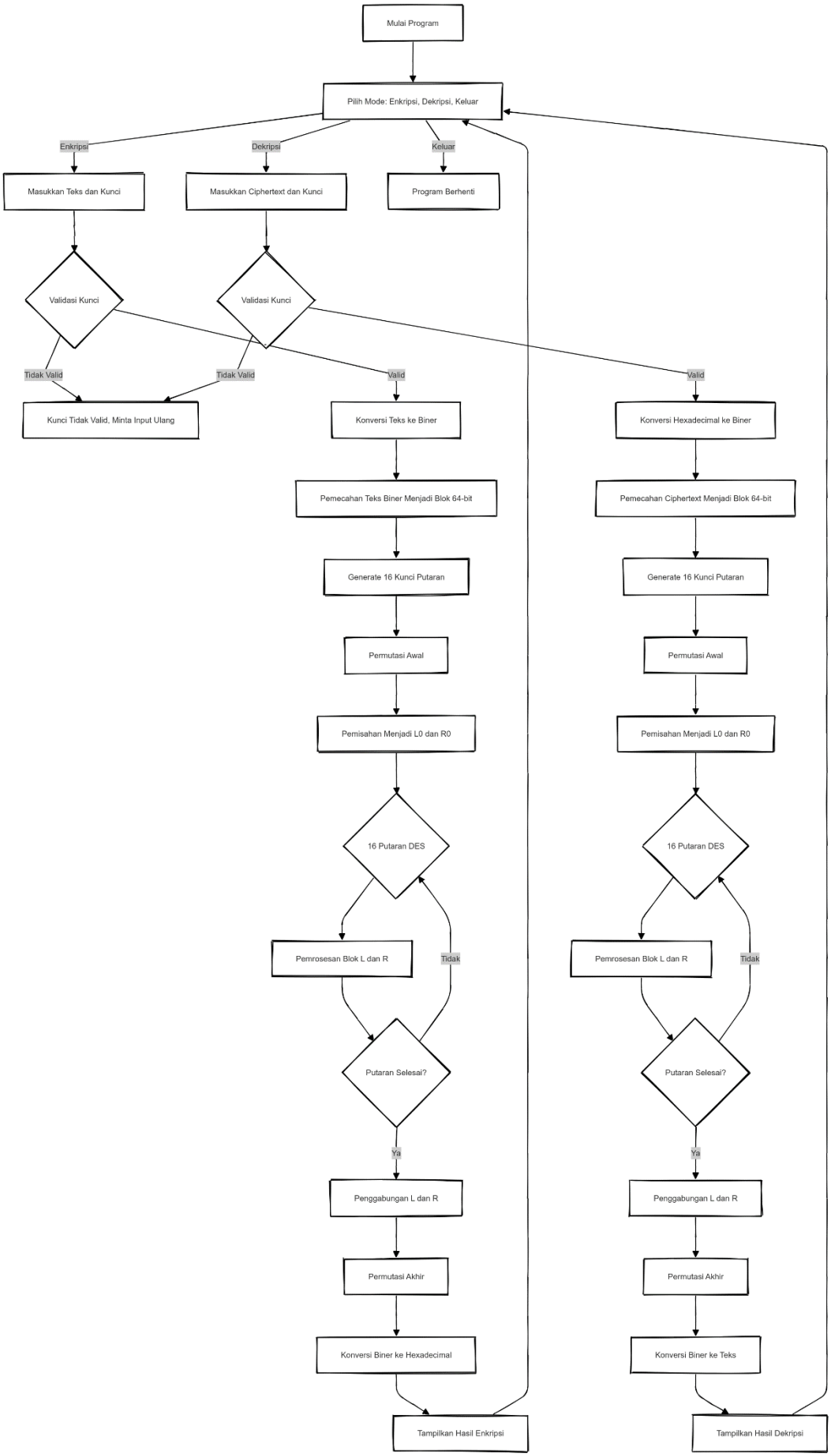
Dokumentasi Program DES Tanpa Library

Algoritma DES (Data Encryption Standard) adalah algoritma enkripsi simetris yang digunakan untuk mengamankan data dengan cara membagi teks asli menjadi blok-blok dan memprosesnya menggunakan serangkaian operasi matematika.

A. Perancangan Sistem

Flowchart Program Des Enkripsi % Dekripsi + Key


Mulai Program
Pilih Mode: Enkripsi, Dekripsi, Keluar
Enkripsi
Dekripsi
Keluar
Masukkan Teks dan Kunci
Masukkan Ciphertext dan Kunci
Program Berhenti
Validasi Kunci
Validasi Kunci
Tidak Valid
Tidak Valid
Valid
Valid
Kunci Tidak Valid, Minta Input Ulang
Konversi Teks ke Biner
Konversi Hexadecimal ke Biner
Pemecahan Teks Biner Menjadi Blok 64-bit
Pemecahan Ciphertext Menjadi Blok 64-bit
Generate 16 Kunci Putaran
Generate 16 Kunci Putaran
Permutasi Awal
Permutasi Awal
Pemisahan Menjadi L0 dan R0
Pemisahan Menjadi L0 dan R0
16 Putaran DES
16 Putaran DES
Pemrosesan Blok L dan R
Tidak
Pemrosesan Blok L dan R
Tidak
Putaran Selesai?
Putaran Selesai?
Ya
Ya
Penggabungan L dan R
Penggabungan L dan R
Permutasi Akhir
Permutasi Akhir
Konversi Biner ke Hexadecimal
Konversi Biner ke Teks
Tampilkan Hasil Enkripsi
Tampilkan Hasil Dekripsi

B. Implementasi

1. Tabel Permutasi Awal (IP)

```
# Tabel permutasi awal
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]
```

- IP (Initial Permutation) adalah tabel yang digunakan untuk mempermutasikan (mengatur ulang) bit-bit dari blok data awal.
- Setiap angka di tabel menunjukkan posisi baru dari bit setelah permutasi. Misalnya, bit ke-58 dari input akan ditempatkan di posisi pertama dari output.

2. Tabel Permutasi Invers (IP_INV)

```
# Tabel permutasi akhir (inverse IP)
IP_INV = [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]
```

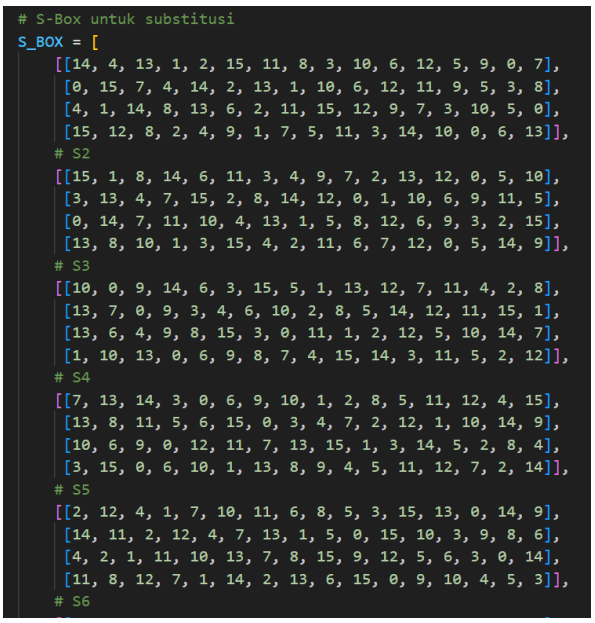
- IP_INV adalah tabel yang membalikkan permutasi awal.
- digunakan pada akhir proses enkripsi untuk mengembalikan bit-bit ke urutan semula mereka setelah semua proses enkripsi dilakukan.

3. Ekspansi ke 48-bit (Expansion Table)

```
# Ekspansi ke 48-bit
E_BOX = [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 ini digunakan untuk memperluas 32-bit menjadi 48-bit dengan cara menggandakan beberapa bit, yang akan memungkinkan penggunaan kunci 48-bit pada langkah berikutnya.

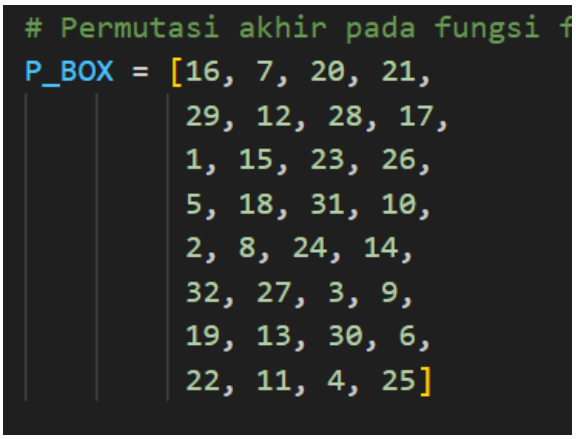
4. S-Box untuk Substitusi



```
# S-Box untuk substitusi
S_BOX = [
    [[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]],
    # S2
    [[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]],
    # S3
    [[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]],
    # S4
    [[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]],
    # S5
    [[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]],
    # S6
```

- S-Box adalah tabel substitusi yang digunakan untuk mengganti 6 bit input dengan 4 bit output. Ada 8 S-Box yang berbeda yang digunakan dalam proses DES, dan setiap S-Box berfungsi pada bagian tertentu dari data yang diolah.

5. Permutasi akhir pada fungsi f



```
# Permutasi akhir pada fungsi f
P_BOX = [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 P digunakan dalam fungsi f untuk menyusun kembali 32 bit output dari S-Box sebelum dikombinasikan dengan bagian kanan dari data.

6. Permutasi Kunci Awal (PC1)

```
# Permutasi kunci awal
PC1 = [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]
```

- Permutasi ini digunakan untuk mengubah kunci 64 bit menjadi dua bagian 56 bit yang digunakan dalam proses enkripsi.

7. Permutasi Kunci Kedua (PC2)

```
# Permutasi kunci kedua
PC2 = [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]
```

- digunakan untuk memilih 48 bit dari 56 bit setelah proses penggeseran dalam pembuatan subkunci.

8. Tabel Pergeseran (Shift Table)

```
# Fungsi untuk mengatur pergeseran (shifts) pada kunci
SHIFT_TABLE = [1, 1, 2, 2, 2, 2, 2, 2,
               1, 2, 2, 2, 2, 2, 2, 1]
```

- Tabel ini mendefinisikan jumlah bit yang harus digeser untuk setiap ronde enkripsi. Ada 16 ronde dalam DES, dan setiap ronde memiliki jumlah pergeseran tertentu.

9. Fungsi Permutasi

```
# Fungsi permutasi menggunakan tabel permutasi yang diberikan
def permute(block, table):
    return [block[i - 1] for i in table]
```

- Fungsi ini melakukan permutasi pada data menggunakan tabel permutasi yang diberikan. block adalah data yang akan dipermutasikan, dan table adalah tabel permutasi yang sesuai.

10. Fungsi XOR

```
# Fungsi untuk melakukan XOR dua list biner
def xor(a, b):
    return [i ^ j for i, j in zip(a, b)]
```

- Fungsi ini melakukan operasi XOR antara dua list biner a dan b.
- Operasi XOR mengembalikan 1 jika hanya satu dari dua bit adalah 1, dan 0 jika keduanya sama.

11. Fungsi Split

```
# Fungsi untuk melakukan pemisahan blok
def split(block):
    return block[:len(block) // 2], block[len(block) // 2:]
```

- Fungsi ini membagi blok menjadi dua bagian yang sama.
- Blok 64-bit dibagi menjadi L (kiri) dan R (kanan).

12. Fungsi Rotasi Kiri

```
# Fungsi rotasi kiri (left circular shift)
def rotate_left(block, n):
    return block[n:] + block[:n]
```

- Fungsi ini menggeser semua bit ke kiri sebanyak n posisi.
- Bit yang terpotong di kiri ditambahkan ke bagian kanan untuk menjaga panjang yang sama.

13. Fungsi Konversi Teks ke Biner

```
# Fungsi konversi teks ke biner (string ke list of integers)
def string_to_bits(text):
    bits = []
    for char in text:
        binval = bin(ord(char))[2:].rjust(8, '0')
        bits.extend([int(x) for x in binval])
    return bits
```

- Fungsi ini mengubah setiap karakter dalam string menjadi representasi biner.
- Setiap karakter diubah menjadi 8 bit, dan semua bit digabungkan menjadi satu list.

14. Fungsi Konversi Biner ke Teks

```
# Fungsi konversi biner ke teks (list of integers ke string)
def bits_to_string(bits):
    chars = []
    for b in range(len(bits) // 8):
        byte = bits[b * 8:(b + 1) * 8]
        chars.append(chr(int(''.join([str(bit) for bit in byte]), 2)))
    return ''.join(chars)
```

- Fungsi ini mengubah list biner kembali menjadi string.

- Setiap 8 bit diambil untuk membentuk satu karakter, yang kemudian digabungkan menjadi satu string.

15. Fungsi Membuat Kunci

```
# Fungsi untuk membuat kunci dari input string
def generate_keys(key):
    key = string_to_bits(key)
    key = permute(key, PC1)
    C, D = split(key)
    keys = []
    for shift in SHIFT_TABLE:
        C = rotate_left(C, shift)
        D = rotate_left(D, shift)
        combined_key = C + D
        round_key = permute(combined_key, PC2)
        keys.append(round_key)
    return keys
```

- Fungsi ini menghasilkan 16 kunci putaran dari kunci awal.
- Kunci diubah menjadi biner, dipermutasikan dengan PC1, dan dibagi menjadi dua bagian (C dan D), lalu dirotasi dan dipermutasikan lagi dengan PC2 untuk mendapatkan kunci setiap putaran.

16. Fungsi Substitusi S-Box

```
# Fungsi substitusi S-Box
def s_box_substitution(block):
    subblocks = [block[k * 6:(k + 1) * 6] for k in range(8)]
    result = []
    for i, subblock in enumerate(subblocks):
        row = int(f"{subblock[0]}{subblock[5]}", 2)
        col = int(''.join([str(x) for x in subblock[1:5]]), 2)
        s_val = S_BOX[i][row][col]
        binval = bin(s_val)[2:].rjust(4, '0')
        result.extend([int(x) for x in binval])
```

- Fungsi ini mengganti setiap subblok 6-bit dengan output 4-bit menggunakan S-Box.
- Subblok dipisahkan, nilai baris dan kolom dihitung, dan hasil dari S-Box ditambahkan ke output.

17. Fungsi f (Transformasi)

```
# Fungsi f untuk menggabungkan dengan kunci dan melakukan transformasi
def function_f(R, K):
    expanded_R = permute(R, E_BOX)
    xor_result = xor(expanded_R, K)
    substituted = s_box_substitution(xor_result)
    return permute(substituted, P_BOX)
```

- Fungsi ini menggabungkan R dengan kunci untuk menghasilkan output untuk putaran enkripsi.
- Ini mencakup ekspansi, XOR dengan kunci, substitusi dengan S-Box, dan akhirnya permutasi menggunakan P-Box.

18. Proses Enkripsi Satu Blok

```
# Proses enkripsi satu blok 64-bit menggunakan 16 putaran DES
def des_encrypt_block(block, keys):
    block = permute(block, IP)
    L, R = split(block)
    for i in range(16):
        L, R = R, xor(L, function_f(R, keys[i]))
    return permute(R + L, IP_INV)
```

- Fungsi ini mengenkripsi satu blok 64-bit.
- Setelah permutasi awal, dilakukan 16 putaran di mana L dan R diperbarui. Hasil akhir juga dipermutasikan kembali menggunakan IP_INV.

19. Proses Dekripsi Satu Blok

```
# Proses dekripsi satu blok 64-bit menggunakan 16 putaran DES
def des_decrypt_block(block, keys):
    block = permute(block, IP)
    L, R = split(block)
    for i in range(15, -1, -1):
        L, R = R, xor(L, function_f(R, keys[i]))
    return permute(R + L, IP_INV)
```

- Fungsi ini mendekripsi satu blok 64-bit dengan cara yang mirip dengan enkripsi, tetapi menggunakan kunci dari belakang (kunci terakhir ke yang pertama).

20. Konversi Biner ke Hexadecimal

```
# Fungsi untuk mengubah biner ke representasi hexadecimal
def bits_to_hex(bits):
    bit_string = ''.join(str(bit) for bit in bits)
    hex_string = hex(int(bit_string, 2))[2:]  # Mengubah ke hexadecimal
    return hex_string.upper()  # Uppercase untuk konsistensi
```

- Fungsi ini mengubah list biner menjadi string hexadecimal untuk representasi

21. Konversi Hexadecimal ke Biner

```
# Fungsi untuk mengubah representasi hexadecimal kembali ke biner
def hex_to_bits(hex_string):
    bit_string = bin(int(hex_string, 16))[2:]  # Mengubah ke biner
    padded_bit_string = bit_string.zfill(len(hex_string) * 4)  # Padding
    return [int(bit) for bit in padded_bit_string]
```

- Fungsi ini mengubah string hexadecimal kembali menjadi biner, dengan padding untuk memastikan panjangnya sesuai dengan 4 bit per digit heksadesimal.

22. Fungsi Utama untuk Enkripsi

```python
# Fungsi utama untuk enkripsi pesan teks menggunakan kunci
def des_encrypt(plaintext, key):
    keys = generate_keys(key)
    plaintext_bits = string_to_bits(plaintext)
    while len(plaintext_bits) % 64 != 0:
        plaintext_bits.extend([0])  # Padding dengan 0 jika tidak cukup
    ciphertext_bits = []
    for i in range(0, len(plaintext_bits), 64):
        block = plaintext_bits[i:i + 64]
        encrypted_block = des_encrypt_block(block, keys)
        ciphertext_bits.extend(encrypted_block)
    # Encode as Hexadecimal
    return bits_to_hex(ciphertext_bits)
```

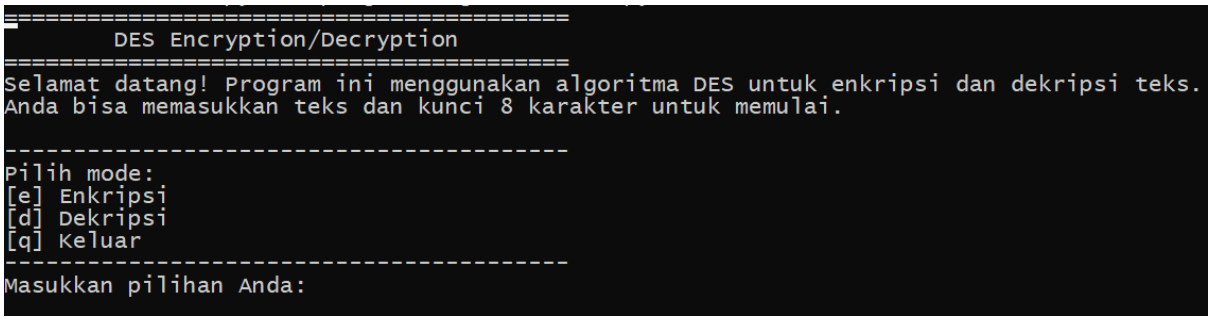
- Fungsi ini mengenkripsi pesan teks menggunakan kunci yang diberikan.
- Setelah mengubah plaintext menjadi biner, ia dibagi menjadi blok 64-bit. Setiap blok dienkripsi, dan hasilnya diubah ke format hexadecimal.

23. Fungsi Utama untuk Dekripsi

```python
# Fungsi utama untuk dekripsi pesan teks menggunakan kunci
def des_decrypt(ciphertext, key):
    keys = generate_keys(key)
    # Decode from Hexadecimal
    decoded_bits = hex_to_bits(ciphertext)
    plaintext_bits = []
    for i in range(0, len(decoded_bits), 64):
        block = decoded_bits[i:i + 64]
        decrypted_block = des_decrypt_block(block, keys)
        plaintext_bits.extend(decrypted_block)
    return bits_to_string(plaintext_bits)
```

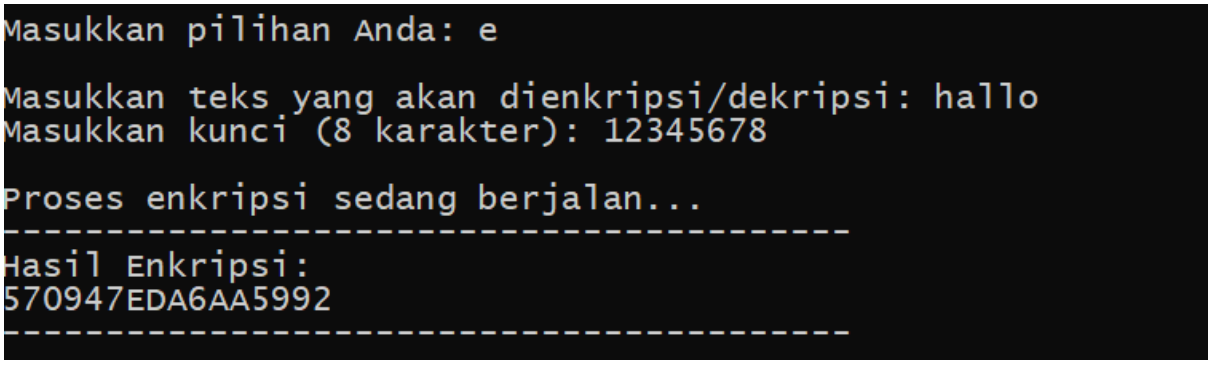
- Fungsi ini mendekripsi ciphertext dengan menggunakan kunci yang sama.
- Ciphertext dikonversi dari format hexadecimal ke biner, dibagi menjadi blok, dan setiap blok didekripsi untuk menghasilkan plaintext.

C. Hasil dan Pengujian

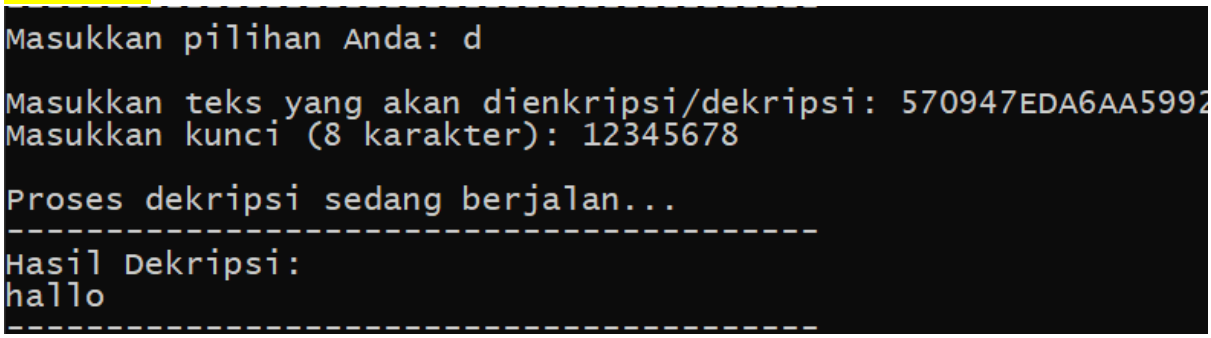


ENKRIPSI : text hallo

Key : 12345678



DEKRIPSI :



D. Running Program

1. save file ke directori C
2. Open cmd admin ketik : cd C:\
3. Ketik : python program_algoritma_des.py
4. Input pilihan