BAB II

LANDASAN TEORI

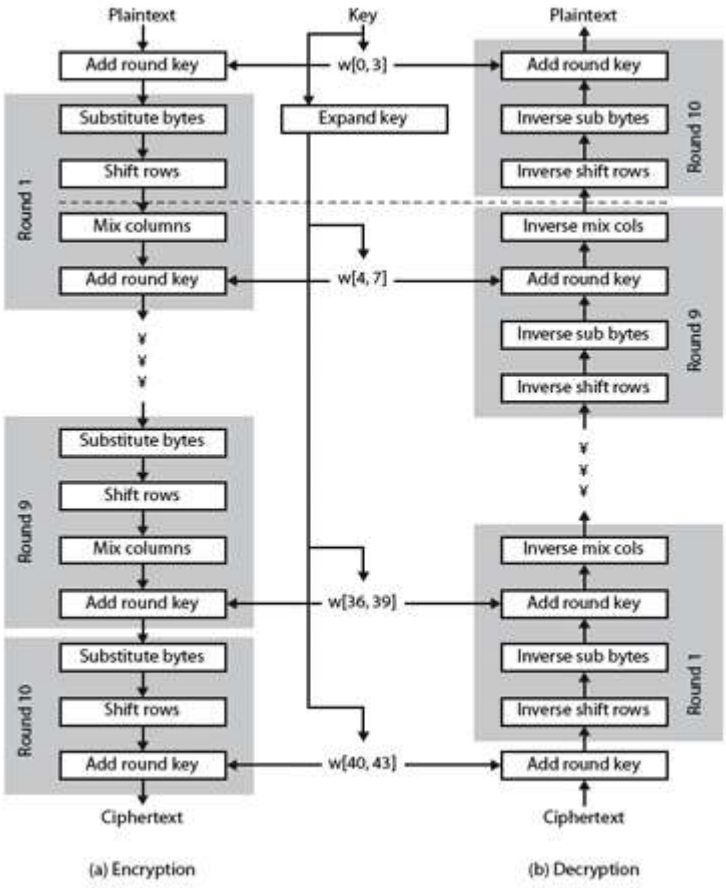
2.1 *Advanced Encryption Standard* (AES)

Advanced Encryption Standard (AES) merupakan algoritma kriptografi simetris yang ditetapkan sebagai standar enkripsi *oleh National Institute of Standards and Technology (NIST)*. AES bekerja menggunakan ukuran blok 128 bit dengan panjang kunci 128 bit, 192 bit, atau 256 bit. Proses enkripsi AES terdiri atas beberapa transformasi utama yaitu *SubBytes, ShiftRows, MixColumns, dan AddRoundKey* yang dijalankan secara berulang dalam sejumlah ronde sesuai panjang kunci yang digunakan (FIPS 197, 2023).

Setiap transformasi dirancang sedemikian rupa untuk menciptakan campuran dan penyebaran informasi, sehingga operasinya tahan terhadap berbagai teknik kriptanalisis. Panjang kunci yang dapat dipilih (128, 192, atau 256 bit) menyediakan tingkat keamanan yang tinggi, dan karena AES menggunakan kunci yang sama untuk enkripsi dan dekripsi, maka pengelolaan kunci yang aman menjadi aspek krusial dalam pemanfaatkan algoritma ini. Karena karakteristiknya tersebut, AES telah banyak digunakan dalam pengamanan data digital pada aplikasi, jaringan, dan sistem informasi di berbagai domain ilmiah dan industri (Hidayatulloh et al., 2023).

AES adalah algoritma enkripsi blok simetris modern dengan struktur *substitution-permutation network* yang bekerja pada blok 128-bit dan

mendukung panjang kunci 128, 192, serta 256 bit, dengan serangkaian transformasi matematis (*SubBytes, ShiftRows, MixColumns, dan AddRoundKey*) untuk menghasilkan *ciphertext* yang aman dan tahan terhadap serangan kriptanalisis. Keunggulan utama AES terletak pada tingkat keamanannya yang tinggi, efisiensi komputasi, serta fleksibilitas panjang kunci yang menjadikannya standar enkripsi global dalam berbagai sistem keamanan informasi (Abdullah, 2017). Gambar 2.1 menunjukkan struktur dasar algoritma AES yang menggambarkan tahapan transformasi pada proses enkripsi dan dekripsi, termasuk mekanisme *key expansion* pada setiap *round*.



Gambar 2.1 Struktur Dasar Algoritma AES (Abdullah, 2017)

2.2 *Advanced Encryption Standard-Galois/Counter Mode* (AES-GCM)

Menurut penjelasan NIST (*National Institute of Standard and Technology*) AES-GCM (*Advanced Encryption Standard – Galois/Counter Mode*) merupakan salah satu mode operasi dari algoritma AES yang digunakan untuk menyediakan mekanisme *authenticated encryption* atau enkripsi yang sekaligus menjamin kerahasiaan dan integritas data. Dalam skema ini, proses enkripsi dilakukan menggunakan mode *counter (CTR)* yang menghasilkan aliran kunci untuk mengenkripsi *plaintext* menjadi *ciphertext*, sedangkan proses autentikasi dilakukan menggunakan fungsi *hash* pada bidang *Galois* yang menghasilkan *authentication tag*. *Galois/Counter Mode (GCM)* merupakan mode operasi block cipher yang menyediakan layanan kerahasiaan (*confidentiality*) dan autentikasi (*authentication*) secara bersamaan. Menurut NIST SP 800-38D, GCM terdiri atas dua fungsi utama yaitu *authenticated encryption* dan *authenticated decryption*. Fungsi *authenticated encryption* digunakan untuk mengenkripsi *plaintext* menjadi *ciphertext* sekaligus menghasilkan *authentication tag*, sedangkan *authenticated decryption* digunakan untuk mendekripsi *ciphertext* dan memverifikasi keaslian data melalui *authentication tag* (Dworkin, 2007).

Kombinasi kedua mekanisme tersebut memungkinkan AES-GCM tidak hanya melindungi kerahasiaan pesan tetapi juga memastikan bahwa data tidak dimodifikasi selama proses transmisi atau penyimpanan. Mode

ini dikenal memiliki performa yang tinggi, telah dioptimalkan secara luas pada berbagai implementasi perangkat lunak maupun perangkat keras, serta banyak digunakan dalam sistem keamanan modern seperti protokol komunikasi jaringan dan layanan komputasi awan. Namun demikian, penggunaan AES-GCM memiliki beberapa persyaratan penting, salah satunya adalah keharusan penggunaan *initialization vector (IV)* atau *nonce* yang unik untuk setiap proses enkripsi, karena penggunaan kembali IV dengan kunci yang sama dapat menyebabkan kegagalan keamanan yang serius dan mengurangi jaminan keamanan dari algoritma tersebut (Kampanakis et al., 2024).

Menurut penelitian Mădălina Chiriţă (2021) AES-GCM (*Advanced Encryption Standard Galois/Counter Mode*) merupakan salah satu mode operasi paling aman yang digunakan bersama algoritma AES. Mode ini dikembangkan dari mode *Counter* (*CTR*), namun memiliki keunggulan tambahan berupa mekanisme autentikasi yang terintegrasi. Dengan kata lain, AES-GCM tidak hanya menyediakan kerahasiaan (*confidentiality*) melalui proses enkripsi, tetapi juga integritas dan autentikasi data (*authentication*) dalam satu skema yang disebut *Authenticated Encryption with Associated Data (AEAD)* (Emil Simion, Alexandru-Mihai Stroie, Mădălina Chiriţă, 2021).

Menurut Muhammad Hanif A. Nasution (2022) *Advanced Encryption Standard Galois/Counter Mode* (AES-GCM) merupakan salah satu mode operasi dari algoritma *Advanced Encryption Standard* (AES)

yang digunakan untuk meningkatkan keamanan dalam proses enkripsi data. AES sendiri merupakan algoritma kriptografi simetris yang dikembangkan untuk menggantikan *Data Encryption Standard* (DES) dan dikenal memiliki tingkat keamanan yang sangat tinggi karena menggunakan panjang kunci 128, 192, atau 256 bit serta proses enkripsi yang terdiri dari beberapa ronde transformasi pada blok data berukuran 128 bit (Hanif & Author, 2022).

AES-GCM bekerja dengan dua komponen utama, yaitu fungsi enkripsi berbasis *counter* (GCTR) dan fungsi autentikasi berbasis operasi medan Galois (GHASH). Prosesnya diawali dengan pembentukan *hash subkey* (H), yang diperoleh dengan mengenkripsi blok nol menggunakan kunci rahasia AES. Selanjutnya, dari *Initialization Vector* (IV) dibentuk *pre-counter block* yang akan digunakan sebagai nilai awal pada proses enkripsi berbasis counter. Pada tahap ini, *plaintext* diproses menggunakan fungsi GCTR untuk menghasilkan *ciphertext*. Keunikan AES-GCM terletak pada tahap autentikasinya. Selain mengenkripsi *plaintext*, mode ini juga dapat mengautentikasi *Additional Authenticated Data (AAD)*, yaitu data yang tidak dienkripsi tetapi tetap diverifikasi integritasnya (misalnya *header* paket jaringan). *Ciphertext* dan AAD diproses oleh fungsi GHASH, yang melakukan operasi polinomial dalam medan biner $GF(2^{128})$. Hasilnya kemudian diproses kembali menggunakan GCTR untuk menghasilkan *authentication tag*. *Tag* inilah yang menjadi bukti keaslian dan integritas data. Jika pada proses dekripsi *tag* tidak valid, maka sistem akan

mengembalikan status gagal (Emil Simion, Alexandru-Mihai Stroie, Mădălina Chiriţă, 2021).

2.3 ChaCha20

ChaCha20 adalah algoritma kriptografi simetris yang termasuk dalam keluarga *stream cipher* (sandi aliran), dikembangkan oleh Daniel J. Bernstein pada tahun 2008 sebagai alternatif yang lebih efisien dan aman dibandingkan AES, khususnya pada perangkat yang tidak memiliki akselerasi perangkat keras. ChaCha20 menggunakan panjang kunci 256-bit dan *nonce* 96-bit, yang dikombinasikan dengan sebuah *counter* 32-bit untuk membentuk keadaan awal (*initial state*) berukuran 16 *word* (masing-masing 32-bit). Struktur awal ini terdiri dari empat konstanta, delapan *word* kunci, satu *word counter*, dan tiga *word nonce* (Iqbal et al., 2026). Dan menurut Ahmad Miftah Fauzi (2025) dalam penelitiaanya, ChaCha20 merupakan algoritma kriptografi kunci simetris yang termasuk dalam kategori *stream cipher* dan dirancang oleh Daniel J. Bernstein sebagai pengembangan dari algoritma Salsa20. Algoritma ini bekerja dengan menghasilkan aliran kunci (*keystream*) yang kemudian dikombinasikan dengan data *plaintext* menggunakan operasi XOR untuk menghasilkan *ciphertext*. ChaCha20 menggunakan panjang kunci 256-bit dan *nonce* 96-bit yang diproses melalui serangkaian operasi aritmatika sederhana seperti penjumlahan modulo 2^{32}, operasi XOR, dan rotasi bit pada struktur matriks internal yang disebut *state*. Proses transformasi tersebut dilakukan dalam beberapa putaran (*round*) sehingga menghasilkan aliran kunci yang sulit diprediksi dan memiliki

tingkat keamanan tinggi. Selain menawarkan keamanan yang kuat, ChaCha20 juga dirancang untuk memberikan performa yang efisien pada berbagai platform komputasi, terutama pada perangkat yang tidak memiliki akselerasi perangkat keras untuk algoritma AES. Oleh karena itu, ChaCha20 banyak digunakan dalam berbagai protokol keamanan *modern* dan sering dipadukan dengan algoritma autentikasi Poly1305 untuk membentuk skema enkripsi terautentikasi yang dikenal sebagai ChaCha20-Poly1305 (Fauzi et al., 2025).

ChaCha20 bekerja dengan menghasilkan *keystream* melalui fungsi *ChaCha20_Block*, yang melakukan 20 putaran transformasi terhadap *state* awal. Setiap dua putaran terdiri dari *column round* dan *diagonal round*, yang diimplementasikan menggunakan fungsi *QuarterRound*. Fungsi ini memanfaatkan operasi aritmetika sederhana namun kuat, yaitu penjumlahan modulo 2^{32}, operasi XOR, dan rotasi bit kiri (ROTL32). Kombinasi operasi ARX (*Addition, Rotation, XOR*) tersebut dirancang untuk meningkatkan difusi dan entropi sehingga menghasilkan aliran kunci yang bersifat acak dan sulit diprediksi. Setelah 20 putaran selesai, hasil transformasi dijumlahkan kembali dengan *state* awal (*feed-forward*) untuk menghasilkan blok *keystream* sepanjang 64 *byte* (Iqbal et al., 2026). Proses enkripsi dilakukan dengan meng-XOR setiap *byte plaintext* dengan *byte keystream* yang dihasilkan. Karena ChaCha20 merupakan *stream cipher* simetris, proses dekripsi dilakukan dengan cara yang identik, yaitu meng-XOR *ciphertext* dengan *keystream* yang sama untuk memperoleh kembali

plaintext asli. Dalam *paper* tersebut juga dijelaskan contoh sederhana enkripsi kata “apa”, di mana setiap karakter dikonversi ke bentuk biner, kemudian di-XOR dengan *keystream* sehingga menghasilkan *ciphertext* dalam representasi heksadesimal (Iqbal et al., 2026).

2.4 ChaCha20 Poly-1305

ChaCha20-Poly1305 merupakan algoritma *Authenticated Encryption with Associated Data* (AEAD) yang menggabungkan algoritma *stream cipher* ChaCha20 dengan message *authentication code* Poly1305. Menurut RFC 8439, ChaCha20 digunakan untuk menyediakan kerahasiaan data melalui proses enkripsi berbasis operasi *Addition, Rotation, dan XOR (ARX)*, sedangkan Poly1305 digunakan untuk menghasilkan *authentication tag* yang menjamin integritas dan autentikasi data (Nir et al., 2018).

ChaCha20-Poly1305 merupakan algoritma kriptografi modern yang termasuk dalam kategori *Authenticated Encryption with Associated Data* (AEAD), yaitu skema enkripsi yang tidak hanya menjamin kerahasiaan data (*confidentiality*), tetapi juga menjamin integritas dan autentikasi pesan secara bersamaan. Algoritma ini merupakan kombinasi dari ChaCha20 sebagai *stream cipher* untuk proses enkripsi dan Poly1305 sebagai *message authentication code* (MAC) untuk menghasilkan *tag* autentikasi (Anggi Susanti, Bayu Ade Prasetiya, Oktafiyan Dyah Pangesti & Saputro, 2024).

ChaCha20-Poly1305 menggunakan kunci simetris 256-bit dan *nonce* 96-bit. Proses dimulai dengan menghasilkan satu blok *keystream* awal dari ChaCha20 yang digunakan sebagai kunci satu kali (*one-time key*)

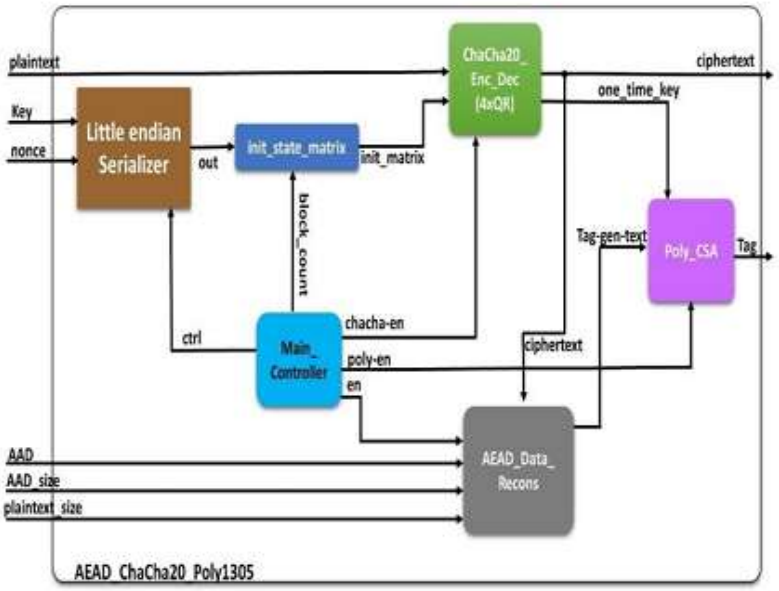
bagi algoritma Poly1305. Setelah itu, ChaCha20 mengenkripsi *plaintext* menjadi *ciphertext* melalui operasi XOR antara *plaintext* dan *keystream*. Selanjutnya, Poly1305 menghitung nilai autentikasi (*authentication tag*) berdasarkan *ciphertext* dan *associated data* (jika ada). Tag ini kemudian dikirimkan bersama *ciphertext* untuk memungkinkan proses verifikasi di sisi penerima (Anggi Susanti, Bayu Ade Prasetiya, Oktafiyan Dyah Pangesti & Saputro, 2024).

Poly1305 merupakan fungsi autentikasi sekali pakai (*one-time authenticator*) yang dirancang untuk menjamin integritas data sekaligus memverifikasi keaslian pengirim. Algoritma ini menerima input berupa kunci 32 *byte* yang digunakan hanya satu kali serta pesan dengan panjang bebas, kemudian menghasilkan tag autentikasi sepanjang 16 *byte* (128-bit). Keunggulan utama Poly1305 dibandingkan skema MAC tradisional seperti HMAC-SHA256 terletak pada komputasinya yang sangat efisien karena berbasis operasi aritmatika modular pada bilangan prima $2^{130} - 5$.

Berbeda dengan AES-GCM yang bergantung pada operasi AES dan dapat memanfaatkan akselerasi AES-NI, ChaCha20-Poly1305 dibangun menggunakan operasi ARX sehingga tidak bergantung pada instruksi AES-NI. Karakteristik ini menjadikan ChaCha20-Poly1305 sering digunakan sebagai alternatif pada perangkat yang tidak memiliki dukungan akselerasi AES berbasis perangkat keras.

Selain itu, tingkat keamanannya bersifat deterministik selama kunci yang digunakan tetap unik (Hendrawan, 2025). Gambar 2.2 menunjukkan

Arsitektur skema AEAD ChaCha20-Poly1305 yang menunjukkan alur koordinasi mekanisme otentikasi Poly1305 divisualisasikan pada gambar berikut ini:



Gambar 2.2 Arsitektur sistem AEAD ChaCha20-Poly1305 (Guard Kanda, 2025)

2.5 PDF *File*

Menurut penelitian oleh Heru Saputra (2023) pada jurnal keamanan dokumen, *Portable Document Format* (PDF) merupakan salah satu format dokumen digital yang umum digunakan untuk mendistribusikan informasi karena mampu menyimpan berbagai jenis data seperti teks, gambar, dan *metadata* dalam satu berkas dokumen elektronik. Format ini banyak digunakan dalam pertukaran dokumen digital di internet, namun juga sering dimanfaatkan sebagai media penyebaran *malware* karena sifatnya yang

fleksibel dan dapat menyimpan berbagai objek di dalam *file* tersebut.(Saputra et al., 2023)

Portable Document Format (PDF) merupakan format dokumen digital yang dikembangkan oleh Adobe untuk merepresentasikan dokumen secara konsisten dan independen terhadap perangkat, sistem operasi, maupun aplikasi pembuatnya. Format ini dirancang untuk mempertahankan struktur tata letak dokumen secara tetap (*fixed layout*), sehingga elemen seperti teks, *font*, gambar, dan grafik dapat ditampilkan secara identik pada berbagai *platform*. Secara teknis, setiap berkas PDF menyimpan deskripsi lengkap halaman dokumen dalam bentuk objek-objek terstruktur yang memungkinkan reproduksi visual yang akurat serta mendukung berbagai fitur tambahan seperti *metadata*, anotasi, formulir interaktif, tanda tangan digital, dan mekanisme enkripsi (Livathinos et al., 2021).

Dalam konteks penelitian dan pengolahan dokumen, penggunaan PDF sangat dominan karena kemampuannya menggabungkan berbagai jenis konten dalam satu berkas serta mempertahankan integritas format dokumen. Namun, kompleksitas struktur internal PDF juga menimbulkan tantangan dalam pemrosesan otomatis, ekstraksi informasi, maupun konversi dokumen. Penelitian Livathinos et al. (2021) menekankan bahwa representasi tingkat rendah dalam PDF berupa rangkaian perintah pencetakan (*printing commands*) dapat dimanfaatkan untuk merekonstruksi struktur dokumen secara lebih akurat menggunakan pendekatan pembelajaran mesin. Hal ini menunjukkan bahwa PDF bukan sekadar

format tampilan, tetapi juga mengandung struktur data yang kaya dan kompleks yang penting untuk berbagai aplikasi komputasi dokumen (Livathinos et al., 2021).

2.6 Advanced Encryption Standard Instructions (AES-NI)

AES-NI adalah set instruksi perangkat keras pada prosesor Intel dan AMD yang mempercepat proses enkripsi dan dekripsi data. Teknologi ini meningkatkan kinerja keamanan (seperti SSL/TLS) dan mengurangi beban *CPU* secara signifikan dengan menangani perhitungan kriptografi langsung di tingkat perangkat keras.

Pada tabel 2.1 menampilkan instrukti yang terdapat pada perangkat keras AES. Menurut penjelasan artikel pada *website* Intel yang ditulis oleh Jeffrey Keith Rott pada tahun 2012 dengan judul artikel ”Intel^{R} Advanced Encryption Standart Instruction (AES-NI) (Rott, 2012), AES-NI menyediakan beberapa instruksi khusus yang digunakan untuk mempercepat tahapan utama algoritma AES.

Tabel 2.1 Intruksi AES-NI

Instruksi	**Fungsi**
AESENC	Melakukan satu ronde enkripsi AES
AESENCLAST	Melakukan ronde terakhir enkripsi AES

AESDEC	Melakukan satu ronde dekripsi AES
AESDECLAST	Melakukan ronde terakhir dekripsi AES
AESKEYGENASSIST	Membantu proses pembentukan *round key*
AESIMC	Membantu transformasi *inverse mix column* pada dekripsi

Melalui instruksi tersebut, operasi seperti *SubBytes, ShiftRows, MixColumns,* dan *AddRoundKey* dapat dieksekusi secara langsung oleh *hardware* sehingga proses komputasi menjadi jauh lebih cepat dibandingkan implementasi perangkat lunak murni.

Berdasarkan pada sumber yang sama pada artikel tersebut, AES-GCM merupakan algoritma AEAD yang menggunakan AES sebagai algoritma enkripsi utama dan GHASH untuk autentikasi data. Karena proses enkripsi dan dekripsi AES-GCM bergantung langsung pada operasi AES, maka seluruh ronde AES dapat dipercepat menggunakan instruksi AES-NI. Intel melaporkan bahwa penggunaan AES-NI dapat meningkatkan performa AES antara 2 hingga 10 kali dibandingkan implementasi perangkat lunak murni, tergantung mode operasi yang digunakan. Berbeda dengan AES-GCM, ChaCha20-Poly1305 tidak menggunakan struktur internal AES. ChaCha20 dibangun menggunakan operasi matematika sederhana yang

dikenal sebagai ARX (*Addition, Rotation,* dan *XOR*), sedangkan Poly1305 digunakan untuk menghasilkan *authentication tag.*

Pada kondisi AES-NI aktif, implementasi OpenSSL memanfaatkan instruksi AES-NI yang tersedia pada prosesor sehingga operasi AES dijalankan menggunakan akselerasi perangkat keras. Sebaliknya, pada kondisi AES-NI non-aktif, penggunaan instruksi AES-NI dinonaktifkan melalui konfigurasi OpenSSL sehingga seluruh proses AES dijalankan menggunakan implementasi perangkat lunak. Dengan pendekatan tersebut, pengaruh langsung AES-NI terhadap performa AES-GCM dan ChaCha20-Poly1305 dapat diamati secara empiris. Instruksi penonaktifan AES-NI mengggunakan instruksi OpenSSL.

Tabel 2.2 instruksi penonaktifan AES-NI

```
$env:OPENSSL_ia32cap="~0x200000200000000"
```

Aktivasi AES-NI memberikan dampak langsung terhadap performa algoritma berbasis AES. AES-NI meningkatkan *throughput* karena lebih banyak data dapat diproses dalam satuan waktu tertentu. AES-NI juga menurunkan latensi atau waktu eksekusi karena sebagian besar operasi AES dijalankan langsung oleh *hardware*. serta, AES-NI mengurangi beban *CPU* karena jumlah instruksi perangkat lunak yang harus dieksekusi menjadi lebih sedikit. Intel menyatakan bahwa akselerasi AES-NI mampu memberikan peningkatan performa yang signifikan sekaligus meningkatkan efisiensi komputasi dibandingkan implementasi *software-only*.

2.7 Parameter Pengukuran Performa

Parameter pengukuran performa merupakan aspek penting dalam penelitian ini karena digunakan untuk mengevaluasi dan membandingkan efisiensi algoritma AES-GCM dan ChaCha20-Poly1305 secara objektif. Pengukuran performa dilakukan dengan pendekatan kuantitatif berdasarkan metrik yang dapat diukur secara langsung selama proses enkripsi dan dekripsi *file* PDF. Parameter yang dipilih mencerminkan kecepatan pemrosesan data serta efisiensi penggunaan *resource*, sehingga hasil evaluasi dapat memberikan gambaran yang komprehensif mengenai karakteristik kinerja kedua algoritma.

Waktu Enkripsi *(Encryption Time*), Waktu yang dibutuhkan untuk mengubah *plaintext* menjadi *ciphertext*.

$$T_{enc} = t_{finish} - t_{start}$$

Persamaan 1

Waktu Dekripsi (*Decryption Time*), Waktu yang dibutuhkan untuk mengembalikan *ciphertext* menjadi *plaintext*.

$$T_{dec} = t_{finish} - t_{start}$$

Persamaan 2

Throughput, *Throughput* menunjukkan jumlah data yang dapat diproses per satuan waktu.

$$Throughput = \frac{Ukuran\ File}{Waktu\ Proses}$$

Persamaan 3

2.8 *State Of The Art*

State of the art dalam penelitian ini mengacu pada perkembangan terkini terkait evaluasi performa algoritma kriptografi *modern*, khususnya perbandingan antara AES-GCM dan ChaCha20-Poly1305 pada berbagai konteks implementasi.

Tabel 2.3 *State Of The Art*

No.	Nama, Tahun Penelitian	Algoritma / Metode	Permasalahan	Solusi
1.	Muhammad Bagus Bintang Timur et al., (2025)(Timur et al., 2025)	AES, Blowfish, ChaCha20	Meningkatnya kebutuhan mekanisme enkripsi yang efisien dan aman untuk berbagai jenis *file* digital mendorong perlunya evaluasi performa algoritma kriptografi yang berbeda.	Melakukan evaluasi komparatif terhadap AES, Blowfish, dan ChaCha20 pada *file* gambar dan dokumen dengan mengukur waktu enkripsi/dekripsi, *CPU Usage* dan *Memory Usage*, serta analisis ketahanan kriptografi.
2.	Muhammad Patria & Dea Andini,	AES-GCM dan ChaCha20-Poly1305	Diperlukan analisis performa algoritma AEAD untuk mengetahui efisiensi pengamanan *file*	Melakukan perbandingan performa AES-GCM dan ChaCha20-Poly1305 pada *file* PDF dengan mengukur waktu eksekusi,

	(2025)(Patria & Andriati, 2025)		PDF berdasarkan waktu proses, *throughput*, dan penggunaan *resource*.	*throughput*, dan konsumsi *resource* untuk menentukan algoritma yang lebih efisien.
3.	Herman, Robby Wijaya, Kenner Farandi, Satriya Miharja, Wilson (2021)(Herman, Robby Wijaya, Kenner Farandi, Satriya Miharja, 2021)	AES vs *Stream Cipher*	Performa algoritma berbeda tergantung jenis *file* dan ukuran	Evaluasi performa berdasarkan ukuran *file*
4.	Asri Prameshwari, Nyoman Putra Sastra, (2023) (Prameshwari & Sastra, 2023)	AES-128 / AES-256	Keamanan *file* dokumen digital belum optimal dan mudah diakses tanpa proteksi	Implementasi AES pada *file* dokumen untuk menjaga kerahasiaan data

5.	I Made Chandra Widjaya & I Komang Ari Mogi, (2024)(Chandra & Ari, 2024)	ChaCha20-Poly1305 + AES-256 (*Key Derivation* SHA-256)	Dokumen UMKM rentan terhadap akses tidak sah dan pencurian data	Implementasi sistem enkripsi-dekripsi dokumen berbasis *web* menggunakan ChaCha20-Poly1305 dengan turunan kunci AES-256
6.	Fitra Rahim, Yusuf Ramadhan Nasution, Supiyandi, (2024)(Rahim & Nasution, 2024)	ChaCha20 (*Stream Cipher*)	*File* citra bitmap (BMP & PNG) tidak memiliki keamanan bawaan dan rentan terhadap akses tidak sah.	Implementasi ChaCha20 untuk enkripsi citra bitmap dan analisis performa eksekusi serta uji ketahanan terhadap *Known-Plaintext Attack.*
7.	Rebwar Khalid Muhammeda, Ribwar Rashid Azizb, Alla Ahmad Hassanb, Aso Mohammed	AES, Blowfish, Twofish, Salsa20, ChaCha20	Meningkatnya ancaman keamanan siber menuntut algoritma enkripsi yang efisien dalam waktu dan *throughput*, namun belum jelas algoritma mana yang paling optimal untuk pemrosesan data besar.	Melakukan analisis komparatif lima algoritma kriptografi menggunakan parameter waktu enkripsi/dekripsi dan *throughput* pada *file* citra berbasis Java

	Aladdinc, Shaida Jumaah Saydahd, Tarik Ahmed. Rashide, Bryar Ahmad Hassan, (2024)(Muhammed et al., 2025)			
8.	Dandi Darmansyah & Abdul Halim Hasugian, (2025)(Darmansyah & Hasugian, 2025)	ChaCha20 vs AES-CBC	AES-CBC memiliki waktu enkripsi lebih lambat dan membutuhkan padding pada komunikasi real-time	Implementasi ChaCha20 pada aplikasi *chat* Android dan analisis performa waktu eksekusi serta *throughput*
9.	Ahmad Miftah Fauzi, Arief Zulianto, Purnomo Yustianto,	AES, ChaCha20, Salsa20	Keterbatasan sumber daya mikrokontroler *IoT* (8-bit & 32-bit) dalam menjalankan algoritma enkripsi.	Implementasi dan analisis performa berdasarkan waktu eksekusi, konsumsi *RAM*, dan estimasi daya pada berbagai arsitektur mikrokontroler.

	(2025)(Fauzi et al., 2025)			
10.	Christian Justin Hendrawan, (2025)(Hendrawan, 2025)	AES-256-GCM (AEAD, *block cipher* + GCM) dan ChaCha20-Poly1305 (AEAD, *stream cipher* + MAC)	Penurunan efisiensi performa AES pada implementasi *software* murni, khususnya pada berkas *multimedia* berukuran besar dan Kebutuhan algoritma yang lebih efisien pada arsitektur tanpa akselerasi *hardware* AES	Implementasi AES-256-GCM berbasis *streaming* AEAD dengan teknik *chunking* (1 MB) untuk menjaga stabilitas *Memory* dan mengukur performa waktu, *throughput*, serta *Memory Usage* dan Implementasi ChaCha20-Poly1305 berbasis *software* murni dan dibandingkan *head-to-head* dengan AES-256-GCM pada skenario *file* 10 MB, 500 MB, dan 1 GB
11.	Jechonia M. Dungog (2025)(Dungog, 2025)	AES-256-CBC dan ChaCha20-Poly1305	AES memiliki *overhead* komputasi dan konsumsi *Memory* tinggi pada perangkat terbatas (*IoT/embedded*) dan Kebutuhan	Implementasi simulasi *Python* untuk mengukur waktu enkripsi, dekripsi, dan peak *Memory* pada berbagai ukuran *file* dan Simulasi ChaCha20 berbasis *software*

			algoritma ringan yang tetap aman untuk transmisi data real-time	untuk membandingkan performa langsung dengan AES pada kondisi identik.

2.9 Matriks Penelitian

Berdasarkan telah terhadap penelitian terdahulu, dapat disusun matriks penelitian untuk memetakan parameter yang telah diteliti serta mengidentifikasi celah penelitian yang masih terbuka.

Tabel 2.4 Matriks Peneltian

No.	Peneliti	Judul	Penerapan Algoritma		Parameter Pengukuran					*Type*	AES-NI
			AES-GCM	ChaCha20 Poly-1305	Enkripsi	Dekripsi	*Throughput*	*CPU Usage*	*Memory Usage*	*Dataset* PDF	
1.	Muhammad Bagus Bintang Timur	*Comparison of Efficiency and Security of*	✓	✓	✓	✓	✓			✓	

	Royansyah, Dewi Kusumaningsih, (2025)	*AES, Blowfish, and ChaCha20 Cryptographic Algorithms on Image and Document Files*									
2.	Muhammad Patria & Dea Andini, (2025)	Analisis Komparatif Performa AES-GCM dan ChaCha20-Poly1305 dalam Enkripsi Dokumen PDF	✓	✓	✓	✓	✓			✓	

		Berbasis AEAD									
3.	Herman, Robby Wijaya, Kenner Farandi, Satriya Miharja, Wilson (2021)	Implementasi Algoritma AES-128 Dan Sha-256 Dalam Perancangan Aplikasi Pengamanan *File* Dokumen			✓	✓					
4.	Asri Prameshwari, Nyoman Putra Sastra, (2023)	Implementasi Algoritma *Advanced Encryption Standard*			✓	✓					

		(AES) 128 Untuk Enkripsi dan Dekripsi *File* Dokumen									
5.	I Made Chandra Widjaya & I Komang Ari Mogi, (2024)	Penerapan Enkripsi dan Dekripsi Dokumen Data UMKM Menggunakan Algoritma ChaCha20-Poly1305		✓	✓	✓					
6.	Fitra Rahim, Yusuf Ramadhan	Implementasi Algoritma ChaCha20			✓	✓					

	Nasution, Supiyandi, (2024)	Pada Pengamanan *File* Citra Bitmap									
7.	Rebwar Khalid Muhammeda, Ribwar Rashid Azizb, Alla Ahmad Hassanb, Aso Mohammed Aladdinc, Shaida Jumaah Saydahd, Tarik Ahmed. Rashide, Bryar	*Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption*			✓	✓	✓				

	Ahmad Hassan, (2024)										
8.	Dandi Darmansyah & Abdul Halim Hasugian, (2025)	Enkripsi Pesan Chat Menggunakan Algoritma Chacha20 Pada Aplikasi Komunikasi *Real-Time*			✓	✓	✓			✓	
9.	Ahmad Miftah Fauzi, Arief Zulianto, Purnomo Yustianto, (2025)	Perbandingan Kinerja Algoritma Enkripsi Chacha20, Salsa20, dan			✓	✓	✓				

		Advanced Encryption Standard (AES) pada Mikrokontroler									
10.	Christian Justin Hendrawan, (2025)	Implementasi dan Analisis Komparasi Kinerja Algoritma ChaCha20-Poly1305 Terhadap AES-GCM pada Sistem Penyimpanan	✓	✓	✓	✓	✓		✓	✓	

		Berkas Multimedia									
11.	Jechonia M. Dungog (2025)	*A Python-Based Simulation and Security Analysis of ChaCha20 and AES for Secure Data Transmission in Resource-Constrained Environments*		✓	✓	✓	✓	✓			
12.	Arul Misbahul Khoer (2026)	Analisis Performa AES-	✓	✓	✓	✓	✓	✓	✓	✓	✓

		GCM Dan Chacha20-Poly1305 Pada *File* Pdf Berdasarkan Waktu, *Throughput*, Dan Penggunaan *Resource*									

2.10 Relevansi Penelitian

Relevansi penelitian terkait algoritma, metode dan topik yang diambil dimana terdapat pembaharuan dari penelitian sebelumnya, relevansi penelitian bisa dilihat pada tabel 2.3 berikut ini.

Tabel 2.5 Relevansi Penelitian

Peneliti	Muhammad Bagus Bintang Timur Royansyah, Dewi Kusumaningsih, (2025)	Muhammad Patria & Dea Andini, (2025)	Arul Misbahul Khoer (2026)
Judul	*Comparison of Efficiency and Security of AES, Blowfish, and ChaCha20 Cryptographic Algorithms on Image and Document Files*	Analisis Komparatif Performa AES-GCM dan ChaCha20-Poly1305 dalam Enkripsi Dokumen PDF Berbasis AEAD	Analisis Performa AES-GCM Dan Chacha20-Poly1305 Pada *File* PDF Berdasarkan Waktu, *Throughput*, Dan Penggunaan *Resource*
Masalah Penelitian	Meningkatnya kebutuhan mekanisme enkripsi yang efisien dan aman untuk berbagai jenis *file* digital mendorong perlunya evaluasi performa algoritma kriptografi yang berbeda.	Diperlukan analisis performa algoritma AEAD untuk mengetahui efisiensi pengamanan *file* PDF berdasarkan waktu proses, *throughput*, dan penggunaan *resource*	Belum adanya analisis komparatif yang lebih terstruktur dan terkontrol pengaruh akselerasi AES-NI terhadapa performa AES-GCM dan ChaCha20-Poly1305 pada *file* PDF dengan parameter

			waktu eksekusi, *throughput*, dan penggunaan *resource.*
Objek Penelitian	*Image* (*JPG dan PNG*), *Document* (*PDF dan DOCX*)	PDF	PDF
Algoritma/Metode	AES, Blowfish, ChaCha20	AES-GCM dan ChaCha20-Poly1305	AES-GCM dan ChaCha20-Poly1305
Dataset	*File* gambar dan dokumen dengan variasi ukuran	*File* PDF dengan variasi ukuran tertentu	*File* PDF ukuran 10 MB sampai 100 MB dengan jarak interval ukuran *file* PDF sebesar 10 MB.
Implementasi	Implementasi aplikasi enkripsi-dekripsi dan pengujian waktu eksekusi serta tingkat keamanan	Implementasi sistem enkripsi AEAD dengan pengukuran waktu, *throughput*, dan *resource*	Implementasi sistem pengujian eksperimen terkontrol dengan aktivasi dan deaktivasi AES-NI dan pengukuran waktu enkripsi, dekripsi, *throughput*, serta *CPU Usage* dan *Memory Usage.*