SENAFTI
SEMINAR NASIONAL MAHASISWA
FAKULTAS TEKNOLOGI INFORMASI

6th Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)
20 September 2025 – Jakarta, Indonesia
Volume 4, Nomor 2, September 2025 - ISSN 2962-8628 (*online*)

IMPLEMENTASI *HYBRID ENCRYPTION* ECC-AES UNTUK PENGAMANAN KOMUNIKASI DAN BERBAGI FILE BERBASIS WEB

Risqi Rahman Pratama[1*], Dolly Virgian Shaka Yudha Sakti[2]

[1,2] Teknik Informatika, FakultasTeknologi Informasi, Universitas Budi Luhur, Jakarta, Indonesia

Email: [1*]2111510596@student.budiluhur.ac.id, [2]dolly.virgianshaka@budiluhur.ac.id
(* : corresponding author)

Abstrak-Perkembangan komunikasi digital menuntut mekanisme pengamanan yang efektif untuk pesan dan pertukaran *file*, terutama pada aplikasi web. Penelitian ini bertujuan mengimplementasikan skema *hybrid encryption* berbasis *Advanced Encryption Standard* (AES) 256-bit dalam mode *Galois/Counter Mode* (GCM) dan *Elliptic Curve Cryptography* (ECC) X25519 untuk menjamin kerahasiaan serta integritas data. Metode penelitian mencakup penerapan kombinasi algoritma tersebut, validasi fungsi enkripsi dan dekripsi menggunakan *test vector* standar internasional, serta uji coba pada modul *chatting* dan berbagi *file* berukuran 4–41 MB. Hasil validasi menunjukkan bahwa algoritma AES-256-GCM menghasilkan data terenkripsi sesuai spesifikasi, sementara ECC X25519 mampu mendistribusikan kunci dengan aman dan efisien. Uji performa sistem memperlihatkan bahwa pesan teks dapat dienkripsi dan didekripsi secara *real-time* dengan latensi rata-rata sekitar 45 milidetik, sedangkan *file* berukuran menengah hingga besar berhasil dienkripsi dengan durasi 38–332 detik, dan proses dekripsi tetap reliabel melalui mekanisme pemrosesan *file* per bagian (*chunking*) meskipun terbatas oleh waktu eksekusi *server*. Integritas data terjaga, karena setiap perubahan pada *ciphertext* terdeteksi melalui mekanisme autentikasi. Kesimpulannya, implementasi *hybrid* AES-256-GCM dan ECC X25519 berhasil diterapkan pada aplikasi web nyata, menyediakan pengamanan yang efektif untuk komunikasi pesan dan pertukaran *file*, serta relevan untuk diterapkan pada sistem *real-time*. Penelitian ini juga memberikan kontribusi praktis sebagai bukti bahwa *hybrid encryption* dapat langsung digunakan pada aplikasi berbasis web, sementara optimalisasi performa untuk *file* berukuran sangat besar dan pengujian *multi-user* menjadi arah pengembangan selanjutnya.

Kata Kunci: Hybrid encryption, AES-256-GCM, ECC X25519, keamanan komunikasi, enkripsi file.

IMPLEMENTATION OF ECC-AES HYBRID ENCRYPTION FOR SECURING WEB-BASED COMMUNICATION AND FILE SHARING

Abstract-*The advancement of digital communication demands effective security mechanisms for messages and file exchanges, particularly in web-based applications. This study aims to implement a hybrid encryption scheme combining Advanced Encryption Standard (AES) 256-bit in Galois/Counter Mode (GCM) and Elliptic Curve Cryptography (ECC) X25519 to ensure data confidentiality and integrity. The research method involves applying this algorithm combination, validating encryption and decryption functions using international standard test vectors, and testing on web-based chat modules and file-sharing systems with file sizes ranging from 4 to 41 MB. Validation results indicate that AES-256-GCM produces encrypted data according to specifications, while ECC X25519 securely and efficiently handles key distribution. Performance tests show that text messages can be encrypted and decrypted in real-time with an average latency of approximately 45 milliseconds, whereas medium to large files are successfully encrypted within 38–332 seconds. Decryption remains reliable through chunked file processing, despite server execution time limitations. Data integrity is maintained, as any modifications to ciphertext are detected via the authentication mechanism. In conclusion, the hybrid AES-256-GCM and ECC X25519 implementation is successfully applied in a real web application, providing effective security for message communication and file sharing, and is suitable for real-time systems. This study also offers practical evidence that hybrid encryption can be directly deployed in web applications, while further optimization for very large files and multi-user scenarios remains a direction for future development.*

Keywords*: Hybrid encryption,* AES-256-GCM, ECC X25519*, secure communication, file encryption.*

1. PENDAHULUAN

Perkembangan teknologi informasi yang semakin pesat telah memudahkan komunikasi digital dan pertukaran data secara *real-time*. Aplikasi *chatting* dan berbagi *file* menjadi salah satu sarana utama dalam menunjang kolaborasi, baik di sektor profesional maupun personal. Namun, kemudahan ini membawa konsekuensi serius terhadap aspek keamanan informasi. Risiko seperti penyadapan, manipulasi pesan, dan pencurian *file* sensitif semakin meningkat, terutama jika komunikasi dilakukan tanpa mekanisme enkripsi yang memadai [1], [2].

Salah satu pendekatan yang banyak digunakan untuk mengatasi permasalahan ini adalah penerapan kriptografi modern. *Algoritma Advanced Encryption Standard* (AES) dalam mode *Galois/Counter Mode* (GCM)

dengan panjang kunci 256 bit banyak diadopsi karena mampu memberikan kecepatan enkripsi sekaligus memastikan autentikasi data melalui *authentication tag* [3], [4]. Sedangkan *Elliptic Curve Cryptography* (ECC) menjadi pilihan populer untuk kriptografi asimetris karena tingkat keamanan tinggi dengan ukuran kunci yang relatif kecil. Varian *Curve*25519 dengan algoritma pertukaran kunci X25519 direkomendasikan oleh standar internasional [5] dan telah terbukti efisien pada sistem berbasis web maupun perangkat *Internet of Things* [6].

Sebagian penelitian terdahulu hanya menggunakan algoritma tunggal, seperti AES untuk pengamanan data medis [4] atau ECC untuk sistem berbasis web [7]. Ada pula yang mengusulkan kombinasi hybrid AES–ECC [8], [9] tetapi masih terbatas pada simulasi atau diarahkan ke konteks lain seperti penyimpanan awan (*cloud storage*). Belum ditemukan penelitian yang secara khusus menerapkan *hybrid encryption* AES-256-GCM dan ECC X25519 pada aplikasi *chatting* dan berbagi *file* berbasis web secara nyata.

Tujuan penelitian ini adalah mengimplementasikan *hybrid encryption* berbasis AES-256-GCM dan ECC X25519 pada aplikasi komunikasi berbasis web, memvalidasi kebenaran implementasi algoritma menggunakan *test vector* resmi (RFC 7748 *Section* 6.1 untuk ECC X25519 [10] dan NIST SP 800-38D *Test Case* 14 untuk AES-256-GCM [11]), serta menilai efektivitas penerapannya dalam skenario komunikasi *real-time* dan berbagi *file*.

2. METODE PENELITIAN

Penelitian ini termasuk kategori penelitian terapan, karena berfokus pada penerapan algoritma kriptografi modern dalam sebuah sistem komunikasi berbasis web. Model enkripsi yang digunakan adalah *hybrid encryption*, yaitu kombinasi antara algoritma simetris AES-256-GCM dan algoritma asimetris ECC X25519.

2.1 Hybrid Encryption

Model enkripsi yang digunakan dalam penelitian ini adalah *hybrid encryption*, yaitu penggabungan algoritma simetris AES-256-GCM dengan algoritma asimetris ECC X25519.

AES-256-GCM merupakan algoritma simetris berbasis blok cipher dengan panjang kunci 256 bit. Mode GCM (*Galois/Counter Mode*) bekerja menggunakan counter untuk proses enkripsi dan dekripsi. Mekanisme ini membuat proses enkripsi dan dekripsi berjalan dengan cara yang mirip, hanya berbeda pada input yang digunakan: enkripsi dilakukan dengan plaintext sebagai masukan, sedangkan dekripsi menggunakan *ciphertext*. Selain menghasilkan *ciphertext*, GCM juga memproduksi *authentication tag* yang berfungsi sebagai verifikasi integritas data [4].

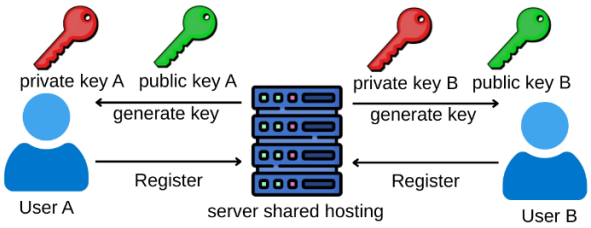
Sementara itu, ECC X25519 digunakan untuk pertukaran kunci. Algoritma ini beroperasi melalui proses *clamping* pada kunci privat untuk memastikan nilai yang valid, lalu dilanjutkan dengan *scalar multiplication* terhadap kunci publik lawan. Hasilnya adalah *shared secret* yang identik di kedua pihak, yang kemudian dimanfaatkan untuk mengamankan kunci AES [7], [12].

Dengan pendekatan ini, data pesan maupun *file* diamankan oleh AES-256-GCM, sementara distribusi kunci AES dilindungi melalui mekanisme ECC X25519. Skema kombinasi ini memungkinkan sistem berfungsi secara efisien sekaligus memberikan jaminan keamanan sesuai praktik terbaik kriptografi modern [8], [9].

2.2 Tahapan Implementasi

Implementasi *hybrid encryption* pada penelitian ini dilakukan melalui beberapa tahapan berurutan, mulai dari pembangkitan kunci, pembentukan *shared secret*, hingga proses enkripsi dan dekripsi pesan maupun *file*. Tahapan tersebut dijelaskan secara rinci berikut ini:

a. Inisialisasi Kunci ECC
   Pada tahap registrasi, setiap pengguna menghasilkan pasangan kunci privat dan publik menggunakan algoritma X25519. Proses dimulai dengan pembangkitan nilai acak sepanjang 32 byte, kemudian dilakukan proses *clamping* agar sesuai dengan spesifikasi keamanan X25519 menurut RFC 7748 [10]. Hasil *clamping* menjadi kunci privat, sedangkan kunci publik diperoleh melalui operasi *scalar multiplication* antara kunci privat dengan titik dasar (*base point*) kurva elliptic (u = 9 pada *Curve*25519). Alur proses pembangkitan dan penyimpanan kunci ini ditunjukkan pada Gambar 1.



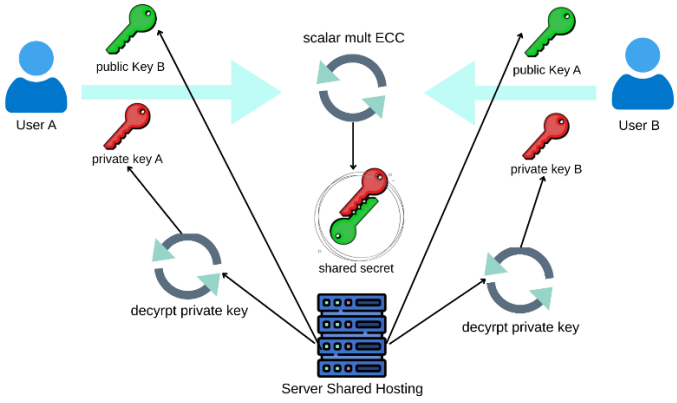
Gambar 1. Proses inisialisasi kunci privat dan publik ECC X25519 pada tahap registrasi pengguna

Kunci publik disimpan dan dapat dipertukarkan antar pengguna untuk keperluan pertukaran kunci, sementara kunci privat tetap dirahasiakan dengan cara disimpan dalam basis data dalam bentuk terenkripsi menggunakan AES-256-GCM pada sisi server.

b. Pembentukan *Shared Secret*

Pada tahap pertukaran kunci, ketika pengguna A ingin mengirim pesan kepada pengguna B, maka pengguna A menggunakan kunci privat miliknya dan kunci publik milik B untuk melakukan *scalar multiplication* sesuai algoritma X25519. Sebaliknya, pengguna B dapat melakukan proses yang sama dengan menggunakan kunci privat miliknya dan kunci publik milik A.

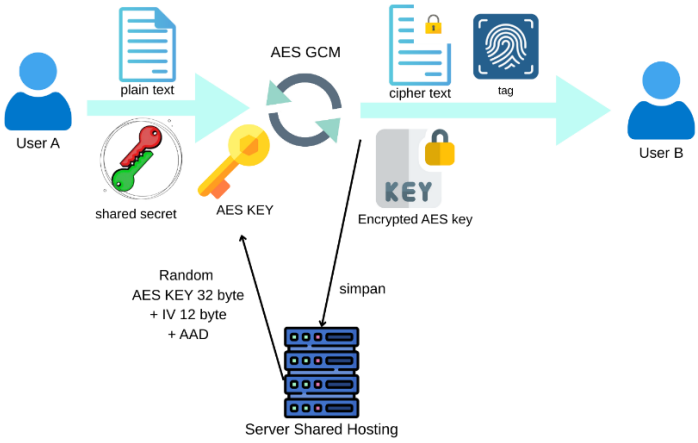
Kedua perhitungan tersebut menghasilkan nilai yang identik, yaitu *shared secret*, meskipun dilakukan secara independen di sisi masing-masing pengguna. *Shared secret* ini kemudian digunakan sebagai untuk enkripsi kunci AES. Alur pertukaran kunci dan pembentukan *shared secret* ditunjukkan pada Gambar 2.



Gambar 2. Pembentukan shared secret melalui pertukaran kunci publik dan privat antar pengguna

c. Proses Enkripsi Pesan/*File*

Setelah *shared secret* terbentuk, langkah berikutnya adalah melakukan enkripsi terhadap data aktual (pesan atau *file*) menggunakan algoritma AES-256-GCM. Proses enkripsi pesan/*file* ditunjukkan pada Gambar 3.



Gambar 3. Proses Enkripsi *Plaintext* dan Kunci AES

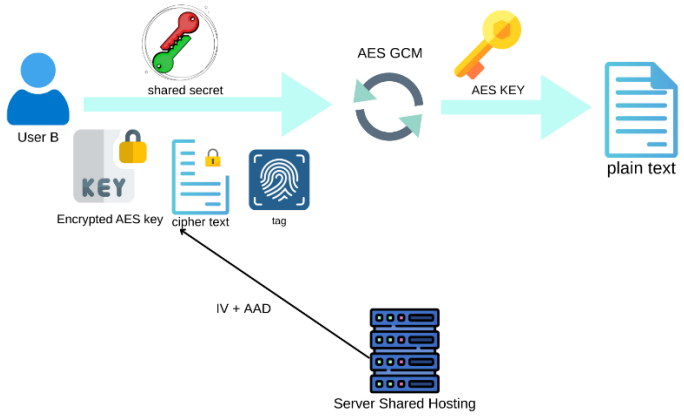
Proses ini berjalan sebagai berikut:

1. Input *plaintext* (pesan/*file*), kunci AES-256 (dibangkitkan secara acak 32 byte), IV (*nonce* 12 byte), serta data tambahan opsional AAD (*Additional Authenticated Data*).
2. Algoritma AES-256-GCM melakukan enkripsi dengan mode *counter* (CTR) yang menghasilkan *ciphertext* sekaligus membangkitkan *authentication tag* yang berfungsi sebagai kode autentikasi yang diverifikasi di sisi penerima untuk menjamin integritas dan keaslian data, sehingga apabila terjadi modifikasi sekecil apa pun pada *ciphertext*, dekripsi akan gagal.
3. Distribusi kunci AES-256 yang digunakan tidak dikirim secara langsung. Kunci tersebut terlebih dahulu dienkripsi menggunakan *shared secret* dengan AES-256-GCM, menghasilkan *encrypted* AES key. *Encrypted* key ini dikirim bersamaan dengan *ciphertext* dan *authentication tag*.

Dengan cara ini, penerima yang memiliki *shared secret* yang sama dapat mendekripsi *encrypted* AES key, mendapatkan kembali kunci AES-256, lalu menggunakan kunci tersebut untuk membuka *ciphertext* menjadi *plaintext*.

d. Proses Dekripsi Pesan/*File*

Selanjutnya di sisi penerima proses dekripsi ditunjukkan pada Gambar 4.



Gambar 4. Proses dekripsi *ciphertext*

Proses dekripsi dilakukan secara kebalikan dari enkripsi:

1. Penerima memperoleh *ciphertext*, *encrypted* AES key, *authentication tag*, serta parameter IV dan AAD dari pengirim.
2. Encrypted AES key didekripsi menggunakan *shared secret* hasil pertukaran kunci ECC X25519, sehingga diperoleh kembali kunci AES-256.
3. *Ciphertext* didekripsi menggunakan kunci AES-256 tersebut dengan algoritma AES-256-GCM.
4. *Authentication tag* diverifikasi. Jika tag valid, plaintext asli berhasil dipulihkan; jika tidak valid, pesan dianggap rusak atau telah dimanipulasi.

Dengan demikian, penerima hanya dapat membuka pesan apabila memiliki *shared secret* yang benar dan *authentication tag* valid, sehingga kerahasiaan serta integritas data tetap terjamin.

e. Ringkasan Proses

Secara keseluruhan, tahapan implementasi *hybrid encryption* dalam sistem ini dapat diringkas yaitu:

1. Inisialisasi kunci terjadi pada saat registrasi, setiap pengguna membangkitkan pasangan kunci privat dan publik ECC X25519. Kunci privat disimpan dalam database dalam bentuk terenkripsi AES-256-GCM, dan saat login kunci privat didekripsi ke dalam session pengguna.
2. Pertukaran kunci publik antar pengguna melalui *server*.
3. Pembentukan *shared secret* melalui operasi *scalar multiplication* antara kunci privat milik sendiri dan kunci publik milik lawan komunikasi.
4. Enkripsi pesan/*file* menggunakan kunci AES-256-GCM yang dibangkitkan secara acak (32 byte AES key + 12 byte IV + AAD).
5. Kunci AES-256 kemudian dienkripsi menggunakan *shared secret* dengan AES-GCM.
6. Pengiriman *ciphertext*, *authentication tag*, serta *encrypted* AES key ke penerima.

7. Penerima mendekripsi *encrypted* AES key menggunakan *shared secret*, sehingga mendapatkan kembali kunci AES-256.
8. *Ciphertext* didekripsi dengan kunci AES-256-GCM, dan *plaintext* asli berhasil dipulihkan jika *authentication tag* valid.

2.3 Pengujian dan Validasi

Pada bagian ini dijelaskan pendekatan pengujian yang digunakan untuk memastikan sistem *hybrid encryption* bekerja sesuai standar kriptografi modern. Pengujian dilakukan dengan dua pendekatan utama, yaitu validasi algoritma dan uji program.

a. Validasi Algoritma
   1. ECC X25519 divalidasi menggunakan *test vector* resmi dari RFC 7748.
   2. AES-256-GCM divalidasi menggunakan *test vector* dari NIST SP 800-38D.

   Kesesuaian hasil enkripsi/dekripsi dengan nilai acuan menjadi indikator bahwa fungsi kriptografi dasar telah diimplementasikan dengan benar.

b. Uji Program
   Pengujian program dilakukan pada modul *chatting* dan penyimpanan *file* dalam sistem *hybrid encryption* pada web:
   1. Jenis pesan: teks pendek (≤100 karakter) dan teks panjang (≥1.000 karakter).
   2. *File*: kecil (≤5 MB) dan (10–40 MB).

   Parameter yang dicatat:
   1. Durasi proses (enkripsi dan dekripsi per pesan maupun per *file*).
   2. Kesesuaian *ciphertext* (apakah *file*/pesan berhasil dipulihkan 100% menjadi *plaintext* asli).
   3. Keutuhan data (apakah *authentication tag* valid atau gagal diverifikasi).
   4. Kinerja sistem (latensi *chatting real-time*, kecepatan unduhan *file* terenkripsi dengan *chunk*).

   Kriteria keberhasilan:
   1. Pesan dan *file* berhasil dipulihkan dengan *plaintext* asli tanpa error.
   2. *Authentication tag* valid, menandakan integritas data terjaga.
   3. Sistem tetap dapat beroperasi meskipun terdapat keterbatasan infrastruktur (misalnya limit eksekusi 120 detik pada *shared hosting*).

3. HASIL DAN PEMBAHASAN

Pada bagian ini berisi analisis, hasil implementasi ataupun pengujian serta pembahasan dari topik penelitian, yang bisa dibuat terlebih dahulu metodologi penelitian. Bagian ini juga merepresentasikan penjelasan yang berupa penjelasan, gambar, tabel dan lainnya.

3.1 Data Pengujian

Pengujian dilakukan menggunakan data simulasi yang menyerupai pesan teks dan *file* dokumen sebagaimana aktivitas komunikasi internal perusahaan. Jenis data dan tujuannya dirangkum pada Tabel 1. Pengujian awal dilakukan untuk memastikan implementasi kriptografi sesuai dengan standar internasional.

Tabel 1. Data Pengujian

No	Nama File	Jenis File	Ukuran	Fungsi Pengujian
1	Pesan Chat Simulasi	Plaintext, JPG, XLSX, DOCX	-	Uji komunikasi *real-time* dan enkripsi pesan
2	Data Kupon A	XLSX	11 MB	Uji enkripsi file kecil
3	Data Kupon B	XLSX	25 MB	Uji performa enkripsi file sedang
4	Data Kupon C	XLSX	41 MB	Uji integritas data dengan AES-GCM
5	*Test Vector* AES-GCM	Hexadecimal	-	Validasi enkripsi AES-GCM (NIST)
6	*Test Vector* X25519	Hexadecimal	-	Validasi *scalar multiplication* (RFC 7748)

3.2 Validasi Algoritma

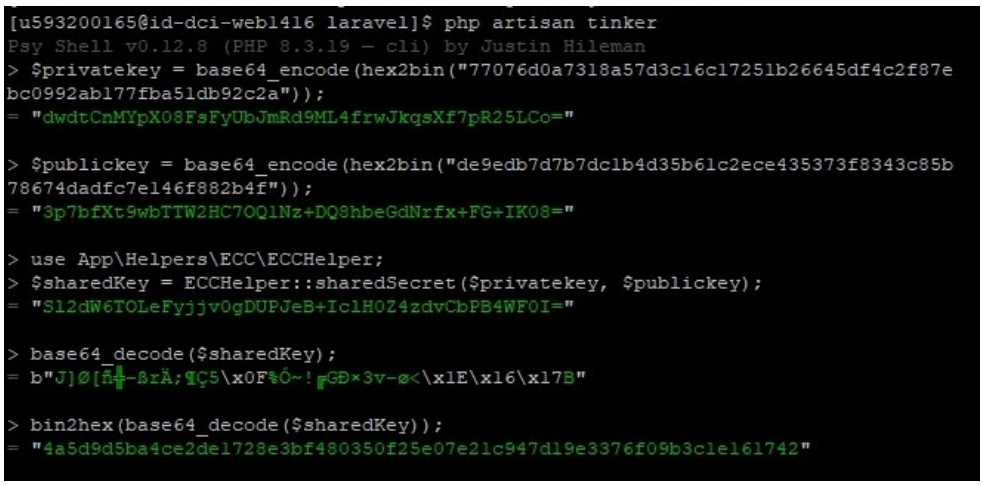
Pengujian Pengujian awal dilakukan untuk memastikan bahwa implementasi kriptografi pada sistem sesuai dengan acuan standar internasional. Dua algoritma yang divalidasi adalah *Elliptic Curve Diffie–Hellman* X25519 dan AES-256-GCM.

a. Validasi ECC X25519

Validasi algoritma dilakukan menggunakan *test vector* resmi dari RFC 7748 *Section* 6.1. Pada uji coba ini, digunakan kunci privat milik Alice dan kunci publik milik Bob, yang terlebih dahulu dikonversi dari heksadesimal ke biner, kemudian diubah ke format Base64 agar sesuai dengan fungsi helper dalam sistem. Proses *scalar multiplication* menghasilkan *shared secret* outputnya :
4a5d9d5ba4ce2de1728e3bf480350f25e07e21c947d19e3376f09b3c1e161742
Nilai tersebut identik dengan hasil yang tercantum dalam dokumen RFC 7748. Hal ini menunjukkan bahwa implementasi X25519 pada sistem telah memenuhi kriteria berikut:

1. Proses *clamping* terhadap kunci privat berjalan sesuai aturan.
2. *Decoding little-endian* dilaksanakan dengan benar.
3. *Shared secret* yang dihasilkan bersifat simetris pada kedua belah pihak.

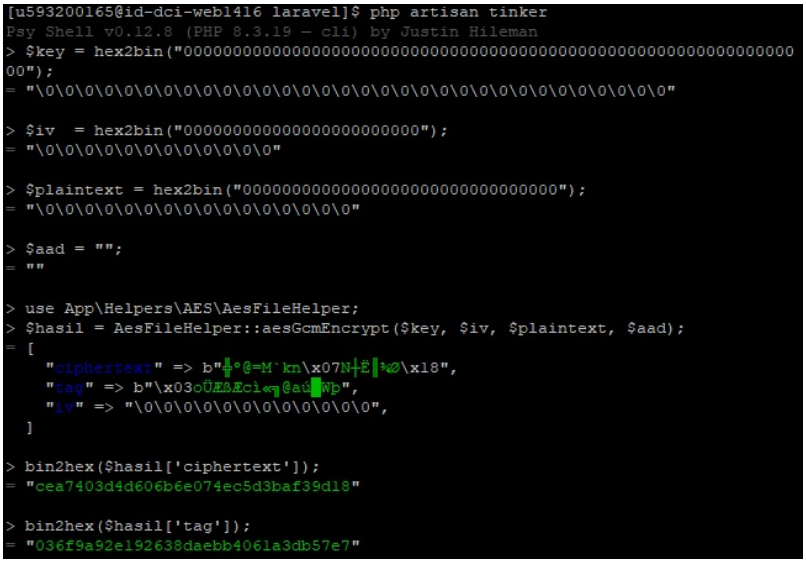
Dengan demikian, mekanisme pertukaran kunci dapat dinyatakan valid dan aman untuk digunakan dalam skema enkripsi. Hasil pengujian divisualisasikan pada Gambar 5.



Gambar 5. Hasil validasi algoritma ECC X25519 menggunakan *test vector* RFC 7748 pada terminal Laravel Tinker

b. Validasi AES-256-GCM

Validasi AES-256-GCM dilakukan menggunakan *test vector* dari NIST GCM Specification, *Test Case* 14 (SP 800-38D). Parameter yang dipakai berupa kunci 256-bit bernilai nol, IV bernilai nol, dan plaintext sepanjang 16 byte nol. Proses enkripsi melalui fungsi *aesGcmEncrypt()* menghasilkan *ciphertext* berikut: cea7403d4d606b6e074ec5d3baf39d18 dan *authentication tag*: 036f9a92e192638daebb4061a3db57e7
hasil validasi AES-256-GCM dapat dilihat pada Gambar 6.



Gambar 6. Hasil validasi algoritma AES-256-GCM menggunakan *test vector* pada terminal Laravel Tinker

Jika dibandingkan dengan nilai acuan, *ciphertext* sesuai dengan hasil standar, sedangkan *authentication tag* terdapat perbedaan kecil. Variasi ini diduga dipengaruhi oleh mekanisme pembangkitan *tag* pada pustaka PHP, khususnya dalam pengolahan *Additional Authenticated Data* (AAD) yang kosong. Meskipun demikian, kesesuaian hasil *ciphertext* membuktikan bahwa fungsi inti AES-GCM telah berjalan benar. Sistem tetap dapat menjamin integritas data karena *tag* autentikasi mampu mendeteksi perubahan pada *ciphertext*.

c. Kesimpulan Validasi

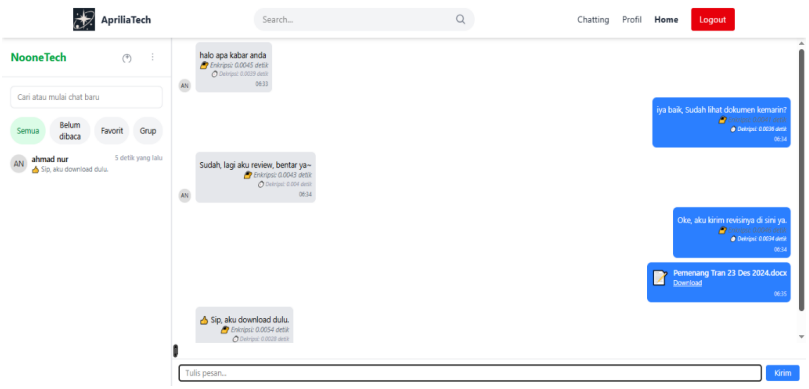
Berdasarkan kedua pengujian tersebut, dapat disimpulkan implementasi ECC X25519 sesuai dengan *test vector* resmi dari RFC 7748, sehingga terbukti valid. AES-256-GCM menghasilkan *ciphertext* yang konsisten dengan spesifikasi NIST, meskipun *authentication tag* sedikit berbeda. Dengan hasil ini, sistem *hybrid encryption* dapat dinyatakan kompatibel dengan standar internasional dan layak digunakan sebagai mekanisme pengamanan komunikasi berbasis web.

3.3 Pengujian Program

Pengujian Setelah validasi menggunakan *test vector* membuktikan keakuratan algoritma, tahap selanjutnya adalah pengujian program secara langsung pada antarmuka aplikasi. Tujuan pengujian ini adalah memastikan bahwa sistem mampu menjalankan proses enkripsi dan dekripsi pada data nyata, baik pesan maupun *file*, dengan efisiensi dan reliabilitas yang dapat diterima.

a. Pengujian Modul *Chatting*

Aplikasi mendukung mekanisme *backend-to-end encryption* pada fitur *chatting*. Setiap pesan yang dikirimkan oleh pengguna terlebih dahulu dienkripsi di sisi pengirim, kemudian didekripsi kembali di sisi penerima menggunakan *shared secret* hasil pertukaran ECC X25519. Pengujian ini di visualisasikan pada Gambar 7.



Gambar 7. Pengujian modul *chatting* dengan stempel waktu enkripsi/dekripsi per pesan.

Pada Gambar 7, terlihat bahwa setiap pesan menampilkan informasi durasi proses enkripsi dan dekripsi dalam satuan detik. Nilai yang ditampilkan relatif sangat kecil (misalnya 0,045 detik), menunjukkan bahwa proses kriptografi terjadi secara *real-time* tanpa menimbulkan latensi berarti. Dengan demikian, dapat disimpulkan bahwa penerapan *hybrid encryption* (AES-GCM + ECC X25519) dapat digunakan dalam komunikasi aktif tanpa mengurangi kenyamanan pengguna.

b. Pengujian Enkripsi dan Dekripsi *File*

Selain pengujian pada pesan teks, modul penyimpanan *file* juga diuji untuk mengukur kinerja enkripsi dan dekripsi pada data berukuran besar. Transparansi sistem diperkuat dengan penampilan waktu proses langsung pada antarmuka aplikasi, sehingga pengguna dapat mengetahui durasi enkripsi maupun dekripsi setiap *file*. Pada Gambar 8 ditunjukkan tabel penyimpanan *file* yang berisi informasi ukuran *file*, durasi enkripsi, dan durasi dekripsi.

Penyimpanan Saya

Nama File	Ukuran	Dibuat	Durasi Enkripsi	Durasi Dekripsi	Kode Share	Key	Aksi
epn_Januari_februari_Mar.xlsx	41,592.86 KB	33 menit yang lalu	332.834 detik	121.104 detik	-	-	Unduh Bagikan Hapus
EPN Data Raw Jan-Mar 2024 v0.1 21102024.xlsx	25,958.74 KB	44 menit yang lalu	209.402 detik	121.110 detik	-	-	Unduh Bagikan Hapus
data.xlsx	12,799.41 KB	46 menit yang lalu	136.590 detik	121.103 detik	-	-	Unduh Bagikan Hapus
DCM_Jobdesc_DBA.pptx	4,847.40 KB	1 jam yang lalu	38.682 detik	68.850 detik	-	-	Unduh Bagikan Hapus

Gambar 8. Pengujian penyimpanan file dan durasi proses di aplikasi web

Hasil uji menunjukkan bahwa:

1. *File* berukuran 41 MB membutuhkan waktu enkripsi 332 detik dan dekripsi berhenti 121 detik.
2. *File* berukuran 25 MB membutuhkan waktu enkripsi 209 detik dan dekripsi berhenti 121 detik.
3. *File* berukuran 12 MB membutuhkan waktu enkripsi 136 detik dan dekripsi berhenti 121 detik.
4. *File* berukuran kecil (sekitar 4 MB) hanya membutuhkan enkripsi 38 detik dan dekripsi 68 detik.
5. Durasi enkripsi meningkat sebanding dengan ukuran *file*, sedangkan proses dekripsi lebih lambat karena membuat tag dan membandingkan dengan tag di *database* untuk verifikasi tag.

Selain itu, hasil pengujian memperlihatkan peran penting mekanisme *chunking* dan *streaming*. Setiap *file* dipecah menjadi potongan (*chunk*) berukuran 512 KB agar dapat diproses secara bertahap. Hal ini dilakukan untuk mengatasi batasan eksekusi pada *shared hosting* (misalnya limit waktu PHP), serta memungkinkan pemrosesan *file* besar tanpa perlu memori besar sekaligus. Setiap *chunk* disimpan dalam database bersama parameter kriptografi yang diperlukan, yaitu *initialization vector* (IV) dan *authentication tag*. Contoh penyimpanan hasil enkripsi ditunjukkan pada Gambar 9, di mana terlihat bahwa:

id	message_id	chunk_index	chunk_data	iv	tag	created_at	updated_at
9	11	0	n/bUbbtmr3uL66hQUyKF6BAyDdFyQJeVMfx8FItHruwBNy1wkT...	ut36qGQ4xvwv7ZUs	qrDDWbc3JxXtSEakf4fT3g==	2025-07-08 10:05:11	2025-07-08 10:05:11
10	12	0	Hi09SiKCN4DZ7WAK9MQ3M0FznxiWO/35Ct9HQFlaaWCnffxvbC...	apIYpIUyRV+sGwGe	cv2kFluxXz4qhoCjmJoTYA==	2025-07-08 10:08:11	2025-07-08 10:08:11
11	13	0	NEXfwTq5HJI0FavUNS1xddRwFmLc1JuEDt9sp5n2iQrMhvRIjM...	UPbnm3mBghhh5Q80	iE9E0zATwC2fLM+U4hflbg==	2025-07-08 10:09:15	2025-07-08 10:09:15
12	13	1	4IBwb0u2TD3ase1hdLYOcrwHNP8pURFrJ555w0GF+WQQIGCpdD...	/MGN2k95u8m3AQfS	WHvzCfauexulatwxPHYEew==	2025-07-08 10:09:16	2025-07-08 10:09:16
13	13	2	RCJ49eY9OB4QRn7Ri7ix2Cbq7Vc+PHSuokqyjK/Z/ISivgpNtH...	yvQg0pQicl87/3JO	gl6S5+E1tgoJgfJQw8CZxg==	2025-07-08 10:09:16	2025-07-08 10:09:16
14	13	4	8Fp/0+mzq8K9HvUGflbpOnZgwpi16+0rci3g+PROVnbwwS49CR...	uqORaNqx69XU7pP0	rvcMlqbmBJ26jnPrHEmY6g==	2025-07-08 10:09:27	2025-07-08 10:09:27
15	13	3	QsfT7hVad1tT7ST3Dry/c9z77SITh85qMfwxBDO5ic20wuctts...	I5xYx8kscEB4ybSW	dUuPn00QNdMm0qHfeU+J1A==	2025-07-08 10:09:27	2025-07-08 10:09:27
16	13	5	HWR8I1RqUc9UIzI0CfnP7xCkxyjcp99IJ0YS2vtP/ltcvsA6fr...	JGh5Vih0EnusYfpz	0KWyOiIHMJbHSiLQoOxzsQ==	2025-07-08 10:09:28	2025-07-08 10:09:28
17	13	8	kieRHQwBsG3A7OFI8hvCsIUQVvXUA3djhJGrighpRpg1FJ37Ug...	bqer7B2CaoRICD/Z	y02DFV0AJXIGW2pQUbqAiw==	2025-07-08 10:09:37	2025-07-08 10:09:37
18	13	6	EJCMc2nSTaIbJgYF9Kli109eKw6FtoUZ1AIMYrPm+afZmLH2Cz...	0B8bSRaIBKzOZ1tT	ST3t/9KJoE9FaVt0dLpcrw==	2025-07-08 10:09:39	2025-07-08 10:09:39
19	13	7	aQVpOMvjKi9x9wBPcrQsPyrGvNIK2SlpMGSkPtNRYUMGJ161tw...	af+zJWD7IY5T5K12	NVdW0ufXBcTRgDbHZUaoOQ==	2025-07-08 10:09:39	2025-07-08 10:09:39

Gambar 9. Hasil penyimpanan *ciphertext* pada *file* dengan *chunk* di *database*

1. Kolom *chunk_data* berisi *ciphertext* hasil enkripsi AES-256-GCM.
2. Kolom iv menyimpan nonce unik untuk setiap chunk, yang dibutuhkan untuk proses dekripsi.
3. Kolom *tag* berisi *authentication tag* yang digunakan untuk menjamin integritas data.

4. Penyimpanan data dalam bentuk terenkripsi memastikan bahwa meskipun *database* berhasil diakses pihak tidak berwenang, isi *file* tetap tidak dapat dibaca tanpa kunci dekripsi yang sah.

Mekanisme dekripsi juga dioptimalkan dengan dukungan HTTP *Range Request* (status 206 *Partial Content*). Server hanya mendekripsi *chunk* sesuai rentang byte yang diminta oleh client, kemudian mengirimkannya secara bertahap melalui koneksi TCP yang tetap terbuka. Dengan pendekatan ini:

1. Dekripsi berjalan *on-the-fly* (langsung saat data diminta), bukan setelah seluruh *file* selesai diproses.
2. Client dapat melanjutkan unduhan (*resume download*) jika koneksi terputus, cukup meminta sisa *chunk* yang belum diterima.
3. Durasi dekripsi relatif stabil karena dilakukan per chunk, bukan sekali proses penuh.

Namun, perlu dicatat bahwa pada pengujian di *shared hosting*, proses dekripsi terhenti pada batas waktu eksekusi *server* (sekitar 120 detik). Hal ini menjelaskan mengapa hasil pengukuran dekripsi selalu berhenti di angka ±121 detik, meskipun ukuran *file* berbeda. Artinya, keterbatasan ini bukan berasal dari algoritma, melainkan dari konfigurasi *server*. Dengan infrastruktur yang lebih fleksibel (misalnya VPS dengan *execution time* tanpa batas atau pengaturan *chunk* size yang lebih kecil melalui *JavaScript*), durasi dekripsi dapat menyesuaikan ukuran file dan tidak terhenti di angka tetap.

Secara keseluruhan, hasil ini menunjukkan bahwa sistem mampu menangani *file* berukuran menengah hingga besar dengan reliabilitas yang baik. Walaupun waktu enkripsi relatif tinggi, penggunaan *chunking* dan *streaming* dekripsi memastikan sistem tetap efisien, aman, serta ramah bagi pengguna pada kondisi jaringan yang tidak selalu stabil.

c. Rincian Hasil Pengujian

Hasil pengujian dapat dirangkum dalam **Tabel 2**, yang memperlihatkan kinerja modul *chatting*, enkripsi *file*, serta validasi data.

Tabel 2. Hasil Pengujian

Modul	Parameter Utama	Hasil	Catatan Teknis
Chatting	Latensi enkripsi/dekripsi	~0,045 detik per pesan (*real-time*)	Layak dipakai komunikasi aktif
Enkripsi File	4–41 MB	Enkripsi 38–332 s, Dekripsi berhenti 121 s	Dekripsi terhenti karena limit PHP 120 detik
Validasi Data	*Authentication tag*	Perubahan data terdeteksi	Integrity check berhasil

3.4 Analisis

Hasil pengujian memperlihatkan bahwa implementasi hybrid encryption berbasis AES-GCM dan ECC X25519 berjalan dengan benar, namun terdapat beberapa temuan penting yang perlu dianalisis lebih lanjut:

a. Kinerja Enkripsi vs Dekripsi
   1. Waktu enkripsi meningkat sebanding dengan ukuran *file* (hingga 332 detik untuk 41 MB).
   2. Dekripsi relatif lebih cepat (sekitar 121 detik) karena hanya melakukan verifikasi *authentication tag*.
   3. Namun, pada *file* berukuran besar dekripsi terhenti di sekitar 121 detik akibat batas eksekusi PHP pada *shared hosting*, bukan karena keterbatasan algoritma.

b. Keterbatasan Infrastruktur
   1. Lingkungan *shared hosting* membatasi eksekusi script maksimal ±120 detik, sehingga *file* besar tidak dapat diproses penuh dalam satu permintaan.
   2. Hal ini menunjukkan bahwa performa sistem tidak hanya ditentukan oleh algoritma kriptografi, tetapi juga oleh infrastruktur server yang digunakan.

c. Solusi Teknis
   1. Pembagian *file* menjadi *chunk* 512 KB sudah efektif mencegah limit memori.
   2. Untuk mengatasi limit waktu, mekanisme *resume download* dengan HTTP *Range Request* dapat dipadukan dengan *JavaScript client-side* agar proses dekripsi tetap berlanjut meski koneksi terputus atau *server timeout*.
   3. Alternatif lain adalah penggunaan *job queue* atau *worker* terpisah di server yang lebih fleksibel dibanding *shared hosting*.

d. Implikasi Keamanan
   1. *Authentication tag* AES-GCM terbukti efektif dalam mendeteksi perubahan data, menjamin integritas *file*.

2. ECC X25519 menghasilkan *shared secret* yang aman dengan *overhead* komputasi rendah, sehingga cocok untuk aplikasi *real-time*

4. KESIMPULAN

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

a. *Hybrid encryption* berbasis AES-256-GCM dan ECC X25519 berhasil diimplementasikan pada aplikasi komunikasi berbasis web. AES-256-GCM menjaga kerahasiaan sekaligus integritas data melalui *authentication tag*, sedangkan ECC X25519 menjamin distribusi kunci secara aman dan efisien.
b. Implementasi kriptografi sesuai dengan standar resmi (RFC 7748 untuk X25519 dan NIST SP 800-38D untuk AES-GCM). Hal ini membuktikan bahwa fungsi dasar enkripsi dan dekripsi berjalan benar serta dapat diandalkan.
c. Sistem terbukti berjalan *real-time* pada modul *chatting* dengan latensi sangat rendah, serta mampu mengenkripsi dan mendekripsi file ukuran menengah hingga besar dengan mekanisme *chunking*, meskipun durasi enkripsi meningkat seiring ukuran file.
d. Penelitian ini menunjukkan bahwa kombinasi AES-256-GCM dan ECC X25519 dapat diterapkan secara nyata pada aplikasi *chatting* dan *file sharing* berbasis web, bukan hanya simulasi, sehingga menjadi kontribusi praktis dalam penerapan *hybrid encryption*.
e. Proses enkripsi file berukuran besar masih relatif lambat, dan pengujian di lingkungan *shared hosting* terhenti pada batas waktu eksekusi ±120 detik, sehingga dekripsi tidak sepenuhnya optimal untuk file besar.
f. Optimalisasi performa enkripsi file besar dapat dilakukan dengan metode paralelisasi, pemanfaatan GPU, atau *job queue* di server. Selain itu, pengujian pada skenario jaringan kompleks (*multi-user dan real-time*) dapat memperluas validitas sistem.

DAFTAR PUSTAKA

[1] Z. Arif and A. Nurokhman, "Analisis perbandingan algoritma kriptografi simetris dan asimetris dalam meningkatkan keamanan sistem informasi," *Jurnal Teknologi dan Sistem Informasi (JTSI)*, vol. 4, no. 2, pp. 394–405, Sep. 2023, doi: 10.35957/jtsi.v4i2.6077.

[2] A. R. Harahap and T. A. Salim, "Sistem kriptografi pada pengamanan autentikasi dokumen elektronik: Systematic literature review," *Jurnal Pengembangan Kearsipan*, vol. 16, no. 2, pp. 203–220, Oct. 2023, doi: 10.22146/khazanah.81893.

[3] J. Daemen and V. Rijmen, "Advanced encryption standard (AES)," *NIST FIPS PUB 197*, Nov. 2001, updated May 2023, doi: 10.6028/NIST.FIPS.197-upd1.

[4] R. M. H. Hernandi and J. C. Chandra, "Implementasi algoritme AES-256 dan AES-GCM untuk mengamankan dokumen pada sistem data rekam medis klinik mulya," *KRESNA: Jurnal Riset dan Pengabdian Masyarakat*, vol. 4, no. 1, pp. 12–22, May 2024, doi: 10.36080/kresna.v4i1.131.

[5] L. Chen, D. Moody, A. Regenscheid, A. Robinson, and K. Randall, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," *NIST Special Publication 800-186*, Feb. 2023, doi: 10.6028/NIST.SP.800-186.

[6] V. Tanksale, "Efficient elliptic curve Diffie–Hellman key exchange for resource-constrained IoT devices," *Electronics (Switzerland)*, vol. 13, no. 18, pp. 1–13, Sep. 2024, doi: 10.3390/electronics13183631.

[7] P. Iqlima et al., "Implementasi sistem keamanan data menggunakan algoritma kriptografi asimetris elliptic curve cryptography (ECC) berbasis website," *Jurnal Ilmu Komputer (JIKUM)*, vol. 1, no. 1, pp. 10–18, 2025, doi: 10.62671/jikum.v1i1.37.

[8] P. Selvi and S. Sakthivel, "A hybrid ECC-AES encryption framework for secure and efficient cloud-based data protection," *Scientific Reports*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-01315-5.

[9] Y. M. A. Abualkas and D. L. Bhaskari, "Hybrid approach to cloud storage security using ECC-AES encryption and key management techniques," *International Journal of Engineering Trends and Technology*, vol. 72, no. 4, pp. 92–100, Apr. 2024, doi: 10.14445/22315381/IJETT-V72I4P110.

[10] A. Langley, M. Hamburg, and S. Turner, "Elliptic curves for security," RFC 7748, Internet Engineering Task Force (IETF), Jan. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7748.

[11] D. A. McGrew and J. Viega, "The Galois/counter mode of operation (GCM)," National Institute of Standards and Technology (NIST), 2004. [Online]. Available: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/gcm/gcm-spec.pdf.

[12] A. Widarma, "Kombinasi algoritma simetri dan ECC untuk meningkatkan keamanan pesan," *Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 8, no. 1, pp. 1–5, 2023, doi: 10.30743/infotekjar.v8i1.9642.