KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 7, No. 1, Januari (2026), pp. 71–80

Perbandingan Algoritma RSA dan AES pada Fitur Transfer dalam Studi Kasus Digital Banking

Ahmad Miftah Fajrin[1]
[1]*Universitas Surabaya, Indonesia*
E-mail: ahmadmiftah@staff.ubaya.ac.id

Abstract

Digital banking services require cryptographic mechanisms that protect sensitive customer data while maintaining low latency and efficient resource utilization. This study evaluates the performance of RSA (asymmetric encryption) and AES (symmetric encryption) in securing digital banking transactions across two common scenarios: login authentication and fund transfer. The evaluation examines encryption time, decryption time, and memory usage during cryptographic processing. Experimental results demonstrate that AES achieves faster encryption than RSA, with 0.7% lower encryption time in the login scenario and 33.6% lower encryption time in the transfer scenario. For login decryption, AES is 65.3% faster than RSA and reduces memory consumption by 65.5%. In contrast, during transfer decryption, AES introduces substantially higher latency, operating 12.2 times slower, which corresponds to a 1119.7% increase, and consumes 2112.7% more memory. These results suggest that AES is highly efficient for encryption and lightweight decryption tasks such as login, but transfer decryption performance may require further optimization to ensure consistent real-time responsiveness in digital banking environments.

Keywords: *AES, RSA, digital banking, encryption, decryption, digital banking*

Abstrak

Keamanan transaksi pada layanan digital banking memerlukan mekanisme kriptografi yang mampu menjaga kerahasiaan data pengguna dan mempertahankan efisiensi sistem. Studi ini membandingkan kinerja algoritma RSA (asimetris) dan AES (simetris) dalam dua skenario utama perbankan digital, yaitu login dan transfer, dengan mengukur waktu eksekusi serta konsumsi memori untuk setiap proses pada RSA. Hasil pengujian memperlihatkan bahwa pada tahap enkripsi, AES memiliki performa lebih baik dibandingkan RSA, yaitu 0,7% lebih cepat pada skenario login dan 33,6% lebih cepat pada skenario transfer. Pada tahap dekripsi login, AES juga lebih efisien dengan kecepatan 65,3% lebih tinggi dibandingkan RSA, serta menurunkan konsumsi memori sebesar 65,5%. Namun, pada dekripsi transfer, AES menunjukkan latensi yang lebih tinggi, yaitu 12,2 kali lebih lambat dibandingkan RSA, disertai kenaikan konsumsi memori sebesar 2112,7%. Temuan ini mengindikasikan bahwa AES lebih unggul pada operasi enkripsi dan dekripsi sederhana seperti login, namun implementasi pada transaksi transfer memerlukan optimasi lebih lanjut untuk menjaga latensi tetap rendah.

Kata Kunci: *AES, RSA, digital banking, enkripsi, dekripsi, keamanan transaksi*

1. Pendahuluan

Perkembangan teknologi mendorong perlunya mekanisme pengamanan data yang andal, terutama pada sistem komunikasi digital di jaringan terbuka. Dalam konteks ini, kriptografi memegang peranan penting dalam menjamin kerahasiaan, integritas, dan autentikasi data guna mencegah ancaman penyadapan, manipulasi, serta penyalahgunaan informasi [1]. Studi-studi sebelumnya menunjukkan bahwa tanpa penerapan kriptografi yang memadai, sistem informasi modern sangat rentan terhadap serangan keamanan, terutama pada layanan berbasis jaringan dan komunikasi real-time[2].

ISSN: 2720-992X

Transformasi layanan perbankan ke kanal digital meningkatkan efisiensi transaksi sekaligus memperluas permukaan serangan siber. Proses seperti login, transfer dana, dan validasi saldo melibatkan data sensitif sehingga memerlukan enkripsi yang kuat dan efisien. Dalam praktik saat ini, sistem pembayaran umumnya mengombinasikan kriptografi simetris untuk perlindungan data karena kecepatan, serta kriptografi asimetris untuk pertukaran kunci atau tanda tangan karena mendukung distribusi kunci publik. Studi pada sistem payment gateway menunjukkan bahwa kombinasi RSA dan AES banyak digunakan untuk menjaga keamanan proses transaksi sekaligus mempertahankan kinerja layanan [3].

Kriptografi kunci publik merupakan salah satu pendekatan utama dalam pengamanan data karena memungkinkan distribusi kunci secara aman tanpa memerlukan saluran rahasia. Algoritma RSA merupakan contoh implementasi algoritma kriptografi yang populer hingga saat ini. RSA masih diterapkan secara luas pada berbagai sistem keamanan informasi, termasuk pengamanan pesan, pertukaran kunci, dan perlindungan data sensitif, karena memiliki dasar matematis yang kuat serta kemudahan implementasi relative [4][5]. Keamanan algoritma RSA bergantung pada tingkat kesulitan faktorisasi bilangan bulat besar. Selama permasalahan ini belum dapat diselesaikan secara efisien dengan komputasi klasik, RSA tetap dianggap aman untuk berbagai aplikasi praktis [6]. Oleh sebab itu, RSA terus menjadi objek penelitian dan pengembangan untuk meningkatkan ketahanan dan efektivitasnya terhadap ancaman keamanan yang semakin kompleks [7].

Advanced Encryption Standard atau AES Adalah contoh dari algoritma simetris yang populer untuk melindungi payload transaksi pada layanan digital banking karena mampu memberikan enkripsi cepat dengan biaya komputasi rendah. Dalam transaksi finansial, AES lazim digunakan untuk menjaga kerahasiaan data sensitif seperti nomor kartu, kredensial, dan detail transfer, baik saat penyimpanan maupun pengiriman, sehingga mendukung keamanan layanan tanpa menambah latensi secara signifikan [8]. Meskipun kuat secara desain, implementasi AES tetap harus memperhatikan risiko serangan kanal-samping, sehingga penerapan praktik implementasi yang aman dan mitigasi menjadi faktor penting dalam sistem pembayaran modern [9].

Dalam praktik digital banking, AES umumnya digunakan untuk melindungi data transaksi ber-volume besar pada kanal API maupun penyimpanan seperti tabel transaksi, log, dan arsip audit. Survei terbaru menunjukkan bahwa AES masih menjadi landasan de facto dalam mengamankan komunikasi dan transaksi finansial karena efisiensi, dukungan akselerasi perangkat keras, serta fleksibilitas mode operasi. Pada sisi penyimpanan, evaluasi kinerja transparent data encryption berbasis AES pada beberapa DBMS menunjukkan overhead yang relatif kecil sehingga tetap kompatibel untuk layanan real-time. Temuan terkait tantangan privasi dan keamanan siber dalam transformasi perbankan digital juga menempatkan enkripsi sebagai kontrol inti untuk menekan risiko kebocoran data nasabah dan penyalahgunaan informasi [10] [11].

Meskipun RSA dan AES sama-sama populer, keduanya memiliki karakteristik komputasi yang berbeda sehingga dampaknya terhadap latensi transaksi perlu dianalisis secara kuantitatif. Kemudian sebagian penelitian terdahulu lebih banyak menilai kinerja secara terpisah yaitu uji enkripsi dan dekripsi pada fungsi tertentu tanpa evaluasi end-to-end pada alur client–server, atau tanpa adanya hasil metrik untuk waktu eksekusi dan konsumsi memori pada skenario transaksi yang berbeda Selain aspek kinerja, RSA juga memiliki risiko pada sisi implementasi apabila perlindungan eksekusi tidak memadai, misalnya terhadap serangan side channel berbasis konsumsi daya [12]. Penelitian ini berkontribusi dengan prototipe client–server yang membandingkan RSA vs AES secara end-to-end, dua skenario yang merepresentasikan beban kerja berbeda yaitu login dan transfer, serta pelaporan berupa matrix waktu dan memori untuk membantu pengambilan keputusan implementatif.

Kontribusi utama dari penelitian ini adalah menyediakan panduan pengambilan keputusan berbasis skenario. Algoritma AES direkomendasikan untuk operasi dengan payload kecil, seperti login, karena menawarkan latensi dan penggunaan memori yang lebih rendah. Untuk transfer payload yang lebih besar, diperlukan audit serta optimasi implementasi agar proses dekripsi tetap memenuhi kebutuhan real-time. Studi ini juga menyoroti pentingnya pelaporan metrik waktu dan memori secara bersamaan untuk menilai trade-off pada sistem dengan sumber daya terbatas.

2. Metodologi Penelitian

2.1. RSA (Rivest Shamir Aldeman)

RSA adalah contoh dari salah satu kriptosistem yang menggunakan public key yang bekerja dengan aritmetika modulo berbasis bilangan bulat besar. Pada tahap pembangkitan kunci, dua bilangan prima besar p dan q dipilih untuk membentuk modulus $n = pq$. Parameter publik e kemudian ditetapkan sehingga relatif prima terhadap $\varphi(n) = (p-1)(q-1)$, lalu nilai privat d dihitung sebagai invers modular dari $e$terhadap $\varphi(n)$. Dengan pasangan kunci (n, e)sebagai kunci publik dan $d$sebagai bagian utama kunci privat, RSA melakukan transformasi kriptografis melalui eksponensiasi modular. Dalam praktik, e sering dipilih bernilai 65537 karena memberikan efisiensi eksponensiasi yang baik tanpa mengorbankan keamanan secara signifikan [13]. Algoritma 1 adalah implementasi dari pseudocode RSA.

Algoritma 1: RSA – Pembentukan Kunci

Input: ukuran kunci kBits
1. pilih prima besar p dan q
2. n ← p × q
3. φ ← (p−1) × (q−1)
4. pilih e (umumnya 65537) dengan gcd(e, φ)=1
5. d ← e^{-1} mod φ
Output: public key (n,e), private key d

Proses enkripsi pada RSA dapat ditunukkan pada Algoritma 2. Inputan pada RSA berupa pasangan kunci publik (n, e) dan pesan m. Pesan m terlebih dahulu direpresentasikan sebagai bilangan bulat yang nilainya lebih kecil dari n, kemudian dienkripsi dengan cara dipangkatkan menggunakan eksponen publik $e$dan diambil hasil modulo n, sehingga diperoleh ciphertext $c = m^e \, mod \, n$. Proses ini menunjukkan bahwa siapa pun yang memiliki kunci publik dapat mengenkripsi pesan, tetapi belum tentu dapat membaca isinya.

Algoritma 2: RSA – Enkripsi

Input: (n, e), pesan m
1. c ← m^e mod n
Output: c

Pada Algorithm 3 menggambarkan proses kebalikan dari enkripsi. Masukan berupa pasangan kunci privat (n, d) dan ciphertext c. Ciphertext tersebut dipangkatkan dengan eksponen privat $d$dan diambil hasil modulo n, sehingga menghasilkan kembali pesan asli $m = c^d \, mod \, n$. Secara matematis, nilai d dipilih sedemikian rupa sehingga operasi pemangkatan dengan $d$akan membatalkan efek pemangkatan dengan e. Maka dari itu, hanya orang yang mempunyai private key yang dapat melakukan dekripsi dan memperoleh kembali pesan asli, yang menjadi inti dari keamanan algoritma RSA.

Algoritma 3: RSA – Dekripsi

Input: (n, d), ciphertext c
1. m ← c^d mod n
Output: m

Implementasi RSA sangat menentukan kekuatan keamanan yang sangat penting. Banyak sistem mempercepat operasi privat menggunakan optimasi seperti CRT-RSA, tetapi optimasi semacam ini juga dapat memperluas permukaan serangan pada sisi implementasi. Penelitian terkini menunjukkan bahwa RSA rentan terhadap serangan kanal-samping yang apabila eksekusi modular exponentiation tidak dilindungi dengan baik maka serangan yang diprofilkan menggunakan pendekatan neural network bahkan dapat mengekstraksi informasi rahasia dari jejak konsumsi daya implementasi RSA tertentu. Studi lain juga menyoroti bahwa kebocoran sebagian parameter privat seperti partial key exposure dapat membuka peluang kriptoanalisis terhadap sejumlah varian RSA dalam kondisi tertentu, sehingga perlindungan implementasi dan pengelolaan parameter tetap menjadi isu penting dalam penerapan RSA modern.

2.2. Advanded Encryption Standard (AES)

AES merupakan symmetric block cipher dengan blok 128-bit dan dengan kunci yang panjangnya 128/192/256-bit. Algoritma ini bekerja melalui serangkaian transformasi per ronde (SubBytes, ShiftRows, MixColumns, dan AddRoundKey) sehingga efisien untuk mengenkripsi payload transaksi yang berulang maupun berukuran besar. Karena biaya komputasinya relatif rendah, AES banyak digunakan pada layanan pembayaran dan perbankan digital untuk melindungi data finansial pada perangkat klien maupun server. Meski demikian, keamanan AES pada level implementasi tetap perlu diperhatikan, misalnya terhadap serangan kanal-samping (Correlation Power Analysis) yang berupaya mengekstrak kunci dari pola konsumsi daya, sehingga penggunaan mitigasi dan praktik implementasi yang benar menjadi penting. Algoritma 4 menunjukkan cara kerja dari AES ketika melakukan enkripsi.

Algoritma 4: AES – Enkripsi

```
Input  : plaintext_block (128 bit), secret_key
Output : encrypted_block
1.  expanded_keys ←  GenerateRoundKeys(secret_key)
2.  current_state ←  plaintext_block
3.  current_state ←  XORRoundKey(current_state, expanded_keys[0])
4.  for i ←  1 to (TotalRound − 1) do
5.      current_state ←  ByteSubstitution(current_state)
6.      current_state ←  RowPermutation(current_state)
7.      current_state ←  ColumnMixing(current_state)
8.      current_state ←  XORRoundKey(current_state, expanded_keys[i])
9.  end for
10. current_state ←  ByteSubstitution(current_state)
```

Pada proses enkripsi AES seperti pada Algoritma 4, pseudocode menggambarkan bahwa algoritma menerima masukan berupa satu blok plaintext berukuran 128-bit dan sebuah kunci rahasia, kemudian kunci tersebut terlebih dahulu diproses melalui tahap key expansion untuk menghasilkan kunci pada setiap ronde. Plaintext direpresentasikan sebagai state dan diawali dengan operasi AddRoundKey, yaitu penggabungan state dengan kunci ronde awal menggunakan operasi XOR. Selanjutnya, state diproses melalui sejumlah ronde utama yang jumlahnya bergantung pada panjang kunci. Pada setiap ronde, dilakukan empat transformasi berurutan, yaitu SubBytes untuk melakukan substitusi non-linear guna meningkatkan konfusi, ShiftRows untuk menggeser baris state sehingga tercapai difusi, MixColumns untuk mencampur nilai pada setiap kolom state, serta AddRoundKey untuk menggabungkan state dengan kunci ronde. Pada ronde terakhir, transformasi MixColumns tidak dilakukan, dan hasil akhir dari state setelah AddRoundKey merupakan ciphertext.

Algoritma 5 : AES – Dekripsi

```
Input  : encrypted_block (128 bit), secret_key
Output : recovered_plaintext
round_keys ← GenerateRoundKeys(secret_key)
internal_state ← encrypted_block
internal_state ← XORRoundKey(internal_state, round_keys[FinalRound])
for r ← (FinalRound − 1) down to 1 do
    internal_state ← InverseRowPermutation(internal_state)
    internal_state ← InverseByteSubstitution(internal_state)
    internal_state ← XORRoundKey(internal_state, round_keys[r])
    internal_state ← InverseColumnMixing(internal_state)
end for
internal_state ← InverseRowPermutation(internal_state)
```

Pada proses dekripsi AES seperti pada Algoritma 2.5, pseudocode menunjukkan bahwa algoritma menerima masukan berupa ciphertext dan kunci rahasia yang sama dengan proses enkripsi. Ciphertext direpresentasikan sebagai state dan terlebih dahulu dikombinasikan dengan kunci ronde terakhir melalui operasi AddRoundKey. Selanjutnya, state diproses melalui ronde-ronde dekripsi dengan urutan terbalik dibandingkan enkripsi, menggunakan transformasi invers, yaitu InvShiftRows, InvSubBytes, AddRoundKey, dan InvMixColumns. Transformasi invers ini dirancang untuk membatalkan efek setiap operasi pada proses enkripsi. Pada ronde terakhir dekripsi, operasi InvMixColumns tidak dilakukan. Setelah seluruh ronde selesai, state digabungkan kembali dengan kunci ronde awal menggunakan AddRoundKey sehingga diperoleh kembali plaintext asli.

2.3. Rancangan Eksperimen dan Parameter Pengujian

Pengujian dilakukan pada prototipe client–server dengan dua varian implementasi yaitu varian RSA yang mengenkripsi payload menggunakan kunci publik server dan didekripsi menggunakan kunci privat serve varian AES yang mengenkripsi dan mendekripsi payload menggunakan kunci simetris yang sama. Pada penilitian ini menggunakan perangkat untuk client dan server adalah satu perangkat yang sama yaitu menggunakan processor AMD Ryzen 5 4600H dengan dukungan sistem operasi Windows

10. Untuk RAM menggunakan 16GB dual channel. Bahasa pemrograman python digunakan untuk implementasi program.

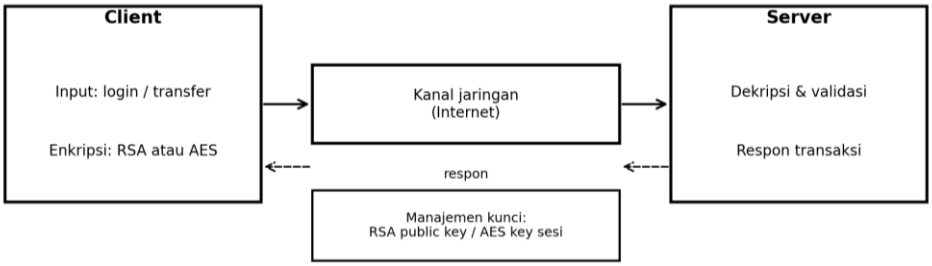
Dalam penelitian ini, algoritma RSA diimplementasikan dengan ukuran kunci 4096-bit dan public exponent 65537, yang dihasilkan menggunakan library cryptography pada Python. Proses enkripsi dan dekripsi RSA menggunakan padding OAEP (Optimal Asymmetric Encryption Padding) dengan SHA-256 . Konfigurasi ini dipilih karena mewakili praktik kriptografi modern yang direkomendasikan untuk tingkat keamanan tinggi serta ketahanan terhadap serangan kriptoanalitik.

Algoritma AES diimplementasikan dengan panjang kunci 128-bit, mode operasi AES-CBC (Cipher Block Chaining), dan padding PKCS7, yang merupakan konfigurasi umum pada sistem keamanan data simetris. AES digunakan untuk enkripsi dan dekripsi payload dalam skenario login dan transfer, dengan tujuan mengevaluasi performa waktu dan penggunaan memori pada berbagai beban kerja. Pencantuman parameter kriptografi ini memastikan bahwa eksperimen bersifat reproducible, transparan, dan sesuai dengan standar evaluasi kriptografi pada sistem client–server.

Pada skenario login, payload berupa string kredensial dengan format “username:password”, sedangkan pada skenario transfer payload berupa string transaksi berformat CSV yang memuat beberapa atribut (jenis transaksi, rekening pengirim, nominal, pesan/berita, PIN, dan rekening penerima). Dengan demikian, payload transfer merepresentasikan beban kerja yang lebih besar dan kompleks dibanding login. Pada implementasi awal, sistem mencatat metrik waktu dan memori untuk setiap operasi enkripsi/dekripsi secara per-event (setiap request) menggunakan pengukuran waktu berbasis time.time() dan pengukuran memori berbasis tracemalloc.

3. Hasil dan Pembahasan

Pada penelitian ini berbasis eksperimen terkontrol dengan membangun prototipe transaksi digital banking berbasis client–server. Dua versi prototipe dibuat dengan alur yang sama: (1) versi RSA, yang mengenkripsi data login dan transfer menggunakan kunci publik penerima; dan (2) versi AES, yang menggunakan satu kunci simetris untuk enkripsi dan dekripsi payload. Pengujian difokuskan pada dua operasi utama, yaitu login dan transfer, karena keduanya merepresentasikan transaksi yang sering terjadi pada layanan perbankan digital.



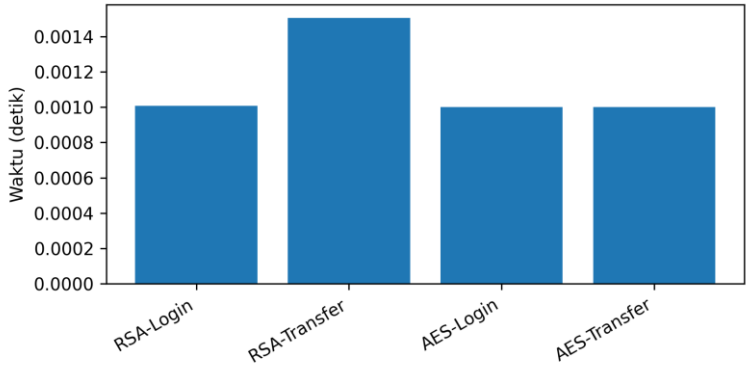
Gambar 1. Prototipe Client – Server

Pada Gambar 1 adalah prototipe client dan server. Dari sisi client, proses dimulai ketika pengguna memasukkan data, seperti informasi login atau permintaan transaksi. Data tersebut kemudian dienkripsi sebelum dikirimkan melalui jaringan internet. Mekanisme enkripsi yang digunakan dapat berupa RSA atau AES, tergantung pada skema keamanan yang diterapkan. Jika menggunakan RSA, client mengenkripsi data menggunakan kunci publik milik server sehingga hanya server yang dapat membukanya dengan kunci privat. Jika menggunakan AES, client mengenkripsi data menggunakan kunci simetris yang telah disepakati sebelumnya. Tujuan utama proses ini adalah memastikan bahwa data yang dikirimkan melalui kanal jaringan private dan hanya bisa diakses oleh pihak yang berwenang.

Dari sisi server, data terenkripsi yang diterima dari client akan melalui proses dekripsi dan validasi. Pada skema RSA, server menggunakan kunci privat untuk mendekripsi data yang sebelumnya dienkripsi dengan kunci publik. Pada skema AES, server menggunakan kunci simetris yang sama dengan client untuk membuka data. Setelah data berhasil didekripsi, server melakukan proses validasi dan eksekusi permintaan, seperti autentikasi pengguna atau pemrosesan transaksi. Hasil dari proses tersebut kemudian dikirimkan kembali ke client dalam bentuk respon melalui jaringan, yang juga dapat dienkripsi menggunakan algoritma yang sama untuk menjaga keamanan komunikasi dua arah.

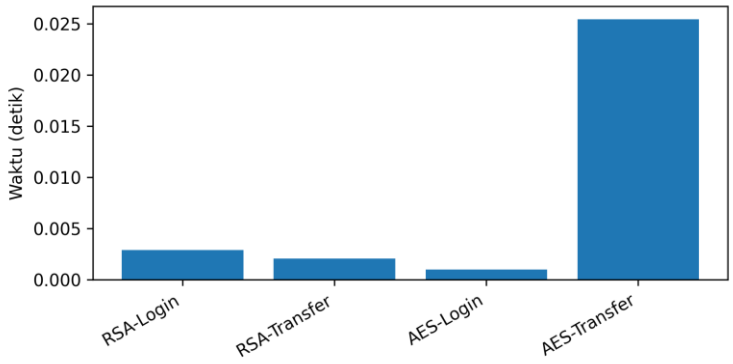
3.1. Perbandingan Waktu Eksekusi pada RSA dan AES.

Bagian ini menyajikan hasil utama pengujian performa. Waktu eksekusi antara RSA dan AES akan diuji pada scenario Login dan Transfer.



Gambar 2. Perbandingan Waktu Eksekusi untuk Enkripsi

Pada Gambar 2 memperlihatkan perbandingan waktu eksekusi algoritma RSA dan AES pada skenario login dan transfer data untuk proses enkripsi. Berdasarkan hasil pengukuran, proses RSA Login membutuhkan waktu sebesar 0,001007 detik, sedangkan RSA Transfer memerlukan waktu yang lebih tinggi, yaitu 0,001505 detik. Sementara itu, algoritma AES menunjukkan kinerja yang lebih efisien, di mana waktu eksekusi untuk AES Login dan AES Transfer masing-masing berada di bawah 0,001 detik. Perbedaan ini mengindikasikan bahwa RSA memiliki overhead komputasi yang lebih besar akibat penggunaan operasi eksponensial modular, khususnya pada proses transfer data. Sebaliknya, AES sebagai algoritma kriptografi simetris mampu memberikan waktu pemrosesan yang lebih cepat dan konsisten. Hasil ini menegaskan bahwa AES lebih sesuai digunakan untuk proses yang menuntut efisiensi dan kecepatan tinggi, sedangkan RSA lebih tepat diterapkan pada skenario yang mengutamakan mekanisme keamanan berbasis kunci publik meskipun dengan biaya komputasi yang lebih besar.



Gambar 3. Perbandingan Waktu Eksekusi untuk Dekripsi.

Pada Gambar 3 tersebut menyajikan hasil pengujian waktu eksekusi algoritma kriptografi RSA dan AES pada dua skenario layanan, yaitu proses login dan transfer data. Sumbu vertikal menunjukkan waktu pemrosesan dalam satuan detik, sedangkan sumbu horizontal merepresentasikan jenis algoritma dan aktivitas yang diuji. Berdasarkan grafik, proses RSA-Login membutuhkan waktu sekitar 0,003 detik, sedangkan RSA-Transfer menunjukkan waktu eksekusi yang lebih rendah, yaitu sekitar 0,002 detik. Pada algoritma AES, proses AES-Login memiliki waktu pemrosesan paling singkat, yaitu sekitar 0,001 detik, sementara AES-Transfer memperlihatkan lonjakan waktu yang sangat signifikan hingga sekitar 0,025 detik.

Perbedaan nilai waktu ini menunjukkan bahwa kinerja algoritma kriptografi sangat dipengaruhi oleh karakteristik operasi yang dilakukan serta skenario penggunaan. Waktu eksekusi AES yang meningkat drastis pada proses transfer mengindikasikan adanya beban pemrosesan tambahan, yang dapat disebabkan oleh ukuran data yang lebih besar atau mekanisme pemrosesan blok yang berulang. Sebaliknya, RSA menunjukkan waktu eksekusi yang relatif lebih stabil pada kedua skenario, meskipun secara teoretis RSA memiliki kompleksitas komputasi yang lebih tinggi. Temuan ini menegaskan bahwa evaluasi performa algoritma kriptografi tidak hanya ditentukan oleh jenis algoritma, tetapi juga oleh konteks dan karakteristik beban kerja yang diterapkan pada sistem.

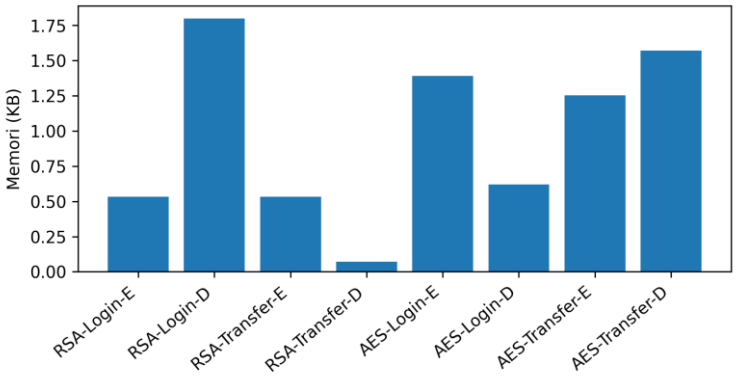
Tabel 1. Hasil Waktu Eksekusi (s)

Algoritma	Skenario	Enkripsi	Dekripsi
RSA	Login	0.001007	0.002883
RSA	Transfer	0.001505	0.002084
AES	Login	0.000001	0.000001
AES	Transfer	0.000001	0.025419

Tabel 1 menunjukkan hasil perbandingan waktu eksekusi dari RSA dan AES. Berdasarkan hasil pengujian, AES menunjukkan performa enkripsi–dekripsi paling cepat pada skenario login (masing-masing sekitar 0,000001 detik), jauh lebih rendah dibanding RSA yang membutuhkan 0,001007 detik untuk enkripsi dan 0,002883 detik untuk dekripsi. Pada skenario transfer, RSA tetap relatif stabil dengan waktu enkripsi 0,001505 detik dan dekripsi 0,002084 detik, sedangkan AES tetap sangat cepat pada enkripsi yaitu 0,000001 detik namun mengalami lonjakan pada proses dekripsi hingga 0,025419 detik. Pola ini mengindikasikan bahwa AES unggul signifikan untuk beban kerja ringan seperti login, tetapi pada transfer terdapat overhead tertentu di sisi dekripsi tergantung pada ukuran payload yang membuat kinerjanya menurun dan perlu ditelusuri lebih lanjut sebelum dijadikan pilihan utama untuk skenario transfer.

3.2. Perbandingan Konsumsi Memori pada RSA dan AES.

Untuk mengetahui performa dari penggunaan memori pada RSA dan AES, dilakukan pengujian seperti pada Gambar 4. Pada pengujian ini menyajikan perbandingan penggunaan memori pada algoritma RSA dan AES untuk dua skenario layanan, yaitu login dan transfer data, masing-masing pada proses enkripsi (E) dan dekripsi (D). Pada algoritma RSA, terlihat bahwa proses dekripsi login (RSA-Login-D) memerlukan penggunaan memori paling besar, yaitu sekitar 1,8 kb, sedangkan proses enkripsi login (RSA-Login-E) dan enkripsi transfer (RSA-Transfer-E) berada pada kisaran 0,53 kb. Proses dekripsi transfer RSA (RSA-Transfer-D) menunjukkan penggunaan memori paling rendah, yaitu 0.07, yang mengindikasikan variasi kebutuhan memori yang signifikan antar tahapan operasi RSA.



Gambar 4. Perbandingan Konsumsi Memori

Sementara itu, pada algoritma AES, penggunaan memori cenderung lebih merata namun tetap menunjukkan perbedaan antara enkripsi dan dekripsi. Proses AES-Login-E dan AES-Transfer-D membutuhkan memori relatif lebih besar, masing-masing sekitar 1,39 kb dan 1,57 kb, dibandingkan proses AES-Login-D yaitu 0,62 kb. Adapun AES-Transfer-E menunjukkan penggunaan memori sekitar 1,25 kb. Hasil ini menunjukkan bahwa kebutuhan memori tidak hanya dipengaruhi oleh jenis algoritma kriptografi, tetapi juga oleh jenis layanan serta tahapan proses yang dijalankan. Temuan ini mengindikasikan bahwa evaluasi efisiensi algoritma kriptografi perlu mempertimbangkan aspek memori secara terpisah dari waktu eksekusi, terutama pada sistem dengan keterbatasan sumber daya. Tabel 2 menunjukkan detail dari hasil konsumsi memori oleh kedua algoritma.

Tabel 2. Hasil Konsumsi Memori (kb)

Algoritma	Skenario	Enkripsi	Dekripsi
RSA	Login	0.53	1.8
RSA	Transfer	0.53	0.07
AES	Login	1.39	0.62
AES	Transfer	1.25	1.57

Berdasarkan Tabel 2, dapat disimpulkan bahwa pola konsumsi memori pada kedua algoritma tidak sepenuhnya sejalan dengan pola waktu eksekusi, karena setiap skenario login dan transfer serta tahapan proses enkripsi dan dekripsi menghasilkan kebutuhan memori yang berbeda. RSA cenderung menunjukkan penggunaan memori yang kontras terutama pada proses dekripsi login yang lebih tinggi, sedangkan AES relatif lebih merata namun tetap memperlihatkan peningkatan pada proses tertentu seperti enkripsi login dan dekripsi transfer. Dengan demikian, pemilihan algoritma kriptografi untuk sistem client–server tidak cukup hanya mempertimbangkan kecepatan, tetapi juga harus memperhatikan efisiensi memori agar implementasi tetap stabil dan sesuai dengan keterbatasan sumber daya, khususnya pada layanan transaksi yang berpotensi memproses data lebih kompleks.

4. Kesimpulan

Berdasarkan pengujian pada skenario digital banking (login dan transfer), AES menunjukkan performa yang lebih baik dibanding RSA pada beberapa tahapan penting. Pada proses enkripsi, AES lebih efisien dengan peningkatan kecepatan 0,7% pada login dan 33,6% pada transfer, sehingga dapat mendukung kebutuhan enkripsi payload transaksi yang memerlukan respons cepat. Pada dekripsi login, AES juga memberikan peningkatan signifikan, yaitu 65,3% lebih cepat sekaligus lebih hemat memori sebesar 65,5%, yang menandakan AES cocok untuk aktivitas autentikasi dan perlindungan data sensitif dengan overhead yang relatif ringan. Akan tetapi, pada dekripsi transfer, AES mengalami penurunan kinerja dengan latensi 12,2 kali lebih lambat dan konsumsi memori

yang meningkat 2112,7% dibanding RSA. Dengan demikian, dapat disimpulkan bahwa AES memiliki keunggulan utama dalam efisiensi enkripsi dan dekripsi pada transaksi sederhana, namun pada transaksi transfer diperlukan optimasi implementasi agar kinerja tetap stabil. Penelitian lanjutan direkomendasikan menggunakan pengujian berulang dengan variasi ukuran payload serta pencatatan waktu presisi tinggi untuk mendapatkan hasil yang lebih representatif terhadap kondisi sistem digital banking nyata.

Daftar Pustaka

[1] Y. Zhong, "An Overview of RSA and OAEP Padding," Highlights Sci. Eng. Technol., vol. 1, pp. 82–86, Jun. 2022, doi: 10.54097/hset.v1i.431.

[2] A. Aminudin, L. Hakim, I. Nuryasin, and H. R. Santiyas, "Kriptosistem Hybrid Algoritme RSA dan El-Gamal Menggunakan Socket TCP pada Instant Messaging," JRST (Jurnal Ris. Sains dan Teknol., vol. 8, no. 1, p. 1, Mar. 2024, doi: 10.30595/jrst.v8i1.17124.

[3] V. Veronica, R. S. Oetama, and A. Ramadhan, "Incorporating rivest-shamir-adleman algorithm and advanced encryption standard in payment gateway system," TELKOMNIKA (Telecommunication Comput. Electron. Control., vol. 22, no. 3, p. 629, Jun. 2024, doi: 10.12928/telkomnika.v22i3.25578.

[4] R. Verma and J. Dhiman, "Implementation of Improved Cryptography Algorithm," Int. J. Inf. Technol. Comput. Sci., vol. 14, no. 2, pp. 45–53, Apr. 2022, doi: 10.5815/ijitcs.2022.02.04.

[5] A. Pratiwi and A. A. Tahir, Muhlis Nawafilillah Alvaradis, "Implementation of RSA Asymmetric Cryptography using GPG and Kelopatra for School Data Security," J. Ris. Inform., vol. 7, no. 3, pp. 170–176, 2025, doi: https://doi.org/10.34288/jri.v7i3.360.

[6] A. Yeboah-Ofori, I. Darvishi, and A. S. Opeyemi, "Enhancement of Big Data Security in Cloud Computing Using RSA Algorithm," in 2023 10th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, Aug. 2023, pp. 312–319. doi: 10.1109/FiCloud58648.2023.00053.

[7] F. O. Mojisola, S. Misra, C. F. Febisola, O. Abayomi-alli, and G. Sengul, "An improved random bit-stuffing technique with a modified RSA algorithm for resisting attacks in information security (RBMRSA)," Egypt. Informatics J., vol. 23, no. 2, pp. 291–301, 2022, doi: 10.1016/j.eij.2022.02.001.

[8] D. Shivaramakrishna and M. Nagaratna, "A novel hybrid cryptographic framework for secure data storage in cloud computing: Integrating AES-OTP and RSA with adaptive key management and Time-Limited access control," Alexandria Eng. J., vol. 84, pp. 275–284, Dec. 2023, doi: 10.1016/j.aej.2023.10.054.

[9] H. Mestiri, "Evaluating AES Security: Correlation Power Analysis Attack Implementation using the Switching Distance Power Model," Eng. Technol. Appl. Sci. Res., vol. 15, no. 1, pp. 20314–20320, Feb. 2025, doi: 10.48084/etasr.9728.

[10] R. Ganesh, B. U. I. Khan, A. R. Khan, and A. Bin Kamsin, "A panoramic survey of the advanced encryption standard: from architecture to security analysis, key management, real-world applications, and post-quantum challenges," Int. J. Inf. Secur., vol. 24, no. 5, p. 216, Oct. 2025, doi: 10.1007/s10207-025-01116-x.

[11] M. Carvalho, F. Sá, and J. Bernardino, "Evaluation of the Impact of AES Encryption on Query Read Performance Across Oracle, MySQL, and SQL Server Databases," Cryptography, vol. 9, no. 4, p. 77, Nov. 2025, doi: 10.3390/cryptography9040077.

[12] A. Barenghi, D. Carrera, S. Mella, A. Pace, G. Pelosi, and R. Susella, "Profiled side channel attacks against the RSA cryptosystem using neural networks," J. Inf. Secur. Appl., vol. 66, p. 103122, May 2022, doi: 10.1016/j.jisa.2022.103122.

[13] S. de la Fe, H.-B. Park, B.-Y. Sim, D.-G. Han, and C. Ferrer, "Profiling Attack against RSA Key Generation Based on a Euclidean Algorithm," Information, vol. 12, no. 11, p. 462, Nov. 2021, doi: 10.3390/info12110462.