[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
    };
```

Tabel untuk mengkompress 56 bit key menjadi 48 bit `round_key`, key-key untuk 16 ronde

```
int table_compression_permutation[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 table_permutation_tmp[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
    };
```

Tabel yang mencatat banyak shift left yang dilakukan pada key untuk 16 ronde

```
int table_shift_key[16]=
    { 1, 1, 2, 2,
        2, 2, 2, 2,
        1, 2, 2, 2,
        2, 2, 2, 1
    };
```

Tabel s box

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


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

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

Tabel untuk suatu permutasi yang akan dilakukan pada tiap ronde

```
int table_permutation_tmp[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
    };
```

Main Function

```
int main(){
  table t1; // struct table
    string plain_text, key;
    cout<<"Masukkan plain text (dalam hexadecimal, 64 bit): ";
    cin>>plain_text; // input 64 bit plain text
    cout<<"Masukkan key(dalam hexadecimal, 64 bit): ";
    cin>>key; // input 64 bit key
```

- Terima masukan 64 bit plain text
- Terima masukan 64 bit key

```
    key= hex_to_bin(key);
    key= permutate(key, t1.table_56_key_permutation, 56);
    string key_left = key.substr(0, 28);
    string key_right = key.substr(28, 28);
```

- Ubah key menjadi biner untuk komputasi
- Permutasi key menjadi 56 bit

- Bagi key menjadi dua, `key_left` dan `key_right`

```
    vector<string> round_key_biner;
    vector<string> round_key_hex;
    generate_key_round(round_key_biner, round_key_hex, key_left, key_right);
```

- Buat 16 48 bit key untuk 16 ronde. `round_key_biner` untuk bentuk biner, dan `round_key_hex` untuk hex.

```
    cout<<"\nEnkripsi DES:\n\n";
    string cipher= DES(plain_text, round_key_biner, round_key_hex);
    cout<<"\nCipher Text: "<<cipher<<endl;
  return 0;
}
```

- Menjalankan fungsi DES dengan argument plain text, dan round key untuk 16 ronde. Fungsi ini menghasilkan chiper text

```
    cout<<"\nDekripsi\n\n";
    string text= DES_Decryption(cipher, round_key_biner, round_key_hex);
    cout<<"\nPlain Text: "<<text<<endl;
```

- Menjalankan fungsi dekripsi. Fungsi ini akan memanggil fungsi yang sama dengan enkripsi hanya saja urutan round key dibalik

Fungsi Mengubah Hex menjadi Bin

```
string hex_to_bin(string hex){
    unordered_map<char, string> map_hex_to_bin;
    map_hex_to_bin['0'] = "0000";
    map_hex_to_bin['1'] = "0001";
    map_hex_to_bin['2'] = "0010";
    map_hex_to_bin['3'] = "0011";
    map_hex_to_bin['4'] = "0100";
    map_hex_to_bin['5'] = "0101";
    map_hex_to_bin['6'] = "0110";
    map_hex_to_bin['7'] = "0111";
    map_hex_to_bin['8'] = "1000";
    map_hex_to_bin['9'] = "1001";
    map_hex_to_bin['A'] = "1010";
    map_hex_to_bin['B'] = "1011";
    map_hex_to_bin['C'] = "1100";
    map_hex_to_bin['D'] = "1101";
    map_hex_to_bin['E'] = "1110";
    map_hex_to_bin['F'] = "1111";
    string bin = "";
    for(int i = 0; i < hex.size(); i++){
        bin += map_hex_to_bin[hex[i]];
    }
    return bin;
}
```

- map dengan key berupa HEX dan value berupa biner. Untuk masing-masing karakter hex diubah ke biner.

Fungsi mengubah Bin menjadi Hex

```
string bin_to_hex(string s){
    unordered_map<string, string> map_bin_to_hex;
    map_bin_to_hex["0000"] = "0";
    map_bin_to_hex["0001"] = "1";
    map_bin_to_hex["0010"] = "2";
    map_bin_to_hex["0011"] = "3";
    map_bin_to_hex["0100"] = "4";
    map_bin_to_hex["0101"] = "5";
    map_bin_to_hex["0110"] = "6";
    map_bin_to_hex["0111"] = "7";
    map_bin_to_hex["1000"] = "8";
    map_bin_to_hex["1001"] = "9";
    map_bin_to_hex["1010"] = "A";
    map_bin_to_hex["1011"] = "B";
    map_bin_to_hex["1100"] = "C";
    map_bin_to_hex["1101"] = "D";
    map_bin_to_hex["1110"] = "E";
    map_bin_to_hex["1111"] = "F";

    string hex = "";

  for(int i = 0; i < s.length(); i += 4){
        string tmp = "";
    for (int j = 0; j < 4; j++){
      tmp += s[i+j];
    }
        hex += map_bin_to_hex[tmp];
    }
    return hex;
}
```

- map yang memetakan 4 digit biner ke 1 karakter Hex.
- Untuk masing-masing 4 digit biner, diubah ke Hex.

Fungsi XOR, 2 String

```
string xor_operation(string x, string y){
    string result = "";
    for(int i = 0; i < x.size(); i++){
        if( x[i] != y[i] ){
            result += "1";
        }else{
            result += "0";
        }
    }
    return result;
}
```

- membandingkan dua string, jika karakter pada index yang sama sama, maka append `"1"` pada variable result, jika beda append `"0"`

Fungsi Permutasi

```
string permutate(string s, int* table_permutation, int n){
    string result = "";
    for(int i = 0; i < n ; i++){
        result += s[table_permutation[i]-1];
    }
    return result;
}
```

Fungsi untuk melakukan permutasi. Menerima parameter `string s` `int* table permutation`.

Fungsi ini bekerja dengan cara: array 2 dimensi yang sebagai argumen, dibaca dari index awal. Nilai dari suatu index adalah index untuk nilai yang akan diambil. Contoh: pada index ke-0 terdapat nilai 32. Maka string hasil pada index 0 adalah karakter index 32 pada string masukan.

Fungsi Membuat 16 Round Key

```
void generate_key_round (vector<string> &round_key_biner, vector<string>
&round_key_hex, string &key_left, string &key_right){
  table t1;
  for(int i = 0; i < 16; i++){
        key_left= shift_left(key_left, t1.table_shift_key[i]);
        key_right= shift_left(key_right, t1.table_shift_key[i]);

        string key_combined = key_left + key_right;
        string round_key= permutate(key_combined,
t1.table_compression_permutation, 48);

        round_key_biner.push_back(round_key);
        round_key_hex.push_back(bin_to_hex(round_key));
    }
}
```

- Mendeklarasi kan variabel `t1` bertipe struct table
- Terdapat 16 iterasi, masing-masing iterasi menghasilkan satu round key
    - shift left pada `key_left` dan `key_right`
    - gabungkan menjadi satu key lagi
    - permutasikan menurut table `table_compression_permutation`

Fungsi DES

```
string DES(string plain_text, vector<string> round_key_biner, vector<string>
round_key_hex){
  table t1;
  plain_text = hex_to_bin(plain_text);
  plain_text= permutate(plain_text, t1.table_initial_permutation, 64);
  cout << "Hasil initial permutation: "<< bin_to_hex(plain_text) << endl;

  string left_plain_text= plain_text.substr(0, 32);
  string right_plain_text= plain_text.substr(32, 32);
  cout<< "Hasil splitting jadi dua: L0="<<bin_to_hex(left_plain_text) << " R0="
<<bin_to_hex(right_plain_text)<<endl<<endl;
```

- deklarasi sebuah variable dengan tipe struct table
- permutasi plain text menurut tabel yang digunakan untuk initial permutation
- Bagi plain text menjadi dua, kiri dan kanan, masing-masing 32 bit

```
    for(int i = 0; i < 16; i++){
        string right_expanded= permutate(right_plain_text,
t1.table_expansion_d_box, 48);
        string xor_result= xor_operation(round_key_biner[i], right_expanded);

        string tmp="";
        for(int i=0;i<8; i++){
            int row= 2*int(xor_result[i*6]-'0')+ int(xor_result[i*6 +5]-'0');
            int col= 8*int(xor_result[i*6 +1 ]-'0')+ 4*int(xor_result[i*6 +2]-
'0')+
                                2*int(xor_result[i*6 +3]-'0')+
int(xor_result[i*6 +4]-'0');
            int val= t1.table_s_box[i][row][col];
            tmp+= char(val/8+ '0');
            val= val%8;
            tmp+= char(val/4+ '0');
            val= val%4;
            tmp+= char(val/2+ '0');
            val= val%2;
            tmp+= char(val+ '0');
        }

        tmp= permutate(tmp, t1.table_permutation_tmp, 32);
        xor_result= xor_operation(tmp, left_plain_text);
        left_plain_text= xor_result;
        if(i!= 15){
            swap(left_plain_text, right_plain_text);
        }
        cout<<"Ronde "<<i+1<<" "<<bin_to_hex(left_plain_text)<<" "
<<bin_to_hex(right_plain_text)<<" ... round key => "<<round_key_hex[i]<<endl;
    }
```

- expand 32 bit plain text kanan menjadi 48 bit, menggunakan permutasi menurut tabel.
- XOR 48 bit key yang telah diexpand dengan round key
- Dilakukan penggunaan s box
- hasilnya dipermutasi menurut tabel yang sudah ada.

- XOR hasil permutasi tersebut dengan plain text kiri
- hasil tersebut menjadi plain text kanan selanjutnya, sementara plain text kanan pada ronde ini menjadi plain text kiri ronde berikutnya

```
string combine= left_plain_text+right_plain_text;
    string cipher= bin_to_hex(permutate(combine, t1.table_final_permutation,
64));
    return cipher;
}
```

- Gabungkan kembali plain text kanan dan kiri
- Lakukan Final Permutation

Fungsi Dekripsi

```
string DES_Decryption(string cipher, vector<string> &round_key_biner,
vector<string> &round_key_hex){
  reverse(round_key_biner.begin(), round_key_biner.end());
    reverse(round_key_hex.begin(), round_key_hex.end());
  return DES(cipher, round_key_biner, round_key_hex);
}
```

Fungsi dekripsi ini membalik urutan ronde untuk 16 round key, lalu menjalankan fungsi DES yang sama dengan yang digunakan untuk enkripsi.

Run Time

```
g++ -o run DES.cpp
./run
```

```
Masukkan plain text (dalam hexadecimal, 64 bit): ABCDEF1234132DEF
Masukkan key(dalam hexadecimal, 64 bit): 918B0ABC2736FFEE

Enkripsi DES:

Hasil initial permutation: 8638D6E787D5C7AD
Hasil splitting jadi dua: L0=8638D6E7 R0=87D5C7AD

Ronde 1 87D5C7AD 0536B784 ... round key => 4945F4777F8E
Ronde 2 0536B784 F3314798 ... round key => 57C4781EBF07
Ronde 3 F3314798 DE7ECA23 ... round key => CEC9A2FE65F4
Ronde 4 DE7ECA23 5A8B8E0C ... round key => BAAB0F69EBCB
Ronde 5 5A8B8E0C A344968E ... round key => 29360BF6F41B
Ronde 6 A344968E B603BD23 ... round key => 611CFCEF176E
Ronde 7 B603BD23 E8CC25D9 ... round key => D4E8D09CFBEA
Ronde 8 E8CC25D9 006ADAC8 ... round key => 16EF3274DE75
Ronde 9 006ADAC8 57FEED51 ... round key => A6EB05EAD8DC
Ronde 10 57FEED51 08FA0716 ... round key => 0B3727C1F7BF
Ronde 11 08FA0716 28E3846D ... round key => E914F9BF1EA9
Ronde 12 28E3846D 32D32458 ... round key => D5C2E8DA5B77
Ronde 13 32D32458 D3918CFA ... round key => 92DB9217EBBC
Ronde 14 D3918CFA D767E429 ... round key => 3C3B47F13DD1
Ronde 15 D767E429 39131C8E ... round key => 23744DEBA23F
Ronde 16 353E3620 39131C8E ... round key => 4C34DDB3C7EB
```

```
Cipher Text: E0365E9AFCD50002

Dekripsi

Hasil initial permutation: 353E362039131C8E
Hasil splitting jadi dua: L0=353E3620 R0=39131C8E

Ronde 1 39131C8E D767E429 ... round key => 4C34DDB3C7EB
Ronde 2 D767E429 D3918CFA ... round key => 23744DEBA23F
Ronde 3 D3918CFA 32D32458 ... round key => 3C3B47F13DD1
Ronde 4 32D32458 28E3846D ... round key => 92DB9217EBBC
Ronde 5 28E3846D 08FA0716 ... round key => D5C2E8DA5B77
Ronde 6 08FA0716 57FEED51 ... round key => E914F9BF1EA9
Ronde 7 57FEED51 006ADAC8 ... round key => 0B3727C1F7BF
Ronde 8 006ADAC8 E8CC25D9 ... round key => A6EB05EAD8DC
Ronde 9 E8CC25D9 B603BD23 ... round key => 16EF3274DE75
Ronde 10 B603BD23 A344968E ... round key => D4E8D09CFBEA
Ronde 11 A344968E 5A8B8E0C ... round key => 611CFCEF176E
Ronde 12 5A8B8E0C DE7ECA23 ... round key => 29360BF6F41B
Ronde 13 DE7ECA23 F3314798 ... round key => BAAB0F69EBCB
Ronde 14 F3314798 0536B784 ... round key => CEC9A2FE65F4
Ronde 15 0536B784 87D5C7AD ... round key => 57C4781EBF07
Ronde 16 8638D6E7 87D5C7AD ... round key => 4945F4777F8E

Plain Text: ABCDEF1234132DEF
```