Tinjauan Algoritma Advanced Encryption Standard (AES-256) dan Mode Operasi Galois/Counter Mode (GCM)

Idham Kholid Nur Azizi

Program Studi Teknik Informatika

Universitas Esa Unggul

Email: idhamkholid2294@student.esaunggul.ac.id

Link YouTube: https://youtu.be/H5h5zdq-bck

I. PENDAHULUAN

Kebutuhan akan keamanan data yang kuat telah menempatkan kriptografi sebagai komponen esensial dalam sistem informasi modern. Advanced Encryption Standard (AES) merupakan algoritma kriptografi simetris yang menjadi standar global setelah ditetapkan oleh National Institute of Standards and Technology (NIST) Amerika Serikat pada tahun 2001 dalam publikasi FIPS 197 (Advanced Encryption Standard (AES), 2023). Dari varian yang ada, AES-256 (menggunakan kunci 256-bit) diadopsi untuk aplikasi dengan kebutuhan tingkat kerahasiaan tertinggi, melampaui standar AES-128 (Advanced Encryption Standard (AES), 2023).

Algoritma block cipher murni seperti AES tidak dapat langsung mengenkripsi data berukuran variabel. Untuk itu, "mode operasi" diperlukan. Mode GCM (Galois/Counter Mode), yang distandarisasi dalam NIST Special Publication 800-38D, telah menjadi standar industry (Dworkin, 2007). Keunggulannya terletak pada penyediaan Authenticated Encryption with Associated Data (AEAD), yang secara simultan memberikan jaminan kerahasiaan dan autentikasi integritas data dalam satu proses (Dworkin, 2007). Tinjauan literatur ini bertujuan untuk mengkaji secara mendalam aspek teknis, properti kriptografis, kinerja, dan tantangan keamanan fundamental dalam implementasi AES-256 yang dipadukan dengan mode GCM, berdasarkan analisis penelitian terbaru.

II. LANDASAN TEORI (ANALISIS KRIPTOGRAFIS MENDALAM)

Untuk menganalisis AES-GCM, kita harus membedah dua komponennya berdasarkan standar teknis yang mendefinisikannya: algoritma block cipher (AES-256) sebagai "inti" enkripsi, dan mode GCM sebagai "kerangka" yang menggunakannya.

2.1 Arsitektur Internal AES-256: Block Cipher

AES adalah block cipher yang beroperasi pada state (matriks internal) berukuran 4x4 byte (128-bit), terlepas dari ukuran kuncinya (Advanced Encryption Standard (AES), 2023). Perbedaan AES-256 terletak pada penggunaan kunci 256-bit yang mengimplikasikan 14 putaran (rounds) transformasi (Advanced Encryption

Standard (AES), 2023). Setiap putaran (kecuali yang terakhir) adalah fungsi yang terdiri dari empat transformasi berbeda, yang didefinisikan dalam FIPS 197:

1) SubBytes (Substitusi Non-Linear):
    - Tujuan: Menyediakan kebingungan (confusion). Ini adalah satu-satunya langkah non-linear dalam AES.
    - Mekanisme: Setiap byte dari state 4x4 diganti dengan byte lain menggunakan tabel substitusi 8-bit yang telah ditentukan, yang disebut "Rijndael S-Box". S-Box ini dirancang khusus untuk memiliki sifat non-linearitas yang tinggi dan resistensi terhadap serangan kriptanalisis linear dan diferensial (Advanced Encryption Standard (AES), 2023).

2) ShiftRows (Transposisi):
    - Tujuan: Menyediakan difusi (diffusion) antar kolom.
    - Mekanisme: Baris-baris dari state 4x4 digeser secara siklis. Baris 0 tidak digeser, baris 1 digeser 1 byte ke kiri, baris 2 digeser 2 byte, dan baris 3 digeser 3 byte. Ini memastikan bahwa byte-byte dari kolom yang berbeda tercampur pada putaran berikutnya (Advanced Encryption Standard (AES), 2023).

3) MixColumns (Pencampuran Matriks):
    - Tujuan: Menyediakan difusi (diffusion) di dalam kolom.
    - Mekanisme: Setiap kolom 4-byte dari state diperlakukan sebagai polinomial dan dikalikan (dalam finite field $GF(2^8)$) dengan matriks tetap. Operasi ini memastikan bahwa setiap byte input pada satu kolom mempengaruhi keempat byte output pada kolom yang sama, menyebarkan perubahan data secara eksponensial (Advanced Encryption Standard (AES), 2023).

4) AddRoundKey (Kombinasi Kunci):
    - Tujuan: Memasukkan materi kunci rahasia ke dalam state.
    - Mekanisme: State 128-bit di-XOR-kan dengan Round Key 128-bit yang unik untuk putaran tersebut (Advanced Encryption Standard (AES), 2023).

2.2 Proses Kunci: KeySchedule (Ekspansi Kunci) AES-256

AES-256 memerlukan total 15 Round Keys 128-bit (1 untuk pra-putaran + 14 untuk setiap putaran) (Advanced Encryption Standard (AES), 2023). Padahal, kita hanya memiliki satu kunci master 256-bit. KeySchedule adalah algoritma yang "mengekspansi" kunci 256-bit tersebut menjadi 15 Round Keys.

Proses ini menggunakan operasi rotasi (RotWord), substitusi (SubWord menggunakan S-Box), dan XOR dengan konstanta putaran (Rcon) untuk menghasilkan materi kunci yang unik untuk setiap putaran. Proses ini dirancang

agar setiap bit dari kunci master mempengaruhi banyak Round Keys, dan untuk mencegah serangan related-key (Advanced Encryption Standard (AES), 2023).

2.3 Konsep Dasar: Mode Operasi AEAD GCM

GCM adalah mode AEAD yang canggih, sebagaimana didefinisikan dalam (Dworkin, 2007). Ini berarti GCM dirancang untuk melakukan dua tugas sekaligus:

1) Authenticated Encryption: Memberikan kerahasiaan (enkripsi) dan jaminan integritas (autentikasi) (Dworkin, 2007).
2) Associated Data (AD): Mampu memberikan jaminan integritas untuk data tambahan (seperti header paket) yang tidak perlu dienkripsi, tetapi tidak boleh diubah (Dworkin, 2007).

GCM mencapai ini dengan menggabungkan dua mekanisme kriptografis secara paralel, yang keduanya dijelaskan dalam standar NIST SP 800-38D.

2.4 Mekanisme GCM (1): Kerahasiaan via Mode CTR

GCM tidak mengenkripsi plaintext secara langsung. Sebaliknya, ia menggunakan AES-256 dalam Mode Counter (CTR) untuk menghasilkan keystream (aliran kunci), yang pada dasarnya mengubah AES (sebuah block cipher) menjadi stream cipher (Dworkin, 2007).

- Input: Sebuah Nonce (nilai unik) 96-bit dan sebuah Counter (pencacah) 32-bit (Dworkin, 2007).
- Proses: GCM menggabungkan Nonce dan Counter (misal: Counter=1) untuk membuat blok 128-bit yang unik.
- Blok 128-bit ini dienkripsi menggunakan AES-256: Keystream_Blok_1 = AES-256(Kunci, Nonce + Counter_1).
- Enkripsi: Ciphertext_Blok_1 = Plaintext_Blok_1 $\oplus$ Keystream_Blok_1.
- Proses diulang dengan menaikkan counter (Counter=2, 3, ...) untuk setiap blok data (Dworkin, 2007).

Keunggulan utamanya adalah kecepatan: karena setiap blok counter independen, enkripsi dapat dilakukan secara paralel pada banyak inti CPU (Dworkin, 2007).

2.5 Mekanisme GCM (2): Autentikasi via GHASH

Secara paralel dengan enkripsi CTR, GCM menghitung tag autentikasi menggunakan fungsi hash universal yang disebut GHASH (Dworkin, 2007).

1) Kunci Autentikasi (H): GCM pertama-tama membuat kunci hash internalnya sendiri (disebut H) dengan mengenkripsi satu blok berisi angka nol: H = AES-256(Kunci, 0^128) (Dworkin, 2007).
2) Proses GHASH: GHASH adalah fungsi hash polimomial yang bekerja di dalam finite field GF(2^{128}) (perkalian Galois) (Dworkin, 2007).
3) GHASH mengambil blok-blok dari Associated Data (AD) dan Ciphertext (bukan Plaintext), lalu "mencampurnya" dengan kunci H (Dworkin, 2007).
4) Tag Final: Output dari GHASH kemudian dienkripsi sekali lagi (menggunakan blok counter ke-0) untuk menghasilkan Authentication Tag 128-bit (Dworkin, 2007).

III. VISUALISASI PROSES AES-GCM

(Bagian ini memvisualisasikan proses yang dijelaskan dalam Bab II, yang didasarkan pada standar (Dworkin, 2007) dan (Advanced Encryption Standard (AES), 2023).

3.1 Visualisasi Proses Enkripsi AES-GCM

Proses enkripsi AEAD GCM mengambil Kunci, Nonce, dan Plaintext, serta Associated Data (AAD) opsional. Proses ini menghasilkan dua output yang terpisah: Ciphertext dan Authentication Tag.

1) Plaintext dienkripsi menggunakan Mode CTR (menghasilkan Ciphertext).
2) Secara bersamaan, GHASH memproses AAD dan Ciphertext.
3) Hasil dari kedua proses digabungkan dan dienkripsi untuk membuat Tag.

3.2 Visualisasi Proses Dekripsi dan Verifikasi AES-GCM

Proses dekripsi adalah "cermin" dari enkripsi dan merupakan langkah paling kritis secara keamanan.

1) Ciphertext didekripsi menggunakan Mode CTR yang sama (menghasilkan Potensial Plaintext).
2) Secara bersamaan, GHASH memproses AAD dan Ciphertext (yang diterima) untuk menghitung Tag Terhitung (Tag internal).
3) Langkah Verifikasi Kritis: Tag Terhitung (internal) dibandingkan bit-per-bit dengan Authentication Tag (yang diterima bersama pesan).
    - Jika Tag COCOK: Integritas data terverifikasi. Potensial Plaintext dianggap aman dan diteruskan ke aplikasi.
    - Jika Tag GAGAL: Integritas data gagal. Sistem harus membuang Potensial Plaintext dan mengembalikan error autentikasi. Data yang didekripsi tidak boleh digunakan.

IV. TINJAUAN PENELITIAN TERDAHULU

Tinjauan ini menganalisis temuan-temuan kunci dari berbagai studi yang berfokus pada kinerja dan keamanan implementasi AES-256 GCM.

Peneliti & Tahun	**Fokus Studi (terkait AES-256/GCM)**	**Hasil & Temuan Kunci**
(Chizoba Eromosele, 2025)	Analisis dampak kinerja AES-256 pada metrik jaringan.	Menemukan bahwa enkripsi AES-256 memiliki dampak yang dapat diabaikan pada ukuran file. *Throughput* (kecepatan transfer) dan *latency* sangat mirip antara *plaintext* dan *ciphertext*, menunjukkan bahwa pada jaringan modern, *overhead* komputasi AES sangat minimal.
(Susanti et al., 2024)	Perbandingan kinerja AES-GCM vs. ChaCha20-Poly1305.	Menunjukkan bahwa AES-GCM memiliki kinerja sangat tinggi pada perangkat yang memiliki **akselerasi perangkat keras (instruksi AES-NI)**. Namun, ChaCha20-Poly1305 (algoritma *software-only*) lebih unggul pada perangkat *low-end* (misal:

		mikrokontroler) yang tidak memiliki instruksi AES-NI.
(Böck et al., n.d.)	Analisis kerentanan keamanan GCM dalam implementasi TLS.	Mengidentifikasi kerentanan "serangan pemalsuan" (*forgery attacks*) yang fatal pada implementasi GCM di dunia nyata. Serangan ini dimungkinkan jika **nonce (nilai counter) digunakan kembali** untuk kunci yang sama. Studi ini menemukan server HTTPS di internet yang rentan terhadap hal ini.
(Mantel et al., n.d.)	Studi sistematis tentang *cache side-channel attacks* pada implementasi AES.	Menunjukkan bahwa banyak *library* kriptografi populer rentan terhadap serangan *side-channel* berbasis *cache*. Serangan ini tidak mengeksploitasi kelemahan matematis AES, tetapi pada cara implementasi *software* (khususnya *lookup tables* S-Box) berinteraksi dengan *cache* CPU, yang dapat membocorkan kunci rahasia.
(Holmgren & Lombardi, n.d.)	Spesifikasi dan analisis AES-GCM-SIV.	Sebagai respons terhadap kerentanan "penyalahgunaan *nonce*" (seperti yang ditemukan (Böck et al., n.d.)), penelitian ini merancang **AES-GCM-SIV**, sebuah varian GCM yang "tahan terhadap penyalahgunaan *nonce*" (*nonce misuse-resistant*). Mode ini mencegah kegagalan keamanan total jika *nonce* tidak sengaja digunakan kembali.

V. ANALISIS DAN SINTESIS

Sintesis dari literatur menunjukkan paradoks: AES-GCM sangat aman secara teoretis, namun sangat "rapuh" (mudah gagal total) dalam implementasi praktis.

5.1 Analisis Kinerja: Peran Akselerasi Perangkat Keras (AES-NI)

Studi (Chizoba Eromosele, 2025) dan (Susanti et al., 2024) mengonfirmasi bahwa overhead kinerja AES-GCM dapat diabaikan. Namun, (Susanti et al., 2024) menjelaskan mengapa hal ini terjadi: bukan karena algoritmanya sederhana, tetapi karena adanya akselerasi perangkat keras. Hampir semua CPU modern (Intel, AMD, ARM) menyertakan set instruksi khusus AES-NI (Advanced Encryption Standard New Instructions). Instruksi ini menjalankan operasi AES (seperti SubBytes, MixColumns) dan GCM (perkalian Galois) langsung di dalam silikon CPU, membuatnya ribuan kali lebih cepat daripada menjalankannya dalam software murni. Inilah alasan (Susanti et al., 2024) menemukan ChaCha20-Poly1305 (yang dirancang untuk software) lebih cepat pada perangkat low-end yang tidak memiliki AES-NI.

5.2 Analisis Kriptografi: Kerentanan Implementasi Fatal

Literatur menyoroti bahwa risiko terbesar AES-GCM bukanlah pada desain algoritmanya, tetapi pada kesalahan implementasi.

1) Bencana Penggunaan Ulang Nonce (Nonce Reuse Catastrophe): Seperti ditekankan oleh standar (Dworkin, 2007) dan dieksploitasi secara praktis oleh (Böck et al., n.d.), penggunaan nonce yang sama sebanyak dua kali

dengan kunci yang sama adalah kesalahan fatal. Ini menyebabkan dua kegagalan:

- Kegagalan Kerahasiaan: Karena mode CTR adalah stream cipher (C = P ⊕ KS), jika nonce dan kunci sama, maka keystream (KS) akan sama. (Böck et al., n.d.) menjelaskan bahwa penyerang dapat mengambil dua ciphertext (C1 dan C2) dan melakukan XOR: C1 ⊕ C2 = (P1 ⊕ KS) ⊕ (P2 ⊕ KS) = P1 ⊕ P2 Penyerang mendapatkan XOR dari dua plaintext. Jika konteks data diketahui, ini seringkali cukup untuk memulihkan kedua plaintext.
- Kegagalan Autentikasi: Yang lebih parah, (Böck et al., n.d.) menunjukkan bahwa penggunaan ulang nonce memungkinkan penyerang untuk memulihkan kunci autentikasi H. Setelah H diketahui, penyerang dapat memalsukan (forge) pesan baru dengan tag autentikasi yang valid, menghancurkan seluruh jaminan integritas GCM.

2) Kerentanan Side-Channel Perangkat Lunak: (Mantel et al., n.d.) menyoroti risiko yang berbeda. Implementasi AES yang "naif" (seringkali untuk optimasi software jika AES-NI tidak ada) menggunakan lookup tables untuk S-Box. Serangan cache side-channel, seperti dijelaskan oleh (Mantel et al., n.d.), bekerja dengan mengukur perbedaan waktu yang sangat kecil saat CPU mengakses data dari cache versus dari RAM. Dengan memantau pola akses cache saat enkripsi berlangsung, penyerang dapat menyimpulkan bit-bit kunci rahasia. Implementasi yang aman, menurut (Mantel et al., n.d.), harus bersifat constant-time (berjalan dalam waktu yang sama persis terlepas dari data atau kunci).

5.3 Evolusi dan Mitigasi: AES-GCM-SIV

Kerapuhan GCM terhadap nonce reuse begitu parah sehingga komunitas kriptografi merancang alternatif. Seperti ditinjau oleh (Holmgren & Lombardi, n.d.), AES-GCM-SIV adalah respons langsung. SIV (Synthetic Initialization Vector) adalah mode nonce misuse-resistant. Seperti dijelaskan dalam penelitian mereka, jika nonce digunakan kembali dalam GCM-SIV, ia hanya akan membocorkan fakta bahwa data yang sama dikirim ulang, namun tidak akan membocorkan plaintext dan tidak akan mengizinkan pemalsuan pesan (Holmgren & Lombardi, n.d.). Ini adalah pertukaran keamanan yang jauh lebih baik.

VI. ARAH DAN PELUANG PENELITIAN

Berdasarkan analisis literatur, beberapa area penelitian masa depan menjadi sangat relevan:

1) Standardisasi dan Adopsi Mode Nonce Misuse-Resistant: Peluang penelitian terbesar adalah dalam analisis, standardisasi, dan adopsi luas mode yang tahan terhadap penyalahgunaan nonce seperti AES-GCM-SIV, yang propertinya diuraikan oleh (Holmgren & Lombardi, n.d.).

2) Implementasi Constant-Time yang Terverifikasi: Masih ada kebutuhan untuk library kriptografi yang tidak hanya cepat tetapi juga dapat dibuktikan secara formal tahan terhadap serangan side-channel yang diidentifikasi oleh (Mantel et al., n.d.).
3) Kriptografi Pasca-Kuantum (PQC): Meskipun AES-256 saat ini dianggap aman terhadap komputer kuantum, kunci AES seringkali dipertukarkan menggunakan algoritma asimetris (seperti RSA) yang rentan. Penelitian sedang berlangsung untuk menggabungkan AES-256 GCM dalam skema hibrida dengan algoritma PQC, sejalan dengan inisiatif yang juga dipimpin oleh NIST (Advanced Encryption Standard (AES), 2023).

VII. KESIMPULAN

Tinjauan literatur ini menegaskan bahwa AES-256 adalah algoritma block cipher yang sangat aman dan efisien, sesuai dengan standar (Advanced Encryption Standard (AES), 2023). Mode operasi GCM melengkapinya dengan menyediakan authenticated encryption (AEAD) berkinerja tinggi yang esensial untuk protokol modern (Dworkin, 2007). Kinerjanya di dunia nyata sangat ditopang oleh akselerasi perangkat keras (AES-NI) (Susanti et al., 2024).

Namun, sintesis penelitian (seperti (Böck et al., n.d.) dan (Mantel et al., n.d.)) secara jelas menunjukkan bahwa tantangan keamanan AES-GCM telah bergeser dari desain algoritmik ke kerapuhan implementasi praktis. Ketergantungan kritisnya pada keunikan nonce (Dworkin, 2007) menjadikannya rentan terhadap kesalahan pengembang yang katastrofal (Böck et al., n.d.), sementara implementasi software-nya rentan terhadap serangan side-channel (Mantel et al., n.d.).

Oleh karena itu, arah penelitian modern tidak lagi berfokus pada "apakah AES-256 aman?", melainkan pada "bagaimana kita mengimplementasikan AES-256 GCM secara aman dan dalam skala besar?" dan "apa mode operasi generasi berikutnya (seperti AES-GCM-SIV) yang dapat melindungi dari kesalahan implementasi?" (Holmgren & Lombardi, n.d.).

VIII. DAFTAR PUSTAKA

Advanced Encryption Standard (AES). (2023). https://doi.org/10.6028/NIST.FIPS.197-upd1

Böck, H., Zauner, A., Devlin, S., Somorovsky, J., & Jovanovic, P. (n.d.). *Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS*. https://github.com/

Chizoba Eromosele, C. (2025). Evaluating the Impact of AES-256 Encryption on Network Performance: An Analysis of Transfer Time, Latency and Throughput. *International Journal of Scientific Research and Modern Technology*, *4*(1), 49. https://doi.org/10.5281/zenodo.14745473

Dworkin, M. J. (2007). *Recommendation for block cipher modes of operation :* https://doi.org/10.6028/NIST.SP.800-38d

Holmgren, J., & Lombardi, A. (n.d.). *Cryptographic Hashing From Strong One-Way Functions * Or: One-Way Product Functions and their Applications*.

Mantel, H., Weber, A., & Köpf, B. (n.d.). *A Systematic Study of Cache Side Channels across AES Implementations*. www.mais.informatik.tu-darmstadt.de/cacheaudit-essos17.html.

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