Analisis Performa Algoritma AES-256 dan Blowfish dalam Enkripsi Pesan

Michael Jeremi Bungaran Simanjuntak - 18221136
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: michaeljeremi244@gmail.com , 18221136@std.stei.itb.ac.id

***Abstract*—Perkembangan pesat teknologi informasi meningkatkan kebutuhan perlindungan data, khususnya pada pengiriman pesan teks melalui internet. Penelitian ini membandingkan performa dua algoritma enkripsi simetrik, yaitu AES-256-GCM dan Blowfish, dalam proses enkripsi dan dekripsi pesan teks. Kedua algoritma diimplementasikan menggunakan Python pada Google Colab dengan library cryptography dan pycryptodome. Pengujian dilakukan pada variasi panjang teks 100, 500, 2.000, 10.000, dan 50.000 karakter dengan mengukur waktu enkripsi-dekripsi, throughput, Avalanche Effect, ukuran ciphertext, serta penggunaan memori. Hasil pengujian menunjukkan bahwa AES-256-GCM secara konsisten unggul dalam hal kecepatan dan throughput, terutama pada data berukuran besar. Sementara itu, kedua algoritma menunjukkan nilai Avalanche Effect yang baik dan setara di kisaran 49%–51%.**

Keywords—AES-256, Blowfish, Algoritma, Kriptografi

I. PENDAHULUAN

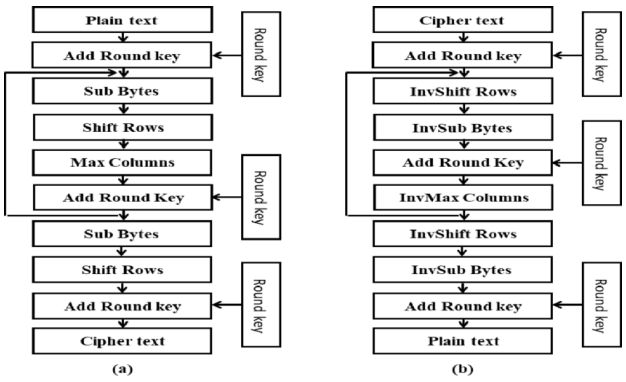
Perkembangan teknologi yang sangat pesat menyebabkan bertambahnya intensitas pertukaran informasi yang terjadi pada internet. Maraknya pertukaran informasi ini tentunya diikuti tingginya risiko keamanan, seperti potensi pencurian dan penyalahgunaan informasi yang dapat terjadi pada siapa saja, termasuk diri kita sendiri. Algoritma Enkripsi dapat menjadi salah satu solusi terbaik untuk mengamankan informasi kita saat kita menggunakan internet. Enkripsi secara umum dibagi menjadi dua, yaitu simetris dan asimetris. Enkripsi simetris memiliki kunci yang sama untuk enkripsi dan dekripsi, sementara enkripsi asimetris memiliki kunci yang berbeda untuk kedua prosesnya. Enkripsi pesan yang ukurannya relatif kecil pada umumnya menggunakan enkripsi simetris, karena membutuhkan *resources* yang lebih sedikit dari enkripsi asimetris.

Algoritma AES-256 / Rijndael merupakan salah satu algoritma enkripsi simetris yang telah dan masih sangat populer digunakan dalam berbagai sektor-sektor tertentu, seperti militer, keuangan, layanan *cloud*, *Virtual Private Network* (VPN), dan masih banyak sektor lainnya. Salah satu algoritma enkripsi simetris yang menarik untuk dibahas adalah Blowfish. Blowfish merupakan algoritma enkripsi simetrik yang dikembangkan oleh Bruce Schneier pada 1993. Algoritma ini dirancang sebagai pengganti Data Encryption Standard (DES) yang dianggap sudah tidak cukup aman. Blowfish menjadi salah satu algoritma enkripsi simetrik yang paling populer pada awal abad ke-21 karena tingkat keamanannya yang baik.

II. LANDASAN TEORI

A. AES-256

Algoritma ini berasal dari kompetisi standard algoritma kriptografi yang diselenggarakan oleh NIST (National Institute of Standard and Technology). Persyaratannya mulai dari merupakan *cipher* blok, rancangannya harus publik, Panjang kunci yang fleksibel, ukuran terenkripsi merupakan 128 bit, dan dapat diimplementasikan baik secara *software* ataupun *hardware*. Algoritma ini kemudian mengadopsi pemenang kompetisi tersebut, yaitu Algoritma Rijndael yang dikembangkan oleh Vincent Rijmen dengan Joan Daemen dari Belgia. Algoritma ini mendukung Panjang kunci 128, 160, 192, 224, dan 256 bit. Algoritma ini juga memiliki Panjang kunci dan ukuran blok yang dapat dipilih secara independent, enkripsi blok dalam sejumlah putaran tertentu, serupa dengan DES, pendahulunya.



Gambar 1 Struktur Arsitektur AES Rijndael

Algoritma ini terdiri dari beberapa tahapan transformasi yang memiliki mekanismenya masing-masing.

a. Tahapan Add Round key bekerja dengan melakukan operasi XOR antara *round key* dan state saat ini. Kunci yang unik dihasilkan dalam proses ini.

b. Tahapan Sub Bytes bekerja dengan menggunakan tabel Substitution Box yang telah tersedia dan mensubstitusikan setiap bytenya dengan box tersebut. Perbandingan yang semakin kompleks menghasilkan tingkat keamanan yang semakin baik.

c. Tahapan Shift Rows bekerja dengan menggeser secara siklik pada 3 baris terbawah yang ada pada state tertentu. Baris teratas tidak mengalami pergeseran, sementara baris dibawahnya mengalami pergesaran sebanyak satu kali lebih banyak dari posisi atasnya.

d. Tahapan Mix Columns bekerja dengan mengalikan matriks state sekarang dengan matriks pengacakan tertentu untuk meningkatkan tingkat keamanan algoritma pada sektor difusinya.

Substitution Box

Elemen yang ada dalam kotak ini bukan diisi secara acak, namun dihasilkan dari proses perhitungan sebagai berikut.

a. Inisialisasi nilai awal

S-Box diinisialisasi terlebih dahulu dalam bentuk matriks 16×16. Setiap baris diisi dengan nilai heksadesimal secara berurutan, dimulai dari baris pertama, baris kedua, hingga baris terakhir.

b. Pencarian Invers Multiplikatif

Setiap nilai pada matriks kemudian diganti dengan invers multiplikatifnya di dalam lapangan Galois GF(2^8). Nilai 00 dipetakan ke dirinya sendiri karena tidak memiliki invers.

c. Kalikan dengan matriks affine

Setelah itu, setiap byte akan dilakukan transformasi affine dengan mengalikan matriks sebuah 8×8 dan menambahkan vektor konstanta.

B. Blowfish

Algoritma Blowfish bekerja dengan ukuran blok **64 bit** dan mendukung berbagai panjang kunci, mulai dari 32 bit hingga 448 bit. Dalam penelitian ini, akan digunakan kunci berukuran 256 bit agar sesuai dengan ukuran Algoritma AES yang telah dipilih. Salah satu keunggulan utama Blowfish adalah proses pembangkitan kunci yang kompleks, sehingga meskipun kunci awal relatif pendek, proses inisialisasi menghasilkan subkunci yang sangat besar dan aman. Proses enkripsi Blowfish terdiri dari dua tahap utama, yaitu *key expansion* dan data *encryption*.

Pada tahap *key expansion*, kunci awal diubah menjadi beberapa array subkunci berukuran besar. Pada tahap enkripsi, data dibagi menjadi dua bagian sesuai prinsip Feistel Network, kemudian melalui 16 ronde yang melibatkan operasi XOR, substitusi menggunakan Substitution Box, dan penambahan subkunci. Walaupun Blowfish termasuk algoritma klasik, hingga saat ini belum ditemukan kriptanalisis yang efektif untuk memecahkannya untuk kunci yang cukup panjang. Oleh karena itu, Blowfish masih relevan dan menarik untuk dijadikan bahan perbandingan dengan algoritma modern seperti AES, terutama dalam mengkaji perbedaan performa, efisiensi, dan karakteristik keamanan antara algoritma generasi lama dan baru.

III. DESAIN PENELITIAN

Penelitian ini dilakukan dengan pendekatan eksperimen kuantitatif melalui *comparative performance analysis*. Penelitian ini bertujuan untuk menganalisis dan membandingkan performa algoritma AES-256 serta Blowfish dalam proses enkripsi dan dekripsi pesan teks. Kedua algoritma dalam penelitian ini diimplementasikan dan akan diujikan pada Google Colab Notebook menggunakan bahasa pemrograman Python 3 dengan memanfaatkan library yang telah tersedia. Spesifikasi komputer yang digunakan adalah sebagai berikut.

- OS : Windows 11 64-bit
- Processor : Intel® Core™ i5-10500H CPU @2.50 GHz (12 CPUs), ~2.5GHz
- Memory : 16 GB
- Bahasa : Python 3.12 atau lebih baru

Objek penelitian dalam studi ini adalah pesan teks berbahasa Indonesia dan Inggris dengan variasi Panjang sebagai berikut.
a. 100 karakter
b. 500 karakter
c. 2.000 karakter
d. 10.000 karakter
e. 50.000 karakter

Setiap ukuran teks akan diuji sebanyak 50 kali pengulangan untuk mendapatkan nilai rata-rata yang representatif dan mengurangi pengaruh variasi waktu eksekusi. Variabel yang diukur meliputi waktu enkripsi, waktu dekripsi, *throughput* (kecepatan pemrosesan data), serta *Avalanche Effect*.

Penelitian ini mengukur performa kedua algoritma berdasarkan beberapa parameter utama, yaitu:

a. Waktu Enkripsi dan Dekripsi — dihitung sebagai indikator performa kecepatan untuk kedua algoritma yang digunakan
b. Throughput — dihitung berdasarkan data yang berhasil dienkripsi atau didekripsi untuk menilai performa kedua algoritma yang digunakan
c. Avalanche Effect — dihitung dengan mengubah satu bit pada plaintext dan melihat berapa persentase bit yang berubah dalam kedua algoritma yang digunakan
d. Ukuran Ciphertext — penambahan ukuran data *post-encryption* pada kedua algoritma yang digunakan
e. Penggunaan memori — seberapa efisien kedua algoritma yang digunakan dalam proses enkripsi dan dekripsi

Proses pengujian dilakukan secara bertahap, dimulai dari inisialisasi kunci, pembacaan teks, proses enkripsi,

pengukuran waktu, proses dekripsi, hingga validasi hasil dekripsi agar sesuai dengan plaintext awalnya.

Jalankan kode ini terlebih dahulu sebelum memulai tes uji pada kedua algoritma yang akan digunakan.

```
!pip install psutil
```

Setelah berhasil ditambahkan, akan dilanjutkan dengan kode berikut.

```
import time
import os
import psutil
from
cryptography.hazmat.primitives.ciphers.aead
import AESGCM
from Crypto.Cipher import Blowfish
from Crypto.Util.Padding import pad, unpad

KEY_AES = os.urandom(32)
KEY_BLOWFISH = os.urandom(16)

def encrypt_aes(plaintext):
    aesgcm = AESGCM(KEY_AES)
    nonce = os.urandom(12)
    start = time.perf_counter()
    ciphertext = aesgcm.encrypt(nonce,
plaintext.encode('utf-8'), None)
    end = time.perf_counter()
    return ciphertext, nonce, (end - start) *
1000

def decrypt_aes(ciphertext, nonce):
    aesgcm = AESGCM(KEY_AES)
    start = time.perf_counter()
    plaintext = aesgcm.decrypt(nonce,
ciphertext, None)
    end = time.perf_counter()
    return plaintext.decode('utf-8'), (end -
start) * 1000

def encrypt_blowfish(plaintext):
    cipher = Blowfish.new(KEY_BLOWFISH,
Blowfish.MODE_CBC)
    padded = pad(plaintext.encode('utf-8'),
8)
    start = time.perf_counter()
    ciphertext = cipher.encrypt(padded)
    end = time.perf_counter()
    return ciphertext, cipher.iv, (end -
start) * 1000

def decrypt_blowfish(ciphertext, iv):
    cipher = Blowfish.new(KEY_BLOWFISH,
Blowfish.MODE_CBC, iv)
    start = time.perf_counter()
    decrypted = cipher.decrypt(ciphertext)
    plaintext = unpad(decrypted, 8)
    end = time.perf_counter()
    return plaintext.decode('utf-8'), (end -
start) * 1000

def calculate_avalanche_effect(plaintext,
algorithm="AES"):
    pt_bytes =
bytearray(plaintext.encode('utf-8'))
    pt_bytes[0] ^= 0x01
    modified_text = pt_bytes.decode('utf-8',
errors='ignore')

    if algorithm == "AES":
        ct1, _, _ = encrypt_aes(plaintext)
        ct2, _, _ =
encrypt_aes(modified_text)
    else:
        ct1, _, _ =
encrypt_blowfish(plaintext)
        ct2, _, _ =
encrypt_blowfish(modified_text)

    diff_bits = sum(bin(a ^ b).count('1') for
a, b in zip(ct1, ct2))
    total_bits = len(ct1) * 8
    return (diff_bits / total_bits) * 100

def get_memory_usage():
    process = psutil.Process(os.getpid())
    return process.memory_info().rss / (1024
* 1024)

def test_performance():
    sizes = [100, 500, 2000, 10000, 50000]
    print(f"{'Ukuran':<8} {'AES Time':<12}
{'Blow Time':<12} {'AES Thrpt':<12} {'Blow
Thrpt':<12} {'AES Avl':<10} {'Blow Avl':<10}
{'AES Size':<10} {'Blow Size':<10} {'Mem
AES':<10} {'Mem Blow':<10}")
    print("=" * 130)

    for length in sizes:
        text = ("Tes. " * (length // 90 +
1))[:length]

        mem_before = get_memory_usage()

        ct_aes, nonce_aes, t_aes_enc =
encrypt_aes(text)
        _, t_aes_dec = decrypt_aes(ct_aes,
nonce_aes)
        total_aes = t_aes_enc + t_aes_dec
```

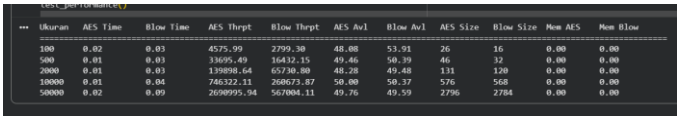
```
        throughput_aes = (length / 1024) /
(total_aes / 1000) if total_aes > 0 else 0
        size_aes = len(ct_aes)

        ct_blow, iv_blow, t_blow_enc =
encrypt_blowfish(text)
        _, t_blow_dec =
decrypt_blowfish(ct_blow, iv_blow)
        total_blow = t_blow_enc + t_blow_dec
        throughput_blow = (length / 1024) /
(total_blow / 1000) if total_blow > 0 else 0
        size_blow = len(ct_blow)

        av_aes =
calculate_avalanche_effect(text, "AES")
        av_blow =
calculate_avalanche_effect(text, "Blowfish")

        mem_after = get_memory_usage()
        mem_usage_aes = mem_after -
mem_before
        mem_usage_blow = mem_after -
mem_before

        print(f"{length:<8}
{total_aes:<12.2f} {total_blow:<12.2f}
{throughput_aes:<12.2f}
{throughput_blow:<12.2f} {av_aes:<10.2f}
{av_blow:<10.2f} {size_aes:<10}
{size_blow:<10} {mem_usage_aes:<10.2f}
{mem_usage_blow:<10.2f}")

test_performance()
```

Kode program di atas digunakan untuk membandingkan performa algoritma AES-256 dan algoritma Blowfish dalam berbagai variable yang telah ditentukan pada bagian sebelumnya.

Fungsi `encrypt_aes` digunakan untuk enkripsi pesan menggunakan algoritma AES-256, sementara `decrypt_aes` digunakan untuk melakukan dekripsi pesan yang tadi sudah dienkripsi. Fungsi `encrypt_blowfish` digunakan untuk enkripsi pesan menggunakan algoritma Blowfish, sementara `decrypt_blowfish` digunakan untuk melakukan dekripsi pesan yang tadi sudah dienkripsi.

Fungsi `calculate_avalanche_effect` digunakan untuk mengukur efek avalanche pada kedua algoritma yang digunakan. Mekanismenya adalah dengan merubah satu bit plaintext dan dilanjutkan dengan menghitung persentase bit yang berbeda antara kedua ciphertext yang telah dienkripsi. Fungsi `get_memory_usage` digunakan untuk mengukur penggunaan memori kedua algoritma yang dipanggil sebelum dan sesudah proses selesai dilakukan. Sementara itu, fungsi `test_performance()` digunakan untuk menguji fungsi-fungsi yang telah ditetapkan untuk berbagai varian *plaintext*. Hasil pengujiannya akan disajikan sebagai berikut.

IV. ANALISIS DAN PEMBAHASAN



Gambar 2 Hasil Pengukuran Performa kedua algoritma

Tabel 1 Tabel waktu yang dibutuhkan (ms)

Karakter	AES	Blowfish
100	0.02	0.03
500	0.01	0.03
2000	0.01	0.03
10000	0.01	0.04
50000	0.02	0.09

Dari hasil pengujian di atas, data menunjukkan AES memiliki waktu yang lebih baik dari Blowfish untuk semua jenis karakter, dengan performwa waktu terdekat pada jenis 100 karakter, yang hanya berbeda 0.01 ms saja. Perbedaan terjauh dapat dilihat pada jenis 50000 karakter, terlihat perbedaan hingga 0.07 ms.

Tabel 2 Tabel data throughput (KB/s)

Karakter	AES	Blowfish
100	4575.99	2799.30
500	33695.49	16432.15
2000	139898.64	65730.80
10000	746322.11	260673.87
50000	2690995.94	567004.11

Dari hasil pengujian di atas, data menunjukkan AES memiliki throughput sekitar 200% dari angka yang didapatkan oleh algoritma Blowfish. Hal ini menandakan bahwa AES memiliki kecepatan pemrosesan data yang sangat konsisten.

Tabel 3 Tabel avalanche effect (%)

Karakter	AES	Blowfish
100	49.04	49.22
500	46.74	50.00
2000	50.19	48.33
10000	49.33	51.56
50000	50.04	49.67

Dari hasil pengujian di atas, cukup menarik bahwa kedua algoritma memiliki nilai avalanche effect yang sangat mirip satu sama lain di semua jenis karakter. Hal ini menunjukkan bahwa keduanya memiliki efek difusi dan tingkat keamanan yang cukup baik.

Tabel 4 Tabel size ciphertext (byte)

Karakter	AES	Blowfish
100	26	16
500	46	32
2000	131	120
10000	576	568

50000	2796	2784

Dari hasil pengujian di atas, dapat dilihat bahwa ukuran ciphertext yang dihasilkan kedua algoritma sebenarnya relatif mirip satu sama lain, dengan algoritma AES memiliki ukuran yang sedikit lebih besar dibandingkan dengan Blowfish. Sementara untuk penggunaan memori, kode yang tadi menghasilkan angka 0.00 untuk kedua algoritma untuk semua jenis karakter.

V. KESIMPULAN

Berdasarkan hasil perbandingan performa antara algoritma AES-256 dan Blowfish yang telah dilakukan tadi, dapat disimpulkan bahwa algoritma AES-256 memiliki performa yang jauh lebih baik dibandingkan dengan algoritma Blowfish, terutama pada waktu yang dibutuhkan dalam melakukan proses enkripsi-dekripsi dan data throughput yang dihasilkan. Selain itu, ditemukan bahwa algoritma Blowfish bisa memberikan avalanche effect dan ukuran ciphertext yang relatif identik dengan AES, walaupun memiliki waktu perilisan yang berbeda hampir satu dekade, yaitu tahun 1993 hingga 2001.

VI. ACKNOWLEDGMENT

Dengan ini, penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas anugerah yang telah memberikan waktu untuk menyelesaikan mata kuliah Kriptografi STI. Penulis juga ingin memberikan terima kasih yang dalam kepada Bapak Prof. Dr. Ir. Rinaldi Munir M.T yang sudah mengajar dan memberikan banyak ilmu dan wawasan, terutama yang terkait dengan kriptografi. Penulis juga ingin mengucapkan terima kasih kepada kelompok Tugas Besar 1 sampai Tugas Besar 4 yang sudah membantu mengerjakan tugas hingga selesai.

REFERENCES

[1] B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," in Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 191-204. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] R. Munir, "Review Beberapa Block Cipher (Bagian 2: AES)," Slide Kuliah II4021 Kriptografi, Institut Teknologi Bandung, 2026.

[3] A. S. Pramono, "RSA, AES, dan SHA-256: Pilar Keamanan Data di Dunia Digital 2020-2025," BSE - Blog Sekolah Elektronik Telkom University, 2020. [Online]. Available: https://bse-sby.telkomuniversity.ac.id/rsa-aes-dan-sha-256-pilar-keamanan-data-di-dunia-digital-2020-2025/. [Diakses: 19 Juni 2026].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Michael Jeremi B. S.

18221136