PENGEMBANGAN APLIKASI KRIPTOGRAFI SEDERHANA MENGGUNAKAN ALGORITMA AES-256 (CBC MODE)

Disusun Oleh:

Amalia Ananda Putri 103052330078

KEAMANAN DATA
PROGRAM STUDI S1 SAINS DATA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2025

DAFTAR ISI

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi digital membuat pertukaran data menjadi semakin cepat dan terbuka. Hal ini tentunya dapat membawa banyak manfaat, tetapi di sisi lain juga dapat meningkatkan risiko kebocoran dan penyalahgunaan data. Informasi-informasi yang bersifat sensitif, seperti data pribadi, dokumen pekerjaan, laporan keuangan, hingga dataset penelitian perlu dilindungi agar tidak mudah diakses oleh pihak yang tidak berwenang.

Salah satu cara utama untuk melindungi atau menjaga keamanan data tersebut adalah dengan menggunakan kriptografi. Terdapat berbagai macam algoritma kriptografi, salah satunya yang paling banyak digunakan saat ini adalah AES (*Advanced Encryption Standard*). Algoritma AES dikenal stabil, aman, dan sudah menjadi standar internasional, khususnya untuk keperluan enkripsi data dalam berbagai aplikasi modern. Di antara berbagai variannya, AES-256 dengan mode CBC merupakan salah satu algoritma AES yang menawarkan tingkat keamanan tinggi karena menggunakan kunci (*key*) 256-bit dan bekerja dalam bentuk blok data. Algoritma ini juga banyak digunakan dalam aplikasi sehari-hari, seperti enkripsi file, koneksi jaringan, hingga penyimpanan data di perangkat.

Melalui tugas ini, akan dijelaskan cara kerja algoritma AES-256 CBC secara lebih mendalam dan bagaimana algoritma tersebut diimplementasikan ke dalam sebuah program sederhana. Dengan membuat aplikasi sendiri, dapat dilihat langsung proses enkripsi dan dekripsi, termasuk bagaimana peran kunci (*key*), IV, dan *padding* digunakan dalam algoritma ini.

1.2 Tujuan

1. Mempelajari konsep dasar enkripsi simetris, khususnya AES-256 dalam mode CBC.
2. Menjelaskan langkah-langkah kerja algoritma AES-256 CBC.
3. Membuat program Python yang dapat melakukan enkripsi dan dekripsi teks maupun *file*.
4. Memahami cara kerja *key*, IV, dan *padding* dalam proses enkripsi.
5. Menguji hasil enkripsi - dekripsi untuk memastikan bahwa program berjalan dengan benar.
6. Menghubungkan pemahaman teori dengan implementasi nyata dalam bentuk aplikasi sederhana.

PEMBAHASAN

2.1 Algoritma Kriptografi AES-256 (CBC Mode)

a. Prinsip Dasar

Berdasarkan dokumen resmi *Advanced Encryption Standard* (AES) yang diterbitkan oleh *National Institute of Standards and Technology* (NIST), *Advanced Encryption Standard* (AES) menetapkan sebuah algoritma kriptografi yang disetujui FIPS yang dapat digunakan untuk melindungi data elektronik. Algoritma AES adalah *symmetric block cipher* (sandi blok simetris) yang dapat mengenkripsi (mengubah menjadi sandi) dan mendekripsi (mengubah dari sandi) informasi digital. Algoritma AES mampu menggunakan kunci kriptografi 128, 192, dan 256 bit untuk mengenkripsi dan mendekripsi data dalam blok 128-bit.

AES merupakan algoritma kriptografi simetris yang digunakan untuk mengamankan data dengan cara mengubah *plaintext* menjadi *ciphertext* menggunakan sebuah kunci (*key*). AES ditetapkan oleh NIST sebagai standar enkripsi pada tahun 2001 setelah melalui kompetisi panjang untuk menggantikan *Data Encryption Standard* (DES) yang saat itu sudah dianggap tidak aman lagi karena ukuran kuncinya terlalu kecil.

Dalam algoritma AES, istilah simetris berarti kunci yang dipakai untuk mengenkripsi juga akan menjadi kunci yang dipakai untuk mendekripsi. Jadi, selama kedua pihak memiliki kunci yang sama, mereka bisa saling bertukar data dengan aman. Jenis kriptografi ini biasanya lebih cepat dibanding kriptografi asimetris dan sering dipakai untuk enkripsi file, *database*, komunikasi jaringan, dan berbagai sistem keamanan modern.

AES bekerja sebagai *block cipher*, yaitu memproses data dalam blok-blok berukuran 128-bit. Walaupun *block size*-nya tetap 128-bit, ukuran kunci bisa berbeda-beda (128-bit, 192-bit, dan 256-bit). Pada laporan ini digunakan AES-256, yang berarti enkripsi dilakukan menggunakan kunci sepanjang 256-bit (32 *byte*). Ukuran kunci yang lebih besar membuat algoritma ini semakin tahan terhadap *brute-force attack* (metode serangan untuk mencoba meretas sebuah sistem keamanan (seperti *password* atau kunci enkripsi) dengan cara mencoba semua kemungkinan kombinasi secara sistematis dan berulang).

Dalam implementasi ini digunakan mode CBC (*Cipher Block Chaining*). Mode ini cara kerjanya adalah dengan menghubungkan setiap blok *plaintext* dengan blok *ciphertext* sebelumnya sehingga meskipun ada dua blok *plaintext* yang isinya sama, hasil *ciphertext*-nya tetap berbeda. Hal ini membuat enkripsi menjadi lebih aman dan mencegah pola data terlihat secara langsung.

Algoritma kriptografi AES-256 (CBC Mode) memiliki beberapa tujuan, di antaranya yaitu:

- Kerahasiaan (*Confidentiality*)

Data yang terenkripsi tidak dapat dibaca oleh pihak lain kecuali mereka memiliki kunci yang benar atau sama.

- Integritas (*Integrity*)
  Walaupun AES-CBC tidak menyediakan integritas secara langsung, penggunaan *padding* dan *chaining* membuat *attacker* sulit memodifikasi data tanpa menghasilkan *ciphertext* yang rusak.
- Autentikasi (*Authentication*)
  Mode CBC sendiri tidak menyertakan autentikasi. Namun, dalam banyak sistem nyata AES-CBC biasanya dipasangkan dengan HMAC untuk memastikan data tidak dimodifikasi.

b. Alur Kerja (*Flowchart*)

- Komponen Utama
  Agar proses enkripsi AES dapat berjalan dengan baik, terdapat beberapa komponen penting, yaitu:
  (1) Kunci 256-bit
  Kunci ini digunakan pada setiap ronde enkripsi. Untuk AES-256, total ada 14 ronde pengolahan data di dalam algoritmanya. Pada setiap ronde, terjadi serangkaian operasi seperti substitusi *byte*, pergantian baris, transformasi kolom, dan penambahan kunci ronde (*round key*).
  (2) IV (*Initialization Vector*)
  IV berukuran 128-bit yang dihasilkan secara acak untuk setiap proses enkripsi. Fungsi IV adalah untuk memastikan bahwa meskipun *plaintext*-nya sama, *ciphertext* yang dihasilkan akan berbeda. Hal ini penting untuk mencegah pola data dapat terlihat.
  (3) *Padding* (PKCS#7)
  Karena AES memproses data per blok 16 *byte*, data yang panjangnya tidak pas dengan ukuran blok harus ditambahkan *padding*. PKCS#7 adalah metode *padding* yang paling umum, di mana *byte* tambahan diisi dengan angka sesuai jumlah *padding*.
  (4) Transformasi AES di tiap ronde
    - *SubBytes* → mengubah setiap *byte* menggunakan tabel substitusi
    - *ShiftRows* → menggeser baris pada blok untuk menciptakan difusi
    - *MixColumns* → mencampur *byte* dalam satu kolom (kecuali ronde terakhir)
    - *AddRoundKey* → XOR blok data dengan kunci ronde

  Transformasi-transformasi ini dirancang agar setiap perubahan kecil pada *plaintext* atau kunci menghasilkan perubahan besar pada *ciphertext*.

- Enkripsi (Mengubah menjadi Sandi)

Mode CBC bekerja dengan konsep *chaining*, yaitu *ciphertext* dari blok sebelumnya digunakan untuk mempengaruhi enkripsi blok berikutnya. Langkah-langkahnya adalah sebagai berikut.

(1) *Plaintext* di*padding* agar panjangnya pas dengan ukuran blok AES.
(2) *Plaintext* dipecah menjadi blok-blok berukuran 16 *byte*: P1, P2, P3, dan seterusnya.
(3) Blok pertama *plaintext* (P1) di XOR dengan IV:

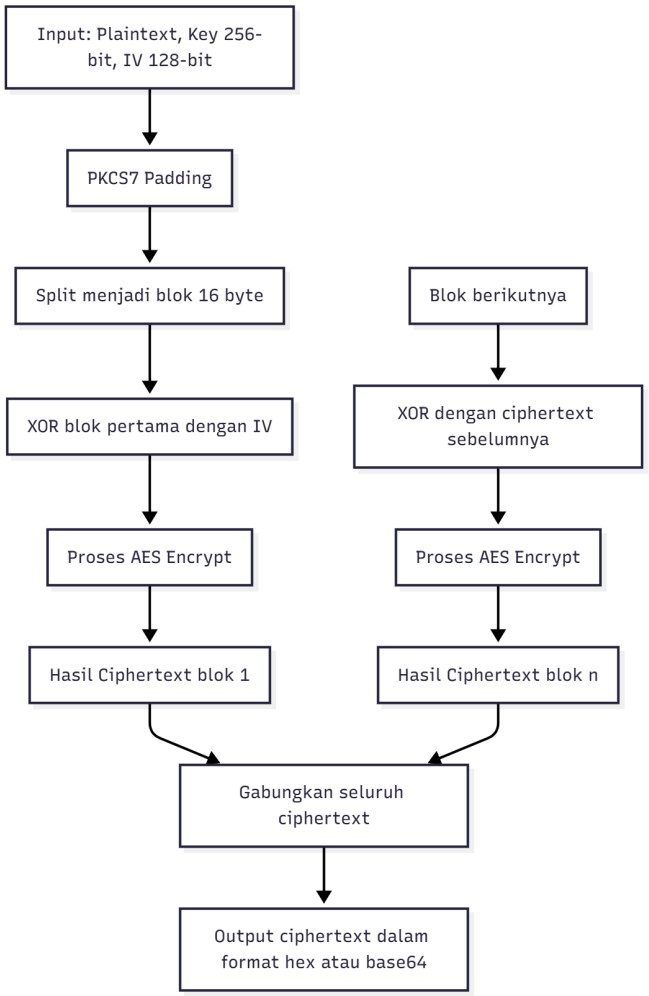
$$C1 = AES(P1 \oplus IV)$$

(4) Blok kedua dan seterusnya di-XOR dengan *ciphertext* sebelumnya:

$$C2 = AES(P2 \oplus C1)$$
$$C3 = AES(P3 \oplus C2)$$

(5) Semua blok *ciphertext* digabung menjadi *ciphertext* akhir.

Karena setiap blok *ciphertext* bergantung pada blok sebelumnya, ini akan membuat enkripsi menjadi lebih kuat dibandingkan dengan beberapa mode lainnya yang hanya mengenkripsi tiap blok secara terpisah.



- Dekripsi (Mengubah dari Sandi)

  Proses dekripsi pada dasarnya adalah membalik proses enkripsi. Langkah-langkahnya adalah sebagai berikut.

  (1) *Ciphertext* dipecah menjadi blok-blok (C1, C2, dan seterusnya).

  (2) Setiap blok didekripsi terlebih dahulu menggunakan AES:

  $$AES^{-1}(C1) \rightarrow \text{menghasilkan data intermediate}$$

  (3) Hasilnya di XOR dengan IV untuk mendapatkan *plaintext* blok pertama:

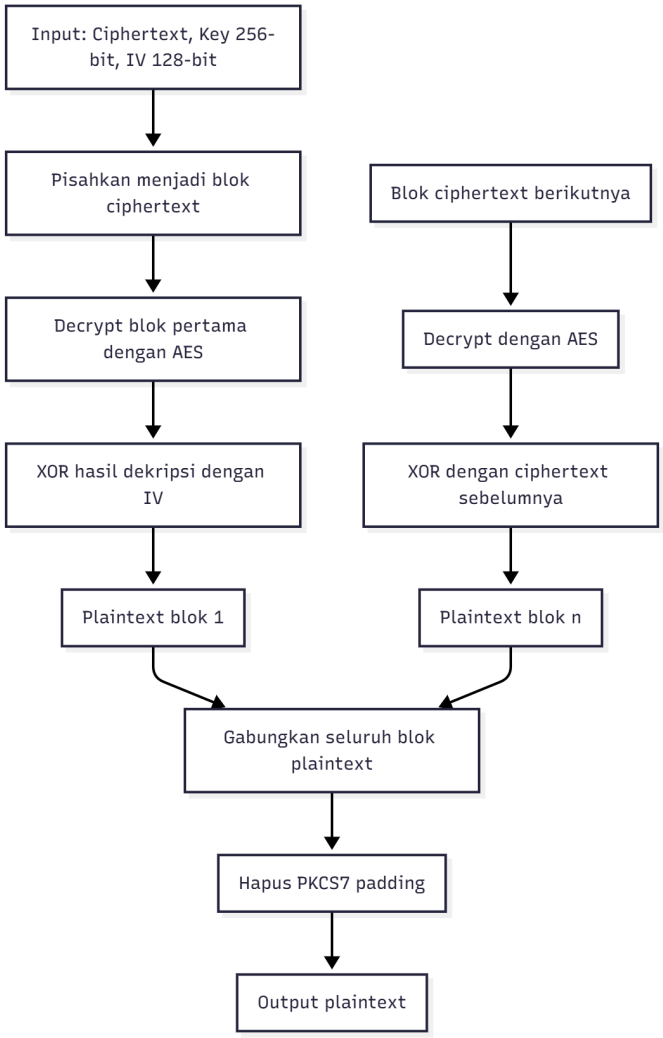
  $$P1 = AES^{-1}(C1) \oplus IV$$

  (4) Untuk blok selanjutnya:

  $$Pn = AES^{-1}(Cn) \oplus C(n-1)$$

  (5) Setelah semua blok *plaintext* didapatkan, *padding* PKCS#7 dihapus.

  Setelah *padding* dihapus, *plaintext* asli dapat diperoleh kembali secara utuh.



c. Contoh Manual Menghitung Enkripsi untuk Input Sederhana

- Misalnya *plaintext*:

  HELLO

- Konversi ASCII (hex), dimana setiap karakter diubah menjadi kode heksadesimal:

  H E L L O

  48 45 4C 4C 4F

  Karena panjangnya hanya 5 *byte* (kurang dari 16 *byte*), maka perlu ditambah *padding* (PKCS#7) 0B sebanyak 11 *byte*:

  48 45 4C 4C 4F 0B 0B 0B 0B 0B 0B 0B 0B 0B 0B 0B

  (0B berarti *padding* sebanyak 11 *byte*)

- *Key* (256-bit) contohnya:

  603deb1015ca71be2b73aef0857d7781

  1f352c073b6108d72d9810a30914dff4

- IV 128-bit contohnya:

  000102030405060708090A0B0C0D0E0F

  IV harus berbeda untuk setiap enkripsi agar *ciphertext* tidak mudah ditebak.

- XOR *Plaintext* Blok Pertama dengan IV

  $\text{Blok\_Plaintext\_1} \oplus \text{IV}$

  Misal hasil XOR ($\oplus$) nya:

  A1 B2 12 12 FF 44 11 23 99 10 44 89 5A 0A 23 F0

- Enkripsi menggunakan AES-256

  Hasil XOR di-enkripsi dengan AES-256 (14 ronde internal).

  Ciphertext blok pertama contohnya:

  76 49 AB AC 81 19 B2 46 CE E9 8E 9B 12 E9 19 7D

- Hasil *Ciphertext* (Hex → Base64)

  Ciphertext biasanya dikirim dalam bentuk Base64 agar mudah dibaca.

  Hex:

  76 49 AB AC 81 19 B2 46 CE E9 8E 9B 12 E9 19 7D

  Base64:

  dkimrIEZskbO6Y6bEukZfQ==

  Ini adalah hasil akhir enkripsi blok tersebut.

d. Contoh Manual Menghitung Dekripsi untuk Input Sederhana

- *Ciphertext* (hex):

  76 49 AB AC 81 19 B2 46 CE E9 8E 9B 12 E9 19 7D

- Ambil komponen yang diperlukan, yaitu *Ciphertext* (C1), Kunci AES-256, IV 128-bit. Proses dekripsi AES dilakukan kebalikan dari proses enkripsinya.

- Dekripsi blok *ciphertext* dengan AES-256

  AES dijalankan sebagai AES^{-1} (*inverse cipher*):

  $\text{Intermediate} = \text{AES}^{-1}(\text{C1})$

Misalnya hasil intermediate:

A1 B2 12 12 FF 44 11 23 99 10 44 89 5A 0A 23 F0

Ini adalah hasil yang sama seperti sebelum enkripsi (hasil XOR *plaintext* ⊕ IV).

- XOR Intermediate dengan IV

  Untuk mendapatkan *plaintext*:

  P1 = Intermediate ⊕ IV

  Jika di XOR kembali:

  A1 B2 12 12 FF 44 11 23 ...
  ⊕
  00 01 02 03 04 05 06 07 ...
  =
  48 45 4C 4C 4F 0B 0B 0B ...

  Hasilnya adalah blok *plaintext* yang sudah di-*padding*:

  48 45 4C 4C 4F 0B 0B 0B 0B 0B 0B 0B 0B 0B 0B 0B

- Hapus *padding* PKCS#7 sehingga tersisa data asli

  48 45 4C 4C 4F

- Konversi kembali ke ASCII

  HELLO

  plaintext berhasil dipulihkan

2.2 Desain Aplikasi Kriptografi

a. Struktur Program

Aplikasi yang dibuat menggunakan bahasa Python dan bertujuan untuk melakukan dua fungsi utama, yaitu enkripsi dan dekripsi terhadap teks maupun *file*. Untuk menjaga keamanan, aplikasi menggunakan *password* yang kemudian diubah menjadi kunci AES-256 melalui proses *key derivation* (PBKDF2). Komponen utama aplikasinya antara lain, yaitu:

- Input
  (1) Mode operasi: *encrypt* atau *decrypt*
  (2) Pilihan jenis input: teks atau *file*
  (3) *Password* (untuk menghasilkan *key*)
- Proses
  (1) Derivasi *key* menggunakan PBKDF2-HMAC-SHA256
  (2) Enkripsi AES-256 CBC (dengan *padding* + IV)
  (3) Dekripsi AES-256 CBC
  (4) Penanganan *file* jika input berupa CSV/JSON/dan lain-lain
- Output
  (1) Jika input teks: *ciphertext* Base64 atau *plaintext* hasil dekripsi
  (2) Jika input *file*: file terenkripsi .enc atau *file* hasil dekripsi

b. *Library* yang Digunakan

Beberapa *library* Python yang digunakan adalah sebagai berikut.

- cryptography → untuk implementasi AES-256 CBC, PBKDF2, *padding*
- os → untuk menghasilkan salt dan IV *random*
- base64 → untuk *encoding*/*decoding ciphertext*
- pandas → untuk membaca *file* CSV/JSON jika ingin integrasi *data science*
- getpass → untuk input *password* tanpa ditampilkan di layar

c. Deskripsi Alur Kerja Aplikasi

- *User* memilih mode
  Encrypt text atau *Decrypt text* atau *Encrypt file* atau *Decrypt file*
- *User* memasukkan data
  Jika teks → input string
  Jika *file* → input *path file*
- *User* memasukkan *password*
  Password secara internal akan diubah menjadi *key* 256-bit melalui PBKDF2
- Proses enkripsi
  (1) *Generate* salt dan IV
  (2) Padding plaintext
  (3) AES-CBC *Encrypt*
  (4) Gabungkan salt + IV + *ciphertext*
  (5) *Encode* ke Base64
  (6) Tampilkan hasil atau simpan ke *file* .enc
- Proses dekripsi
  (1) *Decode* Base64
  (2) Pisahkan salt + IV + *ciphertext*
  (3) *Derive key* dari *password* + salt
  (4) AES *Decrypt*
  (5) Hapus *padding*
  (6) Tampilkan *plaintext* atau simpan *file* hasil dekripsi

d. *Pseudocode* (Sketsa Fungsi Utama)

- Derivasi Kunci (PBKDF2)

```
FUNCTION derive_key(password, salt)
  // Membuat kunci AES dari password dan salt menggunakan PBKDF2
  key = PBKDF2(
    password = password,
    salt = salt,
    hash = SHA256,
    panjang_key = 32 bytes,
```

```
        iterasi = 100000
    )
    RETURN key
END FUNCTION
```

- Membaca File CSV/JSON/dan lainnya

```
FUNCTION load_file_for_encryption(file_path)
    ext = ambil ekstensi file dari file_path

    IF ext == ".csv" THEN
        baca CSV menggunakan pandas
        simpan sementara ke "temp_to_encrypt.csv"
        RETURN "temp_to_encrypt.csv"

    ELSE IF ext == ".json" THEN
        baca JSON menggunakan pandas
        simpan sementara ke "temp_to_encrypt.json"
        RETURN "temp_to_encrypt.json"

    ELSE
        RETURN file_path   // file lain digunakan apa adanya
END FUNCTION
```

- Enkripsi Bytes

```
FUNCTION encrypt_bytes(data, password)
    salt = buat 16 byte random
    key = derive_key(password, salt)
    iv = buat 16 byte random

    padded_data = tambahkan padding agar kelipatan 16 byte

    cipher = buat AES-CBC dengan key dan iv
    ciphertext = enkripsi padded_data dengan cipher

    RETURN gabungkan salt + iv + ciphertext
END FUNCTION
```

- Dekripsi Bytes

```
FUNCTION decrypt_bytes(blob, password)
    salt = ambil 16 byte pertama dari blob
```

```
    iv = ambil byte 17-32 dari blob
    ciphertext = sisa blob setelah byte 32

    key = derive_key(password, salt)
    cipher = buat AES-CBC dengan key dan iv
    padded = dekripsi ciphertext dengan cipher

    plain = hapus padding dari padded
    RETURN plain
  END FUNCTION
```

- Enkripsi Teks

```
  FUNCTION encrypt_text()
    plaintext = minta input teks dari user
    password = minta password dari user (tersembunyi)

    blob = encrypt_bytes(plaintext diubah ke bytes, password)
    encoded = ubah blob ke Base64

    tampilkan encoded ke user
  END FUNCTION
```

- Dekripsi Teks

```
  FUNCTION decrypt_text()
    ciphertext = minta input Base64 dari user
    password = minta password dari user (tersembunyi)

    TRY
      blob = decode Base64 ciphertext
      plain = decrypt_bytes(blob, password)
      tampilkan plain (hasil dekripsi) ke user
    CATCH error
      tampilkan "Gagal mendekripsi" + error
  END FUNCTION
```

- Enkripsi *File*

```
  FUNCTION encrypt_file()
    file_path = minta input path file dari user

    IF file tidak ada THEN
```

```
        tampilkan "File tidak ditemukan"
        RETURN

    IF ukuran file > 1 MB THEN
        tampilkan "Ukuran file melebihi 1 MB"
        RETURN

    actual_path = load_file_for_encryption(file_path)
    data = baca file actual_path sebagai bytes
    password = minta password dari user (tersembunyi)

    blob = encrypt_bytes(data, password)
    encoded = ubah blob ke Base64

    output_path = file_path + ".enc"
    simpan encoded ke output_path

    tampilkan "File terenkripsi disimpan sebagai" + output_path
END FUNCTION
```

- Dekripsi *File*

```
FUNCTION decrypt_file()
    file_path = minta input path file .enc dari user

    IF file tidak ada THEN
        tampilkan "File tidak ditemukan"
        RETURN

    password = minta password dari user (tersembunyi)

    TRY
        encoded = baca file_path
        blob = decode Base64 encoded
        plain = decrypt_bytes(blob, password)

        original_path = hapus ".enc" dari file_path
        simpan plain ke original_path

        tampilkan "File hasil dekripsi disimpan sebagai" + original_path
    CATCH error
```

```
            tampilkan "Gagal mendekripsi" + error
        END FUNCTION
```

- Menu Utama

```
        FUNCTION main()
          LOOP selamanya
            tampilkan menu:
              1. Encrypt Text
              2. Decrypt Text
              3. Encrypt File
              4. Decrypt File
              5. Exit

            choice = input dari user

            IF choice == "1" THEN
              encrypt_text()
            ELSE IF choice == "2" THEN
              decrypt_text()
            ELSE IF choice == "3" THEN
              encrypt_file()
            ELSE IF choice == "4" THEN
              decrypt_file()
            ELSE IF choice == "5" THEN
              tampilkan "Keluar..."
              BREAK
            ELSE
              tampilkan "Pilihan tidak valid. Coba lagi."
        END FUNCTION
```

2.3 Implementasi dan Pengujian

a. Laporan Hasil Pengujian
   - Enkripsi Teks

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 1
Masukkan plaintext: HELLO
Password:

=== Ciphertext (Base64) ===
6nr7jIygIZ4m8SVC2mZL9sNnadO+ZVMKjsGbk+Z3zNhaRXmunHdV3DuY3c8UmxUw
============================
```

- Dekripsi Teks
  Jika *password* benar (sama seperti yang dipakai ketika enkripsi)

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 2
Masukkan ciphertext Base64: 6nr7jIygIZ4m8SVC2mZL9sNnadO+ZVMKjsGbk+Z3zNhaRXmunHdV3DuY3c8UmxUw
Password:

=== Hasil Dekripsi ===
HELLO
======================
```

  Jika *password* salah (beda dengan yang dipakai ketika enkripsi)

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 2
Masukkan ciphertext Base64: 6nr7jIygIZ4m8SVC2mZL9sNnadO+ZVMKjsGbk+Z3zNhaRXmunHdV3DuY3c8UmxUw
Password:
Gagal mendekripsi! Error: Invalid padding bytes.
```

- Enkripsi File
  Jika *file* yang ingin dienkripsi ada (ditemukan)

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 3
Masukkan path file: D:\Amalia Ananda Putri\College\SEM 5\KEMDAT\DatasetLatihan.csv
Password:

File terenkripsi disimpan sebagai: D:\Amalia Ananda Putri\College\SEM 5\KEMDAT\DatasetLatihan.csv.enc
```

Jika *file* yang ingin dienkripsi tidak ada (tidak ditemukan)

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 3
Masukkan path file: D:\Amalia Ananda Putri\College\SEM 5\KEMDAT\DatasetLatihan1.csv
File tidak ditemukan!
```

Jika ukuran *file* melebihi batas (> 1 MB)

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 3
Masukkan path file: C:\Users\Amalia Ananda Putri\nad_here_poi_education_2023.csv
Ukuran file melebihi 1 MB.
```

- Dekripsi File

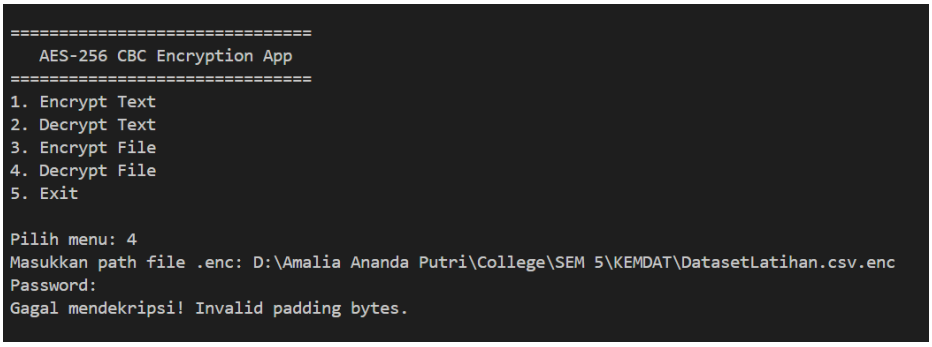
  Jika *password* benar (sama dengan yang dipakai ketika enkripsi)

```
==============================
   AES-256 CBC Encryption App
==============================
1. Encrypt Text
2. Decrypt Text
3. Encrypt File
4. Decrypt File
5. Exit

Pilih menu: 4
Masukkan path file .enc: D:\Amalia Ananda Putri\College\SEM 5\KEMDAT\DatasetLatihan.csv.enc
Password:

File hasil dekripsi disimpan sebagai: D:\Amalia Ananda Putri\College\SEM 5\KEMDAT\DatasetLatihan.csv
```

  Jika *password* salah (beda dengan yang dipakai ketika enkripsi)



2.4 Refleksi

Selama pengerjaan tugas ini, saya banyak belajar tentang konsep enkripsi AES-256 CBC, manipulasi file CSV/JSON dengan Python, serta implementasi antarmuka menu interaktif. Pengerjaan tugas ini tidak hanya menguji pemahaman konsep, tetapi juga kemampuan teknis dalam menangani error, debugging, dan penggunaan pustaka eksternal seperti cryptography.

Melalui tugas ini saya belajar cara membuat kunci dari *password* menggunakan PBKDF2 agar lebih aman terhadap *brute-force attack*, memahami *padding* PKCS7 untuk memastikan panjang data sesuai dengan blok AES, dan mengimplementasikan CBC mode (termasuk penggunaan IV acak untuk setiap enkripsi).

Saya juga belajar bagaimana cara menangani *file* yang berbeda format dengan pandas, memastikan *file* tersimpan ulang dengan format rapi sebelum enkripsi, dan membatasi ukuran *file* untuk mencegah error dan menjaga performa program.

Selama pengerjaan, saya menggunakan AI (ChatGPT) sebagai asisten pembelajaran dan *coding*. Beberapa peran AI antara lain, yaitu:

a. Membantu membuat dan menjelaskan kode
   - Membantu membuat *pseudocode* dari kode Python agar lebih mudah dipahami.
   - Menjelaskan fungsi-fungsi enkripsi/dekripsi dengan bahasa sederhana agar lebih mudah saya pahami.

b. *Debugging* dan *troubleshooting*
   - Membantu menemukan penyebab *error* seperti *file* tidak ditemukan atau *module cryptography* tidak terinstal.
   - Memberikan solusi aman untuk menjalankan program, termasuk cara memasukkan *path file* dan menggunakan terminal Anaconda.

c. Memberikan penjelasan ringkas dan mudah dipahami tentang konsep enkripsi-dekripsi berdasarkan referensi yang saya minta untuk dimasukkan ke laporan (termasuk membantu menyusun kata-kata laporan dan membuat *flowchart*).

PENUTUP

3.1 Kesimpulan

Berdasarkan pembahasan dan pengujian yang telah dilakukan, dapat disimpulkan bahwa algoritma AES-256 dengan mode CBC efektif untuk menjaga keamanan data baik itu data dalam bentuk teks maupun *file*. AES-256 menggunakan kunci panjang 256-bit yang membuat enkripsi sulit ditembus, sedangkan mode CBC memastikan setiap blok *plaintext* terenkripsi berbeda meskipun isinya sama.

Implementasi algoritma melalui Python seperti yang telah dilakukan memperlihatkan pentingnya *key derivation* (PBKDF2), *padding* (PKCS#7), dan penggunaan IV acak untuk keamanan enkripsi. Selain itu, penanganan berbagai jenis *file* (CSV, JSON, atau lainnya) dengan menggunakan *pandas* dan pembatasan ukuran *file* membantu menjaga kelancaran program.

Tugas ini membantu menambah pemahaman tentang kriptografi simetris, pemrograman Python untuk keamanan data, serta pentingnya pengecekan dan *debugging* agar aplikasi berjalan sesuai dengan harapan.

DAFTAR PUSTAKA

National Institute of Standards and Technology (2001) Advanced Encryption Standard (AES). (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 197-upd1, updated May 9, 2023. https://doi.org/10.6028/NIST.FIPS.197-upd1

https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/

Link video demo: https://youtu.be/XBErWSQkvqY