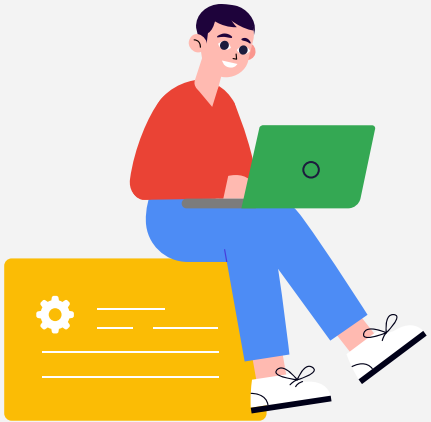
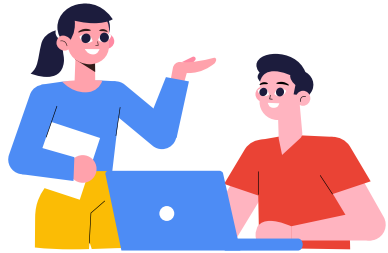
MA4151 - Kriptografi

Algoritma DES (*Data Encryption Standard)*



Team members



Annisa Sekar A

10117004

Rizvan Dwikifirdaus

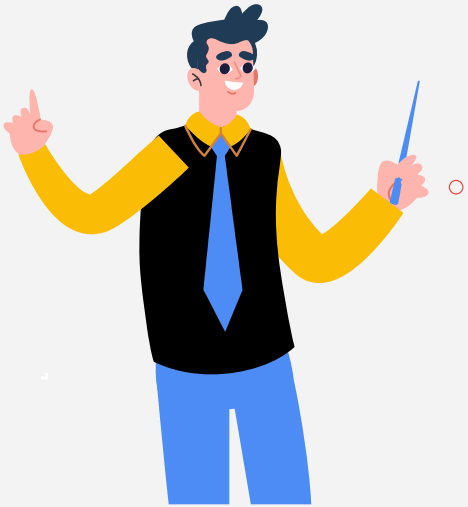
10117060

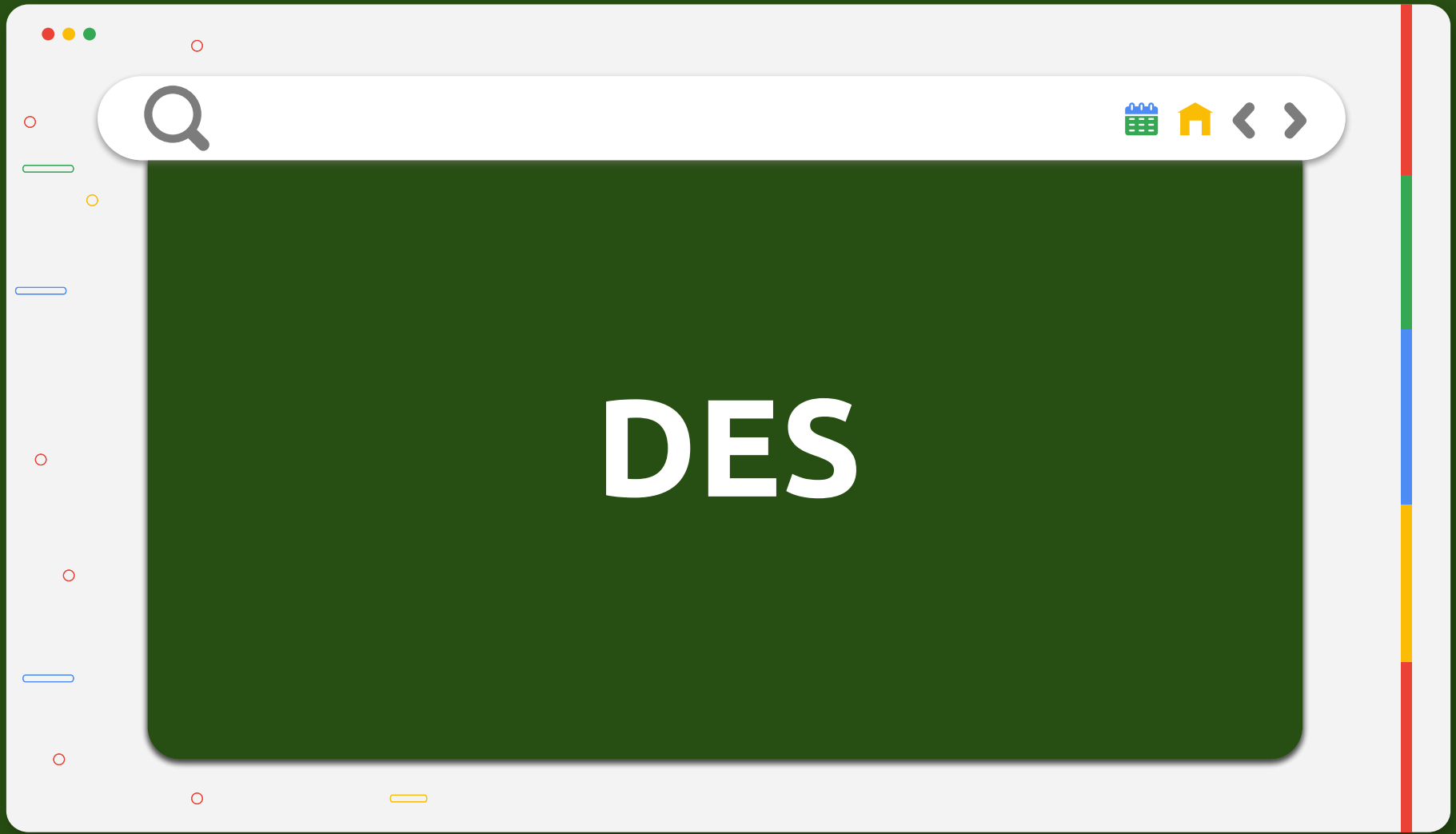
Reynara Ezra P

10117092

Outline

1. DES
2. Algoritma
3. DES Analysis
4. Program




DES

Data Encryption Standard

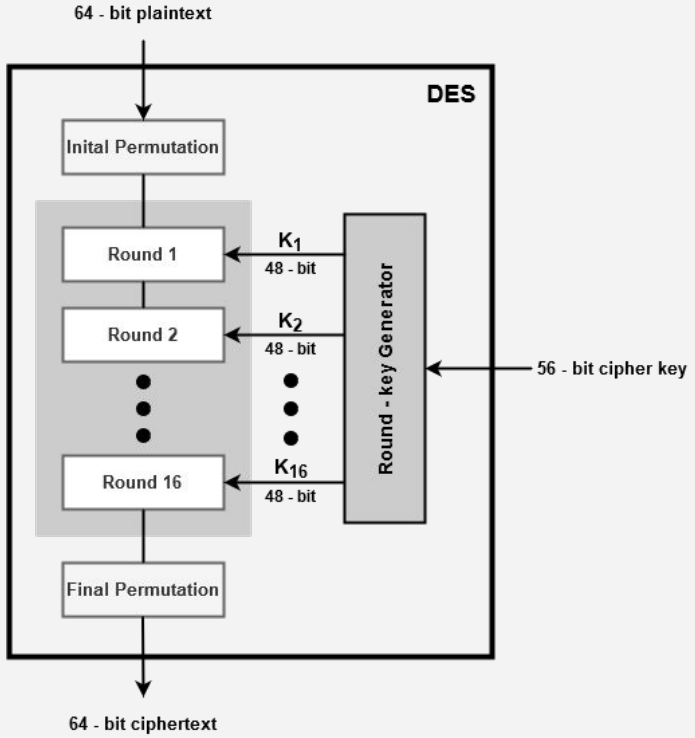
Data Encryption Standard (DES) adalah sandi blok kunci simetris yang dipublikasikan oleh National Institute of Standards and Technology (NIST).

DES merupakan sebuah implementasi dari Feistel Cipher. DES menggunakan 16 round struktur Feistel. Ukuran dari blok adalah 64-bit. Meskipun panjang kunci dari DES adalah 64-bit, tetapi kunci efektif hanya berukuran 56 bit saja, karena 8 dari 64 bit kunci tidak digunakan oleh algoritma enkripsi (berfungsi sebagai bit cek saja).

Algoritma

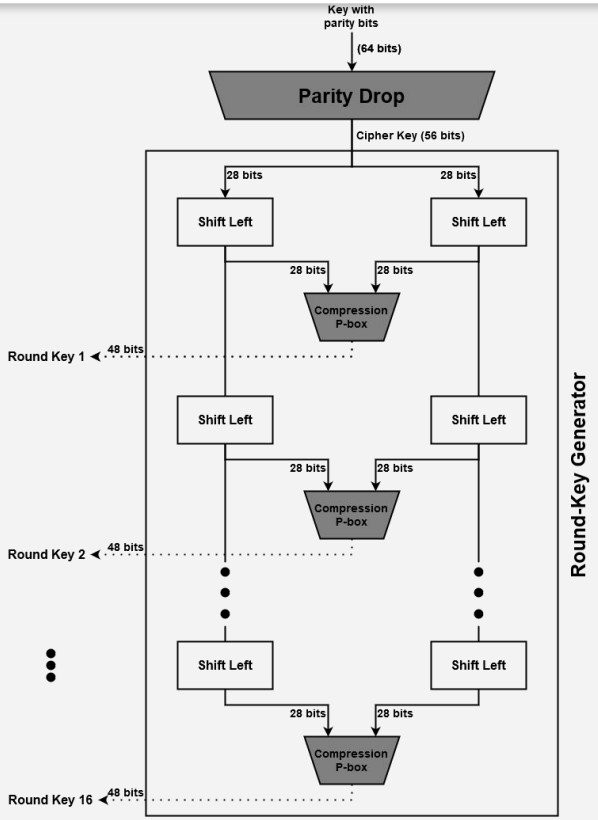
General Structure of DES

- DES bekerja dengan cara **mengenkripsi** pesan berukuran **64-bit** yang setara dengan 16 bilangan hexadecimal.
- DES menggunakan "*key*" berukuran **64-bit** yang juga setara dengan 16 bilangan hexadecimal.
- Setiap bit kelipatan 8 dalam key tidak digunakan (8, 16, …, 64). Sehingga ukuran kunci efektifnya berukuran 56-bit
- Karena DES berdasar pada *Feistel Cipher*, maka DES menggunakan ***Round Function, Key Schedule, Initial Permutation, dan Final Permutation*** dalam algoritma enkripsinya.



Key Generation

- Round-key generator membangkitkan 16 *subkeys*, masing-masing berukuran 48 bits
- Proses untuk membangkitkan *subkeys* terdiri dari permutasi **PC-1, shifting, dan permutasi PC-2.**



Key Generation Example

K : 11001010 10111110 00100000 00000000 00111010 11011010 00010100 00010101
K+ : 00100011 00100001 00010110 11110011 00111100 00100011 00110010

K1 : 000100 001111 010010 011001 001110 101000 000010 111100
K2 : 100111 000110 100010 000100 100000 100110 010011 101110
K3 : 000100 100010 111100 111000 001011 001011 101110 000001
K4 : 110011 000011 010000 100101 101100 100100 010001 110011
K5 : 110000 111000 111001 001100 010011 111000 101100 000010
K6 : 010010 001111 001010 100010 100101 000110 010101 011000
K7 : 101100 001001 110101 101010 011010 011001 001001 000100
K8 : 111000 000110 001001 000011 110100 001110 010010 101010
K9 : 001111 000011 001001 110110 000100 011111 110100 001101
K10 : 101001 100101 010001 001000 001010 100001 010010 110000
K11 : 010010 100100 001101 110100 110010 010110 100100 100111
K12 : 100011 001101 100100 011001 001001 100100 101010 011000
K13 : 000001 110010 001101 101011 110100 010001 000101 010111
K14 : 101010 110101 110010 000001 100001 111000 001010 101000
K15 : 010110 010010 101111 001000 010100 000011 111101 000101
K16 : 001000 110000 100011 111100 011111 001001 000000 000101

Key Generation (PC1)

Kunci 64-bit dipermutasi berdasarkan tabel **PC-1** di samping.
Karena entri pertama dari tabel adalah **"57"**, Ini artinya kunci asli dari **bit ke-57** menjadi **bit pertama** di kunci **K+** yang sudah dipermutasi.
Kemudian kunci asli dari **bit ke-49** menjadi **bit ke-2** pada kunci yang sudah dipermutasi.
Lalu kunci asli pada **bit ke-4** adalah **bit terakhir** dari kunci yang sudah dipermutasi.
Perhatikan bahwa **hanya 56-bit dari kunci asli yang muncul di kunci yang sudah dipermutasi**.

Tabel PC-1

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

Key Generation

Dengan C_0 dan D_0 yang telah didefinisikan, kita akan membuat 16 blok C_n dan D_n, 1<=n<=16.Setiap pasangan blok C_n and D_n terbentuk dari pasangan sebelumnya C_{n-1} dan D_{n-1}, untuk n = 1, 2, ..., 16, menggunakan "left shifts" dari blok sebelumnya. Untuk menggunakan left shift, pindahkan setiap bit yang ada ke kiri, kecuali untuk bit pertama akan dipindahkan ke bit terakhir dari blok.

Iteration	Left Shift	Iteration	Left Shift
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

Key Generation (PC2)

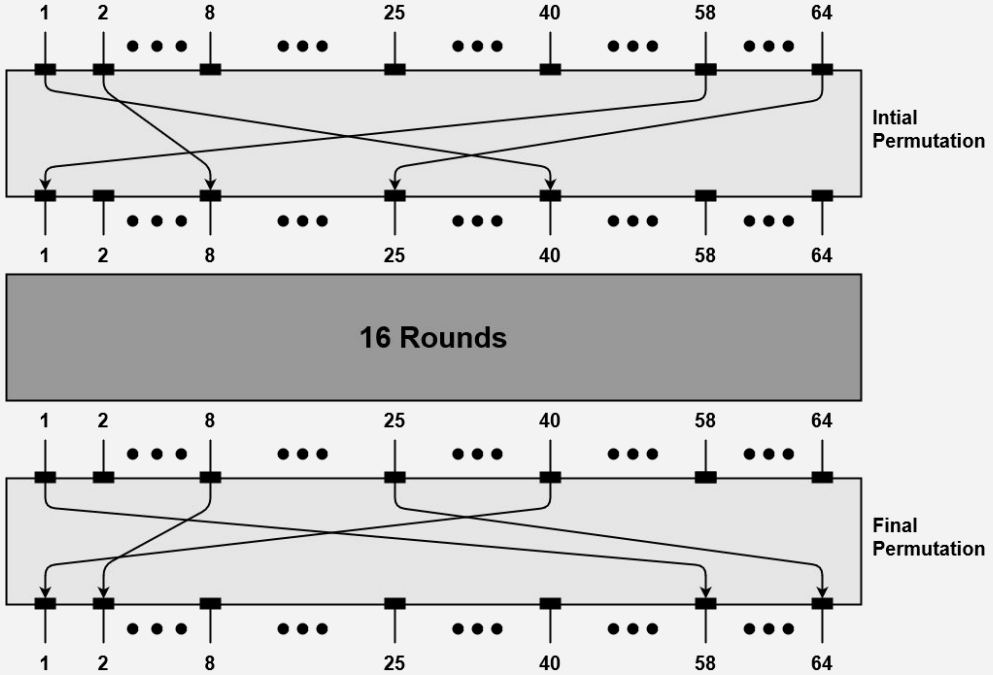
Sekarang kita akan membentuk kunci K_n. Untuk 1<=n<=16, dengan menggunakan permutasi yang ada pada tiap pasangan C_nD_n. Tiap pasangan berukuran 56-bit, namun **PC-2** hanya menggunakan 48-bit. Diperoleh, bit pertama dari K_n adalah 14 dan bit kedua adalah 17 dari C_nD_n. dan seterusnya hingga bit terakhir atau ke 48 dari K_n adalah 32 dari C_nD_n.

Tabel PC-2

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

Initial and Final Permutation

Initial dan Final Permutation merupakan ***straight permutation (P-box)*** yang merupakan **invers satu sama lain**. Keduanya tidak memiliki signifikansi kriptografis dalam DES. Permutasi awal dan akhir diilustrasikan seperti gambar di samping.



Initial Permutation

- ***Initial Permutation* (IP)** hanya dilakukan sekali
- Barisan bit akan berubah sedemikian rupa mengikuti tabel IP
- Contoh :
  34^{th} bit take 4^{th} position
  1^{st} bit take 40^{th} position
- *Output* dari ***Initial Permutation*** akan dibagi menjadi dua bagian sama besar (L_0 dan R_0), masing-masing berukuran **32 bit**

Tabel IP

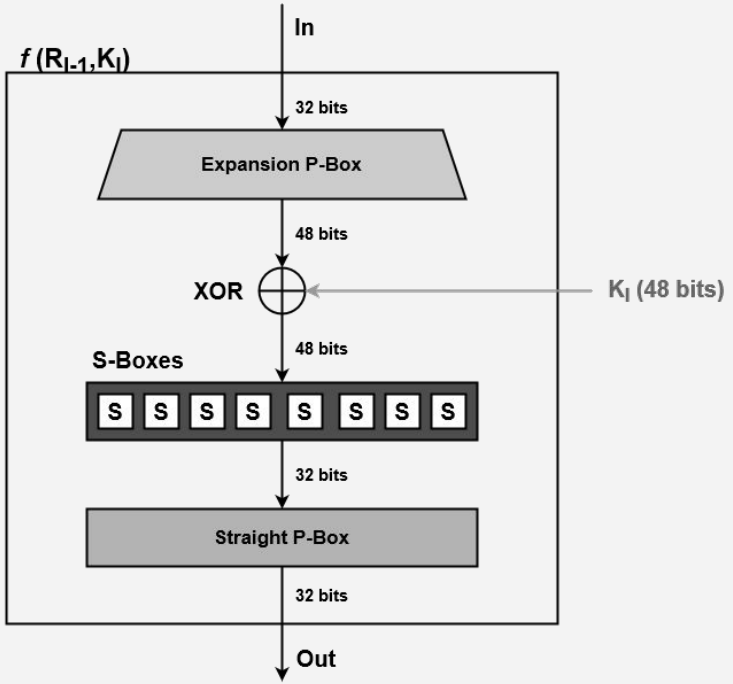
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	53	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

16 Rounds of Encryption

Round Function

1. Expansion Permutation Boxes
2. XOR (Whitener)
3. Substitution Boxes (S-Boxes)
4. Straight Permutation

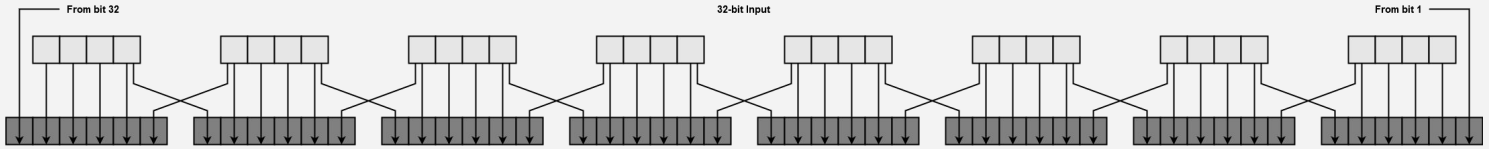
$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$



16 Rounds of Encryption

Expansion Permutation

Dikarenakan **R** berukuran **32-bit** dan ***round key*** berukuran **48-bit**, langkah pertama yang harus kita lakukan adalah **melakukan ekspansi pada R sehingga berukuran 48-bit**. Permutasi yang dilakukan dapat diilustrasikan dengan gambar di bawah ini:



16 Rounds of Encryption

Expansion Permutation

Langkah-langkah *expansion permutation*:

1. 32-bit RPT dibagi menjadi 8 partisi, masing-masing berukuran 4-bits
2. Setiap blok yang berisi 4-bit diekspansi menjadi 6-bit menggunakan **Expansion Permutation Block** sehingga menghasilkan luaran dengan ukuran 48-bits

Contoh :

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

$E(R_0)$= 011110 100001 010101 010101 011110 100001 010101 010101

E - Bit Selection Table

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

16 Rounds of Encryption

XOR (Whitener)

Setelah dilakukan ekspansi permutasi, **DES** menggunakan operator **XOR** pada R dan round key. Round key hanya digunakan satu kali pada operasi ini.

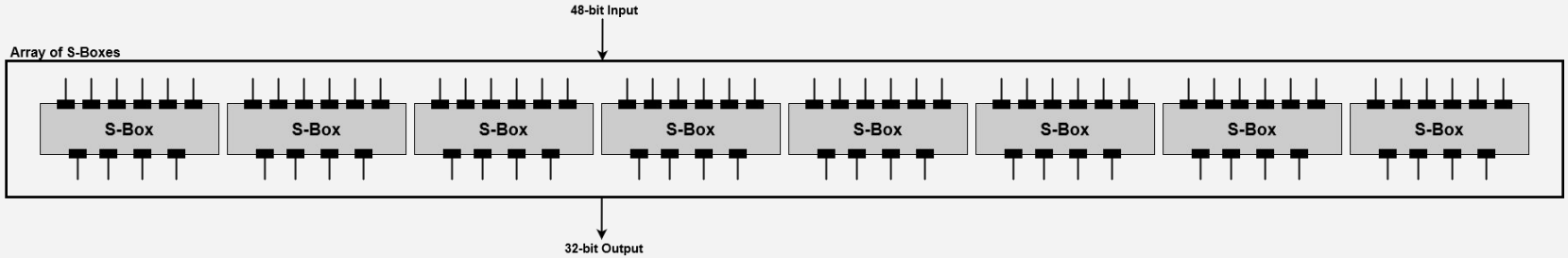
R_{n-1}	0	1	1	0	1	...	1
K_n	0	0	0	1	1	...	1
XOR	0	1	1	1	0	...	0

$$K_n + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$$

16 Rounds of Encryption

Substitution Boxes

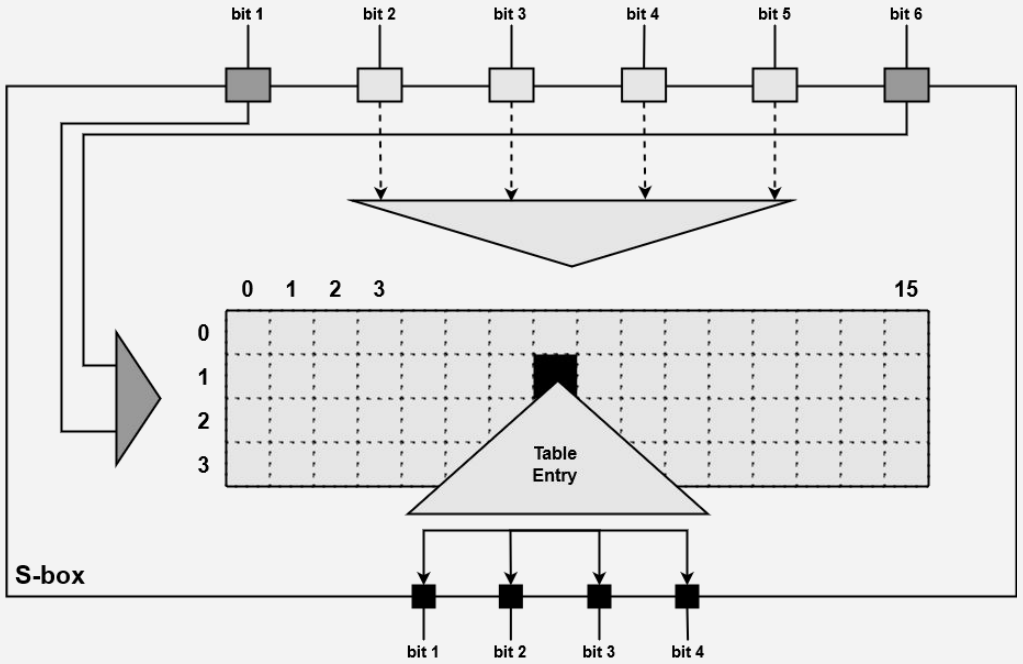
DES menggunakan **8 S-box**, masing-masing dengan **input 6-bit** dan **output 4-bit**. Operasi menggunakan S-box diilustrasikan sebagai berikut.



16 Rounds of Encryption

Substitution Boxes

Visualisasi alur kerja dari Substitution Box dapat dilihat melalui gambar di samping



16 Rounds of Encryption

Substitution Boxes

Contoh : Tabel S_0

Contoh : 101010 -> 0110

Pada 6 bit diatas, untuk bit 1 dan bit 6 adalah 10 yang mana adalah bernilai 2 (baris 3) dan 0101 adalah bernilai 5 (kolom 6) sehingga kita peroleh nilainya adalah 6. Lalu diubah ke dalam bentuk sistem biner

No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

16 Rounds of Encryption

Straight Permutation

Output dari S-box sebesar 32 bit kemudian dikenai *straight permutation* dengan aturan yang ditunjukkan pada ilustrasi berikut:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

DES Analysis

DES Analysis

DES memenuhi dua sifat dari *block cipher*. Kedua sifat ini membuat sandi tidak mudah diretas.

Avalanche Effect

Perubahan kecil pada *plaintext* mengakibatkan perubahan yang drastis pada *ciphertext* nya.

Completeness

Setiap bit dari *ciphertext* bergantung dari jumlah bit *plaintext*.

Pada beberapa tahun terakhir, kriptanalisis menunjukkan bahwa kelemahan DES adalah ketika kunci yang digunakan adalah kunci yang lemah (sehingga mudah dibobol).

DES telah terbukti sebagai *block cipher* dengan desain yang cukup baik. Belum ada serangan kriptanalisis yang signifikan selain pencarian kunci yang lengkap.




Program

Program

Google Colaboratory

Coding dapat diakses pada:
bit.ly/KriptografiDES

Start

Program

Key Generation

```
subkeys = generate_subkeys(16, 'cabe20003ada1415')
for i in range(len(subkeys)):
    print('K_{}: '.format(i+1), subkeys[i])
```

```
K_1:  000100001111010010011001001110101000000010111100
K_2:  100111000110100010000100100000100110010011101110
K_3:  000100100010111100111000001011001011101110000001
K_4:  110011000011010000100101101100100100010001110011
K_5:  110000111000111001001100010011111000101100000010
K_6:  010010001111001010100010100101000110010101011000
K_7:  101100001001110101101010011010011001001001000100
K_8:  111000000110001001000011110100001110010010101010
K_9:  001111000011001001110111000010001111111010000l101
K_10:  101001100101010001001000001010100001010010110000
K_11:  010010100100001101110100110010010110100100100111
K_12:  100011001101100100011001001001100100101010011000
K_13:  000001110010001101101011110100010001000101010111
K_14:  101010110101110010000001100001111000001010101000
K_15:  010110010010101111001000010100000011111101000101
K_16:  001000110000100011111100011111001001000000000101
```

```
def generate_subkeys(n_rounds, key):
    # fungsi pembangkit subkeys dari key sebanyak n_rounds
    C_list = []
    D_list = []
    K_list = []

    K = hex2bits(key)
    K_plus = permutation(K, PC1)

    C_n, D_n = split_half(K_plus)
    C_list.append(C_n)
    D_list.append(D_n)

    #Generate C_n, D_n, 1<=n<=n_rounds
    for i in range(n_rounds):
        if (i == 0) or (i == 1) or (i == 8) or (i == 15):
            C_n = rotate(C_n, 1)
            D_n = rotate(D_n, 1)
        else:
            C_n = rotate(C_n, 2)
            D_n = rotate(D_n, 2)
        C_list.append(C_n)
        D_list.append(D_n)

        #Generate K_n, 1<=n<=n_rounds
        concat = C_n + D_n
        K_n = permutation(concat, PC2)
        K_list.append(K_n)

    return K_list
```

Program

Fungsi Utama DES

Initial Permutation

16 Rounds of Encryption

Final Permutation

```
def des(plaintext, key, decrypt = False):
    # fungsi utama DES, decrypt = True untuk melakukan dekripsi
    n_rounds = 16
    R_list = []
    L_list = []

    PT = hex2bits(plaintext)
    PT = permutation(PT, IP)
    L_n, R_n = split_half(PT)
    L_list.append(L_n)
    R_list.append(R_n)

    subkeys = generate_subkeys(n_rounds = n_rounds, key = key)
    for r in range(n_rounds):
        if decrypt:
            k = n_rounds-r-1
        else:
            k = r

        L_n = R_n
        R_n = xor(L_list[r], f(R_list[r], subkeys[k]))

        L_list.append(L_n)
        R_list.append(R_n)

    RL = R_n + L_n
    ip_inv_bits = permutation(RL, IP_INV)

    ciphertext = bits2hex(ip_inv_bits).zfill(16)

    return ciphertext
```

Program

Fungsi Utama DES

Contoh Enkripsi

```
plaintext = '0857038269471437'
key = 'cabe20003ada1415'

C = des(plaintext = plaintext, key = key, decrypt = False)
print('plaintext :', plaintext, '\nkey:', key, '\nciphertext :', C)

D = des(plaintext = C, key = key, decrypt = True)
print('decrypted :', D,)
```

```
plaintext : 0857038269471437
key: cabe20003ada1415
ciphertext : bf9475a8641aaaa9
decrypted : 0857038269471437
```

Daftar Pustaka

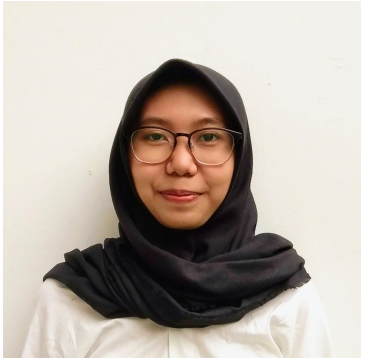
1. youtube.com (2020, 6 Juli). DES Algorithm | Working of DES | DES Encyption Process. Diakses pada 1 Oktober 2020 dari https://www.youtube.com/watch?v=cVhlCzmb-v0&t=1s

2. Tutorialpoint.com. Data Encryption Standard. Diakses pada 1 Oktober 2020 dari https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm

3. Page.math (2006). The DES Algorithm Illustrated. Diakses pada 30 Oktober 2020 dari http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

4. Rosen, K. H. (2006). Description of DES. D. R. Stinson (Ed.), *Cryptography Theory and Practice* (3rd ed., pp. 95-101). Florida, NY: Chapman & Hall.

Terima Kasih!



Nisa (10117004)

Rizvan (10117060)

Ezra (10117092)