Analisis Implementasi Enkripsi DES dalam Python: Tahapan, Penyederhanaan, dan Hasil

Oleh :

Fauzan Azhima (105222003)

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN ILMU KOMPUTER

UNIVERSITAS PERTAMINA

2025

Analisis Implementasi Enkripsi DES dalam Python: Tahapan, Penyederhanaan, dan Hasil

1. Pendahuluan

Data Encryption Standard (DES) adalah algoritma enkripsi kunci simetris yang banyak digunakan di masa lalu. Algoritma ini mengenkripsi data dalam blok 64-bit menggunakan kunci 56-bit (disajikan sebagai 64-bit dengan bit paritas). Meskipun sekarang dianggap tidak aman terhadap serangan *brute-force* karena panjang kuncinya yang relatif pendek, pemahaman cara kerja DES penting sebagai dasar untuk memahami kriptografi modern.

Laporan ini akan menganalisis implementasi DES yang disederhanakan berdasarkan kode Python yang diberikan, merinci tahapan enkripsi yang didefinisikan dalam kode, serta menyajikan hasil enkripsi aktual yang dihasilkan oleh skrip tersebut.

2. Analisis Kode Enkripsi DES yang Disediakan

Kode Python yang diberikan mendefinisikan beberapa konstanta (tabel permutasi seperti IP, FP, EXPANSION, P) dan fungsi-fungsi yang bertujuan untuk melakukan enkripsi DES. Mari kita bedah tahapan yang *seharusnya* terjadi berdasarkan fungsi-fungsi tersebut.

A. Fungsi-Fungsi Pembantu dan Komponen DES yang Didefinisikan:

- IP (Initial Permutation): Tabel untuk permutasi awal blok plaintext.
- FP (Final Permutation): Tabel untuk permutasi akhir, invers dari IP.
- EXPANSION: Tabel untuk memperluas blok 32-bit kanan (R) menjadi 48-bit dalam fungsi Feistel.
- P (P-Box Permutation): Tabel untuk permutasi dalam fungsi Feistel setelah substitusi S-Box. (Catatan: Tabel P ini didefinisikan tetapi tidak digunakan dalam fungsi des_round yang ada).
- string_to_bit_array(text): Mengubah string teks menjadi array bit (setiap karakter menjadi 8 bit).
- bit_array_to_string(array): Mengubah array bit kembali menjadi string teks.
- permute(block, table): Melakukan permutasi pada block berdasarkan table.
- xor(t1, t2): Melakukan operasi XOR antara dua array bit.
- des_round(L, R, key_bits_for_round): Mensimulasikan satu putaran DES.
- encrypt_block(block_bits, key_bits_full): Mengenkripsi satu blok 64-bit.

- des_encrypt(plaintext, key): Fungsi utama yang dimaksudkan untuk mengenkripsi plaintext.

B. Tahapan Proses Enkripsi (Ideal, Berdasarkan encrypt_block **dan** des_round**):**

Berikut adalah tahapan yang akan dilalui oleh sebuah blok plaintext jika fungsi encrypt_block dipanggil dengan benar:

1. Konversi Plaintext ke Bit Array:
    - Plaintext (misalnya, "password") yang berupa string 8 karakter (64 bit) diubah menjadi representasi biner menggunakan string_to_bit_array.
2. Permutasi Awal (IP):
    - Blok plaintext 64-bit yang sudah dalam bentuk bit array kemudian diacak urutan bitnya sesuai dengan tabel IP.
    - Contoh: permuted_block = permute(block_bits, IP)
3. Pembagian Blok:
    - Blok 64-bit hasil permutasi awal dibagi menjadi dua bagian:
        - L (Left): 32 bit pertama.
        - R (Right): 32 bit terakhir.
4. Satu Putaran Fungsi DES (Feistel Network) - Disederhanakan:
    - Kode ini hanya melakukan satu putaran melalui fungsi des_round. DES standar melakukan 16 putaran.
    - Tahapan dalam des_round(L, R, key_bits_for_round):
        - a. Ekspansi (Expansion): Bagian R (32 bit) diperluas menjadi 48 bit menggunakan tabel EXPANSION.
            - expanded_R = permute(R, EXPANSION)
        - b. XOR dengan Kunci Putaran: Hasil ekspansi 48-bit di-XOR-kan dengan key_bits_for_round (dalam kode ini, key_bits_full yang merupakan kunci utama 64-bit, bukan subkunci 48-bit yang dijadwalkan).
            - temp = xor(expanded_R, key_bits_for_round)
        - c. Simulasi S-Box dan P-Box:

            - DES standar menggunakan S-Box (Substitution Boxes) untuk substitusi non-linear dan P-Box (Permutation Box) untuk permutasi.
            - Dalam kode ini, langkah S-Box dan P-Box disimulasikan dengan sangat sederhana dengan hanya mengambil 32 bit pertama dari hasil XOR temp:
                - s_box_p_box_simulated_output = temp[:32]
            - Ini berarti tidak ada operasi S-Box atau P-Box aktual yang terjadi sesuai standar DES. Tabel P yang didefinisikan tidak digunakan di sini.
        - d. XOR dengan Bagian Kiri (L): Output 32-bit dari simulasi S-Box/P-Box (s_box_p_box_simulated_output) di-XOR-kan dengan bagian L (32 bit kiri dari input putaran). Hasilnya menjadi new_R (bagian kanan baru).
            - new_R = xor(L, s_box_p_box_simulated_output)
        - e. Pembaruan Bagian Kiri (L): Bagian L yang baru (new_L) adalah bagian R dari input putaran (sebelum diubah).
            - new_L = R
    - Output dari des_round adalah new_L dan new_R.
5. Penggabungan Blok Setelah Putaran:
    - Setelah satu putaran, L dan R diperbarui. Dalam fungsi encrypt_block, setelah L, R = des_round(L, R, key_bits_full), blok digabungkan kembali dengan urutan R diikuti L (karena L menjadi R_lama dan R menjadi L_lama XOR f(R_lama, K)).
    - final_block_before_fp = R + L
6. Permutasi Akhir (FP):
    - Blok 64-bit yang telah digabungkan (final_block_before_fp) kemudian diacak kembali urutan bitnya menggunakan tabel FP. Tabel FP adalah invers dari IP.
    - result = permute(final_block_before_fp, FP)
    - Ini adalah output ciphertext dalam bentuk bit array.
7. Konversi Ciphertext Bit ke String:

- Idealnya, result (bit array) akan dikonversi kembali menjadi string karakter menggunakan bit_array_to_string.

C. Penyederhanaan Signifikan dan Perbedaan dari DES Standar dalam Kode:

- Hanya Satu Putaran: Fungsi encrypt_block hanya memanggil des_round satu kali. DES standar memiliki 16 putaran.
- Tidak Ada Jadwal Kunci (Key Schedule): Kunci key_bits_full (diasumsikan 64-bit dari konversi string kunci 8 karakter) digunakan secara langsung dalam des_round. DES standar memiliki algoritma penjadwalan kunci yang kompleks untuk menghasilkan 16 subkunci 48-bit yang berbeda untuk setiap putaran dari kunci utama 56-bit.
- S-Box dan P-Box Tidak Diimplementasikan dengan Benar: Bagian paling krusial dari keamanan DES, yaitu S-Box (untuk non-linearitas) dan P-Box (untuk difusi) dalam fungsi Feistel, hanya disimulasikan dengan mengambil 32 bit pertama dari hasil XOR. Ini menghilangkan properti kriptografi penting dari DES. Tabel P yang didefinisikan tidak digunakan dalam des_round.
- Penggunaan Kunci: Fungsi des_round menerima key_bits_for_round. Namun, encrypt_block memanggilnya dengan key_bits_full. Jika key_bits_full adalah hasil konversi string_to_bit_array dari kunci 8 karakter, maka itu adalah 64 bit. Operasi XOR dalam des_round akan menggunakan 48 bit pertama dari key_bits_full karena expanded_R adalah 48 bit.

3. Fungsi Utama des_encrypt **dan Perilaku Aktual Kode**

Meskipun terdapat fungsi-fungsi yang mendefinisikan langkah-langkah DES, fungsi des_encrypt(plaintext, key) yang menjadi titik masuk utama untuk enkripsi memiliki perilaku yang berbeda:

Python

```
def des_encrypt(plaintext, key):
  if len(key) != 8:
    raise ValueError("Kunci harus 8 karakter!")
  if len(plaintext) != 8:
    raise ValueError("Plaintext harus 8 karakter (64 bit) untuk versi sederhana ini.")
  return "Database" # <- Perhatikan baris ini
```

Setelah melakukan validasi panjang input untuk key (harus 8 karakter) dan plaintext (harus 8 karakter), fungsi ini langsung mengembalikan string literal "Database".

Ini berarti:

- Tidak ada konversi plaintext ke bit array yang terjadi melalui string_to_bit_array.
- Fungsi encrypt_block yang berisi logika permutasi dan putaran DES tidak pernah dipanggil oleh des_encrypt.
- Akibatnya, seluruh proses DES yang didefinisikan (IP, pembagian L/R, putaran Feistel, FP) tidak dijalankan untuk menghasilkan output enkripsi.

4. Hasil Enkripsi Aktual Berdasarkan Eksekusi Kode

Karena fungsi des_encrypt selalu mengembalikan string "Database" setelah validasi input berhasil, maka output enkripsi akan selalu sama, terlepas dari isi plaintext atau kunci (selama panjangnya sesuai).

Eksekusi kode pada if __name__ == "__main__": akan menghasilkan:

Plaintext: password

Kunci: 12345678

Hasil enkripsi: Database

Plaintext: abcdefgh

Kunci: 12345678

Hasil enkripsi: Database

Hasil enkripsi yang ditampilkan adalah "Database" untuk kedua kasus tersebut.

5. Kesimpulan

Kode Python yang diberikan mendefinisikan struktur dan beberapa komponen dasar yang mirip dengan algoritma DES, seperti permutasi dan kerangka fungsi putaran. Namun, implementasinya memiliki beberapa penyederhanaan signifikan dibandingkan DES standar, terutama pada jumlah putaran, tidak adanya jadwal kunci, dan simulasi yang sangat kasar pada S-Box dan P-Box.

Lebih penting lagi, fungsi des_encrypt yang seharusnya menjadi pengendali utama proses enkripsi tidak memanfaatkan fungsi-fungsi enkripsi DES yang telah didefinisikan tersebut.

Sebaliknya, ia secara langsung mengembalikan nilai string tetap ("Database") setelah validasi input. Oleh karena itu, hasil enkripsi yang diperoleh dari skrip ini bukanlah hasil dari proses DES yang sebenarnya, melainkan output statis yang telah ditentukan dalam fungsi des_encrypt.

Jika tujuannya adalah untuk mengimplementasikan DES, fungsi des_encrypt perlu dimodifikasi untuk memanggil string_to_bit_array, encrypt_block (yang mungkin juga perlu diperbaiki untuk melakukan 16 putaran dan menggunakan jadwal kunci), dan bit_array_to_string dengan benar.

Link Github : Fauzan-Azh/DES-Algorithm