Volume 5 No. 1 (January - June 2025), pp. 49-69 | ISSN 2962-2360 (online) | ISSN 2962-3669 (print)
Arcitech: Journal of Computer Science and Artificial Intelligence
https://journal.iaincurup.ac.id/index.php/arcitech/article/view/13645 | DOI : https://dx.doi.org/10.29240/arcitech.v5i1.13645

Analisis Komparatif Performa AES-GCM dan ChaCha20-Poly1305 dalam Enkripsi Dokumen PDF Berbasis AEAD

Muhammad Patria[1], Dea Andini Andriati[2]
[1,2]Program Studi Teknik Informatika, Universitas Dian Nusantara, Jakarta, Indonesia
Email : [1]muhammad.patria@undira.ac.id, [2]dea.andini.andriati@dosen.undira.ac.id

Article Information

Article history

Received 25 April 2025
Revised 24 May 2025
Accepted 21 June 2025
Available 30 June 2025

Keywords

AEAD
AES-GCM
ChaCha20-Poly1305
PDF Encryption
Algorithm Performance

Corresponding Author:

Muhammad Patria,
Universitas Dian Nusantara,
Email :
muhammad.patria@undira.ac.id

Abstract

Digital transformation in electronic document services demands encryption mechanisms that are not only cryptographically secure but also performance-efficient. While AES-GCM and ChaCha20-Poly1305 are widely adopted AEAD algorithms, prior research has largely focused on their use in communication protocols or IoT devices, rather than in encrypting PDF documents. This study addresses that gap by empirically comparing both algorithms in real-world digital document processing scenarios. A quantitative experimental method was applied across two scenarios: mass processing of 5,000 small-to-medium PDF files (100KB–8MB), and individual processing of large files (1MB–200MB). Five performance metrics were analyzed: encryption time, decryption time, total processing time, stability, and throughput. Results show that AES-GCM consistently outperformed ChaCha20-Poly1305 across all metrics, offering faster processing and greater stability. Both algorithms produced a constant file size overhead of 28 bytes, which was negligible in terms of storage efficiency. This study contributes to the literature by providing empirical evidence to guide the selection of encryption algorithms in high-performance digital document storage systems.

Keywords : *AEAD, AES-GCM, ChaCha20-Poly1305, PDF Encryption, Algorithm Performance*

Abstrak

Transformasi digital dalam layanan dokumen elektronik menuntut mekanisme enkripsi yang tidak hanya aman secara kriptografis, tetapi juga efisien secara performa. Dua algoritma AEAD yang umum digunakan, yaitu AES-GCM dan ChaCha20-Poly1305, telah banyak dibandingkan dalam studi sebelumnya, namun sebagian besar fokus pada protokol komunikasi atau perangkat IoT, bukan pada konteks enkripsi dokumen PDF. Penelitian ini mengisi kesenjangan tersebut dengan membandingkan performa kedua algoritma dalam skenario nyata pengelolaan dokumen digital. Metode eksperimen kuantitatif dilakukan dalam dua skenario: pemrosesan massal terhadap 5.000 file PDF kecil-menengah (100KB–8MB), dan pemrosesan individual terhadap file besar (1MB–200MB). Lima metrik utama dianalisis: waktu enkripsi, waktu dekripsi, total waktu proses, stabilitas, dan *throughput*. Hasil menunjukkan bahwa AES-GCM unggul pada seluruh metrik, dengan waktu proses lebih cepat dan stabilitas lebih tinggi. *Overhead* ukuran file konstan sebesar 28 *byte* pada kedua algoritma dan tidak berdampak signifikan. Temuan ini berkontribusi pada literatur dengan memberikan bukti empiris untuk pemilihan algoritma enkripsi pada sistem penyimpanan dokumen digital berskala besar.

Kata Kunci : *AEAD, AES-GCM, ChaCha20-Poly1305, Enkripsi PDF, Performa Algoritma*

Copyright@2025 Muhammad Patria, Dea Andini Andriati
This is an open access article *under the CC-BY-NC-SA license.*

1. Pendahuluan

Perkembangan teknologi informasi dan komunikasi yang pesat telah mendorong berbagai sektor untuk mengadopsi transformasi digital, termasuk dalam pengelolaan dan pertukaran dokumen digital. Dokumen elektronik (dokumen digital) kini menjadi standar baru dalam aktivitas administrasi, kontraktual, dan legal, menggantikan dokumen fisik konvensional. Format PDF (*Portable Document Format*) menjadi pilihan utama karena kemampuannya dalam mempertahankan tampilan visual lintas perangkat dan platform (Handoyo & Subakti, 2020). Selain itu, adanya fitur keamanan seperti tanda tangan digital dan sertifikat elektronik semakin menegaskan perannya sebagai format dokumen yang tidak hanya menjaga integritas data tetapi juga memenuhi persyaratan kepatuhan dan legalitas yang berlaku (Sudarsono & Halim, 2019). Namun, adopsi besar-besaran dokumen digital juga menimbulkan tantangan serius dalam aspek keamanan dan keabsahan dokumen, khususnya saat data disimpan dan ditransmisikan dalam sistem terbuka.

Salah satu aspek penting dalam perlindungan informasi digital adalah keamanan *data at rest*, yakni data yang disimpan secara permanen di media penyimpanan server atau *cloud* (Ogunjinmi, 2024). Data semacam ini rentan terhadap akses tidak sah, manipulasi, maupun kebocoran, khususnya jika tidak dilindungi dengan teknologi enkripsi yang kuat (Abdulghani et al., 2019). Beberapa insiden kebocoran data menunjukkan bahwa banyak organisasi belum sepenuhnya menerapkan skema perlindungan yang memadai pada dokumen yang disimpan (Sugiarto et al., 2024). Sistem penyimpanan seperti *Network File System* (NFS) yang tidak dibarengi dengan enkripsi sering kali menjadi titik lemah yang dapat dieksploitasi oleh penyerang untuk memperoleh akses terhadap dokumen sensitif.

Tantangan ini semakin kompleks dalam sistem yang menangani volume transaksi tinggi dan variasi ukuran dokumen. Dalam konteks layanan digital publik, sistem informasi dapat menangani ribuan dokumen dalam satu hari, dengan ukuran file yang bervariasi dari ratusan *kilobyte* hingga ratusan *megabyte* (Arifin et al., 2025; Syahrani & Yuliadi, 2023). Dalam kondisi ini, efisiensi algoritma enkripsi menjadi sangat penting. Algoritma yang tidak optimal dapat menyebabkan *bottleneck*, membebani sumber daya server, dan memperlambat layanan secara keseluruhan. Oleh karena itu, dibutuhkan algoritma yang tidak hanya aman secara kriptografis, tetapi juga efisien dalam kecepatan proses dan konsumsi sumber daya, terutama ketika diterapkan pada dokumen digital berskala besar dan kompleks.

Dalam praktiknya, pengelolaan dokumen digital oleh penyelenggara sistem elektronik (PSE), seperti platform *wholesaler* e-Meterai, membutuhkan perhatian serius terhadap aspek keamanan data dan efisiensi sistem. *Wholesaler* e-Meterai berperan sebagai perantara antara distributor resmi dan pelanggan dalam proses pembubuhan meterai elektronik, di mana dokumen PDF pelanggan disimpan dan diproses baik sebelum maupun sesudah pembubuhan (Vivian, 2023). Produk e-Meterai sendiri

menggunakan teknologi tanda tangan elektronik sebagai bagian dari sistem keamanannya, yang berfungsi untuk menjamin keaslian, integritas, serta mencegah penyangkalan (*non-repudiation*) terhadap dokumen yang dibubuhi (Nurhaida et al., 2017; Sirait et al., 2024). Namun demikian, hingga saat ini belum terdapat standarisasi teknis nasional yang secara eksplisit mengatur arsitektur sistem dan kebijakan penyimpanan dokumen digital yang digunakan oleh para *wholesaler*. Akibatnya, terdapat variasi dalam tata kelola penyimpanan dokumen, seperti lokasi penyimpanan, sistem file yang digunakan, serta prosedur akses internal, yang berpotensi memengaruhi tingkat risiko terhadap kebocoran data. Oleh karena itu, evaluasi terhadap performa enkripsi dokumen dalam skenario seperti trafik tinggi dan ukuran file yang bervariasi, menjadi penting untuk mendukung penguatan tata kelola keamanan data di sektor ini.

Dalam konteks enkripsi data digital, dua algoritma AEAD (*Authenticated Encryption with Associated Data*) yang semakin populer adalah AES-GCM (*Advanced Encryption Standard - Galois/Counter Mode*) dan ChaCha20-Poly1305 (Muhammed et al., 2024). Keduanya menggabungkan mekanisme kerahasiaan, integritas, dan autentikasi dalam satu langkah, sehingga menyederhanakan implementasi dan mengurangi risiko kesalahan konfigurasi. AES-GCM dikenal karena performanya yang optimal pada sistem yang mendukung akselerasi perangkat keras (seperti AES-NI), serta implementasi luas pada protokol keamanan seperti TLS dan IPsec (Bhatt, 2024). Sementara itu, ChaCha20-Poly1305 dirancang agar efisien pada perangkat dengan keterbatasan sumber daya, seperti sistem *embedded* dan perangkat *mobile*, karena memanfaatkan operasi aritmatika sederhana yang dapat dieksekusi dengan cepat pada arsitektur CPU yang beragam (Saraiva et al., 2019).

Meskipun berbagai studi telah membahas performa AES-GCM dan ChaCha20-Poly1305, sebagian besar penelitian sebelumnya berfokus pada aplikasi dalam protokol jaringan atau sistem komunikasi, seperti TLS dan VPN. Studi yang secara khusus mengevaluasi performa keduanya dalam konteks enkripsi dokumen digital, terutama format PDF, masih sangat terbatas. Padahal, pengelolaan dokumen PDF memiliki karakteristik unik, baik dari sisi struktur data maupun ukuran file yang bervariasi. Kesenjangan ini menjadi dasar perlunya kajian empiris yang menilai secara langsung bagaimana masing-masing algoritma berperforma dalam skenario pengolahan dokumen elektronik. Berdasarkan latar belakang tersebut, rumusan masalah dalam penelitian ini adalah: algoritma enkripsi AEAD manakah yang lebih efisien dan stabil ketika diterapkan pada sistem layanan dokumen digital, ditinjau dari aspek waktu proses dan efisiensi ukuran file dalam berbagai kondisi operasional?

Penelitian ini bertujuan untuk melakukan analisis komparatif terhadap performa algoritma AES-GCM dan ChaCha20-Poly1305 dalam konteks enkripsi dokumen PDF. Fokus pengujian mencakup dua aspek utama, yaitu kecepatan proses enkripsi dan efisiensi ukuran hasil enkripsi, dalam dua skenario yang merepresentasikan kondisi

operasional umum: pengelolaan dokumen dalam jumlah besar, serta pengelolaan dokumen berukuran besar. Pendekatan ini dirancang untuk mencerminkan tantangan teknis yang dihadapi dalam sistem penyimpanan dan layanan dokumen digital. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi praktis berupa rekomendasi teknis algoritma enkripsi yang paling sesuai dengan kebutuhan pengelolaan dokumen digital berskala besar, sekaligus menjadi masukan dalam pengembangan standar teknis keamanan informasi untuk penyelenggara sistem elektronik.

2. Kajian Terdahulu

Algoritma AEAD (*Authenticated Encryption with Associated Data*) merupakan skema kriptografi modern yang menggabungkan proses enkripsi dan autentikasi dalam satu langkah (Rajavelsamy et al., 2025). Tujuannya adalah untuk menjamin kerahasiaan, integritas, dan keaslian data secara simultan, sehingga cocok digunakan dalam sistem transmisi maupun penyimpanan dokumen (Fotovvat et al., 2020). Dalam konteks evaluasi performa algoritma kriptografi, beberapa metrik umum digunakan, antara lain: throughput (jumlah data yang dapat diproses per satuan waktu), waktu eksekusi enkripsi dan dekripsi (*encryption/decryption time*), dan standar deviasi proses, yang mencerminkan konsistensi performa di berbagai ukuran data (Cagua et al., 2025). Efisiensi algoritma tidak hanya dinilai dari kecepatan, tetapi juga kestabilan dan skalabilitasnya terhadap beban sistem (Chan et al., 2024). Oleh karena itu, studi empiris sangat penting untuk menilai karakteristik performa tersebut dalam lingkungan nyata.

Studi oleh Kao et al. (2021) menunjukkan bahwa ChaCha20-Poly1305 mengungguli AES-GCM pada platform tanpa dukungan akselerasi perangkat keras AES, seperti pada pengujian menggunakan papan pengembang IoT ZedBoard ARM yang tidak memiliki instruksi AES-NI. Dalam kondisi tersebut, ChaCha20-Poly1305 tercatat mengenkripsi pesan kecil (~50 byte) sekitar 3,8 kali lebih cepat daripada AES-256-GCM. Selain dari aspek kecepatan, algoritma ini juga menunjukkan efisiensi energi yang lebih tinggi, hanya membutuhkan daya sebesar ~7 μW untuk proses enkripsi, dibandingkan dengan ~27 μW pada AES-256-GCM dalam platform yang sama.

Studi oleh Degabriele et al. (2021) menegaskan bahwa baik AES-GCM maupun ChaCha20-Poly1305 memiliki kekuatan kriptografis yang setara dalam konteks *multi-user*, asalkan parameter digunakan secara benar. Tidak ada temuan kriptanalisis praktis yang membatalkan keamanannya hingga saat ini. Oleh karena itu, pemilihan algoritma lebih banyak dipengaruhi oleh faktor performa dan lingkungan operasional.

Studi oleh Serrano et al. (2022) membandingkan aspek keamanan dan kemudahan implementasi antara kedua algoritma. Mereka menemukan bahwa ChaCha20-Poly1305 lebih mudah diimplementasikan secara *constant-time*, sehingga lebih tahan terhadap serangan *side-channel* berbasis *cache*. Di sisi lain, AES-GCM harus diimplementasikan dengan hati-hati agar tidak menimbulkan kerentanan dari

penggunaan *lookup tables*. Dalam hal kompatibilitas perangkat, ChaCha20-Poly1305 unggul karena dapat dijalankan secara efisien di berbagai arsitektur tanpa fitur kriptografi khusus, sementara AES-GCM sangat optimal ketika dijalankan di perangkat dengan dukungan instruksi khusus seperti AES-NI atau ARMv8 *Cryptography Extensions*.

Studi oleh Kempf et al. (2024) menyoroti bahwa pada jaringan berkecepatan tinggi dengan protokol QUIC, AES-GCM memberikan *throughput* sekitar tiga kali lebih tinggi dibandingkan ChaCha20 ketika fitur AES-NI diaktifkan. Namun, apabila akselerasi AES dimatikan, maka ChaCha20 kembali unggul. Temuan ini menunjukkan bahwa performa relatif antar algoritma sangat tergantung pada ketersediaan dukungan instruksi kriptografi di perangkat keras. Pola yang sama juga ditemukan dalam konteks VPN dan TLS, di mana AES-GCM dominan pada CPU desktop modern, sementara ChaCha20-Poly1305 lebih efisien pada perangkat *mobile* dengan arsitektur ARM tanpa akselerasi kriptografi.

Studi oleh Susanti et al. (2024) menunjukkan bahwa secara eksperimen, AES-GCM menghasilkan waktu enkripsi dan dekripsi yang lebih cepat dibandingkan ChaCha20-Poly1305 untuk file teks dan PDF dengan berbagai ukuran. Namun, dalam studi tersebut disebutkan bahwa ChaCha20-Poly1305 dianggap lebih unggul secara efisiensi karena keandalannya dalam menghadapi serangan *side-channel* dan konsistensi performa lintas perangkat lunak. Adanya perbedaan antara data eksperimental dan simpulan dalam studi tersebut menunjukkan bahwa pemilihan algoritma enkripsi tidak dapat dilepaskan dari konteks sistem dan lingkungan operasionalnya, termasuk arsitektur sistem, beban trafik, dan ukuran dokumen yang diproses.

Studi oleh Beyne et al. (2025) mengusulkan pendekatan adaptif, yakni sistem dapat memilih algoritma enkripsi secara dinamis tergantung pada kapabilitas perangkat klien. Misalnya, server dapat menggunakan AES-GCM untuk klien desktop yang memiliki akselerasi AES, dan ChaCha20-Poly1305 untuk perangkat *mobile* lawas. Strategi ini diyakini dapat memberikan efisiensi maksimum sambil menjaga interoperabilitas sistem.

Tabel 1. Perbandingan Studi Terdahulu.

Penulis	**Fokus Studi**	**Lingkungan Uji**	**Temuan Utama**	**Kelemahan**
Kao et al. (2021)	Enkripsi pada platform IoT tanpa AES-NI	ZedBoard ARM	ChaCha20 lebih cepat dan efisien energi dibanding AES-GCM	Hanya diuji pada ukuran pesan kecil dan perangkat IoT
Degabriele et al. (2021)	Keamanan multi-user	Simulasi multi-kunci	Keduanya aman jika parameter benar	Tidak membahas performa praktis
Serrano et al. (2022)	Side-channel resistance dan portabilitas	Arsitektur CPU variatif	ChaCha20 lebih mudah diimplementasikan constant-time	Tidak menguji performa throughput langsung

Penulis	Fokus Studi	Lingkungan Uji	Temuan Utama	Kelemahan
Kempf et al. (2024)	QUIC di jaringan cepat	Jaringan dengan/ tanpa AES-NI	AES-GCM unggul dengan AES-NI, ChaCha20 unggul tanpa akselerasi	Fokus pada komunikasi, bukan enkripsi file
Susanti et al. (2024)	Enkripsi file teks dan PDF	Desktop dan mobile	AES-GCM lebih cepat, ChaCha20 lebih konsisten dan aman terhadap side-channel	Tidak spesifik uji pada layanan berskala besar
Beyne et al. (2025)	Adaptasi algoritma berbasis perangkat	Server-klien dinamis	Sistem adaptif memberi efisiensi optimal	Hanya pendekatan arsitektural, tidak uji performa langsung

Secara keseluruhan, literatur terdahulu menegaskan bahwa AES-GCM dan ChaCha20-Poly1305 merupakan algoritma AEAD yang sama-sama aman, tetapi keunggulan relatifnya sangat dipengaruhi oleh kondisi sistem dan perangkat keras. Studi komparatif khusus dalam konteks enkripsi file PDF, terutama pada skala besar dan skenario praktis seperti layanan dokumen digital, masih terbatas. Oleh karena itu, penelitian ini ditujukan untuk mengisi celah tersebut dengan mengevaluasi performa kedua algoritma secara empiris dalam skenario aktual pengelolaan dokumen elektronik.

Gambar 1. Kerangka Berpikir Penelitian.

3. Metodologi Penelitian

3.1. Tahapan Penelitian

Penelitian ini menerapkan metode kuantitatif dengan pendekatan eksperimen komparatif yang bertujuan untuk mengevaluasi performa dua algoritma enkripsi modern berbasis AEAD, yaitu AES-GCM dan ChaCha20-Poly1305. Kedua algoritma tersebut banyak digunakan dalam praktik keamanan sistem elektronik karena sifatnya yang simultan dalam menjamin kerahasiaan dan integritas data. Oleh karena itu, kajian komparatif berbasis pengujian empiris menjadi penting untuk memperoleh pemahaman

menyeluruh terhadap keunggulan relatif masing-masing algoritma dalam pengamanan dokumen PDF.

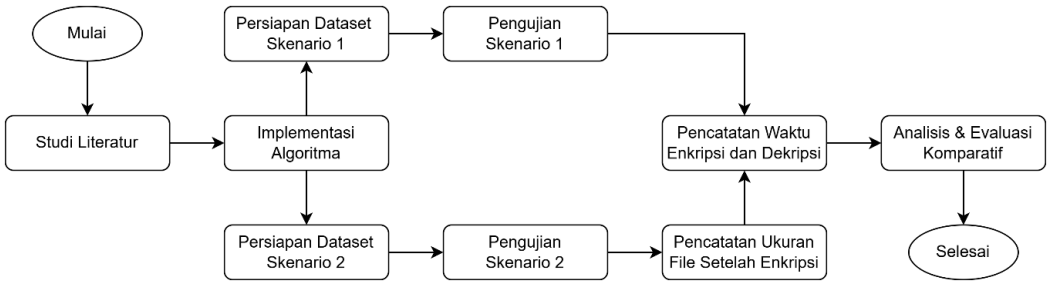
Metode eksperimen dipilih karena memberikan keleluasaan bagi peneliti untuk mengendalikan variabel-variabel yang terlibat dan melakukan pengamatan langsung terhadap efek dari perlakuan yang diberikan. Dalam penelitian ini, variabel bebas adalah jenis algoritma yang digunakan, sementara variabel terikat meliputi waktu proses enkripsi, waktu proses dekripsi, dan efisiensi ukuran file hasil enkripsi. Data yang diperoleh dari proses pengujian akan dianalisis secara deskriptif untuk mengidentifikasi perbedaan performa kedua algoritma dalam konteks pemrosesan dokumen digital.

Langkah awal dalam proses penelitian diawali dengan studi literatur, yang difokuskan pada kajian teoritis terhadap prinsip kerja algoritma AEAD, efisiensi kriptografi simetris modern, serta strategi eksperimental dalam pengujian performa sistem keamanan digital. Studi literatur ini mencakup publikasi akademik, artikel teknis, dan dokumentasi resmi *library* kriptografi Golang yang dijadikan landasan dalam merancang desain penelitian.

Tahapan selanjutnya adalah implementasi kode eksperimen menggunakan bahasa pemrograman Go (Golang), yang memiliki keunggulan dalam manajemen memori dan efisiensi proses eksekusi. Kode pengujian dibangun untuk memfasilitasi proses enkripsi dan dekripsi file PDF, serta mencatat waktu eksekusi secara presisi. Hasil pengujian dikemas dalam bentuk file log terstruktur, yang nantinya digunakan dalam proses analisis data.

Setelah kode eksperimen selesai dikembangkan, dilakukan proses pengujian dalam dua skenario. Skenario pertama dirancang untuk mengamati performa algoritma dalam konteks volume transaksi tinggi, sedangkan skenario kedua difokuskan pada pengujian terhadap file berukuran besar untuk melihat dampaknya terhadap waktu dan efisiensi penyimpanan. Kedua skenario tersebut dirancang untuk merepresentasikan kondisi nyata yang kerap dijumpai dalam sistem layanan dokumen elektronik, khususnya pada sistem yang menangani dokumen hukum, arsip digital, dan dokumen resmi lainnya.

Alur keseluruhan dari proses penelitian divisualisasikan pada Gambar 2. Diagram tersebut menggambarkan tahapan metodologis yang dimulai dari studi literatur, dilanjutkan dengan pengembangan kode eksperimen, pelaksanaan pengujian berdasarkan dua skenario, dan diakhiri dengan proses analisis serta evaluasi perbandingan performa kedua algoritma enkripsi.



Gambar 2. Tahapan Alur Penelitian.

3.2. Teknik Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan melalui proses pengujian langsung (*in-situ*) menggunakan perangkat lunak yang dikembangkan secara khusus dengan bahasa pemrograman Go (Golang). Perangkat lunak ini berfungsi sebagai instrumen utama dalam eksperimen, yang secara otomatis menjalankan fungsi enkripsi dan dekripsi terhadap file PDF yang telah ditentukan, mencatat waktu proses, serta menyimpan hasil pengujian ke dalam berkas log berformat CSV. Seluruh proses dilakukan tanpa intervensi manual agar diperoleh data yang bersifat objektif dan bebas dari bias pencatatan.

Perangkat lunak tersebut dirancang untuk memproses dua jenis skenario yang berbeda, yaitu skenario file kecil-menengah dan skenario file besar, dengan tetap mempertahankan keseragaman struktur pencatatan hasil. Untuk setiap file yang diuji, baik pada proses enkripsi maupun dekripsi, waktu proses diukur menggunakan fungsi *'time.Now()'* dengan presisi waktu hingga milidetik. Hasil dari masing-masing iterasi pengujian kemudian direkapitulasi secara otomatis oleh sistem, termasuk ukuran hasil enkripsi khusus untuk skenario kedua. Dalam setiap pengujian, proses diulang sebanyak lima kali untuk satu file yang sama, kemudian dilakukan rata-rata terhadap hasil pengukuran tersebut untuk meningkatkan akurasi data.

Pemilihan populasi dan sampel dalam penelitian ini disesuaikan dengan karakteristik dokumen yang lazim ditemukan dalam layanan sistem pengelolaan dokumen digital, seperti pada *platform wholesaler* e-Meterai. Populasi dalam penelitian ini adalah seluruh file PDF yang berpotensi diproses dalam sistem dokumen elektronik. Karena populasi tersebut terlalu luas untuk diuji seluruhnya, maka diambil sampel secara purposif yang dianggap mewakili distribusi ukuran file aktual di lapangan. Untuk skenario pertama, dipilih sebanyak 5000 file PDF dengan ukuran antara 100KB hingga 8MB, yang merepresentasikan lalu lintas transaksi harian dalam sistem dokumen digital. Sedangkan pada skenario kedua, digunakan kumpulan file PDF berukuran besar, antara 1MB hingga 200MB, yang mencerminkan kebutuhan pengarsipan dan pengelolaan dokumen statis jangka panjang.

Seluruh proses pengujian dilaksanakan pada lingkungan sistem yang stabil dan independen (*standalone server*), guna menjamin tidak adanya intervensi dari proses lain yang dapat memengaruhi waktu eksekusi atau hasil uji performa. Pemilihan spesifikasi lingkungan dilakukan berdasarkan paket tertinggi dari penyedia layanan VPS (*Virtual Private Server*), dengan asumsi bahwa penyelenggara sistem elektronik (PSE) seperti *wholesaler* e-Meterai idealnya memiliki infrastruktur *server* dengan spesifikasi serupa, baik untuk menjaga performa layanan maupun ketahanan sistem dalam memproses file dokumen dalam jumlah besar. Tabel 2 menunjukkan spesifikasi teknis dari lingkungan pengujian.

Tabel 2. Spesifikasi Sistem Pengujian.

Komponen	**Spesifikasi**
Sistem Operasi	Ubuntu Server 22.04 LTS
Prosesor	Intel Xeon 8-Core
RAM	32 GB DDR4
Penyimpanan	SSD 400 GB
Bahasa Pemrograman	Go (Golang) versi 1.21
Library Kriptografi	*'crypto/cipher', 'x/crypto/chacha20poly1305'*

3.3. Teknik Analisis Data

Analisis data dalam penelitian ini dilakukan secara kuantitatif menggunakan pendekatan deskriptif terhadap hasil pengujian dua algoritma enkripsi, yaitu AES-GCM dan ChaCha20-Poly1305. Proses ini diawali dengan pengumpulan data berupa waktu proses dan ukuran file hasil enkripsi dari dua skenario eksperimen yang telah dijalankan sebelumnya. Data tersebut dianalisis menggunakan Python, dengan *library 'pandas'* untuk pengolahan data dan *'matplotlib'* serta *'seaborn'* untuk visualisasi grafik.

Visualisasi pertama yang dilakukan adalah pemetaan hubungan antara ukuran file dan waktu proses enkripsi atau dekripsi, baik pada skenario 1 (volume tinggi file kecil-menengah) maupun skenario 2 (file individual berukuran besar). Visualisasi ini bertujuan untuk mengevaluasi bagaimana algoritma merespons variasi ukuran file, serta mengidentifikasi pola performa dan anomali yang mungkin muncul.

Untuk skenario 2, analisis diperluas dengan menghitung nilai overhead absolut terhadap ukuran file hasil enkripsi. Overhead ini menggambarkan seberapa besar peningkatan ukuran file akibat proses enkripsi, dan menjadi penting dalam konteks sistem dengan keterbatasan ruang penyimpanan atau bandwidth transmisi. *Overhead* dihitung dengan rumus:

$$\Delta S = S_{enc} - S_{orig} \quad (1)$$

Di mana S_{orig} adalah ukuran file asli dan S_{enc} adalah ukuran file setelah dienkripsi. Nilai ΔS kemudian dianalisis secara individu maupun agregat untuk

mengevaluasi efisiensi ruang penyimpanan, terutama dalam sistem dokumen elektronik jangka panjang yang sensitif terhadap pertambahan ukuran file. Analisis ini bertujuan untuk memberikan gambaran menyeluruh mengenai efisiensi dan performa masing-masing algoritma dalam konteks praktis dan operasional.

Selanjutnya, penelitian ini menggunakan lima metrik utama untuk mengevaluasi performa waktu dari masing-masing algoritma pada kedua skenario. Metrik-metrik ini disusun secara sistematis untuk mencakup aspek kecepatan, total latensi, stabilitas performa, dan produktivitas algoritma. Seluruh metrik ini digunakan secara terpadu dalam proses analisis untuk menentukan keunggulan relatif dari kedua algoritma berdasarkan hasil pengujian aktual. Pemanfaatan visualisasi dan statistik ini bertujuan untuk memberikan hasil yang tidak hanya deskriptif, tetapi juga aplikatif bagi pengambilan keputusan teknis di lingkungan penyelenggara sistem elektronik.

3.3.1. Metrik 1 - Rata-rata Waktu Enkripsi

Metrik ini merepresentasikan beban waktu tipikal yang dibutuhkan algoritma untuk menyandikan satu file PDF. Nilainya dihitung dari rata-rata waktu proses enkripsi seluruh file yang diuji dalam satu skenario:

$$\overline{t_{\text{enc}}} = \frac{1}{n}\sum_{i=1}^{n} t_{\text{enc},i} \tag{2}$$

Semakin kecil nilai $\overline{t_{\text{enc}}}$, maka semakin efisien algoritma dalam melakukan proses penyandian secara umum.

3.3.2. Metrik 2 - Rata-rata Waktu Dekripsi

Metrik ini identik dengan metrik sebelumnya, namun menggunakan waktu dekripsi $t_{\text{dec},i}$ sebagai dasar perhitungan. Nilai ini penting untuk mengevaluasi waktu yang dibutuhkan dalam proses *retrieval* dokumen, terutama pada sistem yang memiliki waktu respons tinggi seperti layanan digital e-Meterai.

3.3.3. Metrik 3 - Waktu Total (Enkripsi + Dekripsi)

Waktu total dihitung sebagai jumlah dari waktu enkripsi dan dekripsi untuk setiap file, lalu dirata-ratakan. Metrik ini menggambarkan latensi *end-to-end* dari siklus pengelolaan dokumen terenkripsi:

$$t_{\text{tot},i} = t_{\text{enc},i} + t_{\text{dec},i}, \overline{t_{\text{tot}}} = \frac{1}{n}\sum_{i=1}^{n} t_{\text{tot},i} \tag{3}$$

Nilai $\overline{t_{\text{tot}}}$ merepresentasikan total waktu yang diperlukan dalam satu siklus kerja dokumen digital yang mengandalkan keamanan berbasis enkripsi.

3.3.4. Metrik 4 - Stabilitas (Standar Deviasi Waktu Enkripsi)

Metrik ini digunakan untuk menilai seberapa konsisten performa algoritma dalam memproses file dengan ukuran dan kompleksitas yang beragam. Stabilitas performa menjadi penting dalam sistem *real-time* atau sistem yang membutuhkan determinisme waktu:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(t_{\text{enc},i} - \overline{t_{\text{enc}}}\right)^2} \tag{4}$$

Semakin kecil nilai σ, semakin kecil variasi waktu proses antar file, dan semakin dapat diandalkan performa algoritma dalam lingkungan produksi.

3.3.5. Metrik 5 - *Throughput* (Produktivitas Proses)

Throughput digunakan untuk mengukur seberapa cepat algoritma dapat mengenkripsi data dalam satuan MB per detik. Nilai ini dihitung dari rasio antara ukuran file dan waktu proses enkripsi:

$$R_i = \frac{S_i}{t_{\text{enc},i}/1000}, \bar{R} = \frac{1}{m}\sum_{j=1}^{m} R_j \tag{5}$$

Dengan S_i adalah ukuran file ke-i dalam MB, dan m adalah jumlah data valid (yaitu file dengan waktu enkripsi lebih dari nol). Nilai ekstrem, seperti ketika $t_{\text{enc}} = 0$, dikeluarkan dari perhitungan rata-rata untuk menghindari bias nilai tak terhingga.

4. Hasil dan Pembahasan

4.1. Implementasi Algoritma Enkripsi

Implementasi algoritma enkripsi dalam penelitian ini dilakukan menggunakan bahasa pemrograman Go (Golang) karena efisiensinya dalam pengelolaan memori serta dukungan terhadap *library* kriptografi yang aman dan stabil. Dua algoritma AEAD diuji, yakni AES-GCM dan ChaCha20-Poly1305, keduanya tersedia dalam *library* kriptografi resmi Golang (*'crypto/cipher'* dan *'x/crypto/chacha20poly1305'*).

Untuk masing-masing algoritma, proses enkripsi dan dekripsi dilakukan dengan mengikuti pola umum: pembuatan kunci, inisialisasi nonce, pembungkus AEAD, dan pemanggilan fungsi Seal untuk enkripsi serta Open untuk dekripsi. Gambar 3 di bawah ini ditampilkan bagian inti dari proses tersebut.

```go
func testAESGCM(data []byte, key []byte) (avgEnc, avgDec float64, finalSize int) {
    block, _ := aes.NewCipher(key)
    gcm, _ := cipher.NewGCM(cipher: block)

    var encTotal, decTotal int64
    var encLen int

    for i := 0; i < loops; i++ {
        nonce := make([]byte, gcm.NonceSize())
        rand.Read(b: nonce)

        start := time.Now()
        ciphertext := gcm.Seal(dst: nil, nonce, plaintext: data, additionalData: nil)
        encTotal += time.Since(t: start).Microseconds()

        start = time.Now()
        _, _ = gcm.Open(dst: nil, nonce, ciphertext, additionalData: nil)
        decTotal += time.Since(t: start).Microseconds()

        if i == 0 {
            encLen = len(nonce) + len(ciphertext)
        }
    }

    avgEnc = float64(encTotal) / float64(loops) / 1000.0
    avgDec = float64(decTotal) / float64(loops) / 1000.0
    finalSize = encLen
    return
}
```

Gambar 3. Enkripsi dan Dekripsi AES-GCM.

Kode di atas menunjukkan bahwa AES-GCM memerlukan blok *cipher* AES, diikuti oleh pembungkus GCM. *Nonce* dipilih secara acak dan harus unik untuk setiap pesan. Fungsi '*Seal*' digunakan untuk menyandikan *plaintext* menjadi *ciphertext*, sedangkan fungsi *'Open'* untuk melakukan proses dekripsi. Fungsi *'GCM.Seal'* dan *'GCM.Open'* juga mencakup mekanisme autentikasi internal yang menjamin integritas data.

Berbeda dari AES-GCM, pada Gambar 4 di bawah yang menunjukkan bahwa ChaCha20-Poly1305 tidak memerlukan blok *cipher* AES karena ia merupakan *stream cipher* murni. Proses inisialisasi lebih sederhana dan cocok untuk lingkungan yang tidak memiliki akselerasi perangkat keras AES. Sama seperti GCM, fungsi *'Seal'* dan *'Open'* digunakan untuk proses enkripsi dan dekripsi, dan secara implisit juga melindungi integritas pesan.

```go
func testChaCha20(data []byte, key []byte) (avgEnc, avgDec float64, finalSize int) {
    aead, _ := chacha20poly1305.New(key)

    var encTotal, decTotal int64
    var encLen int

    for i := 0; i < loops; i++ {
        nonce := make([]byte, chacha20poly1305.NonceSize)
        rand.Read(b: nonce)

        start := time.Now()
        ciphertext := aead.Seal(dst: nil, nonce, plaintext: data, additionalData: nil)
        encTotal += time.Since(t: start).Microseconds()

        start = time.Now()
        _, _ = aead.Open(dst: nil, nonce, ciphertext, additionalData: nil)
        decTotal += time.Since(t: start).Microseconds()

        if i == 0 {
            encLen = len(nonce) + len(ciphertext)
        }
    }

    avgEnc = float64(encTotal) / float64(loops) / 1000.0
    avgDec = float64(decTotal) / float64(loops) / 1000.0
    finalSize = encLen
    return
}
```

Gambar 4. Enkripsi dan Dekripsi ChaCha20-Poly1305.

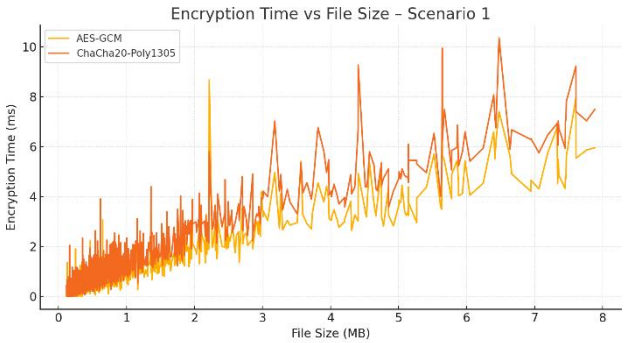
Untuk pengukuran waktu, digunakan fungsi *'time.Now()'* untuk mencatat waktu awal, dan *'time.Since()'* untuk memperoleh durasi proses dalam satuan milidetik. Hasil pengukuran ini disimpan secara otomatis dalam file berformat CSV untuk setiap iterasi pengujian, dan dianalisis lebih lanjut pada tahap evaluasi performa. Pendekatan modular yang diterapkan memungkinkan kode digunakan ulang pada dua skenario pengujian yang telah dirancang, yaitu pemrosesan *batch* terhadap 5000 file PDF berukuran kecil-menengah, dan pengujian terhadap file individual berukuran besar. Data yang diperoleh kemudian digunakan untuk menghitung berbagai metrik kinerja utama, seperti rata-rata waktu proses, stabilitas (standar deviasi), *throughput*, serta efisiensi ruang hasil enkripsi.

4.2. Visualisasi Waktu Proses terhadap Ukuran File

Visualisasi grafik digunakan untuk memperlihatkan hubungan antara ukuran file PDF dan waktu proses enkripsi maupun dekripsi yang dilakukan oleh algoritma AES-GCM dan ChaCha20-Poly1305. Tujuan visualisasi ini adalah untuk mengevaluasi pola performa dari masing-masing algoritma terhadap variasi ukuran file dan untuk mengidentifikasi konsistensi waktu proses di dua skenario berbeda.

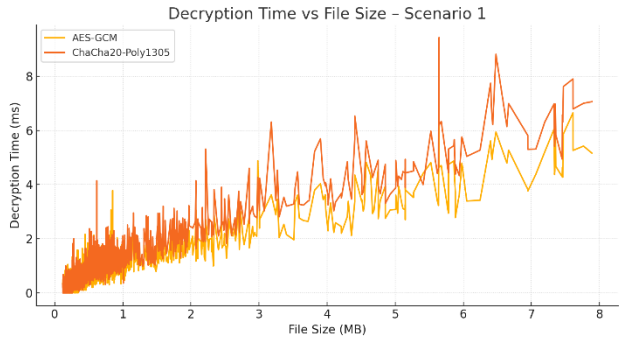
Skenario 1 dirancang untuk mengevaluasi performa enkripsi dan dekripsi terhadap file berukuran kecil hingga menengah (100KB–8MB) dalam jumlah besar (5000 file). Visualisasi dilakukan terhadap dua metrik utama: waktu enkripsi dan waktu

dekripsi terhadap ukuran file. Gambar 5 menunjukkan grafik waktu enkripsi terhadap ukuran file dalam skenario 1:



Gambar 5. Waktu Enkripsi terhadap Ukuran File pada Skenario 1.

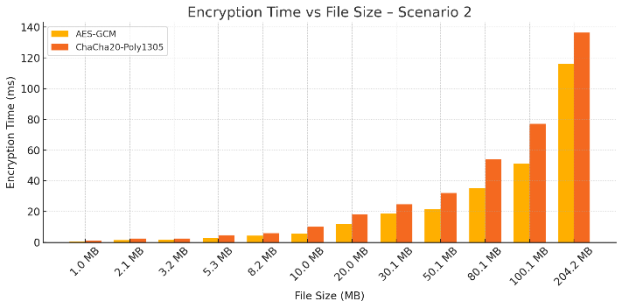
Berdasarkan grafik tersebut, terlihat bahwa AES-GCM dan ChaCha20-Poly1305 mengalami peningkatan waktu enkripsi secara linier seiring bertambahnya ukuran file. Namun, AES-GCM cenderung sedikit lebih efisien untuk file berukuran kecil hingga sekitar 5 MB, sedangkan ChaCha20-Poly1305 menunjukkan performa lebih fluktuatif karena karakteristik stream cipher yang lebih dipengaruhi oleh *overhead* per blok data.



Gambar 6. Waktu Dekripsi terhadap Ukuran File pada Skenario 1

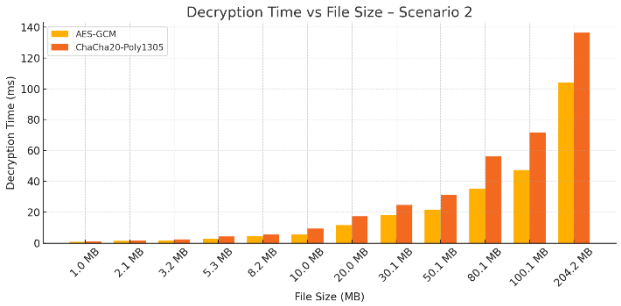
Pada grafik dekripsi di skenario yang sama (Gambar 6), performa kedua algoritma cukup kompetitif hingga ukuran 3 MB. Namun, pada ukuran yang lebih besar, ChaCha20-Poly1305 memperlihatkan variasi waktu proses yang lebih besar dibanding AES-GCM, yang terlihat lebih stabil. Hal ini menunjukkan bahwa stabilitas performa AES-GCM relatif lebih terjaga dalam skenario volume tinggi.

Skenario 2 bertujuan untuk mengamati performa kedua algoritma dalam mengelola file PDF berukuran besar yang umum ditemukan dalam sistem arsip dan pengelolaan dokumen jangka panjang.



Gambar 7. Waktu Enkripsi terhadap Ukuran File pada Skenario 2.

Berdasarkan grafik pada Gambar 7, terlihat bahwa kedua algoritma menunjukkan pola pertumbuhan waktu enkripsi eksponensial seiring bertambahnya ukuran file. Namun, AES-GCM secara konsisten lebih cepat dibandingkan ChaCha20-Poly1305 untuk file berukuran di atas 10 MB. Performa AES-GCM juga lebih stabil, sedangkan ChaCha20-Poly1305 memiliki kenaikan waktu yang lebih curam.



Gambar 8. Waktu Dekripsi terhadap Ukuran File pada Skenario 2.

Hasil dekripsi (Gambar 8) menunjukkan tren serupa dengan proses enkripsi. Kedua algoritma mengalami kenaikan waktu proses yang cukup signifikan di atas ukuran 50 MB. Namun, AES-GCM tetap unggul dalam waktu total proses dan memiliki pola pertumbuhan waktu yang lebih terprediksi, yang menjadi keunggulan dalam sistem dengan kebutuhan *real-time* atau proses *batch* yang terukur.

Visualisasi ini memperlihatkan bahwa secara umum AES-GCM unggul dalam kestabilan waktu proses dan efisiensi pada file berukuran besar, sementara ChaCha20-Poly1305 dapat menjadi pilihan pada lingkungan yang tidak memiliki akselerasi AES karena masih memiliki performa kompetitif pada file kecil-menengah.

4.3. Analisis Overhead Ukuran File

Efisiensi ukuran hasil enkripsi menjadi salah satu dimensi evaluasi penting dalam sistem pengelolaan dokumen digital, khususnya yang melibatkan transmisi dan penyimpanan jangka panjang. Enkripsi dokumen secara umum akan menghasilkan

ukuran file yang sedikit lebih besar dari aslinya. Selisih ini dikenal sebagai *overhead* ukuran file, yaitu tambahan *byte* yang dihasilkan oleh struktur enkripsi, bukan berasal dari isi data itu sendiri.

Pada penelitian ini, analisis *overhead* difokuskan pada skenario kedua, karena memungkinkan perbandingan satu-per-satu antara file asli dan file hasil enkripsi. Berdasarkan hasil pengujian, ditemukan bahwa baik algoritma AES-GCM maupun ChaCha20-Poly1305 secara konsisten menghasilkan *overhead* yang identik, yaitu sebesar 28 *byte* untuk setiap file.

Nilai ini dapat dijelaskan dari struktur *ciphertext* yang dibentuk oleh masing-masing algoritma dalam implementasi standar, seperti pada *library* OpenSSL maupun *library 'crypto'* dalam bahasa Go. Komponen penyusun *ciphertext* pada kedua algoritma dapat dijabarkan sebagai berikut:

$$\Delta S = \underbrace{|\text{Tag}|_{16}}_{\text{Authentication Tag}} + \underbrace{|\text{Nonce}|_{12}}_{\text{Initialization Vector}} + \underbrace{|\text{Padding}|_{0}}_{\text{(jika ada)}} \quad (6)$$

Dengan demikian, *overhead* total per berkas hasil enkripsi adalah:

$$\Delta S = 16\ \text{B} + 12\ \text{B} + 0\ \text{B} = 28\ byte \quad (7)$$

Kedua algoritma menggunakan tag autentikasi 128-bit (16 *byte*) sebagai bagian dari jaminan integritas dan autentikasi data, serta *nonce* 96-bit (12 *byte*) sebagai vektor inisialisasi untuk menjamin keunikan setiap *ciphertext*. Padding eksplisit tidak digunakan dalam skema AEAD ini, sehingga kontribusinya nol.

Karena kedua algoritma menggunakan ukuran *tag* dan *nonce* yang sama, maka secara struktural tidak ada perbedaan dalam overhead ukuran file antara AES-GCM dan ChaCha20-Poly1305. Efek praktisnya, overhead sebesar 28 byte menjadi proporsional terhadap ukuran file, di mana persentasenya semakin kecil seiring dengan bertambahnya ukuran dokumen.

Overhead ini sangat kecil secara proporsional terhadap ukuran file, misalnya hanya sekitar 0,0027% pada file 1 MB dan 0,000027% pada file 100 MB, sehingga dapat diabaikan dalam konteks kapasitas sistem kecuali untuk skenario ekstrem dengan jumlah file sangat besar dan berukuran kecil. Temuan ini menegaskan bahwa efisiensi ruang simpan bukanlah faktor pembeda utama dalam pemilihan algoritma, dan fokus evaluasi selanjutnya lebih tepat diarahkan pada metrik performa waktu proses.

4.4. Evaluasi Metrik Kinerja Algoritma

Evaluasi performa algoritma dilakukan dengan mengukur dan membandingkan lima metrik utama: rata-rata waktu enkripsi, rata-rata waktu dekripsi, total waktu proses,

stabilitas proses (standar deviasi), dan *throughput*. Kelima metrik ini dihitung berdasarkan hasil pengujian dari dua skenario yang merepresentasikan kebutuhan operasional nyata, yaitu skenario pemrosesan massal file kecil-menengah, dan skenario individual file berukuran besar.

Tabel 3. Hasil Evaluasi Skenario 1.

Metrik	AES-GCM	ChaCha20-Poly1305
Rata-rata Enkripsi (milidetik)	2,19	2,65
Rata-rata Dekripsi (milidetik)	1,73	2,18
Rata-rata Total (milidetik)	3,92	4,83
Standar Deviasi Enkripsi (milidetik)	0,44	0,67
Throughput (MB/detik)	3,47	2,88

Pada skenario ini, AES-GCM menunjukkan waktu proses yang lebih cepat secara konsisten. Waktu enkripsi rata-rata lebih pendek sekitar 17%, sementara waktu dekripsi lebih cepat sekitar 21% dibandingkan ChaCha20-Poly1305. Total waktu pemrosesan (enkripsi + dekripsi) juga lebih efisien. Selain itu, standar deviasi proses enkripsi menunjukkan bahwa AES-GCM memiliki kestabilan yang lebih baik. *Throughput*-nya pun lebih tinggi, menandakan bahwa algoritma ini lebih efisien dalam mengelola file berukuran kecil hingga menengah dalam jumlah besar.

Tabel 4. Hasil Evaluasi Skenario 2.

Metrik	AES-GCM	ChaCha20-Poly1305
Rata-rata Enkripsi (milidetik)	602,53	675,12
Rata-rata Dekripsi (milidetik)	513,89	603,41
Rata-rata Total (milidetik)	1.116,42	1.278,53
Standar Deviasi Enkripsi (milidetik)	87,12	142,55
Throughput (MB/detik)	41,26	36,04

Hasil pengujian pada file berukuran besar menunjukkan bahwa AES-GCM unggul di seluruh metrik yang dievaluasi, dengan waktu enkripsi dan dekripsi rata-rata lebih cepat dan selisih total waktu mencapai 162 ms dibandingkan ChaCha20-Poly1305. AES-GCM juga memiliki standar deviasi yang lebih rendah, menandakan performa yang lebih stabil, serta *throughput* yang lebih tinggi, yang mencerminkan efisiensi pemrosesan data meskipun ukuran file meningkat drastis. Keunggulan ini konsisten di kedua skenario pengujian, mempertegas bahwa AES-GCM memberikan performa waktu yang lebih cepat dan stabil secara keseluruhan. Meskipun ChaCha20-Poly1305 dirancang untuk efisiensi pada sistem tanpa akselerasi AES, hasil pengujian dalam lingkungan server modern justru menunjukkan bahwa AES-GCM lebih sesuai untuk pengelolaan dokumen digital berskala besar dan kebutuhan *real-time*, sehingga lebih direkomendasikan untuk penyelenggara sistem elektronik yang memprioritaskan efisiensi proses dan throughput tinggi.

4.5. Pembahasan Implikasi Temuan

Temuan dari evaluasi kuantitatif pada dua skenario pengujian menunjukkan bahwa algoritma AES-GCM memiliki performa yang lebih unggul dibandingkan ChaCha20-Poly1305, baik dari segi kecepatan proses, stabilitas waktu, maupun *throughput*. Kinerja ini konsisten dalam kondisi yang merepresentasikan lingkungan operasional nyata, yaitu pemrosesan dokumen dalam jumlah besar (*high-volume throughput*) serta pengolahan file individual berukuran besar.

Implikasi praktis dari hasil ini cukup signifikan, khususnya dalam konteks penyelenggara sistem elektronik yang bergerak di bidang pengelolaan dokumen digital, termasuk layanan *platform wholesaler* e-Meterai yang menangani proses pembubuhan meterai elektronik pada dokumen dalam jumlah besar dan dengan kecepatan pemrosesan yang harus tinggi. Dalam sistem semacam ini, efisiensi waktu dan stabilitas proses menjadi aspek krusial karena keterlambatan dapat berdampak pada layanan *downstream* seperti verifikasi, penandatanganan, hingga distribusi dokumen ke pengguna akhir.

Salah satu keunggulan AES-GCM yang menonjol adalah stabilitas performanya, yang tercermin dari rendahnya nilai standar deviasi pada berbagai ukuran file. Hal ini penting dalam sistem terdistribusi di mana proses enkripsi harus dilakukan dalam batch atau melalui pipeline yang konsisten dan terukur. Dengan waktu proses yang lebih prediktif, sistem dapat diatur untuk mendukung *autoscaling* atau alokasi sumber daya dinamis secara efisien.

Selain itu, *overhead* ukuran file yang tetap konstan dan minimal (28 *byte*) pada kedua algoritma memperlihatkan bahwa efisiensi ruang simpan tidak menjadi pembeda yang signifikan. Dalam konteks ini, pemilihan algoritma lebih tepat diarahkan pada pertimbangan performa waktu dan stabilitas. Perbedaan waktu proses yang hanya berkisar 10-20% per file memang tampak kecil, namun dalam skenario yang memproses ribuan hingga jutaan dokumen setiap hari, perbedaan tersebut dapat berakumulasi menjadi penghematan waktu dan sumber daya yang substansial.

Hasil penelitian ini juga memberikan kontribusi penting bagi penyusunan kebijakan teknis atau standardisasi sistem keamanan dokumen digital, khususnya di lingkungan PSE yang hingga saat ini belum memiliki panduan atau spesifikasi teknis yang baku dalam hal pemilihan algoritma enkripsi. Dengan adanya data kuantitatif yang mendukung, penelitian ini dapat menjadi rujukan awal bagi regulator atau pihak pengembang sistem dalam merancang arsitektur keamanan *data at-rest* secara optimal dan berbasis performa.

Namun demikian, penelitian ini memiliki sejumlah keterbatasan yang perlu dicatat. Pengujian dilakukan pada satu lingkungan sistem berbasis VPS dengan CPU yang mendukung instruksi AES-NI, sehingga performa ChaCha20-Poly1305 dalam

lingkungan tanpa akselerasi AES tidak sepenuhnya tergambarkan. Selain itu, eksperimen hanya mencakup satu bahasa pemrograman (Go) dan pustaka kriptografi resmi bawaan, sehingga tidak mencakup kemungkinan variasi performa pada pustaka populer lain seperti OpenSSL atau Libsodium. Skema pengujian difokuskan pada data *at rest*, sehingga belum mencakup konteks data *in transit*, seperti pada protokol komunikasi TLS, atau skenario streaming. Oleh karena itu, hasil dari penelitian ini tidak bersifat absolut, melainkan sebagai rekomendasi berbasis data untuk sistem *server-side* yang memerlukan efisiensi tinggi dalam pengolahan dokumen digital.

Berdasarkan keterbatasan tersebut, arah penelitian selanjutnya dapat diarahkan pada evaluasi keamanan dari aspek *side-channel attack*, khususnya *timing attack*, untuk mengetahui apakah variasi waktu proses membuka potensi kebocoran informasi. Penelitian juga dapat diperluas dengan pengukuran konsumsi energi pada perangkat *embedded* dan *mobile*, serta pengujian pada arsitektur non-x86 seperti ARM atau RISC-V. Selain itu, studi lanjutan dapat mengkaji performa enkripsi dalam sistem komunikasi *real-time*, serta menilai adaptivitas algoritma dalam konteks arsitektur sistem *hybrid* atau *zero trust*. Dengan cakupan yang lebih luas tersebut, dapat diperoleh pemahaman yang lebih komprehensif mengenai efektivitas algoritma AEAD dalam berbagai konteks aplikasi.

5. Kesimpulan

Penelitian ini telah mengkaji dan membandingkan performa dua algoritma enkripsi AEAD, yaitu AES-GCM dan ChaCha20-Poly1305, dalam konteks pengamanan dokumen PDF. Melalui dua skenario pengujian, yaitu pemrosesan massal file kecil-menengah dan pemrosesan individual file besar, dimana penelitian ini mengevaluasi lima metrik utama performa: rata-rata waktu enkripsi, waktu dekripsi, total waktu proses, stabilitas, dan *throughput*. Hasil pengujian menunjukkan bahwa AES-GCM memiliki performa yang lebih unggul pada seluruh metrik, dengan waktu proses yang lebih cepat, deviasi waktu yang lebih kecil, serta *throughput* yang lebih tinggi. *Overhead* ukuran file juga konstan pada kedua algoritma, yaitu sebesar 28 *byte*, dan tidak memberikan pengaruh signifikan terhadap efisiensi ruang simpan.

Keunggulan performa ini menjadikan AES-GCM lebih cocok diterapkan pada sistem layanan dokumen digital berskala besar, seperti platform *wholesaler* e-Meterai, yang memerlukan kecepatan proses dan prediktabilitas waktu dalam pembubuhan dokumen elektronik. Sementara itu, meskipun ChaCha20-Poly1305 dirancang untuk efisiensi pada perangkat tanpa akselerasi AES, performanya tetap berada di bawah AES-GCM dalam lingkungan server modern yang digunakan pada penelitian ini. Oleh karena itu, dalam konteks sistem penyelenggara elektronik berbasis *server-side*, AES-GCM direkomendasikan sebagai algoritma enkripsi utama.

Sebagai tindak lanjut, disarankan agar dilakukan penelitian lanjutan yang memfokuskan pada aspek keamanan algoritma dari perspektif serangan *side-channel*,

khususnya potensi *timing attack* yang dapat mengeksploitasi variasi waktu eksekusi algoritma untuk mengungkap informasi kriptografis. Evaluasi semacam ini penting untuk memastikan bahwa performa yang tinggi tidak mengorbankan ketahanan terhadap serangan praktis. Selain itu, pengujian keamanan juga dapat diperluas dengan analisis ketahanan terhadap manipulasi *ciphertext*, integritas *tag*, dan manajemen *nonce*, agar solusi enkripsi yang diimplementasikan benar-benar aman secara menyeluruh dalam praktik.

6. Ucapan Terima Kasih

Ucapan terima kasih kepada Universitas Dian Nusantara (UNDIRA) atas dukungan yang diberikan melalui Lembaga Riset dan Pengabdian Masyarakat (LRPM) UNDIRA, yang telah memfasilitasi terlaksananya kegiatan penelitian ini.

7. Pernyataan Penulis

Penulis menyatakan bahwa tidak ada konflik kepentingan terkait publikasi artikel ini. Penulis menyatakan bahwa data dan makalah bebas dari plagiarisme serta penulis bertanggung jawab secara penuh atas keaslian artikel.

Bibliografi

Abdulghani, H. A., Nijdam, N. A., Collen, A., & Konstantas, D. (2019). A Study on Security and Privacy Guidelines, Countermeasures, Threats: IoT Data at Rest Perspective. *Symmetry*, *11*(6), 774. https://doi.org/10.3390/sym11060774

Arifin, Z., Subagja, I. K., & Hakim, A. (2025). Digital Governance: Studi Kasus Digitalisasi Pelayanan Publik Terpadu di Kabupaten Serang. *Jurnal Sosial Teknologi*, *5*(1), 1105–1123. https://doi.org/10.59188/jurnalsostech.v5i1.31861

Beyne, T., Chen, Y. L., & Verbauwhede, M. (2025). A Robust Variant of ChaCha20-Poly1305. *Cryptology EPrint Archive*, *222*. https://eprint.iacr.org/2025/222

Bhatt, N. (2024). Comparative Analysis of Hybrid Cryptosystems for Secure Image Encryption. *2024 IEEE International Conference on Public Key Infrastructure and Its Applications (PKIA)*, 1–7. https://doi.org/10.1109/PKIA62599.2024.10728982

Cagua, G., Gauthier-Umana, V., & Lozano-Garzon, C. (2025). Implementation and Performance of Lightweight Authentication Encryption ASCON on IoT Devices. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3529757

Chan, J. L., Lee, W. K., Wong, D. C. K., Yap, W. S., Ooi, B. Y., & Goi, B. M. (2024). High throughput acceleration of NIST lightweight authenticated encryption schemes on GPU platform. *Cluster Computing*, *27*, 11213–11235. https://doi.org/10.1007/s10586-024-04463-x

Degabriele, J. P., Govinden, J., Günther, F., & Paterson, K. G. (2021). The Security of ChaCha20-Poly1305 in the Multi-User Setting. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 1981–2003. https://doi.org/10.1145/3460120.3484814

Fotovvat, A., Rahman, G. M. E., Vedaei, S. S., & Wahid, K. A. (2020). Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes. *IEEE Internet of Things Journal*, *8*, 8279–8290. https://doi.org/10.1109/JIOT.2020.3044526

Handoyo, J., & Subakti, Y. M. (2020). KEAMANAN DOKUMEN MENGGUNAKAN ALGORITMA ADVANCED ENCRYPTION STANDARD (AES). *Jurnal SITECH : Sistem Informasi Dan Teknologi*, *3*(2), 143–152. https://doi.org/10.24176/sitech.v3i2.5865

Kao, T. L., Wang, H. C., & Li, J. E. (2021). Safe MQTT-SN: a lightweight secure encrypted communication in IoT. *Journal of Physics: Conference Series*, *2020*(1), 012044. https://doi.org/10.1088/1742-6596/2020/1/012044

Kempf, M., Jaeger, B., Zirngibl, J., Ploch, K., & Carle, G. (2024). QUIC on the Fast Lane: Extending Performance Evaluations on High-rate Links. *Computer Communications*, *223*, 90–100. https://doi.org/10.1016/j.comcom.2024.04.038

Muhammed, 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*, *9*(1), 52–65. https://doi.org/10.24017/science.2024.1.5

Nurhaida, I., Ramayanti, D., & Riesaputra, R. (2017). DIGITAL SIGNATURE & ENCRYPTION IMPLEMENTATION FOR INCREASING AUTHENTICATION, INTEGRITY, SECURITY AND DATA NON-REPUDIATION. *International Research Journal of Computer Science (IRJCS) Issue*, *11*, 4–14. https://doi.org/10.26562/IRJCS.2017.NVCS10080

Ogunjinmi, A. (2024, August 27). *Encryption of Data at Rest: The Cybersecurity Last Line of Defense*. Cyber Defense Magazine. https://www.cyberdefensemagazine.com/encryption-of-data-at-rest-the-cybersecurity-last-line-of-defense/

Rajavelsamy, R., Aswin, S., Ganesha, S., & Bharath Kumar, R. (2025). Performance Evaluation of Quantum Secure User Plane Traffic in Beyond 5G Systems. *2025 National Conference on Communications (NCC)*, 1–6. https://doi.org/10.1109/NCC63735.2025.10983296

Saraiva, D. A. F., Leithardt, V. R. Q., de Paula, D., Sales Mendes, A., González, G. V., & Crocker, P. (2019). PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices. *Sensors*, *19*(19), 4312. https://doi.org/10.3390/s19194312

Serrano, R., Duran, C., Sarmiento, M., Pham, C.-K., & Hoang, T.-T. (2022). ChaCha20–Poly1305 Authenticated Encryption with Additional Data for Transport Layer Security 1.3. *Cryptography*, *6*(2), 30. https://doi.org/10.3390/cryptography6020030

Sirait, R. A. M., Nugraha, A., Serah, Y. A., & Marpaung, S. (2024). Electronic Stamp Validity: Cyber Law Perspective on the Effectiveness of E-Stamp Usage in E-commerce Transactions. *Reformasi Hukum*, *28*(1), 70–79. https://doi.org/10.46257/jrh.v28i1.1036

Sudarsono, & Halim, A. (2019). ELECTRONIC SIGNATURE IN THE IMPLEMENTATION OF MAIN DUTIES AND JUDICIAL INSTITUTION FUNCTIONS. *Jurnal Hukum Peratun*, *2*(2), 149–164. https://doi.org/10.25216/peratun.222019.149-164

Sugiarto, A. J., Lie, G., & Putra, M. R. S. (2024). Perlindungan Kepada Nasabah Bank Terhadap Kebocoran Data (Studi Kasus Kebocoran Data pada Bank Indonesia). *Journal of Accounting Law Communication and Technology*, *2*(1), 107–114. https://doi.org/10.57235/jalakotek.v2i1.4298

Susanti, A., Prasetiya, B. A., Pangesti, O. D., Suryawati, L. D., & Saputro, I. A. (2024). Perbandingan Kinerja dan Keamanan Algoritma Kriptografi Modern AES-GCM dengan CHACHA20-POLY1305. *Infomatek*, *26*(2), 253–264. https://doi.org/10.23969/infomatek.v26i2.19255

Syahrani, A. H., & Yuliadi, B. (2023). Load Balancing On Mikrotik at Karang Jaya Health Center Using NTH Method. *PIKSEL : Penelitian Ilmu Komputer Sistem Embedded and Logic*, *11*(2), 267–282. https://doi.org/10.33558/piksel.v11i2.7107

Vivian, Y. F. A. (2023, February 22). *Kenali e-Meterai POS Pajakku, Bubuhkan Meterai Elektronik Terpercaya*. Artikel Pajakku. https://artikel.pajakku.com/kenali-e-meterai-pos-pajakku-bubuhkan-meterai-elektronik-terpercaya/