ANALISIS KOMPARASI PERFORMANSI ALGORITMA KRIPTOGRAFI RSA DAN AES PADA KEAMANAN DATA TEKS

Revan Sabilillah Saputra, Muhamad Alif Faras Syakir , Siti Aminah, Derisya Febriyanti, Nazwa Althafah Athalia
Informatika, Fakultas Sains dan Teknologi, UIN Sultan Maulana Hasanuddin Banten
Jl. Syech Nawawi Al-Baantani Curug, Kota Serang, Provinsi Banten, Indonesia, 42171.
Revansabil23@gmail.com

ABSTRAK

Keamanan data merupakan aspek prioritas dalam pertukaran informasi digital. Namun, pengembang sistem sering menghadapi dilema dalam memilih antara algoritma simetris *Advanced Encryption Standard* (AES) yang cepat namun sulit dalam distribusi kunci, atau algoritma asimetris *Rivest-Shamir-Adleman* (RSA) yang aman namun memiliki beban komputasi berat. Selain itu, belum banyak studi yang menyoroti inefisiensi (*overhead*) algoritma pada data berukuran mikro. Penelitian ini bertujuan untuk menganalisis perbandingan performansi waktu enkripsi (*encryption time*) antara kedua algoritma tersebut untuk menentukan ambang batas efisiensi yang tepat. Metodologi penelitian dilakukan melalui simulasi eksperimental menggunakan bahasa pemrograman Python dengan pustaka *PyCryptodome* terhadap variasi data uji ukuran 1 KB, 5 KB, dan 10 KB. Hasil pengujian menunjukkan bahwa algoritma AES mengalami *initialization overhead* pada data 1 KB sehingga sedikit lebih lambat dari RSA, namun performansi AES meningkat drastis dan jauh lebih unggul pada data 5 KB dan 10 KB. Penelitian ini menyimpulkan bahwa AES sangat efisien untuk pengamanan data utama (*payload*) dalam jumlah besar, sedangkan RSA lebih efektif direkomendasikan secara terbatas untuk mekanisme pertukaran kunci.

Kata kunci : *Keamanan Data, Kriptografi, AES, RSA, Perbandingan Performansi.*

1. PENDAHULUAN

Transformasi digital yang masif di era Industri 4.0 telah menempatkan teknologi informasi sebagai infrastruktur vital bagi operasional organisasi maupun aktivitas masyarakat sehari-hari. Aliran data digital kini menjadi tulang punggung di berbagai sektor strategis, mencakup layanan perbankan finansial, perdagangan elektronik (*e-commerce*), hingga tata kelola pemerintahan digital. Namun, tingginya ketergantungan terhadap konektivitas ini berbanding lurus dengan eskalasi risiko keamanan siber. Munculnya ancaman seperti penyadapan jalur komunikasi (*eavesdropping*), pencurian identitas, hingga manipulasi data (*tampering*) menuntut implementasi mekanisme proteksi yang tangguh dan adaptif [1]. Dalam perspektif ini, keamanan data (*Data Security*) bukan lagi sekadar fitur pelengkap, melainkan kebutuhan fundamental untuk menjamin aspek kerahasiaan (*confidentiality*), integritas (*integrity*), dan ketersediaan (*availability*) informasi.

Salah satu pilar utama dalam keamanan sistem informasi adalah kriptografi. Kriptografi merupakan ilmu dan seni matematika yang digunakan untuk menyandikan pesan (*plaintext*) menjadi bentuk yang tidak dapat dimengerti (*ciphertext*) tanpa pengetahuan khusus berupa kunci dekripsi [4]. Secara garis besar, algoritma kriptografi modern diklasifikasikan menjadi dua kategori utama berdasarkan manajemen kuncinya, yaitu algoritma simetris dan algoritma asimetris. Pemilihan di antara kedua jenis algoritma ini seringkali menjadi dilema teknis bagi para pengembang sistem, karena masing-masing menawarkan keunggulan dan kelemahan yang bertolak belakang.

Algoritma simetris, dengan *Advanced Encryption Standard* (AES) sebagai standar global yang ditetapkan oleh NIST, menggunakan kunci tunggal yang sama untuk proses enkripsi maupun dekripsi [5]. AES dikenal luas karena efisiensi komputasinya yang tinggi dan kecepatan prosesnya, menjadikannya pilihan utama untuk mengamankan data dalam jumlah besar (*bulk data encryption*). Namun, tantangan utama algoritma simetris terletak pada distribusi kunci (*key distribution problem*); bagaimana cara mengirimkan kunci rahasia kepada penerima tanpa diketahui oleh pihak ketiga yang tidak berwenang [3].

Di sisi lain, algoritma asimetris hadir sebagai solusi atas masalah distribusi kunci tersebut. Algoritma *Rivest-Shamir-Adleman* (RSA), sebagai representasi paling populer dari kriptografi asimetris, menggunakan sepasang kunci (kunci publik dan kunci privat) yang berbeda secara matematis namun saling terkait [2]. RSA memungkinkan pengiriman pesan rahasia tanpa harus mempertukarkan kunci privat terlebih dahulu. Meskipun menawarkan keamanan yang superior dalam hal manajemen kunci, RSA memiliki kelemahan signifikan berupa beban komputasi yang berat karena melibatkan operasi aritmatika modular pada bilangan prima yang sangat besar. Hal ini seringkali menyebabkan kinerja sistem menurun drastis jika RSA dipaksakan untuk mengenkripsi data berukuran besar [6].

Meskipun karakteristik teoritis kedua algoritma ini telah banyak didokumentasikan dalam literatur, masih terdapat kebutuhan untuk melakukan validasi empiris terkait performansi kecepatannya dalam lingkungan pengembangan modern seperti Python.

Beberapa penelitian sebelumnya cenderung berfokus pada analisis keamanan matematis, namun kurang menyoroti aspek *overhead* inisialisasi sistem pada saat memproses data berukuran sangat kecil [7]. Padahal, pemahaman mengenai perilaku algoritma pada berbagai variasi ukuran data sangat krusial untuk menentukan ambang batas efisiensi dalam perancangan sistem keamanan hibrida.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk melakukan analisis komparasi performansi antara algoritma AES dan RSA. Masalah utama yang akan dijawab adalah bagaimana karakteristik kecepatan kedua algoritma saat menangani variasi beban data yang berbeda. Untuk menyelesaikan masalah tersebut, penelitian ini direncanakan melalui metode simulasi eksperimental menggunakan bahasa pemrograman Python untuk mengukur waktu eksekusi enkripsi (*encryption time*) pada data mulai dari ukuran kecil (1 KB) hingga menengah (10 KB). Melalui pendekatan ini, diharapkan dapat dihasilkan rekomendasi teknis yang akurat bagi pengembang sistem dalam memilih algoritma yang tepat sesuai kebutuhan.

2. TINJAUAN PUSTAKA

2.1. Konsep Dasar Kriptografi

Keamanan data merupakan upaya sistematis untuk melindungi informasi dari akses tidak sah, perubahan ilegal, maupun perusakan data selama proses penyimpanan dan transmisi. Dalam sistem informasi modern, keamanan data mencakup tiga aspek utama yang dikenal sebagai CIA Triad, yaitu confidentiality (kerahasiaan), integrity (keutuhan), dan availability (ketersediaan). Kriptografi berperan penting dalam menjaga kerahasiaan data dengan mengubah informasi asli (plaintext) menjadi bentuk tersandi (ciphertext) yang tidak dapat dipahami tanpa kunci tertentu [1].

2.2. Algoritma Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) adalah algoritma kriptografi simetris yang menggantikan standar lama *Data Encryption Standard* (DES). AES ditetapkan oleh *National Institute of Standards and Technology* (NIST) pada tahun 2001 melalui FIPS PUB 197 [5]. Algoritma ini dikembangkan oleh dua kriptografer asal Belgia, Joan Daemen dan Vincent Rijmen (Rijndael).

AES beroperasi pada blok data berukuran 128-bit dan mendukung panjang kunci 128, 192, atau 256-bit. Struktur AES didasarkan pada jaringan substitusi-permutasi (*Substitution-Permutation Network* / SPN). Proses enkripsi AES melibatkan beberapa putaran (*rounds*) transformasi yang terdiri dari empat tahap utama: *SubBytes* (substitusi byte non-linear), *ShiftRows* (pergeseran baris), *MixColumns* (pengacakan kolom), dan *AddRoundKey* (penambahan kunci putaran) [3]. Efisiensi AES terletak pada kesederhanaan operasi aljabar yang digunakannya, sehingga sangat cepat dijalankan baik pada perangkat keras (*hardware*) maupun perangkat lunak (*software*).

2.3. Mode Operasi Electronic Code Book (ECB)

Dalam implementasi *block cipher* seperti AES untuk data yang lebih besar dari satu blok (128-bit), diperlukan mode operasi. Penelitian ini menggunakan mode *Electronic Code Book* (ECB). Mode ECB membagi *plaintext* menjadi blok-blok independen dan mengenkripsi setiap blok secara terpisah menggunakan kunci yang sama [1]. Meskipun mode ini memiliki kelemahan keamanan pola untuk gambar bitmap, ECB dipilih dalam konteks penelitian ini karena kesederhanaannya untuk mengukur kecepatan murni (*raw speed*) algoritma tanpa *overhead* tambahan dari ketergantungan antar-blok (*chaining*) seperti pada mode CBC atau GCM.

2.4. Algoritma Rivest-Shamir-Adleman (RSA)

RSA adalah algoritma kriptografi kunci publik yang dipublikasikan pertama kali oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1978 di MIT [2]. Keamanan RSA bersandar pada kesulitan komputasi untuk memfaktorkan bilangan bulat besar yang merupakan hasil kali dari dua bilangan prima besar (*integer factorization problem*).

Berbeda dengan AES, RSA menggunakan dua kunci:

a. Kunci Publik: Digunakan untuk mengenkripsi pesan dan dapat disebarkan secara bebas.
b. Kunci Privat: Digunakan untuk mendekripsi pesan dan harus dijaga kerahasiaannya oleh pemilik.

2.5. Kompleksitas Komputasi RSA

Proses enkripsi dan dekripsi RSA melibatkan operasi eksponensial modular yang sangat intensif secara komputasi. Jika $\boldsymbol{m}$ adalah pesan, $\boldsymbol{e}$ adalah eksponen publik, dan $\boldsymbol{n}$ adalah modulus, maka proses enkripsi dinyatakan sebagai $\boldsymbol{c = m^e \bmod n}$. Sebaliknya, dekripsi dilakukan dengan $\boldsymbol{m = c^d \bmod n}$, di mana $\boldsymbol{d}$ adalah eksponen privat [6]. Operasi perpangkatan dengan angka ribuan bit inilah yang menyebabkan RSA cenderung lebih lambat dibandingkan algoritma simetris yang hanya menggunakan operasi XOR dan geser bit (*bitwise shift*).

2.6. Skema Padding (PKCS#1 OAEP)

Algoritma RSA murni bersifat deterministik dan rentan terhadap serangan *chosen-ciphertext*. Oleh karena itu, standar keamanan modern mewajibkan penggunaan skema *padding*. Penelitian ini menerapkan *Optimal Asymmetric Encryption Padding* (OAEP) sesuai standar PKCS#1. OAEP menambahkan elemen keacakan (*randomness*) pada *plaintext* sebelum dienkripsi, sehingga pesan yang sama tidak akan menghasilkan *ciphertext* yang sama jika dienkripsi dua kali [6]. Penggunaan OAEP sedikit menambah beban komputasi namun sangat krusial untuk validitas keamanan simulasi.

2.7. Perbandingan Karakteristik AES dan RSA

Secara arsitektur, AES dan RSA memiliki perbedaan mendasar yang memengaruhi kinerjanya. Nawawi, Rahman, dan Putra [7] dalam penelitian terbarunya tahun 2025 menyimpulkan bahwa AES memiliki efisiensi waktu yang jauh lebih unggul dibandingkan RSA pada enkripsi berkas, sehingga lebih direkomendasikan untuk pengamanan data utama (*payload*).

Hal senada diungkapkan oleh Olutola dan Olumuyiwa [9], yang membuktikan bahwa algoritma asimetris memiliki beban komputasi (*computational overhead*) yang tinggi akibat kompleksitas operasi matematika modular. Tijjani dan Isa [8] juga menambahkan bahwa performansi algoritma simetris lebih stabil pada berbagai jenis data (teks, gambar, video) dibandingkan asimetris. Berdasarkan karakteristik tersebut, paradigma keamanan modern kini cenderung menerapkan konsep *Hybrid Cryptosystem*, sebagaimana disarankan oleh Permana dkk. [12], yaitu menggabungkan kecepatan AES untuk data dan keamanan RSA untuk kunci.

2.8. Tinjauan Penelitian Terkait

Sejumlah penelitian terdahulu telah dilakukan untuk membandingkan performansi algoritma kriptografi. Prashant dkk. [10] serta Haque dkk. [11] melakukan komparasi luas antara AES, RSA, dan algoritma lain. Hasilnya konsisten menunjukkan bahwa AES unggul dalam kecepatan (*encryption throughput*), sementara RSA unggul dalam manajemen kunci namun lambat dalam eksekusi. Pentingnya aspek efisiensi waktu ini juga ditegaskan oleh Ariffin dan Rizwan [16] serta Lestari dan Zulfikar [17], yang menyatakan bahwa performansi komputasi adalah parameter krusial dalam implementasi sistem keamanan modern.

Dalam konteks lingkungan yang berbeda, Dhanda dan Kaur [13] meneliti implementasi AES dan RSA di *cloud computing* dan menyatakan bahwa integrasi keduanya memberikan hasil optimal. Sementara itu, Kartit dkk. [15] menguji algoritma ini pada perangkat keras terbatas (Arduino Due) dan menemukan bahwa RSA memakan daya dan waktu yang sangat signifikan dibandingkan AES. Penelitian ini berupaya melengkapi kajian tersebut dengan analisis simulasi berbasis Python untuk menyoroti fenomena *initialization overhead* pada data mikro yang belum banyak dibahas secara spesifik.

3. METODE PENELITIAN

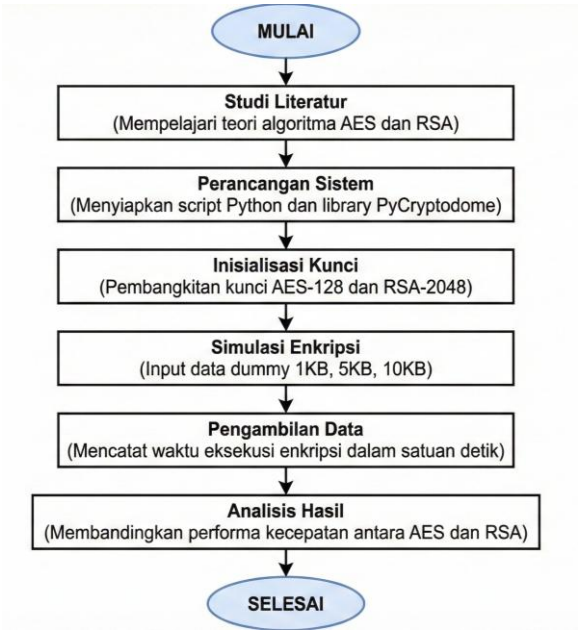
3.1. Lingkungan Pengembangan

Penelitian ini dilakukan menggunakan metode simulasi eksperimental pada perangkat lunak tanpa melibatkan perangkat keras kriptografi khusus (*Hardware Security Module*). Spesifikasi lingkungan pengembangan yang digunakan adalah sebagai berikut:

a. Perangkat Keras: Prosesor Intel/AMD dengan RAM minimal 4 GB.
b. Sistem Operasi: Windows 10/11 (64-bit).
c. Bahasa Pemrograman: Python versi 3.x.
d. Pustaka Kriptografi: *PyCryptodome*, pustaka Python yang menyediakan implementasi algoritma kriptografi standar industri yang stabil, aman, dan efisien [14].
e. Editor Kode: *Visual Studio Code* (VS Code).

3.2. Alur Penelitian

Tahapan penelitian disusun secara sistematis untuk memastikan data yang diperoleh valid dan dapat dipertanggungjawabkan. Proses dimulai dari studi literatur mengenai algoritma AES dan RSA, dilanjutkan dengan perancangan kode program simulasi. Tahap inti penelitian adalah proses inisialisasi kunci (key generation) dan pengukuran waktu enkripsi pada data uji. Alur lengkap penelitian digambarkan dalam diagram alir pada Gambar 1.



Gambar 1. Diagram Alur Penelitian

Penjelasan mengenai tahapan pada Gambar 1 adalah sebagai berikut:

a. Inisialisasi Kunci *(key Initialization) :*
   Pada tahap ini, sistem membangkitkan kunci kriptografi secara otomatis. Untuk algoritma AES, dibangkitkan kunci acak sebesar 128-bit (16 byte). Sedangkan untuk algoritma RSA, dibangkitkan pasangan kunci (key pair) yang terdiri dari kunci publik dan kunci privat dengan panjang 2048-bit.
b. Generasi Data Uji *(Data Generation) :*
   Sistem membuat sampel data teks (dummy text) secara dinamis. Variasi ukuran data yang digunakan dalam penelitian ini adalah 1 KB, 5 KB, dan 10 KB. Data ini berupa karakter berulang yang merepresentasikan payload informasi yang akan diamankan.

c. Simulasi Enkripsi *(Encryption Simulation):*
Setiap sampel data dienkripsi menggunakan algoritma AES dan RSA secara bergantian. Sistem mencatat waktu awal (start time) sebelum proses enkripsi dimulai dan waktu akhir (end time) setelah proses selesai.

d. Analisis Hasil *(Encryption Simulation) :*
Selisih antara waktu akhir dan waktu awal dihitung sebagai durasi eksekusi. Data waktu tersebut kemudian dikompilasi ke dalam tabel dan grafik untuk dianalisis perbandingan kecepatannya.

3.3. Implementasi Sistem

Implementasi sistem dibangun menggunakan bahasa pemrograman Python dalam satu berkas skrip terintegrasi. Algoritma AES diimplementasikan menggunakan mode operasi *Electronic Code Book* (ECB) dengan *padding* standar untuk menyesuaikan blok data. Sedangkan algoritma RSA diimplementasikan menggunakan standar *PKCS1_OAEP* (*Optimal Asymmetric Encryption Padding*) untuk menjamin keamanan skema enkripsi asimetris.

Khusus pada pengujian RSA, dilakukan penyesuaian teknis berupa pengambilan sampel potongan data (chunking) sebesar 100 byte. Hal ini dilakukan mengingat batasan teknis algoritma RSA yang ditujukan untuk enkripsi kunci atau pesan pendek, bukan untuk mengenkripsi berkas berukuran besar secara langsung selayaknya block cipher.

Berikut adalah tangkapan layar (screenshot) implementasi fungsi enkripsi dan logika pengukuran waktu yang dibangun pada editor *Visual Studio Code*:

```
for size in sizes:
    raw_data = generate_data(size)

    start_time = time.time()
    cipher_aes = AES.new(aes_key, AES.MODE_ECB)
    ciphertext_aes = cipher_aes.encrypt(pad(raw_data, AES.block_size))
    aes_duration = time.time() - start_time

    print(f"{size} KB {'':<10} | AES        | {aes_duration:.8f} detik")

    try:
        sample_chunk = raw_data[:100]
        start_time = time.time()
        ciphertext_rsa = rsa_cipher.encrypt(sample_chunk)
        rsa_duration = time.time() - start_time

        print(f"{size} KB (Sampel)  | RSA        | {rsa_duration:.8f} detik")
    except Exception as e:
        print(f"RSA Error: {e}")
```

Gambar 2. Implementasi Kode Pengujian Enkripsi pada Python

Seperti terlihat pada Gambar 2, fungsi time.time() dari pustaka bawaan Python digunakan untuk menangkap penanda waktu (timestamp) dengan presisi tinggi guna memastikan akurasi data pengukuran.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Pengujian

Pengujian dilakukan dengan mengeksekusi skrip simulasi pada terminal *Visual Studio Code*. Variabel yang diukur adalah waktu eksekusi enkripsi (dalam satuan detik) untuk memproses data *dummy* berukuran 1 KB, 5 KB, dan 10 KB. Data waktu diambil dari selisih waktu sistem (*system clock*) sebelum dan sesudah fungsi enkripsi berjalan.

Bukti tampilan hasil eksekusi program yang menunjukkan data waktu enkripsi dapat dilihat pada Gambar 3.

```
==========================================================
SEDANG MENYIAPKAN KUNCI KRIPTOGRAFI...
==========================================================

UKURAN DATA     | ALGORITMA  | WAKTU ENKRIPSI (Detik)
----------------------------------------------------------
1 KB            | AES        | 0.00469398 detik
1 KB (Sampel)   | RSA        | 0.00186348 detik
----------------------------------------------------------
5 KB            | AES        | 0.00011802 detik
5 KB (Sampel)   | RSA        | 0.00155473 detik
----------------------------------------------------------
10 KB           | AES        | 0.00015569 detik
10 KB (Sampel)  | RSA        | 0.00176883 detik
----------------------------------------------------------

SELESAI!
```

Gambar 3. Hasil Eksekusi Program pada Terminal

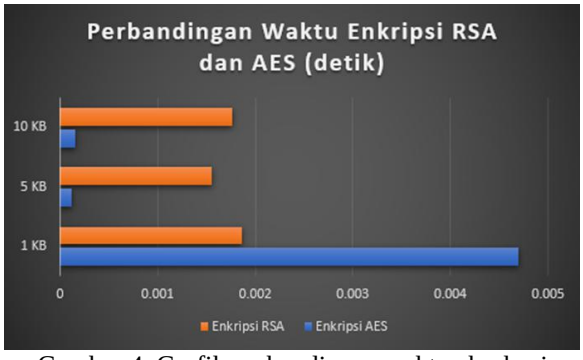
Berdasarkan *output* program di atas, data hasil pengujian dirangkum secara lebih terstruktur pada Tabel 1 berikut:

Tabel 1. Perbandingan Waktu Enkripsi AES dan RSA

Ukuran Data	Waktu Enkripsi APS (detik)	Waktu Enkripsi RSA (detik)
1 KB	0.00469398	0.00186348
5 KB	0.00011802	0.00155473
10 KB	0.00015569	0.00176883

4.2. Pembahasan

Untuk mempermudah analisis tren performansi, data pada Tabel 1 divisualisasikan ke dalam bentuk grafik batang sebagaimana terlihat pada Gambar 4.



Gambar 4. Grafik perbandingan waktu eksekusi enkripsi AES dan RSA

Berdasarkan Gambar 4 dan Tabel 1, dapat dianalisis beberapa temuan penting sebagai berikut:

a. Fenomena *Initialization Overhead* pada AES (1 KB):
Pada pengujian dengan ukuran data terkecil (1 KB), algoritma AES mencatat waktu 0,0046 detik, yang secara menarik justru lebih lambat dibandingkan RSA (0,0018 detik). Hal ini merupakan fenomena wajar yang disebabkan oleh

initialization overhead. Algoritma AES memerlukan waktu awal untuk menyiapkan matriks internal (*State Matrix*) dan tabel substitusi (*S-Box*) di memori komputer sebelum proses enkripsi blok pertama dimulai. Pada data yang sangat kecil, waktu persiapan ini mendominasi total waktu eksekusi.

b. Efisiensi AES pada Data Menengah:
Keunggulan sejati algoritma AES terlihat sangat signifikan pada ukuran 5 KB dan 10 KB. Setelah fase inisialisasi selesai, waktu proses enkripsi AES turun drastis dan stabil di kisaran 0,0001 detik. Hal ini membuktikan bahwa arsitektur AES sangat efisien dalam memproses blok data secara berurutan.

c. Kompleksitas Komputasi RSA:
Sebaliknya, algoritma RSA menunjukkan tren waktu yang lebih lambat dibandingkan fase stabil AES. Pada ukuran 10 KB, RSA membutuhkan waktu 0,0017 detik. Jika dibandingkan dengan AES pada ukuran yang sama (0,0001 detik), RSA lebih lambat sekitar 11 kali lipat. Hal ini terjadi karena RSA menggunakan operasi matematika modular eksponensial pada bilangan prima besar yang memakan banyak siklus CPU, sedangkan AES hanya menggunakan operasi substitusi dan permutasi bit (XOR) yang ringan bagi prosesor.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan bahwa algoritma AES memiliki performansi kecepatan yang jauh lebih unggul dibandingkan RSA untuk enkripsi data di atas 1 KB, dengan waktu eksekusi yang stabil dan sangat singkat (±0,0001 detik). Sebaliknya, RSA membutuhkan sumber daya komputasi yang lebih besar dengan waktu eksekusi yang konsisten lebih lambat dibandingkan AES pada fase stabil, sehingga kurang efisien jika digunakan untuk mengenkripsi data payload berukuran besar secara langsung. Fenomena keterlambatan AES pada data sangat kecil (1 KB) akibat proses inisialisasi (overhead) terbukti hanya bersifat sementara dan dapat diabaikan untuk kepentingan pemrosesan data dalam jumlah besar.

Berdasarkan karakteristik kecepatan yang diperoleh, disarankan untuk menerapkan metode *Hybrid Cryptosystem*. Penggunaan algoritma AES direkomendasikan untuk mengamankan data utama (file/dokumen), sedangkan algoritma RSA lebih tepat digunakan terbatas untuk mengamankan kunci simetris AES (*Key Exchange*) demi mendapatkan kombinasi keamanan dan kecepatan yang optimal.

DAFTAR PUSTAKA

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. Pearson Education, 2017.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[3] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. John Wiley & Sons, 1996.

[5] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS PUB 197, 2001.

[6] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 3rd ed. CRC Press, 2020.

[7] M. Nawawi, A. Rahman, and D. Putra, "Comparison of AES and RSA encryption performance," *G-Tech: Jurnal Teknologi Terapan*, vol. 8, no. 1, pp. 1-10, 2025.

[8] A. Tijjani and A. Isa, "Performance Analysis of Symmetric and Asymmetric Encryption Algorithms," *Int. J. Emerg. Multidiscip. Comp. Sci. & AI*, vol. 3, no. 1, pp. 11-18, 2024.

[9] A. Olutola and M. Olumuyiwa, "Comparative analysis of encryption algorithms (AES vs RSA)," *European Journal of Technology*, vol. 7, no. 1, pp. 1–9, 2023.

[10] K. Prashant, R. Sharma, and S. Verma, "Comparative analysis of AES and RSA with other encryption for secure communication," *Int. J. Sci. Res. in Comp. Sci., Eng. and Info. Tech.*, vol. 10, no. 2, pp. 120-125, 2024.

[11] M. Haque, S. Rahman, and T. Islam, "A comparative analysis of AES, RSA, and 3DES encryption standards," *Management Science Advances*, vol. 2, no. 1, pp. 45-52, 2024.

[12] R. Permana, M. Siregar, and N. Putri, "Mengamankan file rahasia menggunakan hybrid AES dan RSA," *Jurnal Review Pendidikan dan Pengajaran*, vol. 7, no. 1, pp. 2449–2460, 2024.

[13] M. Dhanda and S. Kaur, "Comparative Analysis of RSA and AES Algorithms for Cloud Security," *J. Phys.: Conf. Ser.*, vol. 1950, p. 012089, 2021.

[14] E. Helgeson, "PyCryptodome: The Python Cryptography Toolkit," [Online]. Available: https://www.pycryptodome.org/. [Accessed: 2024].

[15] A. Kartit, M. El Marraki, and A. El Hore, "Evaluation of the RSA and AES algorithms performance on the Arduino Due," *Int. J. Elec. & Comp. Eng.*, vol. 12, no. 1, pp. 589-598, 2022.

[16] S. H. S. Ariffin and M. A. M. Rizwan, "Performance Evaluation of Symmetric Encryption Algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 8, pp. 145-152, 2022

[17] D. P. Lestari and W. B. Zulfikar, "Analisis Perbandingan Algoritma Kriptografi Klasik dan Modern," *Jurnal Informatika*, vol. 10, no. 2, pp. 112-119, 2023.