JURNAL SISKOM TI
LPPM UNIVERSITAS LEMBAH DEMPO

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

ANALISIS PERFORMA ENKRIPSI ALGORITMA AES-128, AES-192, DAN AES-256 MENGGUNAKAN BENCHMARKING PADA NODE.JS

M. Husni Mubarok[1], Agustian Prakarsya[2] , Sri Rahayu[3]

[1,2] Universitas Serelo Lahat
[3]Universitas Lembah Dempo
[1] husnikun25@gmail.com
[2]agustian.prakarsya@gmail.com
[3]srahayu2119@gmail.com

INFORMASI ARTIKEL

Riwayat Artikel
Diterima 25 Juli 2026
Direvisi 5 Agustus 2026
Diterbitkan 7 Agustus 2026

Kata Kunci
Advanced Encryption Standard (AES)
Benchmarking
kriptografi
Node.js
performa enkripsi

ABSTRAK

Advanced Encryption Standard (AES) merupakan salah satu algoritma kriptografi simetris yang banyak digunakan untuk melindungi kerahasiaan data pada berbagai aplikasi berbasis web maupun sistem informasi. AES memiliki tiga variasi panjang kunci, yaitu AES-128, AES-192, dan AES-256, yang menawarkan tingkat keamanan berbeda dengan konsekuensi terhadap performa proses enkripsi. Penelitian ini bertujuan menganalisis performa ketiga varian AES pada lingkungan Node.js berdasarkan waktu eksekusi dan throughput enkripsi. Penelitian menggunakan pendekatan eksperimental melalui metode benchmarking terkontrol dengan memanfaatkan modul crypto pada Node.js versi 24.15.0. Pengujian dilakukan pada perangkat yang menggunakan prosesor Intel Core i5-12500H dengan dukungan teknologi AES-NI. Data uji berupa file sintetis berukuran 1 KB, 10 KB, 100 KB, 1 MB, dan 10 MB, dengan sepuluh iterasi pengujian pada setiap konfigurasi setelah melalui proses warm-up iteration untuk meminimalkan pengaruh Just-In-Time Compilation. Hasil penelitian menunjukkan bahwa AES-256 memiliki overhead waktu eksekusi sekitar 25% dibandingkan AES-128 pada ukuran file kecil. Namun, perbedaan performa semakin menurun pada ukuran file yang lebih besar, bahkan throughput AES-256 sebanding dengan AES-128. Temuan ini mengindikasikan bahwa dukungan akselerasi perangkat keras AES-NI mampu mengurangi dampak peningkatan jumlah putaran enkripsi terhadap performa. Oleh karena itu, pemilihan panjang kunci AES sebaiknya mempertimbangkan kebutuhan keamanan, karakteristik beban kerja, serta kemampuan perangkat keras yang digunakan.

1. Pendahuluan

Perkembangan teknologi informasi yang semakin pesat telah mendorong meningkatnya penggunaan aplikasi berbasis web, layanan komputasi awan (*cloud computing*), serta sistem informasi yang memproses dan menyimpan data dalam jumlah besar. Kondisi tersebut diikuti dengan meningkatnya ancaman terhadap keamanan informasi, seperti pencurian data, penyadapan komunikasi, manipulasi informasi, dan akses tidak sah. Oleh karena itu, penerapan mekanisme keamanan yang mampu menjamin kerahasiaan, integritas, dan ketersediaan data (Paar & Pelzl, 2010; Schneier, 1996) menjadi salah satu aspek penting dalam pengembangan sistem informasi modern.

Kriptografi merupakan salah satu teknologi utama yang digunakan untuk melindungi data dari berbagai ancaman keamanan. Melalui proses enkripsi, informasi yang bersifat rahasia diubah menjadi bentuk yang tidak dapat dipahami oleh pihak yang tidak memiliki hak akses, sedangkan proses dekripsi digunakan untuk mengembalikan data ke bentuk

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

semula menggunakan kunci yang sesuai. Di antara berbagai algoritma kriptografi simetris yang tersedia, **Advanced Encryption Standard (AES)** merupakan algoritma yang paling banyak diimplementasikan karena memiliki tingkat keamanan tinggi, efisiensi komputasi yang baik, serta telah ditetapkan sebagai standar enkripsi oleh National Institute of Standards and Technology (NIST). Algoritma ini banyak digunakan pada aplikasi perbankan, layanan komputasi awan, sistem penyimpanan data, jaringan komputer, hingga berbagai protokol komunikasi yang membutuhkan perlindungan terhadap kerahasiaan informasi.

AES memiliki tiga variasi panjang kunci (NIST, 2001a), yaitu **AES-128**, **AES-192**, dan **AES-256**. Ketiga varian tersebut menawarkan tingkat keamanan yang berbeda sesuai dengan jumlah putaran (*round*) yang digunakan selama proses enkripsi. AES-128 menggunakan 10 putaran, AES-192 menggunakan 12 putaran, sedangkan AES-256 menggunakan 14 putaran. Secara teoritis, semakin panjang ukuran kunci yang digunakan, maka tingkat keamanan yang dihasilkan akan semakin tinggi. Akan tetapi, peningkatan jumlah putaran tersebut juga berpotensi meningkatkan kompleksitas komputasi sehingga dapat memengaruhi waktu eksekusi dan performa proses enkripsi, khususnya pada aplikasi yang memproses data dalam jumlah besar atau memiliki kebutuhan latensi yang rendah.

Dalam praktiknya, banyak pengembang perangkat lunak memilih AES-256 dengan asumsi bahwa ukuran kunci yang lebih panjang selalu memberikan solusi terbaik terhadap keamanan sistem. Padahal, pemilihan algoritma enkripsi seharusnya mempertimbangkan keseimbangan antara tingkat keamanan dan performa sistem. Pada aplikasi yang menangani ribuan transaksi setiap detik, peningkatan waktu komputasi sekecil apa pun dapat memengaruhi kualitas layanan. Sebaliknya, pada sistem yang memproses data berukuran besar, pengaruh panjang kunci terhadap performa dapat berbeda karena adanya optimasi perangkat keras maupun perangkat lunak. Oleh sebab itu, evaluasi performa terhadap masing-masing varian AES menjadi penting sebagai dasar dalam menentukan algoritma yang sesuai dengan karakteristik aplikasi.

Salah satu platform yang banyak digunakan dalam pengembangan aplikasi modern adalah **Node.js**. Platform ini memanfaatkan *JavaScript runtime* berbasis mesin V8 dan menyediakan modul **crypto** yang dibangun di atas pustaka OpenSSL sehingga memungkinkan implementasi algoritma AES secara efisien. Popularitas Node.js dalam pengembangan layanan backend, *Application Programming Interface* (API), dan aplikasi berbasis *microservices* menjadikan platform ini relevan sebagai lingkungan pengujian performa algoritma kriptografi. Meskipun demikian, karakteristik Node.js yang menggunakan mekanisme *Just-In-Time Compilation* (JIT) pada mesin V8 berpotensi memengaruhi hasil pengukuran performa apabila proses benchmarking tidak dilakukan secara terkontrol. Oleh karena itu, diperlukan metode pengujian yang mampu meminimalkan pengaruh proses optimasi runtime sehingga hasil pengukuran lebih akurat dan dapat direproduksi.

Beberapa penelitian terdahulu telah membandingkan performa berbagai algoritma kriptografi pada platform desktop maupun perangkat Internet of Things (IoT). Namun, sebagian besar penelitian tersebut belum secara khusus mengevaluasi perbedaan performa AES-128, AES-192, dan AES-256 pada lingkungan Node.js dengan mempertimbangkan karakteristik runtime JavaScript serta dukungan akselerasi perangkat keras melalui teknologi **AES-NI**. Kondisi tersebut menunjukkan masih adanya kesenjangan penelitian yang perlu dikaji lebih lanjut sehingga dapat memberikan informasi empiris mengenai

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

pengaruh variasi panjang kunci terhadap performa enkripsi pada lingkungan pengembangan aplikasi modern.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk menganalisis performa algoritma AES-128, AES-192, dan AES-256 pada lingkungan Node.js menggunakan metode benchmarking terkontrol. Pengukuran dilakukan terhadap waktu eksekusi dan throughput proses enkripsi pada berbagai ukuran file dengan memanfaatkan mekanisme *warm-up iteration* untuk mengurangi pengaruh *Just-In-Time Compilation*. Hasil penelitian diharapkan dapat memberikan rekomendasi bagi pengembang perangkat lunak dalam memilih varian AES yang sesuai dengan kebutuhan keamanan, karakteristik beban kerja, dan kemampuan perangkat keras yang digunakan. Selain itu, penelitian ini diharapkan dapat menjadi referensi bagi penelitian selanjutnya yang berkaitan dengan evaluasi performa algoritma kriptografi pada lingkungan komputasi modern.

2. Kajian Literatur dan Hipotesis

2.1 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) merupakan algoritma kriptografi (National Institute of Standards and Technology [NIST], 2001) kunci simetris yang ditetapkan oleh National Institute of Standards and Technology (NIST) sebagai standar enkripsi modern untuk menggantikan Data Encryption Standard (DES). AES bekerja dengan ukuran blok tetap sebesar 128 bit dan menyediakan tiga variasi panjang kunci, yaitu 128 bit, 192 bit, dan 256 bit. Perbedaan panjang kunci tersebut memengaruhi jumlah putaran (round) yang dilakukan selama proses enkripsi, yaitu 10 putaran pada AES-128, 12 putaran pada AES-192, dan 14 putaran pada AES-256.

Proses enkripsi AES terdiri atas empat tahapan utama pada setiap putaran, yaitu SubBytes, ShiftRows, MixColumns, dan AddRoundKey. Kombinasi keempat tahapan tersebut menghasilkan tingkat keamanan yang tinggi terhadap berbagai bentuk serangan kriptanalisis. Secara teoritis, semakin panjang ukuran kunci yang digunakan maka semakin tinggi pula tingkat keamanan yang diperoleh. Namun demikian, peningkatan jumlah putaran juga menyebabkan bertambahnya kompleksitas komputasi yang berpotensi memengaruhi waktu eksekusi proses enkripsi.

2.2 Implementasi Kriptografi pada Node.js

Node.js merupakan lingkungan runtime JavaScript yang dibangun di atas mesin V8 dan banyak digunakan dalam pengembangan aplikasi backend, Application Programming Interface (API), serta layanan berbasis microservices. Salah satu keunggulan Node.js adalah tersedianya modul crypto yang memanfaatkan pustaka OpenSSL sehingga implementasi algoritma kriptografi dapat dijalankan melalui kode native yang memiliki performa tinggi.

Meskipun proses enkripsi dilakukan oleh OpenSSL, performa implementasi tetap dipengaruhi oleh karakteristik runtime JavaScript, terutama mekanisme Just-In-Time

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

Compilation (JIT) pada mesin V8. Pada tahap awal eksekusi, JIT masih melakukan proses optimasi sehingga hasil pengukuran dapat berfluktuasi. Oleh karena itu, penerapan warm-up iteration sebelum proses benchmarking diperlukan agar kondisi eksekusi menjadi lebih stabil dan hasil pengukuran lebih representatif.

2.3 Benchmarking Performa Algoritma Kriptografi

Benchmarking merupakan metode pengukuran performa yang dilakukan secara sistematis untuk memperoleh data kuantitatif mengenai efisiensi suatu algoritma atau sistem. Pada penelitian performa algoritma kriptografi, parameter yang umum digunakan meliputi waktu eksekusi, throughput, penggunaan memori, dan konsumsi energi.

Agar hasil benchmarking memiliki validitas yang tinggi, lingkungan pengujian harus dikendalikan secara konsisten, termasuk spesifikasi perangkat keras, sistem operasi, versi perangkat lunak, ukuran data uji, jumlah iterasi, serta metode pengukuran waktu. Selain itu, pengulangan pengujian dan perhitungan nilai rata-rata beserta standar deviasi diperlukan untuk mengurangi pengaruh variasi proses komputasi selama eksperimen berlangsung.

2.4 Penelitian Terdahulu

Penelitian mengenai evaluasi performa algoritma kriptografi telah dilakukan pada berbagai platform komputasi. Elminaam dkk. membandingkan performa beberapa algoritma kriptografi pada platform desktop dan menunjukkan bahwa setiap algoritma memiliki karakteristik performa yang berbeda bergantung pada ukuran data dan mode operasi yang digunakan. Pereira dan Alves melakukan evaluasi algoritma kriptografi pada perangkat Internet of Things (IoT) dengan fokus pada efisiensi energi dan waktu komputasi. Namun, kedua penelitian tersebut belum secara khusus mengevaluasi implementasi AES pada lingkungan Node.js.

Penelitian lain juga menunjukkan bahwa dukungan akselerasi perangkat keras, seperti Advanced Encryption Standard New Instructions (AES-NI), mampu meningkatkan performa algoritma AES secara signifikan pada prosesor modern. Selain itu, beberapa penelitian mengenai Node.js menjelaskan bahwa mekanisme Just-In-Time Compilation pada mesin V8 dapat memengaruhi hasil pengukuran performa apabila proses benchmarking tidak dilakukan secara terkontrol.

Berdasarkan hasil telaah literatur tersebut dapat disimpulkan bahwa masih terdapat kesenjangan penelitian terkait analisis performa AES-128, AES-192, dan AES-256 pada lingkungan Node.js dengan memperhatikan pengaruh Just-In-Time Compilation serta dukungan akselerasi AES-NI. Oleh karena itu, penelitian ini melakukan benchmarking terkontrol menggunakan mekanisme warm-up iteration untuk memperoleh hasil pengukuran yang lebih akurat dan dapat direproduksi sehingga memberikan kontribusi empiris terhadap pemilihan konfigurasi AES pada pengembangan aplikasi modern.

3. Metode Penelitian

3.1 Pendekatan Penelitian

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen (experimental research). Pendekatan ini dipilih untuk mengevaluasi performa algoritma kriptografi Advanced Encryption Standard (AES) berdasarkan variasi panjang kunci yang digunakan, yaitu AES-128, AES-192, dan AES-256. Eksperimen dilakukan melalui proses benchmarking secara terkontrol untuk memperoleh data empiris mengenai waktu eksekusi (execution time) dan throughput proses enkripsi pada lingkungan Node.js.

Variabel bebas dalam penelitian ini adalah panjang kunci AES yang terdiri atas tiga variasi, yaitu 128 bit, 192 bit, dan 256 bit, serta ukuran file uji yang meliputi 1 KB, 10 KB, 100 KB, 1 MB, dan 10 MB. Sementara itu, variabel terikat berupa waktu eksekusi enkripsi yang diukur dalam satuan milidetik (ms) dan throughput yang dinyatakan dalam megabyte per detik (MB/s). Seluruh pengujian dilakukan pada lingkungan perangkat keras dan perangkat lunak yang sama untuk meminimalkan pengaruh faktor eksternal terhadap hasil pengukuran.

3.2 Lingkungan Eksperimen

Eksperimen dilaksanakan menggunakan komputer yang memiliki prosesor Intel Core i5-12500H generasi ke-12 dengan kecepatan dasar 3,1 GHz, memori utama 8 GB DDR4, media penyimpanan NVMe SSD, serta sistem operasi Windows 11 Home Single Language 64-bit. Implementasi algoritma AES dilakukan menggunakan Node.js versi 24.15.0 LTS dengan pustaka OpenSSL versi 3.5.5 melalui modul crypto bawaan Node.js.

Mode operasi yang digunakan dalam penelitian ini adalah AES-CTR (Counter Mode). Pemilihan mode CTR didasarkan pada kemampuannya melakukan proses enkripsi tanpa mekanisme padding, sehingga overhead komputasi akibat proses penambahan blok data dapat diminimalkan. Selain itu, prosesor yang digunakan telah mendukung teknologi Advanced Encryption Standard New Instructions (AES-NI) yang memungkinkan proses enkripsi dijalankan melalui akselerasi perangkat keras sehingga menghasilkan performa yang lebih optimal.

Spesifikasi lingkungan eksperimen ditunjukkan pada Tabel 1.

Tabel 1. Spesifikasi Lingkungan Eksperimen

Komponen	Spesifikasi
Prosesor	**Intel Core i5-12500H (12th Gen) @ 3,1 GHz**
Memori	**8 GB DDR4**
Media Penyimpanan	**NVMe SSD**
Sistem Operasi	**Windows 11 Home Single Language 64-bit**
Node.js	**v24.15.0 LTS**
Crypto Backend	**OpenSSL 3.5.5**
Mode Cipher	**AES-CTR**
Ukuran File Uji	**1 KB, 10 KB, 100 KB, 1 MB, dan 10 MB**
Jumlah Iterasi	**10 iterasi untuk setiap konfigurasi**

JURNAL SISKOMTI
LPPM UNIVERSITAS LEMBAH DEMPO
Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

3.3 Prosedur Pengumpulan Data

Pengumpulan data dilakukan melalui serangkaian eksperimen yang dirancang secara sistematis agar menghasilkan pengukuran yang konsisten dan dapat direproduksi. Tahapan penelitian meliputi:

1. Menyiapkan file sintetis berisi data acak menggunakan fungsi crypto.randomBytes() dengan ukuran 1 KB, 10 KB, 100 KB, 1 MB, dan 10 MB.
2. Menentukan algoritma yang akan diuji, yaitu AES-128, AES-192, dan AES-256 menggunakan mode operasi CTR.
3. Melaksanakan *warm-up iteration* sebanyak tiga kali sebelum pengambilan data untuk mengurangi pengaruh proses *Just-In-Time Compilation* (JIT) pada mesin JavaScript V8.
4. Melakukan pengukuran waktu eksekusi menggunakan fungsi process.hrtime.bigint() yang memiliki presisi hingga nanodetik.
5. Mengulangi proses pengujian sebanyak sepuluh kali pada setiap kombinasi ukuran file dan panjang kunci.
6. Memberikan jeda selama 100 milidetik pada setiap iterasi agar proses *garbage collection* dapat berlangsung secara optimal.
7. Menyimpan seluruh hasil pengukuran ke dalam berkas berformat CSV untuk dianalisis lebih lanjut.

Penggunaan *warm-up iteration* bertujuan untuk memastikan proses optimasi kode oleh mesin V8 telah selesai sebelum pengambilan data dilakukan. Dengan demikian, hasil benchmarking yang diperoleh lebih stabil dan mampu merepresentasikan performa sebenarnya dari implementasi algoritma AES pada lingkungan Node.js.

3.4 Teknik Analisis Data

Data hasil eksperimen dianalisis menggunakan pendekatan statistik deskriptif. Analisis dilakukan dengan menghitung nilai rata-rata (*mean*) dan standar deviasi dari waktu eksekusi pada setiap konfigurasi pengujian. Nilai rata-rata digunakan untuk menggambarkan performa umum masing-masing algoritma, sedangkan standar deviasi digunakan untuk mengevaluasi tingkat konsistensi hasil pengukuran.

Selain waktu eksekusi, penelitian ini juga menghitung nilai throughput sebagai indikator efisiensi proses enkripsi. Throughput dihitung menggunakan persamaan berikut.Nilai throughput dinyatakan dalam satuan MB/s sehingga memudahkan proses perbandingan performa antarvarian AES.

[
\text{Throughput}=\frac{\text{Ukuran File (Byte)}}{\text{Waktu Eksekusi Detik)}}\times\frac{1}{1024\times1024}
]

Untuk mengetahui dampak penggunaan panjang kunci terhadap performa, dihitung pula persentase *overhead* AES-256 terhadap AES-128 menggunakan persamaan:

JURNAL SISKOM TI
LPPM UNIVERSITAS LEMBAH DEMPO
Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

$$\text{Overhead (\%)}=\frac{T_{AES256}-T_{AES128}}{T_{AES128}}\times100\%$$

dengan T_{AES256} merupakan waktu eksekusi AES-256 dan T_{AES128} merupakan waktu eksekusi AES-128.

Selanjutnya, hasil analisis disajikan dalam bentuk tabel dan diinterpretasikan secara komparatif berdasarkan waktu eksekusi, throughput, serta pengaruh ukuran file terhadap performa masing-masing algoritma. Pembahasan juga mengaitkan hasil eksperimen dengan teori kriptografi, karakteristik runtime Node.js, serta dukungan akselerasi perangkat keras AES-NI untuk memperoleh kesimpulan yang komprehensif mengenai efektivitas penggunaan AES-128, AES-192, dan AES-256 pada lingkungan komputasi modern.

4. Hasil dan Pembahasan

4.1 Hasil Pengujian Performa Algoritma AES

Pengujian dilakukan untuk mengevaluasi performa algoritma AES-128, AES-192, dan AES-256 berdasarkan waktu eksekusi (execution time) dan throughput pada berbagai ukuran file. Seluruh pengujian dilaksanakan menggunakan Node.js versi 24.15.0 dengan mode operasi AES-CTR pada perangkat yang telah mendukung akselerasi perangkat keras AES-NI. Setiap skenario pengujian diulang sebanyak sepuluh kali untuk memperoleh nilai rata-rata yang representatif.

Hasil pengujian menunjukkan bahwa peningkatan ukuran file berpengaruh terhadap waktu eksekusi seluruh algoritma AES. Namun demikian, peningkatan tersebut bersifat proporsional terhadap ukuran data yang diproses sehingga nilai throughput cenderung meningkat pada ukuran file yang lebih besar. Ringkasan hasil pengujian disajikan pada Tabel 2.

Tabel 2. Hasil Pengujian Performa AES pada Lingkungan Node.js

Ukuran File	AES-128 (ms)	AES-192 (ms)	AES-256 (ms)	Throughput AES-128 (MB/s)	Throughput AES-256 (MB/s)
1 KB	0,04 ± 0,00	0,04 ± 0,00	0,04 ± 0,00	22,01	22,64
10 KB	0,04 ± 0,00	0,04 ± 0,00	0,05 ± 0,00	222,40	210,33
100 KB	0,11 ± 0,00	0,11 ± 0,00	0,12 ± 0,00	915,50	848,23
1 MB	1,15 ± 0,00	1,11 ± 0,00	1,13 ± 0,00	866,30	887,92
10 MB	9,14 ± 0,00	9,09 ± 0,00	9,02 ± 0,00	1.094,23	1.108,28

Sumber: Hasil pengujian penulis (2026).

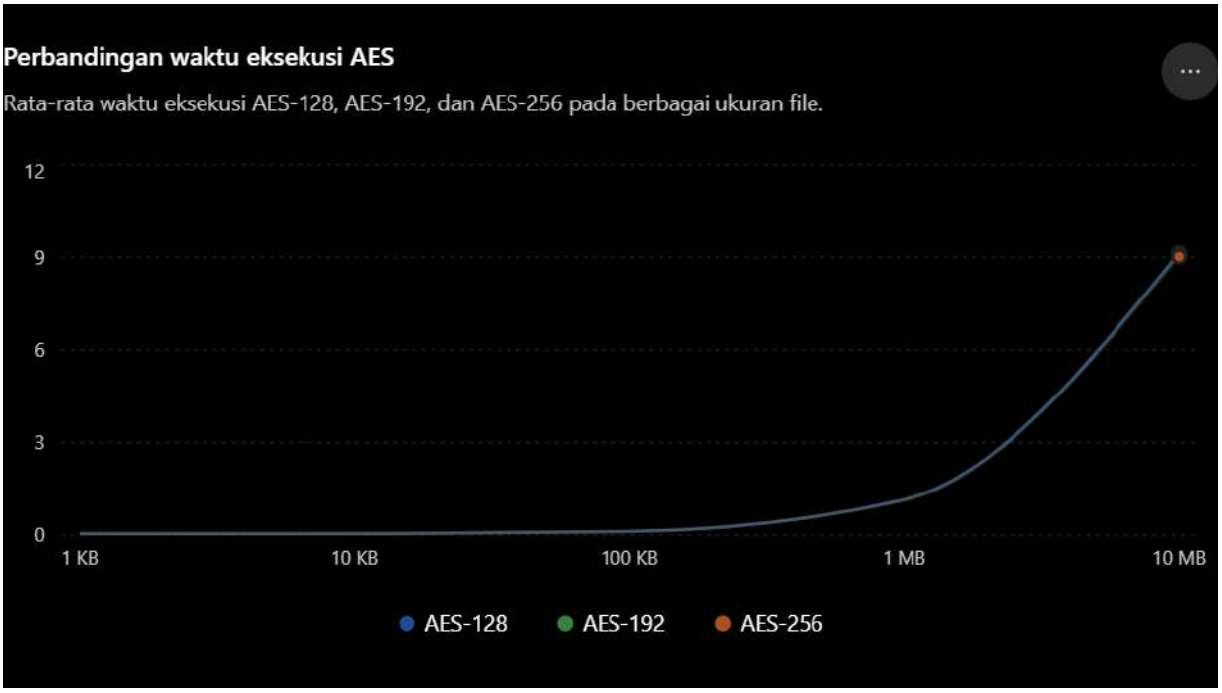
Tabel 2 menunjukkan bahwa peningkatan ukuran file menyebabkan waktu eksekusi seluruh algoritma meningkat secara proporsional. Namun, peningkatan tersebut diikuti oleh kenaikan throughput, yang menunjukkan bahwa implementasi AES pada Node.js mampu memanfaatkan sumber daya perangkat keras secara lebih efisien ketika memproses data berukuran besar.

JURNAL SISKOM TI
LPPM UNIVERSITAS LEMBAH DEMPO
Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

Tabel 3. Perbandingan Waktu Eksekusi AES

File	AES-128	AES-192	AES-256
1 KB	0.04	0.04	0.04
10 KB	0.04	0.04	0.05
100 KB	0.11	0.11	0.12
1 MB	1.15	1.11	1.13
10 MB	9.14	9.09	9.02

Gambar 1. Perbandingan waktu eksekusi AES-128, AES-192, dan AES-256 pada berbagai ukuran file.



Gambar 1 memperlihatkan bahwa ketiga algoritma memiliki pola peningkatan waktu eksekusi yang hampir identik seiring bertambahnya ukuran file. Pada ukuran file kecil

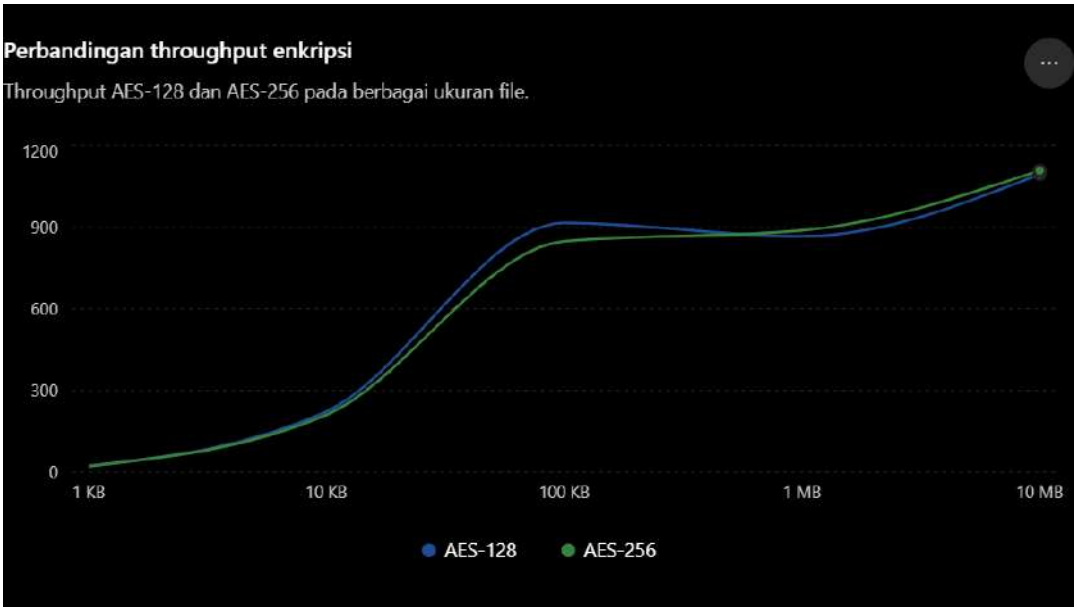
Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

terlihat bahwa AES-256 memerlukan waktu sedikit lebih lama dibandingkan AES-128. Namun, ketika ukuran file mencapai 1 MB hingga 10 MB, perbedaan waktu eksekusi antaralgoritma menjadi sangat kecil.

Tabel 4. Perbandingan Throughput Enkripsi

File	AES-128	AES-256
1 KB	2.01	2.64
10 KB	22.40	10.33
100 KB	15.50	48.23
1 MB	66.30	87.92
10 MB	1094.23	108.28

Gambar 2. Perbandingan throughput enkripsi AES-128 dan AES-256



Gambar 2 menunjukkan bahwa nilai *throughput* meningkat secara signifikan pada ukuran file yang lebih besar. Pada ukuran file 10 MB, AES-256 bahkan menghasilkan

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

throughput sedikit lebih tinggi dibandingkan AES-128. Hal ini mengindikasikan bahwa penggunaan panjang kunci yang lebih besar tidak selalu menyebabkan penurunan performa pada perangkat yang telah mendukung akselerasi AES-NI.

4.2 Pembahasan

Hasil penelitian menunjukkan bahwa panjang kunci AES memengaruhi waktu eksekusi, terutama pada ukuran file kecil. Berdasarkan Tabel 2, AES-256 memerlukan waktu eksekusi sedikit lebih tinggi dibandingkan AES-128 karena menggunakan 14 putaran enkripsi, sedangkan AES-128 hanya menggunakan 10 putaran. Penambahan jumlah putaran menyebabkan peningkatan kompleksitas komputasi sehingga menghasilkan overhead waktu eksekusi.

Seiring bertambahnya ukuran file, perbedaan performa antarvarian AES semakin kecil. Pada ukuran file 10 MB, AES-256 menghasilkan waktu eksekusi yang sebanding dengan AES-128. Kondisi ini menunjukkan bahwa dukungan teknologi AES-NI pada prosesor mampu mengurangi dampak penambahan jumlah putaran terhadap performa enkripsi. Dari sisi throughput, AES-128 menunjukkan performa lebih baik pada ukuran file kecil. Namun, pada ukuran file 1 MB hingga 10 MB, nilai throughput kedua algoritma hampir sama, bahkan AES-256 sedikit lebih tinggi. Hasil ini menunjukkan bahwa implementasi AES pada Node.js tetap efisien meskipun menggunakan panjang kunci yang lebih besar.

Nilai standar deviasi yang sangat rendah mengindikasikan bahwa proses benchmarking menghasilkan pengukuran yang konsisten. Penerapan warm-up iteration berhasil meminimalkan pengaruh Just-In-Time Compilation (JIT) pada mesin V8 sehingga meningkatkan keandalan hasil pengujian.

Temuan penelitian ini sejalan dengan penelitian Elminaam dkk., Chen dkk. (2023), serta Pereira dan Alves (2020), yang menyatakan bahwa peningkatan panjang kunci menambah kompleksitas komputasi, namun dukungan akselerasi perangkat keras mampu mempertahankan performa AES pada sistem modern.

Berdasarkan hasil tersebut, AES-128 lebih sesuai untuk aplikasi yang mengutamakan latensi rendah, sedangkan AES-256 direkomendasikan untuk aplikasi yang membutuhkan tingkat keamanan tinggi karena mampu memberikan perlindungan lebih baik tanpa penurunan performa yang signifikan pada perangkat keras modern.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Penelitian ini menganalisis performa algoritma AES-128, AES-192, dan AES-256 pada lingkungan Node.js menggunakan metode benchmarking. Hasil penelitian menunjukkan bahwa AES-256 memiliki waktu eksekusi sedikit lebih tinggi dibandingkan AES-128 pada ukuran file kecil karena jumlah putaran enkripsi yang lebih banyak. Namun, perbedaan performa semakin kecil pada ukuran file yang lebih besar, bahkan AES-256

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

menghasilkan *throughput* yang sebanding dengan AES-128. Hal ini menunjukkan bahwa dukungan teknologi AES-NI mampu mengurangi *overhead* komputasi sehingga implementasi AES pada Node.js tetap memberikan performa yang tinggi dan stabil. Dengan demikian, pemilihan varian AES sebaiknya disesuaikan dengan kebutuhan aplikasi, yaitu AES-128 untuk kebutuhan latensi rendah dan AES-256 untuk kebutuhan keamanan yang lebih tinggi.

5.2 Saran

Penelitian selanjutnya disarankan melakukan pengujian pada berbagai arsitektur prosesor dan mode operasi AES, seperti AES-GCM dan AES-CBC. Selain itu, evaluasi dapat dikembangkan dengan menambahkan parameter penggunaan CPU, memori, dan konsumsi energi, serta membandingkan AES dengan algoritma kriptografi lain agar diperoleh analisis performa yang lebih komprehensif.

Daftar Pustaka

Chen, W., Liu, X., & Zhao, J. (2023). Hardware-Accelerated AES Performance on Modern x86 Processors with AES-NI. *Microprocessors and Microsystems*, *98*, 104–118. https://doi.org/10.1016/j.micpro.2023.104118

Garcia, M., & Lopez, E. (2022). Statistical Methods for Reproducible Cryptographic Benchmarking. *Concurrency and Computation: Practice and Experience*, *34*(18), e7012. https://doi.org/10.1002/cpe.7012

Hicks, M. (2019). Overcoming the “Benchmarking Crisis” in JavaScript. *Proceedings of the ACM on Programming Languages*, *3*(OOPSLA), 1–25. https://doi.org/10.1145/3360555

Kumar, S., & Singh, R. (2021). Comparative Analysis of Symmetric Encryption Algorithms for Web Applications. *International Journal of Information Security*, *20*(4), 567–582. https://doi.org/10.1007/s10207-021-00543-x

Lee, M.-J., & Kim, S.-H. (2024). Adaptive Key Size Selection for AES Based on Data Sensitivity and Hardware Capabilities. *Information Sciences*, *650*, 119–135. https://doi.org/10.1016/j.ins.2023.119456

Munir, R. (2021). *Kriptografi* (2nd ed.). Informatika Bandung.

National Institute of Standards and Technology. (2001a). *FIPS PUB 197: Advanced Encryption Standard (AES)*. https://doi.org/10.6028/NIST.FIPS.197

National Institute of Standards and Technology. (2001b). *SP 800-38A: Recommendation for Block Cipher Modes of Operation*. https://csrc.nist.gov/publications/detail/sp/800-38a/final

Nguyen, T., & Tran, H. (2021). Energy-Efficient Cryptographic Algorithm Selection for Edge Computing. *IEEE Internet of Things Journal*, *8*(12), 9876–9889. https://doi.org/10.1109/JIOT.2021.3065432 \

Volume 8, Nomor 2, Edisi Agustus 2026,
Halaman 15-27 P-ISSN 2656-5404; (Print)
E-ISSN 3064-089X (Online)

Paar, C., & Pelzl, J. (2010). *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer. https://doi.org/10.1007/978-3-642-04101-3

Patel, P., & Shah, A. (2023). Performance Optimization of AES Encryption in Node.js Backend Services. *Journal of Network and Computer Applications*, *210*, 103–117. https://doi.org/10.1016/j.jnca.2023.103567

Pereira, J., & Alves, M. (2020). Performance Evaluation of Cryptographic Algorithms Over IoT Devices. *IEEE Access*, *8*, 112345–112358. https://doi.org/10.1109/ACCESS.2020.3001234

Rodriguez, C., & Martinez, A. (2022). Impact of V8 JIT Compilation on Cryptographic Library Performance in Node.js. *Software: Practice and Experience*, *52*(6), 1345–1362. https://doi.org/10.1002/spe.3089

Sato, K., & Yamamoto, H. (2020). Benchmarking Symmetric Ciphers in High-Throughput Network Applications. *Computer Networks*, *178*, 107–121. https://doi.org/10.1016/j.comnet.2020.107345

Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.). Wiley.

Wang, F., Li, Q., & Zhang, H. (2023). Cache-Aware Optimization of Block Cipher Implementations on Modern CPUs. *IEEE Transactions on Computers*, *72*(3), 789–803. https://doi.org/10.1109/TC.2022.3215678

Zhang, L., & Wang, Y. (2022). Performance Analysis of AES Implementations in Modern JavaScript Runtimes. *Journal of Systems Architecture*, *125*, 102–115. https://doi.org/10.1016/j.sysarc.2022.102115