INFOMATEK: Jurnal Informatika, Manajemen dan Teknologi
Volume 26 Nomor 2, Desember 2024

PERBANDINGAN KINERJA DAN KEAMANAN ALGORITMA KRIPTOGRAFI MODERN AES-GCM DENGAN CHACHA20-POLY1305

Anggi Susanti* ,Bayu Ade Prasetiya, Oktafiyan Dyah Pangesti, Liling Daru Suryawati, Indrawan Ady Saputro

Program Studi Informatika, STMIK AMIKOM Surakarta, Indonesia

Abstrak: Di era digital, kebutuhan akan algoritma kriptografi yang andal semakin penting untuk melindungi data sensitif. AES-GCM dan ChaCha20-Poly1305 adalah dua algoritma populer yang digunakan, namun keduanya memiliki trade-off antara kinerja dan keamanan. Penelitian ini bertujuan mengevaluasi kedua algoritma melalui pengujian kinerja dan simulasi serangan timing. Pengujian kinerja dilakukan dengan mengukur waktu enkripsi dan dekripsi pada berbagai ukuran data, sedangkan simulasi serangan timing menilai kerentanan terhadap serangan tersebut. Hasil penelitian menunjukkan bahwa AES-GCM lebih lambat dibandingkan ChaCha20-Poly1305, terutama pada data berukuran besar, tetapi menawarkan ketahanan lebih baik terhadap serangan timing. Sebaliknya, ChaCha20-Poly1305 unggul dalam kecepatan, menjadikannya pilihan ideal untuk aplikasi dengan kebutuhan performa tinggi tanpa mengorbankan keamanan dasar.

Kata kunci: AES-GCM, ChaCha20-Poly1305, kriptografi, kinerja keamanan, serangan timing

I. PENDAHULUAN

Keamanan informasi dalam dunia digital sangat penting, terutama karena semakin banyaknya transaksi dan komunikasi data yang dilakukan secara online (Waruwu & Sundari, 2024). Salah satu komponen utama dalam menjaga keamanan data adalah algoritma kriptografi, yang bertanggung jawab untuk memastikan kerahasiaan dan integritas informasi. Di antara algoritma kriptografi yang banyak digunakan saat ini, AES-GCM (Advanced Encryption Standard in Galois/Counter Mode) dan ChaCha20-Poly1305 adalah dua algoritma yang populer karena keduanya menawarkan keamanan yang kuat dan efisiensi yang tinggi (Wang et al., 2021); (Almeida & Pereira, 2020)

AES-GCM (Advanced Encryption Standard - Galois/Counter Mode) merupakan salah satu mode operasi dari algoritma AES yang menggabungkan enkripsi dan autentikasi data dalam satu langkah (Widyastuti et al., 2019). Dengan memanfaatkan *counter mode* untuk proses enkripsi dan algoritma Galois (GHASH) untuk autentikasi, AES-GCM memastikan data tetap aman serta terlindungi dari upaya manipulasi (Kelana et al., 2021). Mode ini sangat unggul dalam hal efisiensi karena memungkinkan enkripsi dan autentikasi dijalankan secara bersamaan, sehingga cocok untuk aplikasi dengan kebutuhan performa tinggi seperti protokol keamanan modern (misalnya, TLS dan IPsec). Selain itu, AES-GCM memanfaatkan kunci kriptografi, *nonce* (angka unik), data tambahan terautentikasi

*) ccnqqi5@gmail.com

Diterima: 21 Oktober 2024
Direvisi: 20 Desember 2024
Disetujui: 24 Desember 2024
DOI: 10.23969/infomatek.v26i2.19255

(AAD), serta menghasilkan tag autentikasi untuk menjamin keutuhan data. (Marsiani et al., 2021),(Zhang & Wang, 2023). Namun, penelitian menunjukkan bahwa meskipun AES-GCM memiliki tingkat keamanan yang tinggi, ada beberapa kelemahan dalam hal kinerjanya, terutama pada perangkat dengan sumber daya terbatas, seperti perangkat mobile atau IoT (Huo & Wang, 2023),(Hassan et al., 2022).

Sebagai alternatif, ChaCha20-Poly1305 menawarkan keuntungan dalam hal efisiensi di perangkat dengan kemampuan komputasi terbatas, seperti yang banyak digunakan pada perangkat mobile dan IoT (Pfau et al., 2019),(Serrano et al., 2022). Algoritma ini dianggap lebih tahan terhadap beberapa jenis serangan kriptanalisis dan tidak bergantung pada akselerasi perangkat keras, yang membuatnya lebih fleksibel di berbagai platform. ChaCha20-Poly1305 telah diadopsi dalam protokol TLS 1.3 dan aplikasi lain seperti WireGuard VPN, menunjukkan bahwa algoritma ini mampu memberikan keamanan yang setara dengan AES-GCM, namun dengan kinerja yang lebih konsisten di berbagai perangkat (Serrano et al., 2021),(Ghosh & Dey, 2024).

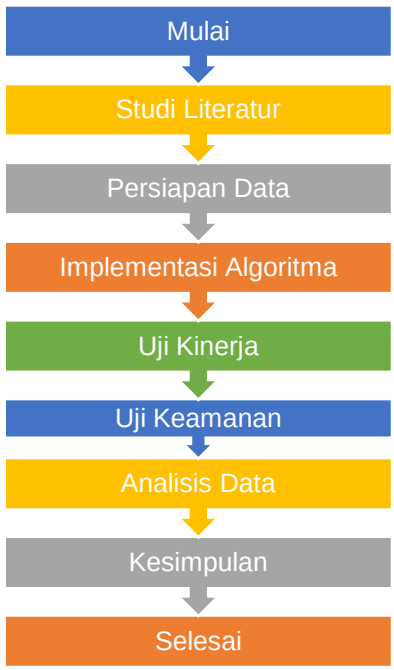
Meskipun kedua algoritma ini banyak digunakan dalam berbagai aplikasi, perbandingan langsung antara kinerja dan keamanan AES-GCM dan ChaCha20-Poly1305 dalam berbagai kondisi perangkat dan serangan kriptanalisis masih terbatas (Muhammad et al., 2024) (Donzilio Antonio Meko, 2018). Penelitian mengenai perbandingan secara menyeluruh dalam konteks perangkat keras dengan sumber daya terbatas, serta dalam menghadapi serangan yang semakin canggih, masih jarang dilakukan (Singh et al., 2019). Selain itu, meskipun kedua algoritma ini terkenal kuat dalam aspek keamanan, riset lebih lanjut diperlukan untuk menilai bagaimana mereka menangani potensi kerentanannya, seperti dalam serangan saluran samping (side-channel attacks) atau serangan terhadap implementasi perangkat keras (Hassan et al., 2022).

Penelitian ini sangat penting untuk memberikan wawasan lebih dalam tentang kelebihan dan kelemahan AES-GCM serta ChaCha20-Poly1305, khususnya dalam konteks perangkat dengan daya terbatas dan aplikasi yang memerlukan kecepatan tinggi. Dengan meningkatnya penggunaan perangkat mobile dan IoT dalam berbagai sektor, pemilihan algoritma kriptografi yang tepat akan sangat mempengaruhi keamanan dan kinerja sistem secara keseluruhan (Zhao et al., 2021). Oleh karena itu, penelitian ini diharapkan dapat memberikan kontribusi yang berarti dalam membantu pengembang sistem memilih algoritma yang paling sesuai dengan kebutuhan keamanan dan performa di dunia yang semakin terhubung ini.

II. METODOLOGI

2.1. Metode Pengumpulan Data

Metode penelitian ini bertujuan untuk membandingkan kinerja dan keamanan dua algoritma kriptografi modern, yaitu AES-GCM dan ChaCha20-Poly1305.Uji kinerja dilakukan untuk mengukur waktu enkripsi dan dekripsi menggunakan kedua algoritma, sementara uji keamanan dilakukan dengan mensimulasikan serangan timing untuk melihat sensitivitas masing-masing algoritma terhadap serangan tersebut. Alur Penelitian tersaji pada Gambar 1.



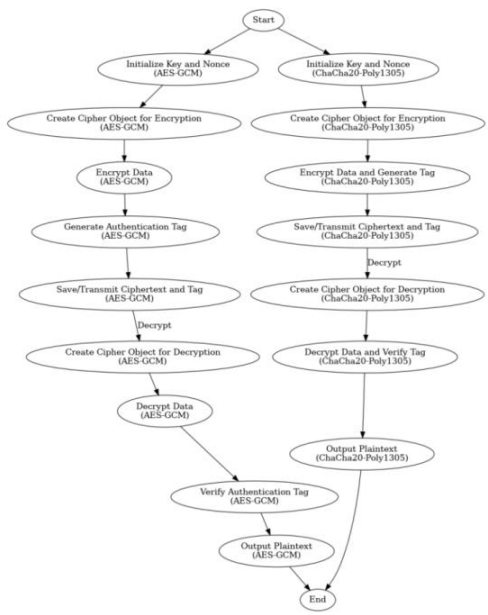
Gambar 1. Alur Penelitian

Penelitian dimulai dengan menentukan topik penelitian dan tujuan utama, yaitu membandingkan kinerja dan keamanan algoritma AES-GCM dan ChaCha20-Poly1305. Langkah pertama melibatkan peninjauan literatur yang relevan. Ini termasuk mempelajari kedua algoritma secara mendalam, memahami prinsip enkripsi dan dekripsi, serta autentikasi. Literatur juga akan mencakup penelitian sebelumnya yang berkaitan dengan uji kinerja dan keamanan kedua algoritma ini. Kemudian pada tahap persiapan data disiapkan beberapa dataset untuk uji kinerja, beberapa dataset dengan ukuran yang bervariasi akan disiapkan. Sedangkan untuk uji keamanan, dua set data yang hampir identik (misalnya, dengan perbedaan 1 bit) akan disiapkan untuk mensimulasikan serangan timing.

Kedua algoritma, AES-GCM dan ChaCha20-Poly1305, diimplementasikan menggunakan bahasa pemrograman Python. Implementasi ini termasuk enkripsi, dekripsi, dan autentikasi data. Untuk Uji kinerja dilakukan dengan mengenkripsi dan mendekripsi dataset yang telah disiapkan menggunakan kedua algoritma. Waktu yang dibutuhkan untuk setiap proses akan dicatat dan dianalisis, tahapan uji kinerja tersaji pada gambar 2.

Gambar 2. Alur Uji Kinerja Algoritma AES-GCM dan ChaCha20-Poly1305.

Langkah awal dalam alur uji kinerja adalah menyiapkan dataset, Dataset dengan berbagai ukuran (misalnya, 1 KB, 1 MB, 10 MB) disiapkan untuk diuji. Data ini akan digunakan untuk mengukur kinerja enkripsi dan dekripsi dari kedua algoritma menggunakan python.Dataset dienkripsi dan didekripsi menggunakan kedua algoritma. Proses ini diulang untuk setiap ukuran dataset untuk mengukur waktu yang dibutuhkan oleh masing-masing algoritma yang nantinya akan dicatat.Hasil analisis ini akan menunjukkan efisiensi masing-masing algoritma dalam berbagai skenario penggunaan. Secara lebih detail, berikut bagaimana cara kerja enkripsi dan dekripsi masing masing algoritma bekerja pada uji kinerja yang akan tersaji pada Gambar 3.



Gambar 3. Flowchart Alur Enkripsi dan Dekripsi Uji Algoritma AES-GCM dan ChaCha20-Poly1305

Untuk AES-GCM, proses dimulai dengan inisialisasi kunci enkripsi dan nonce, yang merupakan nilai unik yang digunakan untuk memastikan keamanan enkripsi. Setelah itu, sebuah objek cipher dibuat menggunakan mode AES-GCM, dengan kunci dan nonce yang telah diinisialisasi. Data kemudian dienkripsi menggunakan objek cipher ini, menghasilkan ciphertext. Langkah berikutnya adalah menghasilkan tag autentikasi, yang penting untuk memastikan integritas data selama proses dekripsi. Ciphertext dan tag tersebut kemudian disimpan atau dikirimkan bersama dengan nonce. Dalam proses dekripsi, objek cipher baru dibuat menggunakan kunci dan nonce yang sama. Data yang telah dienkripsi kemudian didekripsi, dan integritas serta keasliannya diverifikasi menggunakan tag yang telah dihasilkan. Jika tag autentikasi valid, data asli yang terdekripsi akan dihasilkan. Untuk ChaCha20-Poly1305, proses dimulai dengan inisialisasi kunci dan nonce, mirip dengan AES-GCM. Objek cipher kemudian dibuat menggunakan mode ChaCha20- Poly1305, di mana kunci dan nonce yang telah diinisialisasi digunakan. Proses enkripsi dilakukan, dan berbeda dengan AES-GCM, ChaCha20-Poly1305 menghasilkan ciphertext dan tag autentikasi dalam satu langkah tunggal. Setelah proses enkripsi selesai, ciphertext, tag, dan nonce disimpan atau dikirimkan untuk proses dekripsi. Dalam dekripsi, objek cipher baru dibuat dengan kunci dan nonce yang sama. Ciphertext kemudian didekripsi, dan pada saat yang sama, tag autentikasi diverifikasi. Jika tag valid, data asli akan dihasilkan. Simulasi serangan timing dalam konteks uji keamanan bertujuan untuk mengukur seberapa rentan algoritma kriptografi terhadap serangan yang memanfaatkan perbedaan waktu proses enkripsi untuk memperoleh informasi rahasia. Kemudian untuk alur uji keamanan akan ditampilkan pada Gambar 4.

Gambar 4. Flowchart Alur Uji Keamanan Algoritma AES-GCM dan ChaCha20-Poly1305

Dalam uji ini, serangan timing bekerja dengan memperhatikan perbedaan kecil dalam waktu yang dibutuhkan oleh algoritma untuk memproses dua set data yang hampir identik. Langkah pertama adalah menyiapkan dua set data yang hanya berbeda dalam satu bit atau sangat kecil. Misalnya, jika satu set data adalah "Hello, World!" maka set data kedua mungkin adalah "Hello, Worlc!", di mana satu karakter diubah. Tujuannya adalah untuk menciptakan dua input yang secara kriptografi mirip tetapi berbeda sedikit, yang memungkinkan serangan timing mengamati perbedaan dalam cara data tersebut diproses oleh algoritma. Pada tahap implementasi algoritma, baik AES-GCM maupun ChaCha20-Poly1305 dienkripsi dengan kedua set data. Meskipun perubahan data tersebut sangat kecil, perubahan ini bisa menyebabkan variasi dalam pola akses memori, jumlah operasi komputasi, atau penanganan internal lainnya dalam algoritma. Perbedaan ini mungkin tercermin dalam waktu yang dibutuhkan oleh algoritma untuk menyelesaikan enkripsi. Simulasi serangan timing kemudian dilakukan dengan menjalankan proses enkripsi pada kedua set data berulang kali, dan setiap kali mencatat waktu yang dibutuhkan untuk menyelesaikan proses enkripsi. Dengan melakukan enkripsi berkali-kali, peneliti dapat mengumpulkan sejumlah besar data waktu yang kemudian digunakan untuk mengidentifikasi pola atau perbedaan yang muncul antara waktu enkripsi dari dua set data tersebut. Perbedaan dalam waktu enkripsi ini bisa disebabkan oleh berbagai faktor internal algoritma, seperti pengoptimalan yang berbeda dalam penanganan data atau kondisi seperti adanya cabang kondisi yang bergantung pada nilai input. Dalam konteks serangan timing, seorang penyerang bisa mencoba mengeksploitasi perbedaan kecil ini untuk mengekstrak informasi tentang kunci enkripsi atau data lain yang seharusnya tidak dapat diakses. Data yang dikumpulkan dari simulasi ini kemudian dianalisis untuk melihat apakah perbedaan waktu enkripsi tersebut konsisten dan signifikan. Jika satu algoritma menunjukkan perbedaan waktu yang signifikan antara dua set data, maka algoritma tersebut dianggap lebih rentan terhadap serangan timing. Sebaliknya, jika perbedaan waktu tidak signifikan, algoritma dianggap lebih aman dari perspektif serangan timing. Kemudian setelah dilakukanya uji kinerja dan uji keamanan adalah melakukan analisis data, Data yang dikumpulkan dari uji kinerja dan uji keamanan akan dianalisis. Analisis ini akan mencakup perbandingan antara kedua algoritma dalam hal kecepatan dan ketahanan terhadap serangan. Dan langkah terakhir dalam penelitian ini adalah menarik kesimpulan. Berdasarkan hasil analisis, kesimpulan akan dibuat mengenai algoritma mana yang lebih unggul dalam hal kinerja dan keamanan. Rekomendasi akan diberikan mengenai skenario penggunaan yang tepat untuk masing-masing algoritma.

III. HASIL DAN PEMBAHASAN

Dalam bagian ini, peneliti akan membahas hasil pengujian kinerja dan keamanan dari algoritma kriptografi AES-GCM dan ChaCha20-Poly1305 secara mendetail. Tujuan dari analisis ini adalah untuk menilai efektivitas dan efisiensi kedua algoritma tersebut. Untuk mencapai tujuan ini, kami menggunakan metode pengujian berbasis kode yang melibatkan pengukuran waktu enkripsi dan dekripsi serta simulasi serangan timing.sebelum-sebelumnya untuk mencari literatur dalam memperkaya pembahasan. Hasilnya akan disajikan melalui tabel dan grafik, yang memungkinkan analisis kualitatif dan kuantitatif terhadap performa dan ketahanan keamanan masing-masing algoritma.

3.1 Uji Kinerja

Pada penelitian ini, uji kinerja dilakukan untuk membandingkan waktu enkripsi dan dekripsi antara algoritma AES-GCM dan ChaCha20-Poly1305. Uji kinerja dilakukan dengan menggunakan data statis berukuran 1 KB, 1 MB, dan 10 MB. Berikut adalah kode yang digunakan untuk mengukur waktu enkripsi dan dekripsi tersaji pada gambar 5.

uji kinerja

```
import time
from Crypto.Cipher import AES, ChaCha20_Poly1305
import matplotlib.pyplot as plt

# Fungsi untuk menghasilkan data string dengan panjang tertentu
def generate_data(size):
    return b'A' * size

def aes_gcm_encrypt_decrypt(data):
    key = b'0123456789abcdef0123456789abcdef'  # 256-bit key (32 bytes)
    nonce = b'0123456789ab'  # 96-bit nonce (12 bytes)

    # Enkripsi
    cipher = AES.new(key, AES.MODE_GCM, nonce=nonce)
    start_time = time.time()
    ciphertext, tag = cipher.encrypt_and_digest(data)
    encrypt_time = time.time() - start_time

    # Dekripsi
    cipher = AES.new(key, AES.MODE_GCM, nonce=nonce)
    start_time = time.time()
    decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
    decrypt_time = time.time() - start_time

    return encrypt_time, decrypt_time

def chacha20_poly1305_encrypt_decrypt(data):
    key = b'0123456789abcdef0123456789abcdef'  # 256-bit key (32 bytes)
    nonce = b'0123456789ab'  # 96-bit nonce (12 bytes)

    # Enkripsi
    cipher = ChaCha20_Poly1305.new(key=key, nonce=nonce)
    start_time = time.time()
    ciphertext, tag = cipher.encrypt_and_digest(data)
    encrypt_time = time.time() - start_time

    # Dekripsi
    cipher = ChaCha20_Poly1305.new(key=key, nonce=nonce)
    start_time = time.time()
    decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
    decrypt_time = time.time() - start_time

    return encrypt_time, decrypt_time
```

Gambar 5. Kode Uji Kinerja Algoritma AES-GCM dan ChaCha20-Poly1305.

. Pada impor modul terdapat Modul time digunakan untuk mengukur waktu eksekusi, Crypto.Cipher dari pustaka pycryptodome menyediakan implementasi algoritma AES dan ChaCha20-Poly1305, sementara matplotlib.pyplot digunakan untuk visualisasi grafik. Kemudian pada Fungsi 'generate_data(size)' digunakan untuk menghasilkan data berupa string byte yang panjangnya ditentukan oleh parameter size. Data ini diisi dengan karakter A dan digunakan untuk pengujian enkripsi dan dekripsi Untuk Fungsi 'aes_gcm_encrypt_ decrypt(data)' terdapat Inisialisasi Kunci ('key') dan nonce ('nonce') yang berupa nilai statis, dengan panjang masing-masing 256-bit dan 96-bit. Kemudian Sebuah objek 'AES' baru dibuat dengan mode GCM dan nonce yang ditentukan. Fungsi 'encrypt_and_digest()' digunakan untuk mengenkripsi data, menghasilkan 'ciphertext' dan 'tag' (yang digunakan untuk verifikasi). Waktu yang diperlukan untuk enkripsi diukur. Objek 'AES' baru dibuat dengan mode GCM dan nonce yang sama. Fungsi 'decrypt_and_verify()' digunakan untuk mendekripsi data yang sudah dienkripsi dan memverifikasi 'tag'. Waktu yang diperlukan untuk dekripsi diukur. Selanjutnya Fungsi 'chacha20_poly1305_encrypt_decrypt(data)', Sama seperti AES, kunci dan nonce adalah statis dengan panjang yang sama, Sebuah objek 'ChaCha20_Poly1305' baru dibuat dengan kunci dan nonce yang ditentukan. Fungsi 'encrypt_and_digest()' digunakan untuk mengenkripsi data, menghasilkan 'ciphertext' dan 'tag'. Waktu yang diperlukan untuk enkripsi diukur.Objek 'ChaCha20_Poly1305' baru dibuat dengan kunci dan nonce yang sama. Fungsi 'decrypt_and_verify()' digunakan untuk mendekripsi data yang sudah dienkripsi dan memverifikasi 'tag'. Waktu yang diperlukan untuk dekripsi diukur. Kemudian untuk menjalankan benchmark kinerja dari algoritma AES-GCM dan ChaCha20-Poly1305 dengan berbagai ukuran data dan memvisualisasikan hasilnya dalam bentuk grafik, maka kodenya dapat dilihat pada Gambar 6.

Gambar 6. Kode Uji Kinerja Algoritma AES-GCM dan ChaCha20-Poly1305

Fungsi ‘run_benchmark()’ dimulai dengan mendefinisikan beberapa ukuran data yang akan digunakan untuk pengujian: 1 KB, 1 MB, dan 10 MB. Kode ini menyimpan ukuran data dalam daftar ‘data_sizes’ dan mempersiapkan dua kamus, ‘aes_times’ dan ‘chacha_times’, untuk menyimpan waktu enkripsi dan dekripsi masing-masing algoritma. Untuk setiap ukuran data dalam data_sizes, fungsi ini melakukan hal berikut:

1. Menghasilkan Data : Fungsi ‘generate_data(size)’ dipanggil untuk membuat data byte dengan ukuran yang sesuai. Data ini akan digunakan sebagai input untuk enkripsi dan dekripsi.
2. Pengujian AES-GCM : Fungsi ‘aes_gcm_encrypt_decrypt(data)’ dipanggil untuk mengenkripsi dan mendekripsi data.Waktu yang diperlukan untuk proses enkripsi dan dekripsi disimpan dalam variabel ‘aes_encrypt_time’ dan ‘aes_decrypt_time’, kemudian dimasukkan ke dalam kamus ‘aes_times’.
3. Pengujian ChaCha20-Poly1305 : Fungsi ‘chacha20_poly1305_encrypt_decrypt(dat a)’ dipanggil untuk mengenkripsi dan mendekripsi data.Waktu yang diperlukan untuk proses enkripsi dan dekripsi disimpan dalam variabel ‘chacha_encrypt_time’ dan ‘chacha_decrypt_time’, kemudian dimasukkan ke dalam kamus chacha_times.
4. Menampilkan Hasil : Waktu enkripsi dan dekripsi untuk setiap algoritma dan ukuran data dicetak ke layar. Separator garis ("-" * 50) digunakan untuk memisahkan hasil pengujian untuk setiap ukuran data.

Setelah semua ukuran data diuji, fungsi run_benchmark() mengembalikan data_sizes, aes_times, dan chacha_times yang berisi waktu yang diperlukan untuk enkripsi dan dekripsi untuk kedua algoritma. Bagian selanjutnya dari kode ini adalah visualisasi hasil menggunakan matplotlib, pada matpqotlib Fungsi ‘plt.plot()’ digunakan untuk menggambar grafik waktu enkripsi dan dekripsi untuk AES-GCM dan ChaCha20-Poly1305. Setiap algoritma memiliki dua kurva: satu untuk waktu enkripsi dan satu untuk waktu dekripsi. Kemudian label sumbu X dan Y, judul grafik, dan legenda ditambahkan untuk memperjelas informasi yang ditampilkan, terakhir Fungsi ‘plt.show()’ digunakan untuk menampilkan grafik.

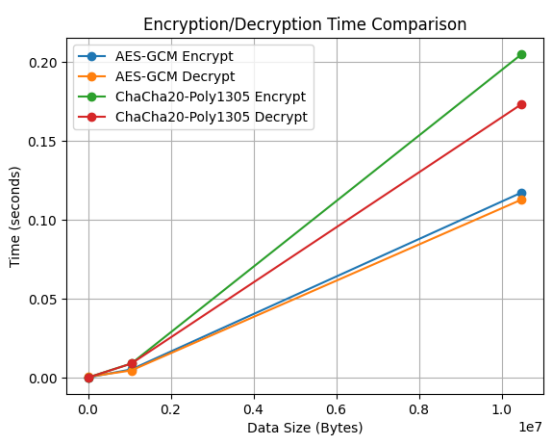
Grafik yang dihasilkan akan menunjukkan bagaimana waktu enkripsi dan dekripsi berubah seiring dengan perubahan ukuran data untuk kedua algoritma kriptografi, memungkinkan perbandingan kinerja yang jelas.

Penjelasan untuk Kode di atas menghasilkan waktu enkripsi dan dekripsi untuk berbagai ukuran data. Hasil pengujian ini dapat dilihat pada Tabel 1.

Tabel 1. Waktu Enkripsi dan Dekripsi AES-GCM dan ChaCha20-Poly1305

Ukuran Data (Bytes)	AES-GCM Encrypt (s)	AES-GCM Decrypt (s)	ChaCha20-Poly1305 Encrypt (s)	ChaCha20-Poly1305 Decrypt (s)
1 KB	0,000134s	0,000542s	0,000153s	0,000096s
1 MB	0,005258s	0,004607s	0,008981s	0,008941s
10 MB	0,117246s	0,112792s	0,204848s	0,173236s

Dari tabel di atas, dapat disimpulkan bahwa ChaCha20-Poly1305 memiliki waktu enkripsi dan dekripsi yang lebih cepat dibandingkan AES-GCM, terutama ketika ukuran data yang dienkripsi dan didekripsi meningkat. Berikut grafik perbandingan Enkripsi dan Dekripsi tersaji pada gambar 7.



Gambar 7. Grafik Perbandingan Uji Kinerja Algoritma AES-GCM dan ChaCha20-Poly1305

3.2 Uji Keamanan

Selain uji kinerja, uji keamanan juga dilakukan untuk mengukur potensi kerentanan terhadap serangan timing pada kedua algoritma. Uji ini memeriksa perbedaan waktu enkripsi antara dua data yang hampir identik. Berikut adalah kode yang digunakan untuk melakukan simulasi serangan timing tersaji pada gambar 8.

```python
import time
from Crypto.Cipher import AES, ChaCha20_Poly1305

def timing_attack_simulation(cipher_func, data1, data2):
    # Mengukur waktu enkripsi untuk data1
    start_time = time.perf_counter()
    cipher_func(data1)
    time1 = time.perf_counter() - start_time

    # Mengukur waktu enkripsi untuk data2
    start_time = time.perf_counter()
    cipher_func(data2)
    time2 = time.perf_counter() - start_time

    return time1, time2
```

Gambar 8. Kode Uji Keamanan Algoritma AES-GCM dan ChaCha20-Poly1305

Hal pertama yang dilakukan adalah mengimpor modul, modul yang digunakan adalah ‘time’ Modul ini digunakan untuk mengukur waktu yang diperlukan untuk mengeksekusi suatu fungsi, dalam konteks ini adalah waktu enkripsi. Kemduian modul ‘Crypto.Cipher’ Digunakan untuk mengimpor algoritma enkripsi AES dan ChaCha20-Poly1305 dari pustaka PyCryptodome.

Defini Fungsi ‘timing_attack_simulation’, Fungsi ini menerima tiga parameter yaitu cipher_func, data1, dan data2. ‘cipher_func’ adalah fungsi enkripsi yang akan diuji. data1 dan data2 adalah dua data yang mirip tetapi sedikit berbeda, yang akan dienkripsi untuk melihat apakah ada perbedaan waktu yang signifikan.

Mengukur waktu enkripsi untuk ‘data1’ dan ‘data2’ dilakukan dalam dua langkah yang serupa, di mana keduanya bertujuan untuk menghitung waktu yang dibutuhkan oleh fungsi enkripsi untuk mengenkripsi masing-masing data.

Untuk setiap data, proses dimulai dengan mencatat waktu saat enkripsi dimulai menggunakan ‘time.perf_counter()’, yang memberikan presisi tinggi dalam pengukuran waktu. Setelah waktu mulai tercatat, fungsi enkripsi (‘cipher_func’) dieksekusi untuk mengenkripsi data yang diberikan (‘data1’ atau ‘data2’). Setelah enkripsi selesai, waktu dihentikan dengan mencatat waktu saat itu dan mengurangi waktu mulai dari waktu akhir untuk mendapatkan total waktu enkripsi (‘time1’ untuk ‘data1’ dan ‘time2’ untuk ‘data2’). Berikut kode yang digunakan tersaji pada gambar 9.

```
def run_timing_attack_simulation():
    # Data kunci dan nonce statis
    key = b'0123456789abcdef0123456789abcdef'  # 256-bit key (32 bytes)
    nonce = b'0123456789ab'  # 96-bit nonce (12 bytes)

    # Data yang hampir sama untuk memeriksa perbedaan waktu
    data1 = b'Hello, World!' + b'\x00' * 100
    data2 = b'Hello, World!' + b'\x01' * 100

    # AES-GCM Timing Attack
    def aes_encrypt(data):
        cipher = AES.new(key, AES.MODE_GCM, nonce=nonce)
        return cipher.encrypt_and_digest(data)

    aes_time1, aes_time2 = timing_attack_simulation(aes_encrypt, data1, data2)

    # ChaCha20-Poly1305 Timing Attack
    def chacha_encrypt(data):
        cipher = ChaCha20_Poly1305.new(key=key, nonce=nonce)
        return cipher.encrypt_and_digest(data)

    chacha_time1, chacha_time2 = timing_attack_simulation(chacha_encrypt, data1, data2)

    print("AES-GCM Timing Attack Simulation:")
    print(f"Waktu enkripsi data1 (AES-GCM): {aes_time1:.9f}s")
    print(f"Waktu enkripsi data2 (AES-GCM): {aes_time2:.9f}s")
    print(f"Perbedaan waktu (AES-GCM): {abs(aes_time1 - aes_time2):.9f}s\n")

    print("ChaCha20-Poly1305 Timing Attack Simulation:")
    print(f"Waktu enkripsi data1 (ChaCha20-Poly1305): {chacha_time1:.9f}s")
    print(f"Waktu enkripsi data2 (ChaCha20-Poly1305): {chacha_time2:.9f}s")
    print(f"Perbedaan waktu (ChaCha20-Poly1305): {abs(chacha_time1 - chacha_time2):.9f}s")

run_timing_attack_simulation()
```

Gambar 9. Kode Uji Keamanan Algoritma AES-GCM dan ChaCha20-Poly1305.

Fungsi run_timing_attack_simulation() bertujuan untuk mensimulasikan serangan waktu terhadap dua algoritma kriptografi, yaitu AES-GCM dan ChaCha20-Poly1305. Fungsi ini secara khusus mengukur perbedaan waktu enkripsi antara dua set data yang hampir identik, yang dapat menjadi indikator kerentanan terhadap serangan waktu.

Pada pembuatan Kunci dan Nonce Statis dimulai dengan mendefinisikan kunci simetris berukuran 256-bit (32 bytes) dan nonce berukuran 96-bit (12 bytes). Kunci dan nonce ini digunakan secara statis dalam proses enkripsi untuk kedua algoritma. Kemudian dua set data, ‘data1’ dan ‘data2’, dibuat dengan konten yang hampir sama. Data ini terdiri dari string "Hello, World!" yang diikuti oleh 100 byte tambahan, di mana data1 memiliki byte tambahan berupa nilai 0x00, sedangkan data2 memiliki byte tambahan berupa nilai 0x01. Perbedaan kecil ini memungkinkan pengujian terhadap perbedaan waktu enkripsi.

Untuk Simulasi Serangan Waktu AES-GCM menggunakan fungsi lokal ‘aes_encrypt’ yang didefinisikan untuk mengenkripsi data menggunakan algoritma AES-GCM dengan kunci dan nonce yang telah ditentukan. Fungsi ini kemudian digunakan dalam ‘timing_attack_simulation’ untuk mengukur waktu yang dibutuhkan untuk mengenkripsi data1 dan data2. Hasilnya adalah dua waktu enkripsi yang disimpan dalam ‘aes_time1’ dan ‘aes_time2’. Sedangkan untuk Simulasi Serangan Waktu untuk ChaCha20-Poly1305, Proses yang sama dilakukan untuk ChaCha20-Poly1305. Fungsi lokal ‘chacha_encrypt’ didefinisikan untuk mengenkripsi data menggunakan algoritma ChaCha20-Poly1305, dan kemudian digunakan dalam ‘timing_attack_simulation’ untuk mengukur waktu enkripsi untuk data1 dan data2. Hasil waktu enkripsi disimpan dalam ‘chacha_time1’ dan ‘chacha_time2’.

Hasil pengukuran waktu enkripsi untuk kedua algoritma dicetak ke konsol. Waktu enkripsi untuk ‘data1’ dan ‘data2’ dibandingkan untuk masing-masing algoritma, dan perbedaan waktu ini dicetak sebagai indikasi potensi kerentanan terhadap serangan waktu. Penjelasan kode di atas digunakan untuk

mengukur waktu enkripsi untuk dua data yang hampir identik, kemudian membandingkan perbedaan waktu enkripsi antara data-data tersebut untuk menilai kerentanannya terhadap serangan timing. Hasilnya disajikan dalam bentuk tabel 2.

Tabel 2. Hasil Simulasi Serangan Timing pada AES-GCM dan ChaCha20-Poly1305

Algoritma	Waktu Enkripsi Data 1 (s)	Waktu Enkripsi Data 2 (s)	Perbedaan Waktu (s)
AES-GCM	0,000287	0,000294	0,000007
ChaCha20-Poly1305	0,000243	0,000251	0,000008

Berdasarkan tabel di atas, terlihat bahwa perbedaan waktu enkripsi antara dua data yang hampir identik sangat kecil, yang menunjukkan bahwa baik AES-GCM maupun ChaCha20-Poly1305 memiliki ketahanan yang cukup baik terhadap serangan timing dalam konteks uji ini. Meskipun perbedaan waktu enkripsi dapat dideteksi, nilainya sangat kecil sehingga sulit untuk dieksploitasi secara praktis dalam kondisi normal.

V. KESIMPULAN

Berdasarkan hasil simulasi pada pembahasan, maka diperoleh hasil berikut :

1. Berdasarkan hasil pengujian kinerja, ChaCha20-Poly1305 menunjukkan waktu enkripsi dan dekripsi yang lebih cepat dibandingkan dengan AES-GCM. Perbedaan ini lebih terasa pada data berukuran besar, yang menjadikan ChaCha20-Poly1305 lebih efisien untuk digunakan dalam aplikasi yang membutuhkan pemrosesan data secara cepat.
2. Dari sisi keamanan terhadap serangan timing, baik AES-GCM maupun ChaCha20-Poly1305 menunjukkan ketahanan yang cukup baik, dengan perbedaan waktu yang minimal ketika data yang hampir sama dienkripsi. Namun, perbedaan waktu ini tetap harus dipertimbangkan dalam implementasi praktis untuk menghindari potensi risiko keamanan, terutama pada aplikasi yang memerlukan tingkat keamanan yang sangat tinggi.
3. Perbandingan ini menunjukkan bahwa pilihan antara AES-GCM dan ChaCha20-Poly1305 harus didasarkan pada kebutuhan spesifik dari aplikasi yang akan digunakan. ChaCha20-Poly1305 mungkin lebih disukai dalam konteks yang membutuhkan kinerja tinggi tanpa dukungan akselerasi perangkat keras, sedangkan AES-GCM tetap menjadi pilihan utama pada aplikasi yang mengutamakan keamanan, terutama jika tersedia akselerasi perangkat keras yang dapat mengurangi waktu enkripsi

DAFTAR PUSTAKA

Almeida, R. P., & Pereira, A. L. (2020). Performance of AES-GCM and ChaCha20-Poly1305 in IoT systems. *Journal of Information Security and Applications*, *55*, 102562.

Donzilio Antonio Meko. (2018). Perbandingan Algoritma DES , AES , IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data. *Jurnal Teknologi Terpadu*, *4*(1), 8–15.

Ghosh, A., & Dey, S. (2024). Optimizing cryptographic performance: AES-GCM vs ChaCha20-Poly1305 in cloud-based environments. *Journal of Cloud Computing: Advances, Systems, and Applications*, *15*(1), 101–116.

Hassan, N., Khalil, I., Dahiya, S., & Ahmed, A. (2022). Side-channel analysis of AES-GCM and ChaCha20-Poly1305: A comparative approach. *International Journal of Information Security*, *21*(3), 367–385.

Huo, X., & Wang, X. (2023). Internet of things for smart manufacturing based on advanced encryption standard (AES) algorithm with chaotic system. *Results in Engineering*, *null*, null. https://doi.org/10.1016/j.rineng.2023.101589

Kelana, O. H., Irawan, P., & Halim, P. (2021). Implementasi One Time Password dengan Metode Advanced Encryption Standard. *Prosiding Seminar Nasional Universitas Ma Chung*, *null*, null. https://doi.org/10.33479/snumc.v1i.233

Marsiani, E. S., Setiadi, I., & Cahyo, A. (2021). Implementasi Sistem Keamanan AES 256-Bit GCM Guna Mengamankan Data Pribadi. *JRKT (Jurnal Rekayasa Komputasi Terapan)*, *null*, null. https://doi.org/10.30998/jrkt.v1i02.4096

Muhammad, R. K., Aziz, R. R., Hassan, A. A., Aladdin, A. M., Saydah, S. J., Rashid, T. A., & Hassan, B. A. (2024). Comparative Analysis of AES , Blowfish , Twofish , Salsa20 , and ChaCha20 for Image Encryption. *Kurdistan Journal of Applied Research (KJAR)*, *9*(1).

Pfau, J., Reuter, M., Harbaum, T., Hofmann, K., & Becker, J. (2019). A Hardware Perspective on the ChaCha Ciphers: Scalable Chacha8/12/20 Implementations Ranging from 476 Slices to Bitrates of 175 Gbit/s. *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, *null*, 294–299. https://doi.org/10.1109/SOCC46988.2019.1570548289

Serrano, R., Duran, C., Hoang, T.-T., Sarmiento, M., Tsukamoto, A., Suzaki, K., & Pham, C. (2021). ChaCha20-Poly1305 Crypto Core Compatible with Transport Layer Security 1.3. *2021 18th International SoC Design Conference (ISOCC)*, *null*, 17–18. https://doi.org/10.1109/ISOCC53507.2021.9614016

Serrano, R., Duran, C., Sarmiento, M., Pham, C., & Hoang, T.-T. (2022). ChaCha20-Poly1305 Authenticated Encryption with Additional Data for Transport Layer Security 1.3. *Cryptogr.*, *6*, 30. https://doi.org/10.3390/cryptography6020030

Singh, R., Kumar, A., & Shukla, A. (2019). A comprehensive comparison of AES-GCM and ChaCha20-Poly1305 in securing online applications. *Security and Privacy*, *2*(6), e88.

Wang, Y., Zhang, H., Li, J., & Chen, Y. (2021). Comparative study on the security and performance of AES-GCM and ChaCha20-Poly1305 in mobile devices. *Journal of Cryptographic Engineering*, *11*(2), 147–159.

Waruwu, G., & Sundari, J. (2024). AUDIT TEKNOLOGI INFORMASI MENGGUNAKAN COBIT 5 STUDI KASUS PT . GLOBAL NETWORK DHARMA JAYA. *INFOMATEK:Jurnal Informatika, Manajemen Dan Teknologi*, *26*, 69–74. https://doi.org/10.23969/infomatek.v26i1.13333

Widyastuti, S., Ariandi, W., & Sulistiono, V. (2019). *Implementasi Kriptografi AES Dalam Pengamanan Data Seleksi*

Peserta JAMKESMAS. https://doi.org/10.46772/INTECH.V1I02.66

Zhang, T., & Wang, X. (2023). Secure and efficient encryption algorithms for secure communication: A comparative study of AES-GCM and ChaCha20-Poly1305. *Future Generation Computer Systems*, *132*, 223–239.

Zhao, X., Zhang, B., Lin, J., & Liu, W. (2021). Evaluating the security and performance of cryptographic algorithms in IoT environments. *Security and Privacy*, *4*(2), e122.