

Enkripsi menggunakan *DES (Data Encryption Standard)*



Oleh :

- 1. Athirah Rashida Naima (105222027)**
- 2. Arshanda Geulis Nawajaputri (105222045)**
- 3. Jihan Fadila (105222022)**

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS SAINS DAN ILMU KOMPUTER
UNIVERSITAS PERTAMINA
2025
BAB 1

PENDAHULUAN

1.1 Pendahuluan

Keamanan informasi dalam era digital saat ini menjadi aspek penting yang tidak dapat diabaikan dalam pengembangan sistem teknologi informasi. Setiap hari, jutaan data pribadi dan rahasia bisnis berpindah tangan melalui berbagai media komunikasi elektronik, mulai dari email, pesan instan, transaksi perbankan online, hingga penyimpanan data di cloud. Tanpa adanya sistem keamanan yang memadai, informasi-informasi berharga ini dapat dengan mudah diakses oleh pihak-pihak yang tidak bertanggung jawab, yang tentunya dapat menimbulkan kerugian besar baik secara finansial maupun reputasi.

Salah satu metode yang telah lama digunakan untuk menjaga kerahasiaan data adalah Data Encryption Standard (DES). DES merupakan algoritma kriptografi simetris yang dikembangkan oleh IBM pada tahun 1970-an dan diadopsi sebagai standar enkripsi oleh National Institute of Standards and Technology (NIST) pada tahun 1977. Algoritma ini menggunakan kunci sepanjang 56-bit dan memproses data dalam blok 64-bit melalui 16 putaran transformasi yang kompleks, termasuk substitusi dan permutasi, untuk menghasilkan ciphertext yang aman [1].

Meskipun DES telah digantikan oleh algoritma yang lebih kuat seperti Advanced Encryption Standard (AES) dalam banyak aplikasi, penelitian dan implementasi DES masih relevan, terutama dalam konteks pendidikan dan sistem dengan sumber daya terbatas. Beberapa studi telah menunjukkan penerapan DES dalam berbagai konteks. Misalnya, penelitian oleh Widiarti dkk. mengimplementasikan DES untuk mengamankan data nilai siswa di SD Negeri 064979 Medan, memastikan bahwa data tersebut tidak dapat diakses atau dimanipulasi oleh pihak yang tidak berwenang [2]. Panjaitan et al. juga mengembangkan sistem enkripsi berbasis web menggunakan DES untuk mengamankan dokumen PDF, menyoroti fleksibilitas algoritma ini dalam berbagai platform [3].

Berdasarkan latar belakang tersebut, tujuan dari laporan ini adalah untuk mengimplementasikan algoritma DES dalam sistem keamanan data. Dengan memahami struktur dan mekanisme kerja DES, diharapkan dapat dibangun sistem enkripsi yang mampu melindungi data dari akses yang tidak sah, khususnya dalam lingkungan dengan keterbatasan sumber daya.

BAB 2

KAJIAN PUSTAKA

2.1 Struktur Umum

DES menggunakan struktur Feistel cipher dengan 16 putaran (rounds). Setiap putaran menggunakan subkey yang berbeda yang diturunkan dari kunci utama. Proses enkripsi melibatkan beberapa tahap utama:

- 1) Initial Permutation (IP): Permutasi awal pada blok 64-bit input
- 2) 16 Rounds Feistel: Setiap round terdiri dari:
 - Pembagian data menjadi left (L) dan right (R) 32-bit
 - Aplikasi F-function pada bagian kanan
 - XOR hasil F-function dengan bagian kiri
 - Pertukaran posisi left dan right
- 3) Final Permutation (FP): Permutasi akhir yang merupakan invers dari IP

2.2 Key Generation

Proses pembangkitan subkey melibatkan:

- 1) PC-1 (Permuted Choice 1): Memilih 56 bit dari 64-bit kunci utama
- 2) Left Circular Shifts: Pergeseran kiri pada setiap round
- 3) PC-2 (Permuted Choice 2): Menghasilkan subkey 48-bit untuk setiap round

2.3 F-Function

F-function adalah inti dari algoritma DES yang terdiri dari:

- 1) Expansion (E): Memperluas 32-bit menjadi 48-bit
- 2) XOR dengan Subkey: Operasi XOR dengan subkey 48-bit
- 3) S-Box Substitution: 8 S-box yang mengonversi 6-bit input menjadi 4-bit output
- 4) P-Permutation: Permutasi akhir pada hasil S-box

BAB 3

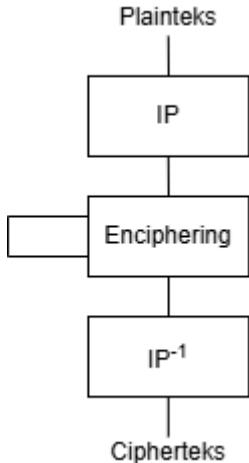
ISI DAN PEMBAHASAN

3.1 Skema Algoritma DES

Skema global dari algoritma DES adalah sebagai berikut:

1. Blok plainteks dipermutasi dengan matriks permutasi awal (initial permutation atau IP).

2. Hasil permutasi awal kemudian di-enciphering- sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil enciphering kemudian dipermutasi dengan matriks permutasi balikan (invers initial permutation atau IP-1) menjadi blok cipherteks.



3.2 Implementasi

3.1.1 Struktur kelas

Implementasi dibuat dalam bentuk kelas DES yang berisi seluruh tabel permutasi dan s-box, beserta fungsi-fungsi utama.

3.1.2 Fungsi Utama

1) Konversi Data

- `string_to_bits()`: Mengkonversi string ke array bit
- `bits_to_string()`: Mengkonversi array bit ke string
- `pad_data()`: Menambahkan padding untuk membuat data kelipatan 64-bit

2) Operasi Dasar

- `permute()`: Melakukan permutasi berdasarkan tabel
- `xor()`: Operasi XOR pada dua array bit
- `left_shift()`: Pergeseran kiri sirkuler

3) Key Generation

- `generate_keys()`: Membangkitkan 16 subkey dari kunci utama

4) Core Algorithm

- `f_function()`: Implementasi F-function
- `des_round()`: Satu putaran DES
- `encrypt_block()`: Enkripsi satu blok 64-bit
- `decrypt_block()`: Dekripsi satu blok 64-bit

3.1.3 Interface Publik

Dua fungsi utama yang dapat digunakan:

- encrypt_des(plaintext, key): Enkripsi teks dengan kunci
- decrypt_des(ciphertext_bits, key): Dekripsi hasil enkripsi

3.1.4 Penggunaan

```
if __name__ == "__main__":
    # Test enkripsi dan dekripsi
    plaintext = "TUBES? 17 HARI?"
    key = "CAPEKASLI" # 8-character key

    print("BASIC TEST (without detailed output):")
    print(f"Plaintext: {plaintext}")
    print(f"Key: {key}")

    # Basic encryption
    encrypted = encrypt_des(plaintext, key, verbose=False)
    print(f"Encrypted (bits): {len(encrypted)} bits")
    print(f"Encrypted (hex): {hex(int(''.join(map(str, encrypted)), 2))}")

    # Basic decryption
    decrypted = decrypt_des(encrypted, key, verbose=False)
    print(f"Decrypted: {decrypted}")
    print(f"Encryption successful: {plaintext == decrypted}")

    print("\n" + "="*80)
    print("DETAILED TEST (with step-by-step output):")
    print("="*80)

# Detailed decryption
print("\n")
decrypted_verbose = decrypt_des(encrypted_verbose, key, verbose=True)

print(f"\nFinal Result: {decrypted_verbose}")
print(f"Verification: {plaintext == decrypted_verbose}")
```

Enkripsi

Overview Hasil Enkripsi & Dekripsi

```
Plaintext: TUBES? 17 HARI?
Key: CAPEKASLI
Encrypted (bits): 128 bits
Encrypted (hex): 0xc5339268d4c6a58a0a9542e7dfe361ef
Decrypted: TUBES? 17 HARI?
Encryption successful: True
```

Key Preparation

Key 64 bit → PC1 56 bit → Split menjadi C₀(28 bit) dan D₀(28 bit)

Original key	:	0100 0011 0100 0001 0101 0000 0100 0101 0100 1011 0100 0001 0101 0011 0100 1100
After PC1	:	0000 0000 1111 1111 0000 0000 0100 0101 0001 1000 1000 1001 0000 0100
Left half (C ₀)	:	0000 0000 1111 1111 0000 0000 0100
Right half (D ₀)	:	0101 0001 1000 1000 1001 0000 0100

SubKey generation

Untuk setiap round (1-16):

- Left shift C dan D
- Gabung C dan D → PC2 → Subkey 48 bit

K1	:	1010 0000 1001 0010 0100 0010 0010 1001 0000 1000 0100 0110
K2	:	1011 0000 0001 0010 0101 0010 0100 0010 0000 0011 1010 0000
K3	:	0010 0100 0101 0010 0101 0000 1001 0000 0010 1001 0000 1001
K4	:	0000 0110 0101 0001 0101 0100 0110 0010 0001 0010 0001 0000
K5	:	0000 1110 0100 0001 0101 0001 0101 0001 0010 0001 0010 1010
K6	:	0000 1111 0100 0001 0010 1001 0010 0100 0001 1000 0000 1000
K7	:	1000 1011 0000 0001 1000 1001 0100 0000 0011 0000 0111 0110
K8	:	0001 1001 0000 1010 1000 1001 0010 0101 1000 1000 1010 1000
K9	:	0011 1001 0000 1000 1000 0000 0010 0110 0110 0000 1000
K10	:	0001 0000 0010 1000 1000 1100 1111 1000 0001 0001 0000 0000
K11	:	0001 0000 0010 1100 0001 0100 1000 0000 0100 0010 0010 1010
K12	:	0100 0100 0010 1100 0010 0100 0101 0100 0011 1010 0000 0000
K13	:	1100 0010 1010 0100 0010 0100 1011 0000 0000 0000 0111 1000
K14	:	1100 1000 1000 0110 0010 0010 0000 0001 1011 1010 0000 0010
K15	:	1110 0000 1001 0010 0010 1010 0011 0100 0010 0100 0011 0000
K16	:	1010 0000 1001 0010 1010 0010 1001 0010 0101 0000 0000 0101

PlainText : TUBES? 17 HARI?

Konversi Karakter ke Binary :

T = 84 = 01010100

U = 85 = 01010101

B = 66 = 01000010

E = 69 = 01000101

S = 83 = 01010011

? = 63 = 00111111

= 32 = 00100000

1 = 49 = 00110001

7 = 55 = 00110111

= 32 = 00100000

H = 72 = 01001000

A = 65 = 01000001

R = 82 = 01010010

I = 73 = 01001001

? = 63 = 00111111

Penggabungan bit :

01010100 01010101 01000010 01000101 01010011 00111111 00100000

00110001

01010100 01010101 01000010 01000101 01010011 00111111 00100000

00110001

Pembagian Blok (64 bit per blok):

1) Blok 1

01010100 01010101 01000010 01000101 01010011 00111111 00100000
00110001

T U B E S ? space 1

2) Blok 2

00110111 00100000 01001000 01000001 01010010 01001001 00111111
00000000

7 space H A R I ? padding

Proses Enkripsi Blok 1

Enkripsi Blok

Plaintext 64 bit → Initial Permutation (IP) 64 bit → Split menjadi L₀(32 bit) dan R₀(32 bit)

Input block	:	0101 0100 0101 0101 0100 0010 0100 0101 0101 0011 0011 1111 0010 0000 0011 0001
After IP	:	0001 1111 1011 0011 0010 1011 1011 1010 0000 0000 1110 0000 0010 0000 0011 0100
Initial Permutation (IP):		
L ₀	:	0001 1111 1011 0011 0010 1011 1011 1010
R ₀	:	0000 0000 1110 0000 0010 0000 0011 0100

Iterasi Round 16

- 1) Round Feitsel

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

- 2) F-Function

R (32 bit) + Subkey (48 bit) → Expansion (E): 32 bit ke 48 bit → XOR dengan Subkey: 48 bit \oplus 48 bit = 48 bit → S-Box: 48 bit → 32 bit (8 S-box, masing-masing 6 ke 4 bit) → Permutasi (P): 32 bit ke 32 bit

--- Round 1 ---		
Expanded R ₀ (E)	:	0000 0000 0001 0111 0000 0000 0001 0000 0000 0001 1010 1000
Subkey K ₁	:	1010 0000 1001 0010 0100 0010 0010 1001 0000 1000 0100 0110
After XOR (E \oplus K)	:	1010 0000 1000 0101 0100 0010 0011 1001 0000 1001 1110 1110
After S-Box	:	1101 0110 0101 1101 0110 0000 1000 0010
After P-box Perm	:	1100 0100 1000 0111 1001 0000 1101 0011
L ₁ = R ₀	:	0000 0000 1110 0000 0010 0000 0011 0100
R ₁ = L ₀ \oplus F	:	1101 1011 0011 0100 1011 1011 0110 1001
--- Round 2 ---		
Expanded R ₁ (E)	:	1110 1111 0110 1001 1010 1001 0101 1111 0110 1011 0101 0011
Subkey K ₂	:	1011 0000 0001 0010 0101 0010 0100 0010 0000 0011 1010 0000
After XOR (E \oplus K)	:	0101 1111 0111 1011 1111 1011 0001 1101 0110 1000 1111 0011
After S-Box	:	1011 1100 0111 0111 1100 0100 1011 1100
After P-box Perm	:	1000 1111 1100 1101 0001 0110 0011 1111
L ₂ = R ₁	:	1101 1011 0011 0100 1011 1011 0110 1001
R ₂ = L ₁ \oplus F	:	1000 1111 0010 1101 0011 0110 0000 1011
.....		
--- Round 16 ---		
Expanded R ₁₅ (E)	:	0111 1010 1010 1010 0101 0101 0100 0101 0001 0101 0000 1101
Subkey K ₁₆	:	1010 0000 1001 0010 1010 0010 1001 0010 0101 0000 0000 0101
After XOR (E \oplus K)	:	1011 1010 0011 1000 1111 0111 1101 0111 0100 0101 0000 1000
After S-Box	:	0111 1000 1010 1011 0000 0100 1001 0110
After P-box Perm	:	1000 0010 0100 1010 1000 0011 0110 1111
L ₁₆ = R ₁₅	:	1111 0101 0100 1010 1000 1000 1010 0110
R ₁₆ = L ₁₅ \oplus F	:	0011 1001 0001 0110 0111 0001 0100 0011

Finalisasi

Setelah 16 rounds: L₁₆, R₁₆ → Swap: R₁₆ + L₁₆ → Final Permutation (FP): 64 bit ke 64 bit (Ciphertext)

Before final perm	:	0011 1001 0001 0110 0111 0001 0100 0011 1111 0101 0100 1010 1000 1000 1010 0110
After final perm	:	1100 0101 0011 0011 1001 0010 0110 1000 1101 0100 1100 0110 1010 0101 1000 1010

Proses Enkripsi Blok 2

Enkripsi Blok

Input block	:	0011 0111 0010 0000 0100 1000 0100 0001 0101 0010 0100 1001 0011 1111 0000 0000
After IP	:	0011 1100 0101 0001 0100 0001 0110 1001 0000 0000 0100 0011 0110 0100 0101 0001
Initial Permutation (IP):		
L ₀	:	0011 1100 0101 0001 0100 0001 0110 1001
R ₀	:	0000 0000 0100 0011 0110 0100 0101 0001

Iterasi Round 16

```

--- Round 1 ---
Expanded R0 (E)      : 1000 0000 0000 0010 0000 0110 1011 0000 1000 0010 1010 0010
Subkey K1            : 1010 0000 1001 0010 0100 0010 1001 0000 1000 0100 0110
After XOR (E ⊕ K)   : 0010 0000 1001 0000 0100 0100 1001 1000 1000 1010 1110 0100
After S-Box          : 0010 1111 1101 1110 1011 1110 0100 0100
After P-box Perm    : 0111 0101 0111 1001 0101 0011 1111 1000
L1 = R0              : 0000 0000 0100 0011 0110 0100 0101 0001
R1 = L0 ⊕ F         : 0100 1001 0010 1000 0001 0010 1001 0001

--- Round 2 ---
Expanded R1 (E)      : 1010 0101 0010 1001 0101 0000 0000 1010 0101 0100 1010 0010
Subkey K2            : 1011 0000 0001 0010 0101 0010 0100 0010 0000 0011 1010 0000
After XOR (E ⊕ K)   : 0001 0101 0011 1011 0000 0010 0100 1000 0101 0111 0000 0010
After S-Box          : 0111 0000 0011 1101 0101 0100 0110 0010
After P-box Perm    : 1010 0100 0001 0110 1001 0110 0100 1110
L2 = R1              : 0100 1001 0010 1000 0001 0010 1001 0001
R2 = L1 ⊕ F         : 1010 0100 0101 0101 1111 0010 0001 1111

.....
--- Round 16 ---
Expanded R15 (E)     : 1101 1111 0101 0111 0101 0001 0100 1010 0011 1101 1111 1011
Subkey K16           : 1010 0000 1001 0010 1010 0010 1001 0010 0101 0000 0000 0101
After XOR (E ⊕ K)   : 0111 1111 1100 0101 1111 0011 1101 1000 0110 1101 1111 1110
After S-Box          : 1000 0010 1110 0100 0101 1111 1111 1000
After P-box Perm    : 0111 1010 1011 0101 0011 0101 0000 1101
L16 = R15            : 1011 1010 1110 1000 1001 0001 1011 1101
R16 = L15 ⊕ F       : 1111 1100 0001 0010 1001 1010 1111 1010

```

Finalisasi

Before final perm	: 1111 1100 0001 0010 1001 1010 1111 1010 1011 1010 1110 1000 1001 0001 1011 1101
After final perm	: 0000 1010 1001 0101 0100 0010 1110 0111 1101 1111 1110 0011 0110 0001 1110 1111

3.3 Analisis dan pembahasan

3.2.1 Keunggulan Implementasi

Kelengkapan Algoritma Implementasi ini mencakup semua komponen standar DES tanpa penyederhanaan. Semua tabel permutasi, S-box, dan jadwal pergeseran kunci mengikuti spesifikasi resmi FIPS 46-3. Hal ini memastikan kompatibilitas penuh dengan implementasi DES standar lainnya.

Modularitas Kode Struktur kelas yang digunakan memungkinkan pemisahan yang jelas antara berbagai komponen algoritma. Setiap fungsi memiliki tanggung jawab yang spesifik, sehingga memudahkan pemahaman, testing, dan maintenance kode.

Fleksibilitas Input Implementasi dapat menangani teks dengan panjang arbitrary melalui sistem padding otomatis. Data akan secara otomatis di-pad menjadi kelipatan 64-bit sebelum diproses.

3.2.2 Kompleksitas Algoritma

1) Kompleksitas Waktu

- Key generation: $O(1)$ untuk setiap subkey, $O(16) = O(1)$ untuk semua subkey
- Enkripsi/dekripsi per blok: $O(1)$ karena jumlah operasi tetap (16 rounds)
- Total untuk n bit data: $O(n/64) = O(n)$

2) Kompleksitas Penyimpanan

- Penyimpanan tabel: $O(1)$ - ukuran tabel tetap

- Penyimpanan subkey: $O(1)$ - 16 subkey dengan ukuran tetap
- Buffer data: $O(n)$ - sesuai ukuran input

BAB 4

PENUTUP

4.1 Kesimpulan

Berdasarkan hasil yang diperoleh, bahwa memiliki beberapa poin kesimpulan yaitu:

1. Data Encryption Standard (DES) merupakan algoritma enkripsi simetris klasik yang menggunakan struktur *Feistel cipher* dengan 16 putaran enkripsi, serta kunci sepanjang 56 bit efektif.

2. Kekuatan kunci yang digunakan menentukan seberapa aman algoritma DES. Tetapi DES sekarang rentan terhadap serangan karena kurang kuat dalam kriptanalisis dan perhitungan komputasi
3. Hasil yang didapatkan menguji seberapa baik kinerja algoritma DES ketika mengenkripsi plaintext menjadi ciphertext dengan menggunakan berbagai macam teknik, termasuk pembuatan kunci internal, permutasi awal, kompresi permutasi dan fungsi pengganti
4. Dalam pembuatan DES, menggunakan kunci “CAPEKASLI” dan plainteks “TUBES? 17 HARI?” dan menghasilkan “After final perm : 0011 0111 0010 0000 0100 1000 0100 0001 0101 0010 0100 1001 0011 1111 0000 0000”.

DAFTAR PUSTAKA

- [1] P. Loshin dan M. Cobb, "Data Encryption Standard (DES)," *TechTarget*, 2025. [Online]. Tersedia: <https://www.techtarget.com/searchsecurity/definition/Data-Encryption-Standard/>
- [2] Widiarti R. M., Azanuddin, dan Elfitriani, "Implementasi Kriptografi Pengamanan Data Nilai Siswa Menggunakan Algoritma DES," *Jurnal SAINTIKOM*, vol. 21, no. 1, 2022. [Online]. Tersedia: <https://ojs.trigunadharma.ac.id/index.php/jis/article/view/4764E-Journal> STMIK Triguna Dharma+2E-Journal STMIK Triguna Dharma+2SLOT GACOR+2

- [3] A. W. Panjaitan, I. Zufria, dan Y. R. Nasution, "Implementation of Data Encryption Standard (DES) Algorithm for Data Security on PDF Documents," *ZERO: Jurnal Sains, Matematika dan Terapan*, vol. 6, no. 2, 2022. [Online]. Tersedia: <http://dx.doi.org/10.30829/zero.v6i2.17365>

LAMPIRAN

<https://github.com/athirahtira/Kriptografi>

https://github.com/ArshandaGN/Kuis3_Kripto.git