ANALISIS PERBANDINGAN PERFORMA GALOIS COUNTER MODE (GCM) PADA AES DAN CAMELLIA UNTUK ENKRIPSI DAN DEKRIPSI DATA

Usman Hidayatulloh Sidiq[1], Akhmad Zaini[2], Danang Aditya Nugraha[3]

[1,2,3]Fakultas Sains dan Teknologi, Program Studi Teknik Informatika, Universitas PGRI Kanjuruhan Malang
[1,2,3]Jl. S. Supriadi No.48, Bandungrejosari, Kec. Sukun, Kota Malang, Jawa Timur 65148

Email : [1]usmanhdsidiq@gmail.com, [2]zaini@unikama.ac.id, [3]danang.adty@unikama.ac.id

ABSTRAK

Keamanan data digital telah menjadi kebutuhan yang kritis seiring dengan meningkatnya ancaman terhadap kerahasiaan dan intergitas informasi di era modern. *Advanced Encryption Standard* (AES) dan Camellia merupakan dua algoritma *block cipher* yang telah distandarisasi secara internasional dengan tingkat keamanan yang setara, namun memiliki struktur internal yang berbeda. *Substitution-Permutation Network* (SPN) pada AES dan *Feistel Network* pada Camellia menghasilkan karakteristik kinerja yang berbeda. Belum adanya penelitian secara komprehensif yang membandingkan keduanya dalam mode *Galois Counter Mode* (GCM) dengan parameter *multithreading* dan *hardware acceleration* menjadi permasalahan yang melatarbelakangi penelitian ini. Penelitian ini bertujuan untuk membandingkan kinerja AES dan Camellia pada mode GCM berdasarkan parameter ukran kunci, ukuran file, mode pengoperasian, *multithreading*, dan *hardware acceleration*. Implementasi dilakukan dengan bahasa pemrograman Java dan *library* BouncyCastle. Hasil penelirian menunjukkan bahwa AES lebih unggul dengan *processing rate* 30% lebih tinggi. Efisiensi *multithreading* optimal pada 2 *threads* dan menurun pada 8 *threads* akibat *overhead* pada manajemen *thread*. *Hardware acceleration* memberikan dampak yang bervariasi, seperti mempercepat waktu enkripsi dan memperlambat waktu dekripsi pada AES. Secara keseluruhan, AES menunjukkan performa yang lebih unggul, sementara Camellia tetap kompetitif dalam kondisi tertentu.

Keywords: *AES, Camellia, GCM, enkripsi, kriptografi.*

ABSTRACT

Digital data security has become a critical necessity in tandem with the increasing threats to information confidentiality and integrity in the modern era. The Advanced Encryption Standard (AES) and Camellia are to internationally standardized block cipher algorithms with equivalent security levels but distinct internal structures. The Substitution-Permutation Network (SPN) in AES and the Feistel Network in Camellia result in different performance characteristics. The lack of comprehensive research comparing both algorithms in Galois Counter Mode (GCM) using multithreading and hardware acceleration parameters serves as the background for this study. This research aims to compare the performance of AES and Camellia in GCM based on parameters including key size, file size, mode of operation, multithreading, and hardware acceleration. The implementation was conducted using the Java programming language and the BouncyCastle library. The result indicate that AES is superior, exhibiting a 30% higher processing rate. Multithreading efficiency is optimal at 2 threads but decreases at 8 threads due to thread management overhead. Hardware acceleration yields varied impacts, such as accelerating encryption time and slowing down the decryption time for AES. Overall, AES demonstrates superior performance, whereas Camellia remains competitive under specific conditions.

Keywords: *AES, Camellia, GCM, encryption, cryptography.*

1. PENDAHULUAN

Perkembangan teknologi informasi yang sangat cepat telah mengubah cara manusia dalam mengakses, mengelola, dan menyebarkan data digital. Namun, seiring dengan meningkatnya pertukaran data, muncul berbagai ancaman kemanan seperti penyadapan, modifikasi data, hingga pencurian identitas

digital. Oleh karena itu, diperlukan metode pengamanan data untuk menjaga kerahasiaan, integritas, dan keaslian data, di mana kriptografi memilki peran yang penting untuk mengamankannya.

Beberapa algoritma kriptografi yang umum digunakan yaitu AES (*Advanced Encryption Standard*) dan Camellia. Kedua algoritma tersebut telah distandarisasi secara internasional serta diakui memiliki tingkat keamanan dan efisiensi yang tinggi [1]. Algoritma AES menggunakan struktur *Substitution Permutation Network*, sednagkan Camellia menggunakan *Feistel Network*. Perbedaan ini menjadikan keduanya menarik dibandingkan dari sisi kinerja, terutama ketika diimplementasikan pada mode operasi yang sama.

Dalam algoritma *block cipher*, mode operasi sangat dibuthuhkan untuk mengatur bagaimana setiap blok data diproses. Beberapa mode operasi yang banyak digunakan diantaranya *Electronic Codebook* (ECB), *Counter Mode* (CTR), dan *Galois Counter Mode* (GCM). Dalam penelitian ini, mode GCM digunakan karena menggabungkan keunggulan dari mode CTR dalam hal kecepatan serta pemrosesan secara paralel dengan kemampuan autentikasi data melalui operasi *Galois Field Multiplication* [2].

Penelitian sebelumnya oleh Khan dkk. Menunjukkan bahwa penggunaan AES-GCM dapat meningkatkan keamanan komputasi pada perangkat IoT dengan waktu komputasi dan *processing rate* yang efisien [3]. Selanjutnya, Panahi dkk. Membandingkan beberapa algoritma *block cipher* termasuk AES dan Camellia pada perangkat IoT berbasis Zigbee. Penelitian tersebut berfokus pada aspek konsumsi energi, *processing rate*, dan efisiensi memori [4]. Sementara itu, Dinev dan Maleshkov melakukan analisis perbandingan terhadap AES dan Camellia dengan berfokus pada kecepatan enkripsi dan dekripsi, serta ketahanan terhadap *brute-force* dan *dictionary attack*. Penelitian tersebut menunjukkan bahwa keduanya memiliki performa yang tinggi dalam berbagai skenario [5].

Berbeda dengan ketiga penelitian sebelumnya yang berfokus pada keamanan dan performa secara umum dengan berbagai mode operasi, penelitian ini secara spesifik membahas kinerja algoritma AES dan Camellia dengan mode operasi GCM. Mode GCM dipilih karena memiliki karakteristik autentikasi serta kemampuan dalam pemrosesan secara paralel yang belum banyak diteliti dalam konteks perbandingan kedua algoritma tersebut. Oleh karena itu, penelitian ini bertujuan untuk mengisi *research gap* mengenai analisis performa AES dan Camellia dalam mode operasi GCM, terutama dari parameter ukuran kunci, ukuran file, mode pengoperasian (enkripsi/dekripsi), penggunaan *multithreading*, serta penggunaan *hardware acceleration.*

2. TINJAUAN PUSTAKA

2.1. Advanced Encryption Standard (AES)

Algoritma AES merupakan algoritma kriptografi simetris yang termasuk ke dalam kategori block cipher yang dikembangkan untuk mengatasi kelemahan algoritma DES (*Data Encryption Standard*) [6]. Algoritma ini menggunakan struktur *Substitution Permutation Network* (SPN). AES dikenal memiliki efisiensi dan kecepatan pemrosesan yang tinggi. Salah satu keunggulan AES yaitu terdapat teknologi akselerasi perangkat keras seperti AES-NI (*AES New Instructions*) yang tersedia pada prosesor Intel dan AMD. Teknologi ini secara signifikan dapat mempercepat proses enkripsi dan dekripsi tanpa mengurangi tingkat keamanan. Oleh karena itu, AES banyak digunakan pada TLS (*Transport Layer Security*), VPN (*Virtual Private Network*), serta sistem penyimpanan terenkripsi.

2.2. Struktur Internal AES

AES beroperasi dengan panjang blok tetap 128-bit, yang menunjukkan bahwa data diproses dalam blok berukuran 128-bit. Selain itu, AES mendukung tiga variasi panjang kunci, yaitu 128-bit, 192-bit, dan 256-bit [7]. AES menggunakan struktur *Substitution Permutation Network* (SPN) yang merupakan kombinasi dari operasi substitusi dan permutasi yang dilakukan secara berulang (*round*) untuk mencapai tingkat keamanan yang tinggi [8].

Setiap *round* pada proses enkripsi AES, terdiri dari empat tahapan, yaitu:

1. *SubBytes*: Setiap *byte* pada blok data digantikan menggunakan tabel substitusi (S-box) untuk menciptakan kebingungan data (*confusion*).
2. *ShiftRows*: Setiap baris dalam *state matrix* digeser sebanyak jumlah tertentu untuk meningkatkan (*diffusion*).

3. *MixCollumns*: Setiap kolom data dikombinasikan menggunakan operasi matriks hingga 8 ($GF(2^8)$) untuk menyebarkan pengaruh pada setiap *byte* ke seluruh blok.
4. *AddRoundKey*: Setiap *byte* dalam blok data dikombinasikan dengan *round key* melalui operasi XOR.

Pada putaran terakhir, tahap *MixCollumns* tidak digunakan untuk menjaga struktur hasil supaya efisien dan mudah dikembalikan saat proses dekripsi. Melalui serangkaian proses tersebut, algoritma AES dapat menghasilkan *ciphertext* yang cukup sulit diprediksi meskipun hanya terjadi perubahan yang sangat kecil pada *plaintext*.

2.3. Algoritma Camellia

Selain AES, Camellia juga merupakan algoritma kriptografi simetris yang memberikan tingkat keamanan yang setara dengan AES [9]. Namun, Camellia tetap memiliki perbedaan, terutama pada struktur *Feistel Network*. Perbedaan struktur ini dinilai menjadikan Camellia menjadi lebih fleksibel dalam proses dekripsi. Selain itu, Camellia dirancang untuk mencapai efisiensi yang tinggi, baik pada perangkat lunak maupun perangkat keras dengan sumber daya terbatas seperti IoT.

2.4. Feistel Network Pada Camellia

Camellia menggunakan struktur *Feistel Network* dengan 18-*rounds* untuk kunci 128-bit, serta 24 rounds untuk kunci 192-bit dan 256-bit [10]. Feistel Network bekerja dengan membagi blok data menjadi dua bagian, yaitu kiri (L) dan kanan (R) dengan ukuran yang sama. Proses transformasi dilakukan secara berulang melalui sejumlah putaran (*rounds*) menggunakan fungsi non-linear F dan subknuci yang berbeda pada setiap putaran [10].

Secara matematis, proses enkripsi pada struktur *Feistel* adalah sebagai berikut:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1,} K_i) \quad (1)$$

Dengan keterangan:

- Li dan Ri adalah bagian kiri dan kanan dari blok data setelah putaran ke-i,
- Li – 1 dan Ri – 1 merupakan hasil dari putaran sebelumnya,
- Ki adalah subkunci yang digunakan pada putaran ke-i,
- F merupakan fungsi non-linear yang mencakup operasi substitusi, permutasi, serta manipulasi bit menggunakan S-box dan P-box.

Simbol $\oplus$ menunjukkan operasi XOR (*Exclusive Or*).

Pada proses dekripsi, struktur *Feistel* memungkinkan algoritma untuk melakukan proses secara terbalik. Karakteristik ini membuat *Feistel Network* efisien untuk diimplementasikan, karena algoritma enkripsi dan dekripsi dapat menggunakan fungsi dan komponen yang sama tanpa adanya perubahan struktur yang signifikan [10].

2.5. Mode Operasi Kriptografi

Kedua algoritma tersebut membutuhkan mode operasi supaya dapat mengenkripsi data berukuran besar. Mode operasi digunakan untuk mengatur bagaimana setiap blok data diproses. Tanpa mode operasi, maka proses enkripsi akan bersifat determinstik. Yang artinya, *plaintext* akan menghasilkan *ciphertext* yang hampir sama, sehingga menyebabkan pola data asli masih dapat terlihat [8].

2.6. Mode Operasi Galois Counter Mode (GCM)

Galois Counter Mode (GCM) merupakan salah satu mode operasi block cipher yang banyak digunakan, karena kemampuannya dalam melakukan proses enkripsi dan autentikasi secara bersamaan. Hal ini membuat mode GCM menjadi lebih efisien, karena mode GCM tidak hanya mengurangi beban komputasi, namun juga meminimalkan risiko kesalahan implememtasi [11].

2.7. Parameter dalam GCM

1. Nonce (*Number used once*): Merupakan nilai unik sepanjang 96-bit yang digunakan bersama dengan kunci enkripsi. Nonce (*Number used once*) tidak bersifat rahasia, tetapi harus dipastikan bahwa

nonce tidak pernah digunakan lagi untuk kunci yang sama. Penggunaan *nonce* yang sama akan menyebabkan terjadinya kebocoran informasi.

2. *Counter*: Merupakan nilai yang digunakan untuk membedakan setiap blok enkripsi. *Counter* dikombinasikan dengan *nonce* untuk menghasilkan input yang unik bagi setiap blok pada proses enkripsi.
3. *Authentication Tag*: Dihasilkan oleh fungsi GHASH dengan ukuran tetap 128-bit, berfungsi sebagai tanda autentikasi untuk memverifikasi integritas *ciphertext* dan *associated data*.
4. *Associated Data* (AD): Informasi tambahan yang tidak dienkripsi, tetapi diverifikasi integritasnya. AD sering digunakan untuk header paket atau metadata yang perlu dilindungi dari modifikasi.

2.8. Alur Enkripsi dan Autentikasi GCM

Berikut merupakan proses enkripsi dan autentikasi pada GCM:

1. Nilai *nonce* digabungkan dengan *counter* awal untuk membentuk input enkripsi pertama.
2. Selanjutnya, input dienkripsi dengan *block cipher* untuk menghasilkan *keystream block*.
3. Kemudian *plaintext* di-XOR dengan *keystream block* untuk menghasilkan *ciphertext*.
4. Secara paralel, fungsi GHASH memproses *associated data* dan *ciphertext* untuk menghasilkan *authentication tag*.
5. *Authentication tag* dikombinasikan dengan hasil enkripsi dari *nonce* awal untuk menghasilkan tag akhir berukuran 128-bit.
6. Proses dekripsi dilakukan dengan menghitung ulang GHASH dan memverifikasikan kesesuaian tag untuk memastikan keaslian data.

2.9. Metrik Pengukuran Kinerja.

Metrik pengukuran yang digunakan dalam penelitian ini antara lain:

a. Rata-rata waktu pemrosesan

$$T_{avg} = \frac{T_1 \ + \ T_2 + \ldots T_i}{n} \tag{1}$$

b. Processing rate

$$Processing\ Rate\ \left(\frac{MB}{s}\right) = \frac{Ukuran\ File\ (MB)}{Waktu\ Pemrosesan\ (s)} \tag{2}$$

c. Speedup

$$Speedup\ = \frac{T_1}{T_n} \tag{3}$$

d. Efficiency

$$Efficiency\ (\%) = \frac{Speedup}{thread_n} \times 100 \tag{4}$$

e. HwSpeedup

$$HwSpeedup\ (ms) = \frac{T_{off}}{T_{on}} \tag{5}$$

f. Performance ratio

$$Performance\ Ratio = \frac{PR_{Camellia}}{PR_{AES}} \tag{6}$$

2.10. Penelitian Terdahulu

Beberapa penelitian telah dilakukan terkait implementasi dan analisis kinerja algoritma kriptografi simetris, terutama AES dan algoritma sejenis. Panahi & Bayılmış[4] menggunakan enkripsi AES-GCM untuk mengamankan data pada IoT dengan berbasis Wireless Sensor Networks (WSL) dan menyimpulkan bahwa AES-GCM memberikan tingkat keamanan yang tinggi dan efisiensi komputasi yang baik. Semantara itu, Khan dkk. mengimplementasikan protokol AES-GCM pada sistem komunikasi IoT dan membuktikan bahwa AES-GCM mampu meningkatkan keamanan sekaligus menurunkan *latency* daripada protokol konvensional. Kedua penelitian tersebut berfokus pada konteks IoT, sedangkan pada penelitian ini bertujuan untuk membandingkan performa AES dan Camellia tanpa batasan perangkat tertentu.

Penelitian oleh Anindita[12] membandingkan AES-CBC dengan ChaCha20-Poly1305 pada konteks enkripsi *livestreaming* dan menyimpulkan bahwa ChaCha20 lebih cepat, namun AES lebih aman untuk sistem komunikasi *real-time*. Arifandi melakukan studi literatur mengenai implementasi GCM untuk keamanan data *cloud* dan menyimpulkan bahwa GCM lebih unggul dalam hal kecepatan dan autentikasi, namun membutuhkan perangkat keras yang mendukung. Perbedaannya, penelitian yang dilakukan oleh Arifandi bersifat konseptual tanpa pengujian terhadap algoritma yang spesifik, sedangkan penelitian ini melakukan pengujian empiris terhadap AES dan Camellia.

Muhammed dkk.[13] membandingkan AES, Twofish, Salsa20, dan ChaCha20 untuk enkripsi citra menggunakan Java dan menyimpulkan bahwa AES dan ChaCha20 memiliki *throughput* tertinggi. Selanjutnya, penelitian oleh Algherini[14] mengimplementasikan AES, Camellia, dan RC6 untuk enkripsi pesan SMS di Android dan menyimpulkan bahwa Camellia memiliki kecepatan enkripsi yang tinggi, sedangkan RC6 lebih cepat untuk dekripsi. Sementara itu, Dinev & Maleshkov[5] melakukan analisis performa pada berbagai algoritma simetris termasuk AES dan Camellia menggunakan *educational benchmarking tool* dan menyimpulkan bahwa keduanya menunjukkan performa yang kompetitif dengan efisiensi CPU yang baik.

Berdasarkan penelitian-penelitian di atas, belum terdapat penelitian yang secara khusus membandingkan AES dan Camellia pada mode GCM dengan parameter pengujian yang mencakup ukuran kunci, ukuran file, *multithreading*, dan *hardware acceleration* secara bersamaan. Oleh karena itu, penelitian ini hadir untuk mengisi celah tersebut.

3. METODE PENELITIAN

3.1. Jenis Penelitian

Penelitian ini menggunakan pendekatan kuantitatif eksperimental, karena penelitian ini berfokus pada pengukuran dan analisis hasil berupa data numerik, seperti kecepatan pemrosesan, penggunaan *multithreading*, serta pengaruh dari *hardware acceleration*.

3.2. Alat Penelitian

Alat penelitian yang digunakan dalam penelitian ini berupa sebuah laptop dengan spesifikasi berikut, diantaranya:

a. Prosesor: Intel Core i5-1155G7 @ 2.50Ghz
b. RAM: 16 GB DDR4
c. Penyimpanan: SSD NVMe 500 GB
d. Sistem Operasi: Windows 11
e. Bahasa Pemrograman: Java dengan *library* BouncyCastle

3.3. Bahan Penelitian

Bahan yang digunakan dalam penelitian ini meliputi:

a. Algoritma kriptografi: AES dan Camellia dengan mode operasi GCM.
b. Dataset uji: File lokal dengan ukuran 1 MB, 10 MB, 100 MB, 500 MB, dan 1 GB.
c. Parameter Pengujian:
   - Ukuran kunci: 128-bit dan 256-bit
   - Mode pengoperasian: Enkripsi dan Dekripsi
   - *Multithreading*: 1, 2, 4, dan 8 threads

- *Hardware acceleration*: on dan off

3.4. Teknik Pengumpulan Data

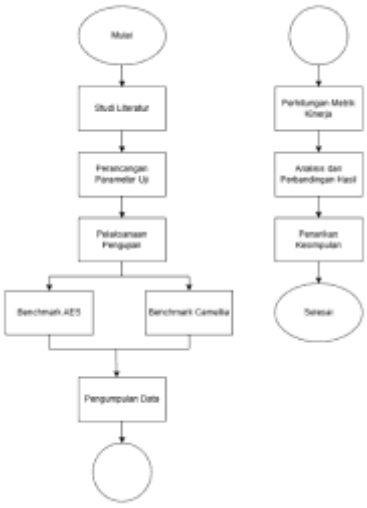
Teknik pengumpulan data pada penelitian ini dilakukan dengan uji eksperimental yang dikembangkan secara langsung menggunakan bahasa pemrograman Java dengan library BouncyCastle karena telah mendukung berbagai algoritma kriptografi dan mode operasi, termasuk AES dan Camellia dengan mode GCM.

Langkah-langkah pengumpulan data sebagai berikut:

1. Menjalankan program AESBench.java dan CamelliaBench.java.
2. Memilih *file* dengan ukuran seperti pada 3.3. Bahan Penelitian.
3. Melakukan pengujian dengan parameter seperti pada 3.3. Bahan Penelitian.
4. Hasil pengukuran dicatat secara otomatis ke dalam sebuah file CSV.
5. Mengolah data hasil pengujian untuk memperoleh nilai rata-rata, serta analisis perbandingan performa pada setiap konfigurasi.

3.5. Kerangka Penelitian

Kerangka penelitian menjelaskan proses yang dilalui dalam peenlitian, mulai dari pengumpulan data hingga evaluasi hasil. Kerangka penelitian dirancang untuk memberikan pemahaman sistematis mengenai langkah-langkah yang dijalanakan selama penelitian.



Gambar 1. Kerangka Penelitian

Diagram alur penelitian dijelaskan sebagai berikut:

1. Mulai: Menandai dimulainya penelitian.
2. Studi Literatur: Mengkaji teori-teori yang serta penelitian terdahulu.
3. Perancangan Parameter Uji: Menentukan variasi ukuran kunci, ukuran file, jumlah *threads*, dan kondisi *hardware acceleration*.
4. Pelaksanaa Pengujian: Melakukan *benchmark* menggunakan bahasa pemrograman Java.
5. Pengumpulan Data: Mencatat hasil pengujian.
6. Perhitungan Metrik Kinerja: Menghitung nilai *processing rate*, *thread efficiency*, dan *speedup*.
7. Analisis dan Perbandingan Hasil: Membandingkan dan melakukan analisis terhadap hasil pengujian.
8. Penarikan Kesimpulan: Menentukan algoritma yang memiliki performa optimal.
9. Selesai.

4. PEMBAHASAN

4.1 Pengaruh Ukuran Kunci

Hasil pengujian menunjukkan bahwa ukuran kunci 256-bit menghasilkan waktu pemrosesan yang lebih lama dibandingkan 128-bit pada kedua algoritma. Peningkatan waktu tersebut disebabkan oleh

kompleksitas pambangkitan *round key* yang lebih besar pada kunci 256-bit, sehingga setiap putaran kriptografi memerlukan komputasi yang lebih besar. Hal ini sejalan dengan persamaan (1) yang digunakan sebagai acuan dalam metrik pengukuran.

4.2 Pengaruh Ukuran File

Tabel berikut menunjukkan rata-rata waktu enkripsi dan dekripsi pada maisng-masing ukuran *file* dalam satuan milidetik (ms), pada ukuran kunci 128-bit, 1 *thread*, serta kondisi *hardware acceleration* nonaktif.

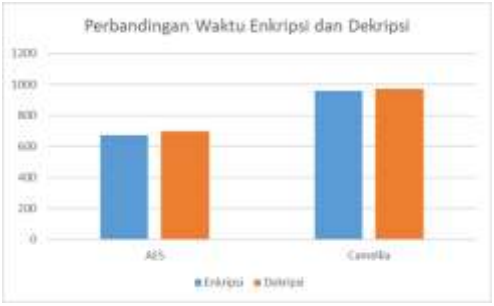
Tabel 1. Rata-rata waktu pemrosesan berdasarkan ukuran file

Ukuran File	AES		Camellia	
	Enc	Dec	Enc	Dec
1 MB	6,618	6,738	9,598	9,266
10 MB	67,640	71,026	96,086	100,768
100 MB	672,968	696,064	957,166	973,556
500 MB	3.392,144	3.467,062	4.835,666	4.911,482
1 GB	8.897,726	9.051,884	12.601,244	12.670,006

Berdasarkan tabel di atas, AES secara keseluruhan menunjukkan waktu pemrosesan yang lebih cepat dibandingkan Camellia pada seluruh ukuran *file*. Waktu pemrosesan mengalami peningkatan seiring dengan bertambahnya ukuran *file*, hal ini terjadi karena algoritma harus memproses blok data yang semakin banyak. Selain itu, mode GCM dapat menambah *overhead* autentikasi pada setiap blok, sehingga dampaknya menjadi semakin jelas pada *file* berukuran besar.

4.3 Perbandingan Waktu Enkripsi dan Dekripsi

Grafik berikut menunjukkan perbandingan waktu enkripsi dan dekripsi yang dihasilkan oleh algoritma AES dan Camellia pada ukuran *file* 100 MB serta kondisi lain yang sama.



Gambar 2. Grafik perbandingan waktu enkripsi dan dekripsi

Pada kedua algoritma, waktu dekripsi sedikit lebih lama dibandingkan waktu enkripsi, namun selisih yang dihasilkan tidak terlalu signifikan. Pada AES, waktu enkripsi sebesar 672,968 ms dan dekripsi 696,064 ms. Sedangkan pada Camellia, waktu enkripsi sebesar 957,166 ms dan dekripsi 973,556 ms. Hal ini terjadi karena pada mode GCM, proses dekripsi memerlukan proses verifikasi *authentication tag* terlebih dahulu sebelum menghasilkan *plaintext*, sehingga menambah sedikit *overhead* komputasi. Secara keseluruhan, karakteristik enkripsi dan dekripsi pada kedua algoritma tersebut cenderung seimbang.

4.4 Pengaruh Multithreading

Tabel berikut menunjukkan nilai *speedup* dan *efficiency* enkripsi berdasarkan jumlah *thread* yang digunakan.

Tabel 2. Speedup dan efficiency enkripsi berdasarkan jumlah *threads*

Jumlah Threads	AES		Camellia	
	Speedup	Efficiency (%)	Speedup	Efficiency (%)
1	1.00	100.00	1.00	100.00
2	1.91	95.5	1.90	95.0
4	3.17	79.25	3.36	84.0
8	3.57	44.62	4.33	54.12

Berdasarkan tabel di atas, penambahan jumlah *thread* terbukti dapat mempercepat waktu pemrosesan, namun hal tersebut tidak bersifat linear. Efisiensi tertinggi dicapai pada 2 *threads*, yaitu mencapai 95% pada kedua algoritma. Hal tersebut menunjukkan bahwa pemanfaatan pemrosesan paralel yang terdapat dalam mode GCM hampir berjalan dengan optimal. Sedangkan pada 8 *threads*, efisiensi menurun secara signifikan menjadi 44,62% pada AES, dan 54,42% pada Camellia, yang mengindikasikan terjadinya *bottleneck* akibat *overhead* pada manajemen *thread* serta keterbatasan pada inti prosesor.

4.5 Pengaruh *Hardware Acceleration*

Tabel berikut menampilkan nilai *HwSpeedup* yang dihitung berdasarkan perbandingan waktu pemrosesan tanpa dan dengan *hardware acceleration*.

Tabel 3. Nilai *HwSpeedup* berdasarkan Hardware Acceleration

Algoritma	HwSpeedup Enkripsi	HwSpeedup Dekripsi
AES	1.016	0.991
Camellia	1.031	1.008

Dari tabel di atas, nilai *HwSpeedup* enkripsi pada AES dan Camellia menunjukkan adanya peningkatan performa, meskipun hasilnya tidak signifikan. Nilai *HwSpeedup* dekripsi pada AES sebesar 0,991 (<1) menunjukkan bahwa penggunaan *hardware acceleration* sedikit memperlambat proses dekripsi, hal ini bisa terjadi akibat adanya *overhead* pada instruksi AES-NI pada pustaka BouncyCastle. Di sisi lain, Camellia menunjukkan peningkatan yang lebih konsisten pada kedua jenis operasi, karena tidak bergantung pada instruksi perangkat keras khusus. Temuan ini mengonfirmasi bahwa dampak dari *hardware acceleration* dapat bervariasi tergantung pada algoritma dan jenis operasi yang dilakukan [1].

4.6 Perbandingan Keseluruhan Kinerja

Tabel berikut menyajikan nilai *Performance Ratio* sebagai metrik perbandingan keseluruhan kinerja antara algoritma AES dan Camellia.

Tabel 4. Performance Ratio (PR) AES dan Camellia

Metrik	AES (MB/s)	Camellia (MB/s)	Performance Ratio
PR Enkripsi	148.342	104.298	0.703
PR Dekripsi	143.442	102.542	0.714

Berdasarkan tabel di atas, AES memiliki nilai *Performance Ratio* enkripsi sebesar 0,703 (<1), hal ini menunjukkan bahwa AES memiliki nilai *Processing Rate* 30% lebih tinggi dibandingkan Camellia pada kondisi *baseline* pengujian. Perbedaan tersebut dipengaruhi oleh tingkat optimasi pada AES yang lebih matang pada perangkat lunak modern serta dukungan *hardware acceleration* yang lebih baik. Temuan ini sejalan dengan hasil penelitian oleh Dinev & Maleshkov [5] yang menyebutkan bahwa AES dan Camellia memiliki tingkat keamanan yang kompetitif, namun AES cenderung lebih unggul dalam implementasi perangkat lunak. Di sisi lain, Camellia tetap relevan pada lingkungan yang tidak terdapat dukungan AES-NI, terutama pada perangkat berdaya rendah seperti IoT [4].

5. KESIMPULAN

Penelitian ini membandingkan kinerja AES dan Camellia pada mode GCM dengan lima parameter utama. Hasil penelitian menunjukkan bahwa AES memiliki performa yang lebih unggul dengan nilai

processing rate sekitar 30% lebih tinggi pada sebagian besar skenario pengujian. *Multithreading* secara efektif dapat meningkatkan kecepatan pemrosesan pada kedua algoritma dengan efisiensi yang optimal pada 2 *threads* (95%) lalu mengalami penurunan pada 8 *therads* (±54%) akibat *overhead* sistem. *Hardware acceleration* memberikan dampak yang bervariasi, seperti mempercepat waktu enkripsi dan sedikit memperlambat waktu dekripsi pada AES. Di sisi lain, Camellia tetap kompetitif, terutama pada lingkungan tanpa dukungan perangkat keras secara khusus. Penelitian selanjutnya dapat membandingkan dengan mode operasi lain seperti ECB, CBC, dan OCB, menambah parameter tipe *file*, atau melakukan analisis keamanan secara mendalam pada masing-masing algoritma.

DAFTAR PUSTAKA

[1] O. Kuznetsov, Y. Kuznetsova, E. Frontoni, M. Arnesano, and O. Smirnov, "Performance analysis of symmetric encryption algorithms for time-critical cybersecurity application ⋆," 2024. Accessed: Nov. 24, 2025. [Online]. Available: CEUR-WS.org/vol-3826/paper8.pdf

[2] A. Susanti, B. A. Prasetiya, O. D. Pangesti, L. D. Suryawati, and I. A. Saputro, "Perbandingan Kinerja dan Keamanan Algoritma Kriptografi Modern AES-GCM dengan CHACHA20-POLY1305," *Infomatek*, vol. 26, no. 2, pp. 253–264, Dec. 2024, doi: 10.23969/infomatek.v26i2.19255.

[3] A. B. F. Khan, S. Kalpana Devi, and K. R. Devi, "An enhanced AES-GCM based security protocol for securing the IoT communication," *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 23, no. 4, pp. 711–719, Jul. 2023, doi: 10.17586/2226-1494-2023-23-4-711-719.

[4] U. Panahi and C. Bayılmış, "Enabling secure data transmission for wireless sensor networks based IoT applications," *Ain Shams Engineering Journal*, vol. 14, no. 2, Mar. 2023, doi: 10.1016/j.asej.2022.101866.

[5] D. Dinev and V. Maleshkov, "Comparative Analysis of Symmetric Cryptographic Algorithms and Cryptanalysis Methods Through a Practical Software Tool for Educational Purposes †," *Engineering Proceedings*, vol. 104, no. 1, 2025, doi: 10.3390/engproc2025104066.

[6] P. Manikandaprabhu and M. Samreetha, "A Review of Encryption and Decryption of Text Using the AES Algorithm," 2024. Accessed: Nov. 24, 2024. [Online]. Available: https://www.researchgate.net/publication/379871849

[7] N. Febitri and H. Witriyono, "Application of AES 256 Cryptography Algorithm OCB Mode on Student Data Penerapan Algoritma Kriptografi AES 256 Mode OCB pada Data Mahasiswa," *JURNAL KOMITEK*, vol. 3, no. 2, pp. 423–432, 2023, doi: 10.53697/jkomitek.v3i2.

[8] A. khaled Alregabo, Y. Hikmat, and A. Khaled Alregabo, "BLOCK CIPHER PERFORMANCE AND RISK ANALYSIS," 2023. [Online]. Available: www.csmj.mosuljournals.com

[9] M. G. Ar Romandhon, A. Junaidi, and A. N. Sihananto, "PENERAPAN HYBRID CRYPTOGRAPHY MENGGUNAKAN CAMELLIA DAN DUAL MODULUS RSA PADA PERTUKARAN FILE," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 3S1, Oct. 2024, doi: 10.23960/jitet.v12i3s1.5218.

[10] C. Wei Ci, S. Z. M. Naziri, R. C. Ismail, R. Hussin, M. N. M. Isa, and M. S. S. M. Basir, "Crypto-Core Design using Camellia Cipher," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Mar. 2021. doi: 10.1088/1742-6596/1755/1/012019.

[11] A. M. Sitorus, "Analisis Dampak Reuse Nonce pada Keamanan AES-GCM dalam Aplikasi Web," 2025.

[12] M. Anindita and A. A. 18219086, "Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming," 2022.

[13] M. Patria and D. A. Andriati, "Analisis Komparatif Performa AES-GCM dan ChaCha20-Poly1305 dalam Enkripsi Dokumen PDF Berbasis AEAD," *Arcitech: Journal of Computer Science and Artificial Intelligence*, vol. 5, no. 1, pp. 49–69, Jun. 2025, doi: 10.29240/arcitech.v5i1.13645.

[14] M. Algherini, "Securing Mobile Messaging on Android," 2025. Accessed: Nov. 25, 2025. [Online]. Available: https://easrjournals.com/index.php/AJAPST/index