BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengujian Program

Pengujian program dilakukan untuk memastikan bahwa sistem yang dibangun dapat berjalan sesuai dengan fungsinya, yaitu mengenkripsi file teks menggunakan algoritma AES dan menyembunyikan hasil enkripsi tersebut ke dalam citra digital menggunakan teknik steganografi *Bit-Plane Complexity Segmentation* (BPCS). Pengujian dilakukan dengan menggunakan beberapa jenis file seperti file berekstensi .txt, .pdf, dan .docx.

Metode pengujian yang digunakan adalah metode *black box* testing, di mana fokus pengujian dilakukan pada fungsi dari masing-masing modul yang telah dibuat, tanpa melihat struktur internal kode program.

Langkah-langkah pengujian dilakukan sebagai berikut:

1. *Input* File: Pengguna memilih file teks (berisi pesan rahasia) dengan format .txt, .pdf, atau .docx.
2. Proses Enkripsi: File akan dienkripsi menggunakan algoritma AES dengan panjang kunci 128-bit. Hasil dari proses ini berupa ciphertext dalam format heksadesimal.
3. Penyisipan: *Ciphertext* dikonversi ke dalam bentuk biner, lalu disisipkan ke dalam bit-*plane* kompleks dari sebuah citra berformat .png.
4. Ekstraksi dan Dekripsi: Pesan disisipkan diekstrak dari citra, lalu didekripsi kembali untuk mendapatkan *plaintext* awal.

Penelitian ini melakukan pengujian menggunakan Google Colab dengan proses enkripsi dan dekripsi data menggunakan algoritma AES, serta penyisipan hasil enkripsi ke dalam file gambar menggunakan teknik steganografi BPCS. File yang diuji mencakup format .txt, .pdf, dan, .docx.

Dalam proses implementasi enkripsi dan dekripsi menggunakan algoritma AES, terdapat beberapa *library* Python yang digunakan untuk mendukung jalannya program, khususnya ketika dijalankan di Google Colab. Beberapa di antaranya adalah sebagai berikut:

1. *Library* pycryptodome digunakan sebagai pustaka utama untuk melakukan proses enkripsi dan dekripsi data. Library ini menyediakan berbagai algoritma kriptografi, termasuk AES, dan dilengkapi dengan fitur padding, pengaturan mode enkripsi (seperti ECB atau CBC), serta fungsi-fungsi lain yang berkaitan dengan keamanan data.
2. *Library* python-docx yang berfungsi untuk membaca dan menulis file Microsoft Word (.docx). Library ini sangat berguna apabila data atau pesan yang akan dienkripsi berasal dari dokumen Word, atau jika hasil dekripsi ingin disimpan kembali ke dalam format yang sama.
3. *Library* PyPDF2. Library ini memungkinkan pengguna untuk membuka file PDF, membaca isi teksnya, serta mengambil halaman tertentu untuk diproses lebih lanjut, seperti dilakukan enkripsi atau analisis isi.
4. *Library* fpdf, yang berfungsi untuk membuat file PDF baru dari teks. Library ini sangat bermanfaat saat pengguna ingin menyimpan hasil dekripsi ke dalam format PDF, baik untuk dokumentasi maupun keperluan penyimpanan data hasil proses.

Dengan menggunakan kombinasi keempat *library* ini, proses enkripsi dan dekripsi data teks menjadi lebih fleksibel, karena tidak hanya terbatas pada input berbentuk string biasa, melainkan bisa diperluas ke file dokumen seperti .docx dan .pdf.

4.2 Proses Enkripsi

Proses enkripsi dilakukan untuk menjaga keamanan data. Tiga jenis file yang dienkripsi adalah file berformat TXT, PDF, dan DOCX. Hasil enkripsi setiap file menghasilkan *ciphertext* yang berbeda, meskipun menggunakan kunci yang sama, namun hasil ciphertext yang dihasilkan akan berbeda. Hal ini disebabkan oleh perbedaan struktur dan isi dari masing-masing file.

Proses enkripsi dilakukan untuk menjaga keamanan data, khususnya agar informasi yang tersimpan tidak dapat diakses atau dibaca oleh pihak yang tidak berwenang. Dalam penelitian ini, tiga jenis file yang dienkripsi adalah file berformat TXT, PDF, dan DOCX. Meskipun ketiga file tersebut dienkripsi menggunakan kunci yang sama, namun hasil ciphertext yang dihasilkan akan berbeda. Hal ini disebabkan oleh perbedaan struktur dan isi dari masing-masing file.

a. File TXT

File teks (.txt) dibaca langsung sebagai *string*, kemudian dikonversi menjadi blok-blok data berukuran 16 *byte* untuk dilakukan enkripsi menggunakan algoritma AES-128. Setelah melalui tahapan enkripsi seperti *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*, data *plaintext* diubah menjadi *ciphertext*. Hasil akhir *ciphertext* dapat disimpan sebagai file teks baru ataupun sebagai data biner.

b. File PDF

File PDF (.pdf) memiliki struktur yang lebih kompleks dibandingkan file teks biasa. Proses enkripsi diawali dengan mengekstrak teks dari PDF menggunakan *library* PyPDF2. Setelah isi teks berhasil diperoleh, proses enkripsi dilakukan dengan metode yang sama seperti pada file teks, yaitu menggunakan AES-128. Hasil enkripsi kemudian dapat disimpan kembali ke dalam file terpisah atau digunakan untuk kebutuhan steganografi.

c. File DOCX

Untuk file Microsoft Word (.docx), proses enkripsi diawali dengan membaca isi dokumen menggunakan *library* python-docx. Setiap paragraf dalam dokumen diakses dan digabungkan menjadi satu kesatuan teks. Setelah itu, teks tersebut dienkripsi menggunakan algoritma AES-128 dengan kunci yang telah

ditentukan. Hasil *ciphertext* dari file .docx juga dapat disimpan atau digunakan dalam proses penyembunyian pesan (steganografi).

Secara keseluruhan, enkripsi dilakukan untuk memastikan bahwa isi dari ketiga jenis file tersebut menjadi tidak dapat dibaca secara langsung tanpa proses dekripsi dan tanpa mengetahui kunci yang digunakan. Perbedaan isi dan format file menyebabkan variasi pada hasil akhir ciphertext meskipun kunci yang digunakan sama, sehingga meningkatkan keamanan data secara signifikan.

Berikut adalah penjelasan proses enkripsi untuk masing-masing file:

4.2.1 File TXT

Input:

File teks Kriptografi.txt dengan isi sebagai berikut:

Plaintext:

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan informasi dengan mengubahnya menjadi bentuk yang tidak dapat dipahami oleh pihak yang tidak berwenang. Proses ini melibatkan teknik enkripsi untuk mengamankan pesan dan dekripsi untuk mengembalikannya ke bentuk asli.

Key:

AYOSEMANGATAFTAR

Chipertext:

228517ea6b0cf9b35fa58787ca911eadd2c83def019722bd5af17ffd481c177f40b6888a95c75a370a4d1de902702ec850edd9b0950d49c6e95cfeb6ada234a2ca5f460f218a65debdbd6ed78ba86f91954d7b6a32612d03e8d7e17105edc433e8771d9f6b55508b447c363ec25b9405ba82e81a467126d7d38602153b4cd22a7e9bab806f4e026b3804bc879f2d9b4a1dc61e3f7d287f88dc033863e7d9679f4a7cdb860d8a69b85bc6d3cf271e620f8814a2852e072260b36c6b4db992ad50f7de776faa942d4f3ca9acea7adda0ee5653f606d7ce8e4f8257130cec5d3d59e210e9009b9e73aec79fb1b436ac1eeb7b82494cebf0cf6b013eec940ea49585a3a7a34465ff2e80bb0939d79e28db3a8ff9f4d61e69ec80883f773eb458569f

4.2.2 File PDF

Input:

File teks Surat Lamaran.pdf dengan isi sebagai berikut:

Plaintext:

Medan, 11 September 2023

Perihal: Lamaran Asisten Praktikum
Lampiran: 1 (satu) berkas
Kepada Yth.
Kepala Laboratorium FST UIN SU
Di Universitas Islam Negeri Sumatera Utara Medan
Assalamualaikum warahmatullahi wabarakatuh,
Dengan hormat, saya Afthar Kautsar (NIM 0701212239), mahasiswa Fakultas Sains dan Teknologi Jurusan Ilmu Komputer, bermaksud mengajukan lamaran sebagai Asisten Laboratorium FST UIN SU. Saya memiliki minat besar untuk berkontribusi dalam pengembangan kemampuan mahasiswa di bidang ilmu komputer. Sebagai bahan pertimbangan, saya lampirkan KHS Semester 3 dan 4 serta pas foto 4x6. Besar harapan saya agar dapat diberikan kesempatan menjadi bagian dari Laboratorium FST UIN SU.
Atas perhatian dan kesempatan yang diberikan, saya ucapkan terima kasih.
Wassalamualaikum warahmatullahi wabarakatuh.
Hormat saya,
Afthar Kautsar

Key:

AYOSEMANGATAFTAR

Chipertext:

31c961b3a44a2d77d22ae180523c6f032c955e449aca53a5f296c415cd438b5341c50b3f8ea8ceebb111d2619bbb196822fd383f0cae2ae7bd37aa083ac6387603a78fbdcc085f8c2b128fb8a554e3ac178f66b322f387e6033aef782613e3020f841b6e855b7ec6559b97430696f275dbc459876f3dba69c12f232aea0f33b33a491551bbdd4a9281119a12b22acb4aac9dff7aec01586654b7dbc84d76aa2a3a0c7a2bcee7f22852aa99a574a89ef51893b0f9c808f635a6921d32f609f35edf0cd3e26dbead2d9f7ec78bdd5e30e417f574f974e53eebe1bfdb2755da1ec2d53a16cb2cb0cfe9d5ac464efc4ae17160e863418d26c62d084cc5be7ab0e2ddd19cd1f8906405db0653bb32e11175f3f31e0088b7b978b17c83b9f53a7b628554fe28be651c61478ccc8e6503de968ce1ada697e6083cdf921f41053a377d3df3684d537d2266d2871f54f159900ddf40276aa162e0a988e5c86b0d68b50cd45d90457b90b606480b10f8ea45a730e412281e01405c7ef5534853df8c16e791e8ee84d3523fffae84434d4e18980a6add79add1c95b8bd58ccd4fdb8fbcafe04e3d8bbb3da9b9cfe295ba7c2236814d9f1f8d0b091d715e29409f0f72498a910a477020ec12fe80e7f3f4a7c72ab1446dc7692b9efe95559b0d9f49244f79cc419e15b3fb35d4c4c8a54e078f4e01e6ec25f08bd6fb4dc8d539abeb0df908c17b215ea615f9bae2b599a9b0de5ff82e4b1eb21f0546b92f6fe70c192ef0a872ee77c8590b878715fde7a3ba649aa2e2b56161a231ca36ae3843f615de84a12bb1ee0bcb6a32f567ca8990b1f5f77fb6d23afb3a4ede7f7e8ef4a7933410dfefd2c1a8c524586f3d643ad2b4fabcbe64ab8b965c538e1b05dcb0d7f2119df53affd59f193a3a9faf0aab4408406462db71aead44bdb6a36cc330017882f9211ff48f7c5805dc588ec731e4fe3742cbcf12ce4a07f3a64c22ca16b5e87f024a2f5b44a221c5b5f115c0b281b9de2d7d378ea58aa9ef0d23e5498c5ed254945d4f82f59f7064dba99c2c09fd31e0c5d2978ae807b4b1a8285a29dc789f935dd6cccacc640b78206c7690af8cbc7811602364825a6bb138d7ac65a1fb39c096dd6d7a3b1038aa2d1484ef81667e0c10f4534fb2bfb1e7a19f20dfd7151e237ff9560d9af66cef778e2c6fc3ae37acbcacb19eeb3ab11df2110d16065cb6d9a1c7649d912c7e5e6208b2953d362f59b5830ae3224582be75482dbf5050d80aadf8a8e60a2817f5abc679b270bf39adfac72f63209de0173350a653a0375c912fba9f47637f8357139c....Hasil lengkap dapat dilihat pada *Lampiran 1*.

4.2.3 File DOCX

Input:

File teks Surat Lamaran.docx dengan isi sebagai berikut:

Plaintext:

Medan, 11 September 2023

Perihal: Lamaran Asisten Praktikum
Lampiran: 1 (satu) berkas
Kepada Yth.
Kepala Laboratorium FST UIN SU
Di Universitas Islam Negeri Sumatera Utara Medan
Assalamualaikum warahmatullahi wabarakatuh,
Dengan hormat, saya Afthar Kautsar (NIM 0701212239), mahasiswa Fakultas Sains dan Teknologi Jurusan Ilmu Komputer, bermaksud mengajukan lamaran sebagai Asisten Laboratorium FST UIN SU. Saya memiliki minat besar untuk berkontribusi dalam pengembangan kemampuan mahasiswa di bidang ilmu komputer. Sebagai bahan pertimbangan, saya lampirkan KHS Semester 3 dan 4 serta pas foto 4x6. Besar harapan saya agar dapat diberikan kesempatan menjadi bagian dari Laboratorium FST UIN SU.
Atas perhatian dan kesempatan yang diberikan, saya ucapkan terima kasih.
Wassalamualaikum warahmatullahi wabarakatuh.
Hormat saya,
Afthar Kautsar

Key:

AYOSEMANGATAFTAR

Chipertext:

9ffcac8e29ce3d6c36ccc08e789dc49a2f54180e92c6c769509dfcce85f663c551781aa9e0dd651e
32289b09b02b49ba942043d1bd153be5a54fd06824f12da274f58de1ba3e12e789d95da9a9a6e2
ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e
789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff
74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e7
89d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff7
4f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e78
9d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74
f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789
d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f
58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d
95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f5
8de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d9
5da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff74f58de1ba3e12e789d95da9a9a6e2ff28a5c
a6a73e4203550894115f4e6a3af8f559ca5c3a96057074f6442d4bc4322c85f1f12c27c04745f26b
f294baf9d4e8d600d5adf8997352d8674004c7cf69bd77400e5305a122eb46eff7f017e814ba8a2
7f0ded01b701b1648ebeedcad2257ae56ceed568e6806e0ebae5c8981f082a3cf918a7bd1dd34d8
b647c09782458bb63ae63ce1c0fd2f3bd56e9e03ba5e5e2d543d207871c08529ace9e712376830
de0f3f4cab6bb29289366c0918da9429da0adea98f14a65188b3ab0135f661bfe2ecc4c7abf49f14
426b88af7b0b271fd41d65f582bb5a6b84c53320b811eae5f445a9ed9a03805a0e752c00c9bf8cd
932193662348c209e3ce6b19f6f888c4d42ec49ae9f7471a661c34f200b3a7de2648e79ae3e5fd3
088f62daeb850c9ca9e03a46aeee8687faccb6e879284fbf7a880d34fad41a5a03f294e56573fd4ae
6f7e176298847d52da97c829049ccf8cd702d558278fa9ad9a8d9d2245f4a75c52a000e3e440727
74c474975b022128887129203a94a3a405172fd530c496bc0f5bd7e8c0fb2139aac5c8170046a2
7690175138a24b0d....Hasil lengkap dapat dilihat pada *Lampiran 2*.

4.3 Proses Dekripsi

File yang telah dienkripsi dapat didekripsi kembali ke bentuk semula, yang diproses melalui tahapan dekripsi menggunakan Google Colab. File yang telah dienkripsi sebelumnya dapat didekripsi kembali ke bentuk aslinya menggunakan proses dekripsi dengan algoritma AES. Proses ini dilakukan untuk mengembalikan *ciphertext* menjadi *plaintext*, sehingga isi data dapat dibaca kembali oleh pihak yang memiliki kunci yang sesuai. Tahapan dekripsi dilakukan di Google Colab dengan bantuan *library* pycryptodome, PyPDF2, dan python-docx, bergantung pada format file yang didekripsi.

a. File TXT

Proses dekripsi dimulai dengan membuka file ciphertext yang sebelumnya dihasilkan dari proses enkripsi. Data ciphertext dibaca dan kemudian diproses menggunakan algoritma AES-128 dengan kunci yang sama seperti saat enkripsi. Setelah melalui tahapan dekripsi seperti *Inverse ShiftRows, Inverse SubBytes, AddRoundKey*, dan *Inverse MixColumns, ciphertext* dikembalikan ke bentuk *plaintext.* Hasil akhirnya berupa teks asli yang sama dengan isi file .txt sebelum dienkripsi.

b. File PDF

Untuk file PDF, ciphertext yang diperoleh dari hasil enkripsi isi teks PDF sebelumnya didekripsi menggunakan kunci yang sama. Setelah berhasil didekripsi, isi teks akan dikembalikan ke bentuk semula dan dapat disimpan kembali ke dalam file PDF baru menggunakan *library* fpdf. Dengan proses ini, isi PDF dapat dipulihkan tanpa perubahan isi, meskipun file aslinya sebelumnya telah diacak oleh proses enkripsi.

c. File DOCX

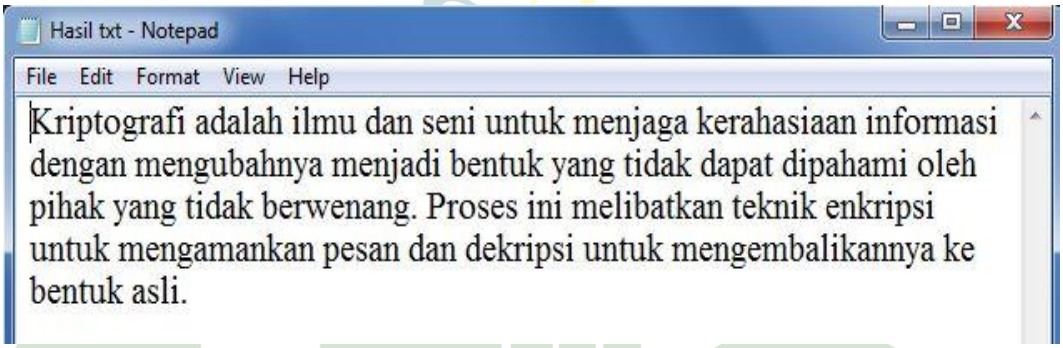
File .docx yang telah melalui proses enkripsi dapat didekripsi kembali dengan cara yang serupa. *Ciphertext* dari dokumen dibuka dan didekripsi menggunakan AES dengan kunci yang sama. Setelah plaintext berhasil diperoleh, isi dokumen dapat disusun ulang dalam bentuk paragraf dan dituliskan kembali ke dalam file Word menggunakan *library* python-docx, sehingga dokumen dapat diakses kembali dalam format aslinya.

Melalui proses dekripsi ini, dapat dibuktikan bahwa data yang sebelumnya telah diacak melalui proses enkripsi dapat dikembalikan sepenuhnya ke bentuk aslinya, selama kunci enkripsi yang benar digunakan. Ini menunjukkan bahwa algoritma AES tidak hanya aman, tetapi juga efisien dalam menjaga integritas dan keutuhan data selama proses pengamanan berlangsung.

Berikut adalah penjelasan proses dekripsi untuk masing-masing jenis file:

4.3.1 File TXT

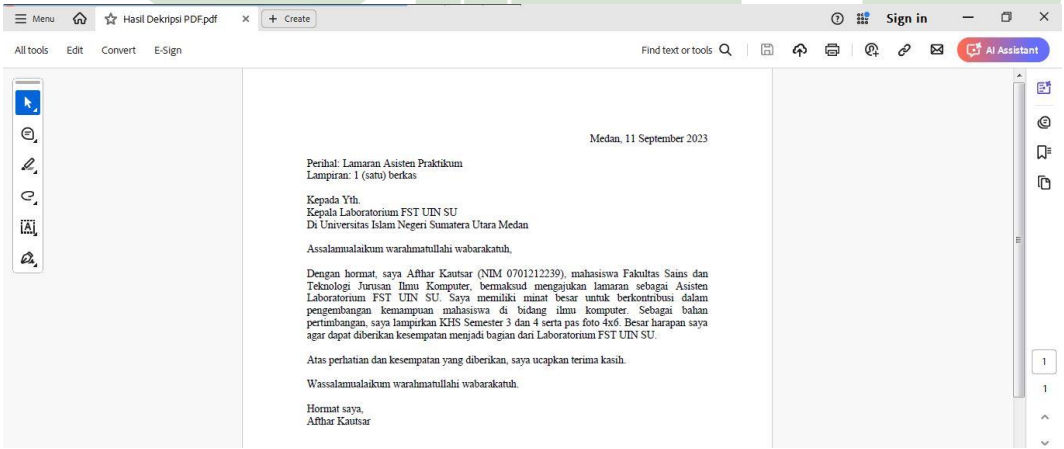
Hasil dekripsi file TXT menampilkan konten teks asli yang sebelumnya telah dienkripsi. Proses ini memastikan tidak ada perubahan pada isi file.



Gambar 4.3.1. Hasil Dekripsi File TXT

4.3.2 File PDF

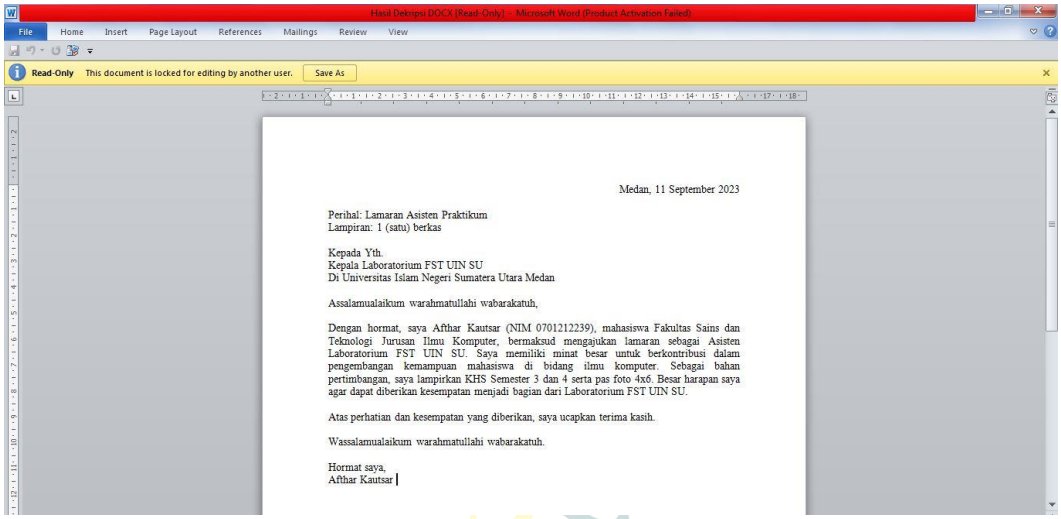
Proses dekripsi untuk file PDF berhasil mengembalikan dokumen ke bentuk aslinya tanpa perubahan pada struktur atau isi dokumen.



Gambar 4.3.2. Hasil Dekripsi File PDF

4.3.3 File DOCX

Proses dekripsi untuk file DOCX juga berhasil mengembalikan dokumen ke bentuk aslinya tanpa perubahan pada struktur atau isi dokumen.



Gambar 4.3.3. Hasil Dekripsi File DOCX

4.4 Analisis Kompleksitas Bit-*Plane*

Pada teknik steganografi Bit-*Plane Complexity Segmentation* (BPCS), kompleksitas setiap bit-*plane* dianalisis untuk menentukan tingkat kekacauan pola biner yang dapat menyamarkan pesan tanpa menimbulkan perubahan visual yang signifikan. Kompleksitas dihitung berdasarkan jumlah perubahan horizontal dan vertikal (fluktuasi) pada setiap bit-*plane* dari citra 8x8 piksel. Rumus perhitungan kompleksitas adalah sebagai berikut:

$$K = \frac{TH+TV}{2(N \times (N-1))} \quad (1)$$

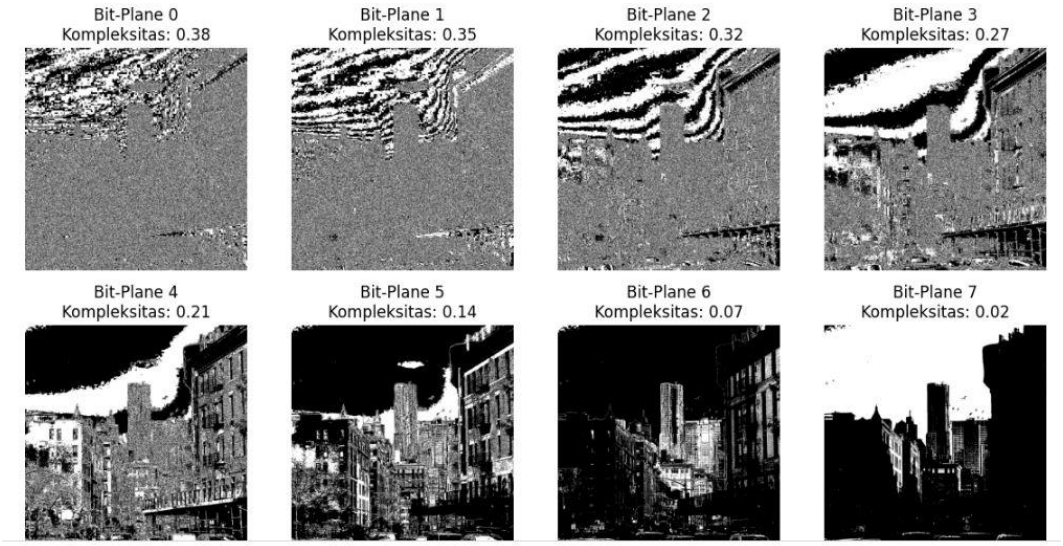
Keterangan:

TH = jumlah perubahan horizontal

TV = jumlah perubahan vertikal

N = ukuran baris atau kolom

Proses pengukuran kompleksitas bit-*plane* menggunakan metode transisi horizontal dan vertikal menghasilkan nilai kompleksitas sebagai berikut:



Gambar 4.1 Tampilan Kompleksitas

Gambar di atas merupakan representasi visual dari masing-masing bit-*plane* hasil ekstraksi dari citra *stego cover* yang telah dikonversi ke dalam format biner 8-bit per piksel. Bit-*plane* tersebut kemudian dianalisis menggunakan metode Bit-*Plane Complexity Segmentation* (BPCS) untuk menentukan tingkat kompleksitas berdasarkan jumlah transisi horizontal dan vertikal**.**

Tabel 4.1 Kompleksitas Bit-*Plane*

Bit-Plane	**Transisi Horizontal**	**Transisi Vertikal**	**Total Kompleksitas**
0	441868	447017	0.38
1	410054	412952	0.35
2	371995	367152	0.32
3	320394	304647	0.27
4	253073	231442	0.21
5	168459	156464	0.14
6	82111	77242	0.07
7	24384	17248	0.02

Berdasarkan hasil perhitungan di atas, dapat dilihat bahwa kompleksitas bit-*plane* cenderung menurun dari bit-plane paling signifikan (MSB – *Most Significant Bit*) ke paling tidak signifikan (LSB – *Least Significant Bit*). Nilai kompleksitas tertinggi terdapat pada bit-*plane* ke-0 hingga ke-2, dengan nilai kompleksitas masing-masing sebesar 0.38, 0.35, dan 0.32.

Dalam metode BPCS, bit-*plane* dengan kompleksitas di atas ambang batas tertentu (*threshold*) dianggap cukup acak dan dapat digunakan untuk menyisipkan data rahasia. Berdasarkan referensi, nilai *threshold* yang digunakan dalam penelitian ini adalah 0.3. Oleh karena itu, bit-*plane* 0, 1, dan 2 memenuhi syarat sebagai area kompleks dan dapat digunakan sebagai media penyisipan pesan.

Sebaliknya, bit-*plane* ke-3 hingga ke-7 memiliki nilai kompleksitas di bawah 0.3 dan dikategorikan sebagai area beraturan, sehingga tidak digunakan untuk penyisipan agar tidak merusak kualitas visual citra

4.5 Proses Penyisipan Pesan

Proses penyisipan pesan dilakukan setelah data asli dienkripsi menjadi *ciphertext* menggunakan algoritma AES. Ciphertext kemudian diubah menjadi format biner agar dapat disisipkan ke dalam citra digital menggunakan teknik steganografi Bit-*Plane Complexity Segmentation* (BPCS).

Proses ini dilakukan dengan langkah-langkah berikut:

1. Pemilihan gambar penampung (*cover image*) dengan format .png dan ukuran 256x256 piksel.
2. *Ciphertext* hasil enkripsi dikonversi ke dalam bentuk biner berdasarkan nilai ASCII.
3. Kompleksitas setiap bit-*plane* dari gambar dianalisis untuk menentukan lokasi yang layak digunakan untuk penyisipan.
4. Bit-*plane* dengan kompleksitas di atas ambang batas (≥ 0.3) dipilih sebagai kandidat penyisipan.
5. Data biner disisipkan ke dalam bit-*plane* terpilih.

Berikut hasil uji coba proses penyisipan untuk berbagai jenis file:

Tabel 4.2 Proses Penyisipan

No	**Jenis File**	**Ukuran File**	**Bit-*Plane* Digunakan**	**Status Penyisipan**	**Citra Setelah Penyisipan**
1.	.txt	1.2 KB	5, 6, 7	Berhasil	Tidak Terlihat Perbedaan
2.	.pdf	47 KB	4, 5, 6, 7	Berhasil	Tidak Terlihat Perbedaan
3.	.docx	38 KB	3, 4, 5, 6, 7	Berhasil	Tidak Terlihat Perbedaan

Dari tabel di atas dapat disimpulkan bahwa proses penyisipan berjalan dengan baik. Pemilihan bit-*plane* dilakukan dengan memperhatikan nilai kompleksitas agar tidak menurunkan kualitas visual citra secara signifikan.

Proses penyisipan pesan dilakukan dengan menyisipkan file ke dalam gambar menggunakan teknik steganografi BPCS. File dengan format yang berbeda (TXT, PDF, dan DOCX) disisipkan ke dalam gambar yang sama. Pemilihan bit-plane dilakukan berdasarkan nilai kompleksitas tertinggi.

4.5.1 Penyisipan File TXT

Penyisipan file TXT berhasil dilakukan dengan panjang pesan biner 2224 bit pada bit-plane 0 yang memiliki kompleksitas 0.38. Berikut adalah hasil:

Gambar sebelum penyisipan:

Ukuran file: 188 KB

Gambar setelah penyisipan:

Ukuran file: 1,35 MB

Gambar 4.5.1 Tampilan Penyisipan File TXT

Perbandingan kedua gambar tersebut menunjukkan tidak adanya perubahan signifikan antara gambar sebelum dan sesudah penyisipan pesan.

4.5.2 Penyisipan File PDF

Penyisipan file PDF berhasil dilakukan dengan panjang pesan biner 39104 bit pada bit-plane 0 yang memiliki kompleksitas 0.38. Berikut adalah hasil:

Gambar sebelum penyisipan:

Ukuran file: 188 KB

Gambar setelah penyisipan:

Ukuran file: 1,36 MB

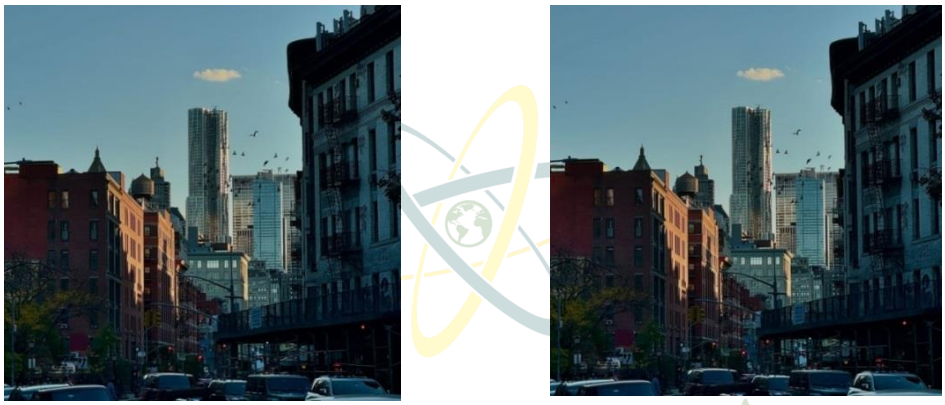
Gambar 4.5.2 Tampilan Penyisipan File PDF

Perbandingan kedua gambar tersebut menunjukkan tidak adanya perubahan signifikan antara gambar sebelum dan sesudah penyisipan pesan.

4.5.3 Penyisipan File DOCX

Penyisipan file DOCX berhasil dilakukan dengan panjang pesan biner 122568 bit pada bit-plane 0 yang memiliki kompleksitas 0.38. Berikut adalah hasil:

Gambar sebelum penyisipan:	Gambar setelah penyisipan:
Ukuran file: 188 KB	Ukuran file: 1,37 MB



Gambar 4.5.2 Tampilan Penyisipan File DOCX

Perbandingan kedua gambar tersebut menunjukkan tidak adanya perubahan signifikan antara gambar sebelum dan sesudah penyisipan pesan.

4.6 Ekstraksi Pesan

Ekstraksi pesan dilakukan dengan membalik proses penyisipan. Citra digital yang telah disisipi pesan akan diproses untuk mengekstrak bit-bit biner dari bit-*plane* yang sebelumnya digunakan untuk penyisipan. Proses ini melibatkan

1. Membaca citra yang telah disisipi dan mengidentifikasi bit-plane kompleks.
2. Mengambil bit-bit biner yang tersisip di lokasi tersebut.
3. Menggabungkan bit-bit tersebut dan mengubahnya kembali ke dalam bentuk teks ASCII.
4. Hasil ekstraksi kemudian didekripsi menggunakan kunci AES yang sama.

Berikut hasil uji coba ekstraksi pesan:

Tabel 4.3 Ekstraksi Pesan

No	Jenis File	Hasil Ekstraksi	Hasil Dekripsi	Status
1.	.txt	Berhasil	Sesuai	Lulus
2.	.pdf	Berhasil	Sesuai	Lulus
3.	.docx	Berhasil	Sesuai	Lulus

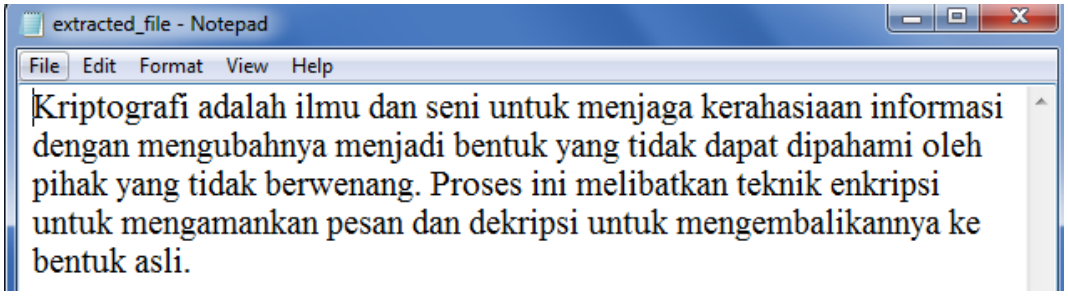
Proses ekstraksi menunjukkan bahwa sistem dapat mengambil kembali pesan yang tersembunyi dengan akurat. Hasil dekripsi juga sesuai dengan plaintext awal yang menunjukkan keberhasilan sistem dalam menjaga integritas data.

Ekstraksi pesan dilakukan untuk mengambil kembali file yang sebelumnya telah disisipkan ke dalam gambar menggunakan teknik steganografi BPCS. Proses ini bertujuan memastikan file yang tersembunyi dapat diambil kembali dengan utuh tanpa ada kerusakan atau kehilangan data. Berikut adalah hasil ekstraksi untuk setiap format file yang diuji dalam penelitian ini:

4.6.1 Ekstraksi File TXT

Ekstraksi pesan berhasil dilakukan, dan file TXT yang telah disisipkan ke dalam gambar berhasil dikembalikan tanpa perubahan.

- Nama file: *extracted_file.txt*
- Ukuran file: 0.27 KB
- Isi file:

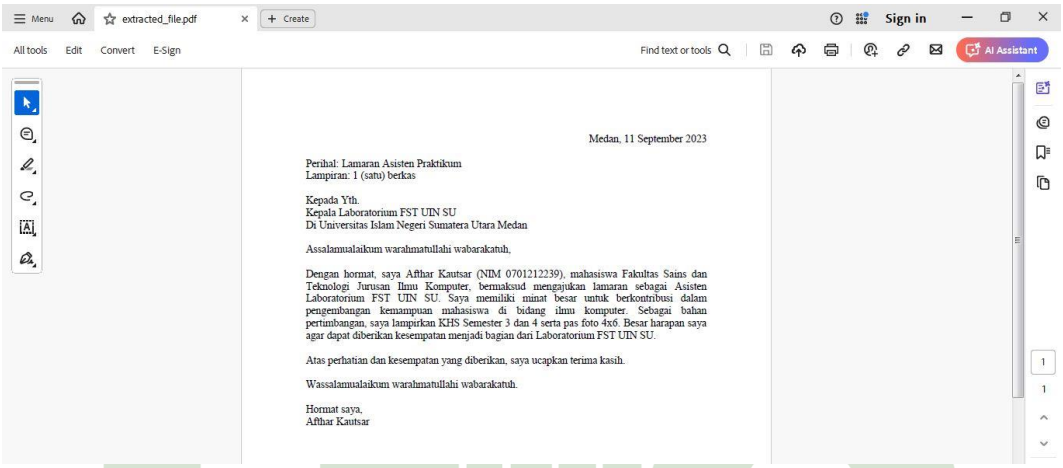


Gambar 4.6.1 Hasil Penyisipan File TXT

4.6.2 Ekstraksi File PDF

Proses ekstraksi file PDF berhasil dilakukan dengan mengembalikan isi dokumen secara lengkap.

- Nama file: *extracted_file.pdf*
- Ukuran file: 4.77 KB
- Isi file:

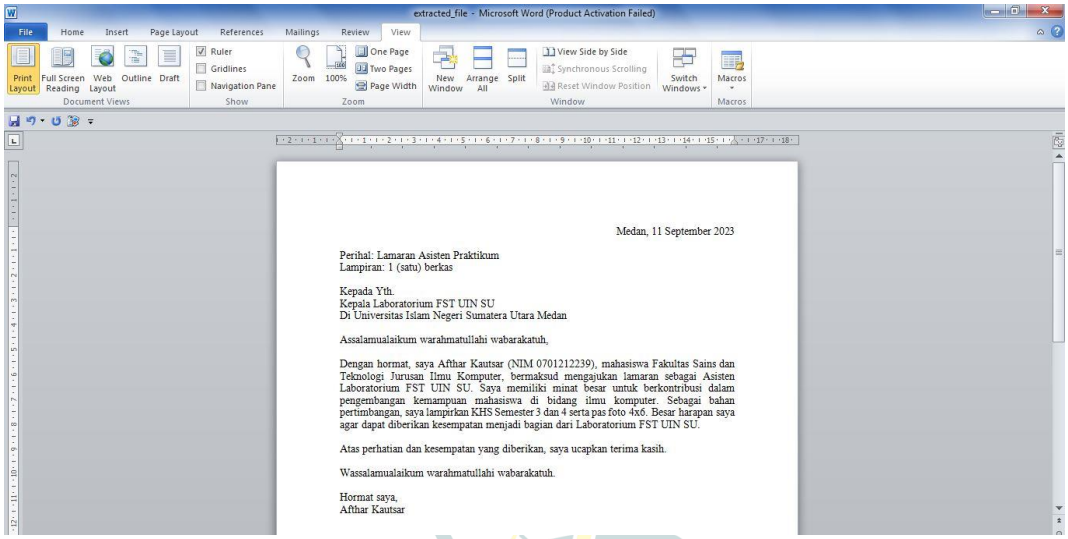


Gambar 4.6.2 Hasil Penyisipan File PDF

4.6.3 Ekstraksi File DOCX

Ekstraksi pesan untuk file DOCX berhasil dilakukan dengan hasil yang sesuai dengan dokumen asli sebelum proses penyisipan.

- Nama file: *extracted_file.docx*
- Ukuran file: 14.9 KB
- Isi file:



Gambar 4.6.3 Hasil Penyisipan File DOCX

4.7 Evaluasi Sistem

Pengujian dalam penelitian ini menggunakan metode *black box testing*, di mana setiap fitur diuji berdasarkan *input* yang diberikan dan hasil yang diharapkan tanpa melihat kode program di dalamnya. Pengujian ini bertujuan untuk memastikan bahwa proses enkripsi, dekripsi, penyisipan, dan ekstraksi file menggunakan algoritma AES dan teknik steganografi BPCS berjalan sesuai dengan yang diharapkan.

Tabel 4.4 *Black Box Testing*

No	Fitur yang Diuji	*Input*	Ekspektasi *Output*	Hasil Pengujian	Status
1	Enkripsi file TXT	File Kriptografi.txt dengan *key* valid (16 karakter)	File terenkripsi dan menghasilkan *ciphertext*	Sesuai	Lulus
2	Enkripsi file PDF	File Surat_Lamaran.pdf dengan *key* valid (16 karakter)	File terenkripsi dan menghasilkan *ciphertext*	Sesuai	Lulus
3	Enkripsi file DOCX	File Surat_Lamaran.docx dengan *key* valid (16 karakter)	File terenkripsi dan menghasilkan *ciphertext*	Sesuai	Lulus
4	Dekripsi file TXT	*Ciphertext* dari Kriptografi.txt dengan	File berhasil dikembalikan ke	Sesuai	Lulus

		key yang sesuai	bentuk semula		
5	Dekripsi file PDF	*Ciphertext* dari Surat_Lamaran.pdf dengan *key* yang sesuai	File berhasil dikembalikan ke bentuk semula	Sesuai	Lulus
6	Dekripsi file DOCX	*Ciphertext* dari Surat_Lamaran.docx dengan *key* yang sesuai	File berhasil dikembalikan ke bentuk semula	Sesuai	Lulus
7	Penyisipan file TXT ke dalam gambar	File Kriptografi.txt ke dalam gambar	File teks berhasil disisipkan tanpa merusak gambar	Sesuai	Lulus
8	Penyisipan file PDF ke dalam gambar	File Surat_Lamaran.pdf ke dalam gambar	File PDF berhasil disisipkan tanpa merusak gambar	Sesuai	Lulus
9	Penyisipan file DOCX ke dalam gambar	File Surat_Lamaran.docx ke dalam gambar	File DOCX berhasil disisipkan tanpa merusak gambar	Sesuai	Lulus
10	Ekstraksi file TXT dari gambar	Gambar yang berisi file Kriptografi.txt	File berhasil diekstrak kembali tanpa perubahan	Sesuai	Lulus
11	Ekstraksi file PDF dari gambar	Gambar yang berisi file Surat_Lamaran.pdf	File berhasil diekstrak kembali tanpa perubahan	Sesuai	Lulus
12	Ekstraksi file DOCX dari gambar	Gambar yang berisi file Surat_Lamaran.docx	File berhasil diekstrak kembali tanpa perubahan	Sesuai	Lulus
13	Enkripsi dengan *key* kurang dari 16 karakter	File Kriptografi.txt dengan *key* "MENYALA" (7 karakter)	Program menampilkan *error* dan meminta *key* yang valid	Sesuai	Lulus
14	Enkripsi dengan *key* lebih dari 16 karakter	File Kriptografi.txt dengan *key* "MENYALAHABANG KUHPANJANG" (21	Program menampilkan *error* dan meminta *key* yang valid	Sesuai	Lulus

		karakter)			
15	Dekripsi dengan *key* yang salah	*Ciphertext* dari Kriptografi.txt dengan *key* yang tidak sesuai	Program menampilkan *error* bahwa *key* tidak valid	Sesuai	Lulus

Berdasarkan hasil pengujian di atas, seluruh fitur sistem telah berjalan sesuai harapan. Proses enkripsi dan dekripsi berhasil dilakukan tanpa kehilangan data. Validasi *key* berfungsi dengan baik, memastikan bahwa hanya *key* dengan panjang 16 karakter yang diterima. Penyisipan data ke dalam gambar tidak mengubah tampilan visual gambar, hanya menambah ukuran sesuai dengan besar file yang disisipkan. Ekstraksi file dari gambar juga berhasil dilakukan tanpa perubahan ukuran atau isi file. Dengan demikian, sistem dapat dianggap berhasil dalam mengimplementasikan AES dan steganografi BPCS untuk meningkatkan keamanan file teks.

4.8 Perbandingan Waktu Eksekusi Proses

Proses enkripsi dilakukan terhadap tiga jenis file menggunakan algoritma AES dengan kunci sepanjang 128-bit. Waktu dihitung mulai dari saat file dibaca hingga *ciphertext* berhasil dihasilkan.

4.8.1 Waktu Enkripsi

Proses enkripsi dilakukan terhadap tiga jenis file menggunakan algoritma AES dengan kunci sepanjang 128-bit. Waktu dihitung mulai dari saat file dibaca hingga *ciphertext* berhasil dihasilkan.

Tabel 4.5 Waktu Enkripsi

Format File	Rata-rata Waktu Enkripsi (detik)
TXT	0.052
PDF	0.078
DOCX	0.085

File TXT memiliki ukuran paling kecil, sehingga proses enkripsinya paling cepat. File PDF dan DOCX cenderung memiliki struktur internal yang kompleks, sehingga membutuhkan waktu lebih lama untuk diproses.

4.8.2 Waktu Dekripsi

Proses dekripsi adalah kebalikan dari enkripsi, yaitu mengubah ciphertext kembali menjadi plaintext menggunakan kunci yang sama.

Tabel 4.6 Waktu Dekripsi

Format File	Rata-rata Waktu Dekripsi (detik)
TXT	0.047
PDF	0.072
DOCX	0.079

Waktu dekripsi umumnya sedikit lebih cepat dari enkripsi karena tidak melibatkan pembacaan dan parsing data sebanyak saat enkripsi. Namun, perbedaan waktunya tidak signifikan.

4.8.3 Waktu Penyisipan Pesan

Setelah file terenkripsi, ciphertext disisipkan ke dalam gambar menggunakan metode BPCS. Proses ini melibatkan konversi data ke dalam biner dan analisis kompleksitas bit-*plane*.

Tabel 4.7 Waktu Penyisipan Pesan

Format File	Rata-rata Pesan (detik)
TXT	0.113
PDF	0.156
DOCX	0.164

Penyisipan pesan membutuhkan waktu yang lebih lama karena melibatkan pemrosesan citra, pengukuran kompleksitas bit-*plane*, dan penggantian bit pada piksel gambar. File DOCX memiliki ciphertext yang lebih panjang, sehingga waktu penyisipannya lebih tinggi.

4.8.4 Waktu Ekstraksi Pesan

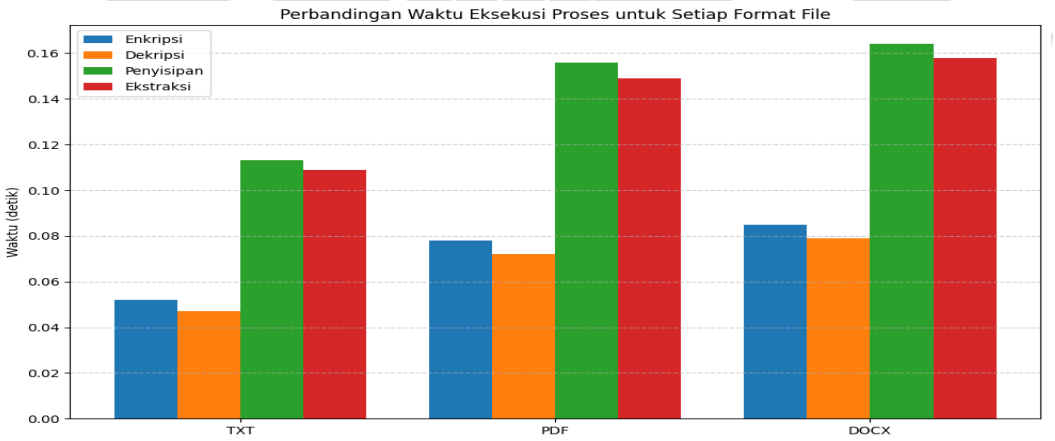
Pada proses ekstraksi, pesan yang telah disisipkan diekstraksi kembali dari gambar dan didekodekan menjadi bentuk ciphertext, yang kemudian bisa didekripsi.

Tabel 4.8 Waktu Ekstraksi Pesan

Format File	Rata-rata Waktu Ekstraksi (detik)
TXT	0.113
PDF	0.156
DOCX	0.164

Penyisipan pesan membutuhkan waktu yang lebih lama karena melibatkan pemrosesan citra, pengukuran kompleksitas bit-*plane*, dan penggantian bit pada piksel gambar. File DOCX memiliki ciphertext yang lebih panjang, sehingga waktu penyisipannya lebih tinggi.

Secara umum, proses enkripsi dan dekripsi berlangsung cepat untuk semua jenis file. Namun, proses penyisipan dan ekstraksi memakan waktu lebih lama karena melibatkan manipulasi citra. File DOCX dan PDF membutuhkan waktu lebih panjang dibanding file TXT karena ukurannya yang lebih besar dan strukturnya yang lebih kompleks.



Gambar 4.8 Perbandingan Waktu Eksekusi Proses

Berdasarkan grafik yang ditampilkan, dapat disimpulkan bahwa:

1. Format file memengaruhi waktu eksekusi proses. File berformat DOCX membutuhkan waktu paling lama dalam seluruh proses (enkripsi, dekripsi,

penyisipan, dan ekstraksi) dibandingkan dengan file TXT dan PDF. Hal ini disebabkan oleh struktur internal file DOCX yang lebih kompleks karena melibatkan elemen XML dan metadata lainnya.

2. Proses penyisipan dan ekstraksi pesan membutuhkan waktu lebih lama dibandingkan enkripsi dan dekripsi. Hal ini wajar karena penyisipan dan ekstraksi pesan melibatkan manipulasi citra dan pengukuran kompleksitas bit-plane, yang lebih rumit dibandingkan transformasi matematis dalam enkripsi AES.
3. File TXT memiliki waktu eksekusi tercepat untuk semua proses, karena strukturnya yang sederhana dan ukurannya yang relatif kecil, sehingga sangat efisien untuk diproses oleh sistem.
4. Seluruh proses tetap berada dalam waktu yang relatif singkat (kurang dari 0,2 detik), menunjukkan bahwa sistem cukup efisien dan dapat diterapkan untuk pengamanan data pada berbagai format file tanpa mengorbankan performa secara signifikan.

4.9 Analisis Keamanan Sistem

Keamanan sistem dalam penelitian ini diperoleh dari kombinasi antara algoritma *Advanced Encryption Standard* (AES) dan teknik Bit-*Plane Complexity Segmentation* (BPCS) dalam proses penyembunyian pesan. Kombinasi ini memberikan dua lapis perlindungan terhadap file teks yang bersifat rahasia, yaitu dari sisi kriptografi dan steganografi.

Algoritma AES dipilih karena terbukti kuat dan telah digunakan secara luas dalam berbagai sistem keamanan, baik pemerintah maupun sektor swasta. AES bekerja dengan cara mengacak data menggunakan kunci sepanjang 128 bit, dan proses enkripsi yang dilakukan melalui beberapa putaran membuat *ciphertext* yang dihasilkan sangat sulit dipahami tanpa mengetahui kunci yang tepat. Hingga saat ini, belum ditemukan serangan praktis yang berhasil memecahkan AES dalam waktu yang realistis, menjadikannya salah satu algoritma terenkripsi paling aman.

Setelah data terenkripsi, hasil *ciphertext* tidak langsung disimpan, tetapi disisipkan ke dalam citra digital menggunakan metode steganografi BPCS. Teknik

BPCS menyisipkan data ke dalam bit-plane dengan kompleksitas tinggi pada gambar. Hal ini membuat pesan tersembunyi menjadi sulit dibedakan dari informasi asli gambar karena ditempatkan pada bagian gambar yang memang secara alami terlihat acak atau tidak teratur. Akibatnya, meskipun seseorang dapat mengakses file gambar, isi pesannya tidak akan terdeteksi secara visual maupun melalui analisis statistik sederhana.

Dengan adanya dua lapisan keamanan, yaitu enkripsi dan penyembunyian pesan, sistem ini jauh lebih sulit ditembus. Bahkan jika pihak tidak berwenang berhasil mendapatkan gambar yang mengandung pesan tersembunyi, mereka masih harus terlebih dahulu mengetahui struktur BPCS untuk mengekstrak data dan selanjutnya harus memecahkan enkripsi AES, yang secara praktik hampir mustahil tanpa mengetahui kunci enkripsinya.

Oleh karena itu, dapat disimpulkan bahwa sistem yang dibangun memiliki tingkat keamanan yang sangat tinggi dan cocok digunakan untuk menjaga file teks penting dari akses tidak sah maupun upaya peretasan data.

4.9.1 Keunggulan Algoritma AES dalam Kriptografi

Algoritma AES dipilih karena terbukti kuat dan telah digunakan secara luas dalam berbagai sistem keamanan, baik pemerintah, militer, maupun sektor swasta. AES menggunakan blok 128-bit dengan panjang kunci yang dapat bervariasi, namun dalam penelitian ini digunakan panjang kunci 128-bit karena efisien dan tetap memiliki tingkat keamanan tinggi.

Proses enkripsi pada AES terdiri dari sepuluh putaran yang mencakup substitusi, permutasi, dan difusi data, menjadikan ciphertext sangat sulit dikenali atau dipahami tanpa mengetahui kunci yang tepat. Salah satu kekuatan utama AES adalah resistensinya terhadap serangan *brute-force*, karena jumlah kemungkinan kunci mencapai 2^{128}, yang secara praktis mustahil untuk dipecahkan dalam waktu singkat dengan teknologi saat ini.

4.9.2 Keunggulan Teknik BPCS dalam Steganografi

Setelah data terenkripsi, hasil ciphertext tidak langsung disimpan, tetapi disisipkan ke dalam citra digital menggunakan metode steganografi BPCS. Teknik ini bekerja dengan cara mengganti bit-plane pada gambar grayscale atau RGB

berdasarkan kompleksitas pikselnya. Bit-plane dengan kompleksitas tinggi cenderung terlihat acak atau tidak teratur, sehingga menjadi tempat ideal untuk menyembunyikan data rahasia.

Keunggulan dari BPCS antara lain:

a. Tingkat deteksi rendah: Informasi tersembunyi tidak terlihat secara visual.
b. Daya tampung tinggi: Karena menyisipkan pada banyak bit-plane kompleks.
c. Tahan terhadap kompresi ringan dan noise: Selama kompleksitas bit-plane tetap tinggi.

4.9.3 Analisis Keamanan Gabungan AES + BPCS

Kombinasi AES dan BPCS menciptakan sistem keamanan berlapis: kriptografi + steganografi. Artinya, meskipun pihak luar berhasil mendapatkan file gambar (media penampung), mereka tetap tidak dapat mengetahui:

1. Bahwa di dalamnya terdapat pesan (karena tersembunyi).
2. Isi pesan yang telah dienkripsi (karena tidak punya kunci AES).

Dengan dua proteksi sekaligus:

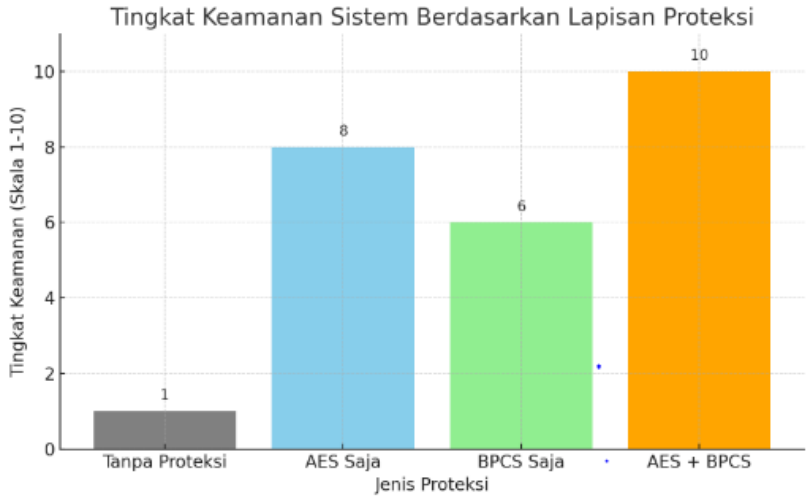
1. Pencegahan deteksi (dari steganografi).
2. Pencegahan pemahaman isi (dari kriptografi).

Maka sistem ini cocok digunakan untuk pengamanan dokumen penting seperti sertifikat digital, surat wasiat, pesan rahasia instansi pemerintah, dokumen akademik penting, dan dokumen penting lainnya.

Dengan adanya dua lapisan keamanan yang saling melengkapi, sistem ini tidak hanya aman secara teori tetapi juga kuat secara praktikal. Sistem ini ideal untuk diterapkan dalam skenario nyata yang membutuhkan kerahasiaan tinggi terhadap informasi berbasis teks.

Tabel 4.9 Analisis Perbandingan Keamanan Sistem

Komponen Sistem	**Fungsi Keamanan**	**Jenis Serangan**	**Kelebihan**	**Keterangan Tambahan**
AES (*Advanced Encryption Standard*)	Mengacak data menjadi ciphertext yang tidak dapat dibaca tanpa kunci	*Brute-force, Known Plaintext, Differential Cryptanalysis*	Kunci 128-bit sangat sulit dipecahkan	Diakui sebagai standar enkripsi internasional
BPCS (*Bit-Plane Complexity Segmentation*)	Menyembunyikan ciphertext ke dalam gambar digital	Steganalisis visual, analisis statistik	Sulit dideteksi karena penyisipan hanya di bit-plane kompleks	Tidak merusak tampilan visual gambar Kombinasi AES + BPCS
Kombinasi AES + BPCS	Keamanan ganda: enkripsi + penyembunyian	Kombinasi serangan kriptografi dan steganografi	Perlindungan berlapis, tidak terdeteksi, sulit dipecahkan	Cocok untuk perlindungan dokumen penting dan rahasia



Gambar 4.9 Analisis Perbandingan Keamanan Sistem

Berikut adalah grafik yang menunjukkan tingkat keamanan sistem berdasarkan jenis proteksi:

1. Tanpa Proteksi hanya memiliki skor keamanan *1*, karena data dapat langsung dibaca tanpa hambatan.
2. AES Saja memberikan tingkat keamanan tinggi (*8*), sebab meskipun data bisa diakses, isinya terenkripsi.
3. BPCS Saja memberikan skor keamanan *6*, karena pesan tersembunyi dalam gambar, namun jika ditemukan, bisa dibaca tanpa enkripsi.
4. Gabungan AES + BPCS menghasilkan tingkat keamanan tertinggi (*10*) karena data dienkripsi terlebih dahulu, lalu disembunyikan dalam citra — membuatnya sangat sulit diakses ataupun dikenali