BINER
Jurnal Ilmiah Informatika dan Komputer
Vol. 5 | No. 2 | Juli 2026

p-ISSN : 2828-0210 | e-ISSN : 2828-0229
https://ojs.unsiq.ac.id/index.php/biner

ANALISIS KOMPARASI KINERJA ALGORITMA MERGESORT DAN QUICKSORT PADA PEMBANGKITAN KUNCI KRIPTOGRAFI ELGAMAL

Muh. Farizzi [1], Moh. Alhaji Putra Lede [2]
[1] Teknik Informatika, Fakultas Teknik, Universitas Tadulako
[2] Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin
Email: ezhifarizzi@gmail.com [1], moh.alhaji74@gmail.com [2]

Dikirim: 18 April 2026; Disetujui: 29 Mei 2026; Dipublikasikan: 31 Juli 2026

ABSTRAK

Fase pembangkitan kunci pada algoritma kriptografi asimetris Elgamal merupakan tahapan paling krusial yang bergantung pada efisiensi proses pengurutan bilangan acak untuk menyeleksi bilangan prima. Penelitian sebelumnya mengimplementasikan Quicksort pada fase ini, namun algoritma tersebut memiliki kerentanan penurunan performa drastis menjadi $O(n^2)$ pada skenario terburuk. Penelitian ini bertujuan mengusulkan Mergesort sebagai substitusi dan melakukan analisis komparatif waktu eksekusi antara kedua algoritma. Pengujian disimulasikan menggunakan C++ pada lingkungan komputasi terisolasi dengan variasi populasi data (n) dari 1.000 hingga 100.000 elemen pada kondisi data acak dan terurut terbalik. Hasil pengujian menunjukkan bahwa meskipun Quicksort lebih cepat pada kondisi rata-rata, algoritma tersebut mengalami kelumpuhan rekursif (*Stack Overflow*) pada kondisi terurut terbalik. Sebaliknya, Mergesort terbukti konsisten mempertahankan kompleksitas $O(n \log n)$ di seluruh skenario; pada $n = 10.000$ *worst case*, Mergesort 65 kali lebih cepat dari Quicksort. Validasi *end-to-end* membuktikan kunci Elgamal yang dibangkitkan sepenuhnya valid secara kriptografis.

Kata Kunci : Kriptografi, Elgamal, Pembangkitan Kunci, Mergesort, Quicksort.

ABSTRACT

The key generation phase in the Elgamal asymmetric cryptographic algorithm is the most critical stage, relying heavily on the efficiency of sorting random numbers to select prime numbers. Previous research implemented Quicksort for this phase; however, Quicksort suffers a drastic performance degradation to $O(n^2)$ in the worst-case scenario. This study proposes Mergesort as a substitution and conducts a comparative analysis of execution time between the two algorithms. Testing was simulated using C++ in an isolated computing environment, varying the data population (n) from 1,000 to 100,000 elements under random and reverse-sorted data conditions. Results show that while Quicksort is faster in average-case conditions, it experiences a recursive breakdown (Stack Overflow) under reverse-sorted data. In contrast, Mergesort consistently maintains $O(n \log n)$ complexity across all scenarios; at $n = 10{,}000$ in the worst case, Mergesort is 65 times faster than Quicksort. End-to-end validation confirms that the generated Elgamal keys are fully valid cryptographically.

Keywords *: Cryptography, Elgamal, Key Generation, Mergesort, Quicksort.*

1. PENDAHULUAN

Tingginya mobilitas pertukaran informasi digital menuntut adanya mekanisme pengamanan data yang kuat terhadap berbagai dokumen penting, sehingga kerahasiaan, integritas, serta keasliannya tetap terjaga utuh dan terhindar dari modifikasi pihak yang tidak bertanggung jawab [1]. Perlindungan data atau dokumen menuntut adanya sebuah alat bantu khusus yang mampu menjamin kerahasiaan, integritas, pengenalan identitas pengirim, sekaligus mencegah penyangkalan pengiriman [2]. Salah satu metode perlindungan data yang terbukti kuat adalah kriptografi modern. Dalam perkembangannya, sistem kriptografi modern terbagi menjadi dua kategori utama berdasarkan penggunaan kuncinya: kriptografi simetris yang menggunakan satu kunci identik untuk enkripsi dan dekripsi, serta kriptografi asimetris yang menggunakan sepasang kunci berbeda (kunci publik dan kunci privat) [3]. Analisis komparatif modern membuktikan bahwa meskipun kriptografi simetris lebih cepat, kriptografi asimetris lebih unggul dalam menjamin keamanan sistem informasi saat proses distribusi kunci karena ketiadaan kanal komunikasi kunci rahasia yang rentan disadap [4]. Oleh karena itu, perlindungan infrastruktur berskala besar menuntut penerapan metode asimetris dengan jaminan stabilitas algoritma yang tinggi [5]. Salah satu standar algoritma yang diimplementasikan adalah algoritma Elgamal.

Algoritma Elgamal merupakan algoritma kriptografi asimetris yang tingkat keamanannya didasarkan pada kerumitan penyelesaian masalah logaritma diskrit pada grup perkalian bilangan bulat modulo [6]. Penggunaan metode kriptografi seperti algoritma ElGamal terbukti mampu menjamin keaslian sebuah tanda tangan atau dokumen digital. Dengan memanfaatkan kompleksitas logaritma diskrit, algoritma ini menawarkan perlindungan kerahasiaan yang ketat, di mana data yang diamankan terhindar dari potensi manipulasi atau modifikasi oleh pihak eksternal [7]. Sistem ini beroperasi melalui tiga tahapan komputasi utama: pembangkitan kunci, enkripsi, dan dekripsi. Keandalan algoritma ini sangat bergantung pada fase pembangkitan kunci, di mana sistem harus memilih bilangan prima bernilai besar (p) dan akar primitif (g) secara acak dari sebuah populasi data [8]. Sehingga, kekuatan utama dari algoritma Elgamal terletak pada kerumitan perhitungan logaritma diskrit dan sangat bergantung pada kualitas bilangan prima acak yang dibangkitkan untuk membentuk kunci publik dan kunci privat. Oleh karena itu, fase pembangkitan kunci (*key generation*) menjadi tahapan yang sangat krusial sebelum proses enkripsi dan dekripsi dapat dilakukan.

Dalam proses pencarian dan penyeleksian bilangan prima untuk kunci Elgamal, diperlukan sebuah metode pengurutan (*sorting*) kumpulan bilangan acak agar sistem dapat memilih nilai batas atas dan batas bawah secara presisi. Penelitian sebelumnya telah mengimplementasikan algoritma Quicksort untuk mengurutkan nilai-nilai acak tersebut dalam proses pembangkitan kunci dokumen (merujuk pada penelitian Al-Amansyah, 2022). Berdasarkan analisis kinerja komparatif struktur data, Quicksort secara empiris menunjukkan waktu eksekusi tercepat pada ukuran data kecil dan kondisi rata-rata dengan kompleksitas $O(n\,log\,n)$ [9], [10]. Sehingga, algoritma Quicksort dipilih karena secara umum memiliki performa yang baik dengan pendekatan *divide-and-conquer*.

Meskipun Quicksort efektif pada kondisi rata-rata (average case), algoritma ini memiliki kerentanan arsitektural pada kondisi anomali data (misalnya himpunan terurut terbalik) yang memicu partisi array asimetris, sehingga kompleksitas waktu komputasinya memburuk secara signifikan hingga menyentuh batas atas $O(n^2)$ [11]. Pada kondisi ini, pemilihan pivot menjadi tidak optimal sehingga kompleksitas waktu eksekusi Quicksort anjlok drastis dari $O(n \log n)$ menjadi $O(n^2)$. Dalam konteks keamanan data berskala besar yang membutuhkan pembangkitan kunci secara cepat dan *real-time*, ketidakstabilan waktu eksekusi ini dapat menjadi celah penurunan performa (*bottleneck*) pada sistem.

Modifikasi dan pengembangan arsitektur kriptografi Elgamal merupakan salah satu area penelitian yang aktif dikaji dalam komunitas keamanan siber global. Berbagai pendekatan telah diusulkan, mulai dari penggunaan konsep matriks untuk meningkatkan kompleksitas logaritma diskrit [12] hingga generalisasi sistem kunci publik secara matematis [13]. Meskipun demikian, mayoritas penelitian terdahulu lebih

berfokus pada optimasi operasi matematis (fase enkripsi), namun masih mengesampingkan optimasi pada ranah struktural sistem (fase pembangkitan kunci awal). Oleh karena itu, celah inilah yang menjadi fokus utama dalam penelitian ini.

Berdasarkan identifikasi celah tersebut, terdapat 3 unsur kebaruan yang belum pernah dikaji secara bersamaan oleh penelitian sebelumnya. Pertama, penelitian ini merupakan kajian pertama yang secara eksplisit mengidentifikasi kerentanan arsitektural Quicksort pada fase pembangkitan kunci Elgamal sebagai ancaman keamanan nyata. Kedua, penelitian ini mengusulkan dan memvalidasi secara empiris algoritma Mergesort sebagai substitusi yang menjamin kompleksitas $O(n \log n)$ di seluruh kondisi data dalam konteks kriptografi Elgamal yang belum pernah ditawarkan oleh penelitian-penelitian modifikasi Elgamal sebelumnya yang berfokus pada fase enkripsi [12][13]. Ketiga, penelitian ini pertama kali menghubungkan ketidakstabilan algoritma pengurutan pada fase pembangkitan kunci dengan resiko *Algoritmic Complexity Attack* terhadap infrastruktur kriptografi.

Untuk mengatasi kelemahan tersebut, penelitian ini mengusulkan algoritma Mergesort sebagai substitusi pengurutan dalam pembangkitan kunci Elgamal. Berbeda dengan Quicksort, Mergesort secara konsisten menjamin kompleksitas waktu $O(n \log n)$ di segala kondisi data. Sebagai alternatif untuk mengatasi ketidakstabilan performa pada skenario *worst-case*, Mergesort hadir sebagai algoritma pengurutan yang juga mengimplementasikan teknik *divide-and-conquer* [11]. Berbagai studi komparatif berskala internasional terhadap paradigma *divide-and-conquer* menyimpulkan bahwa algoritma Mergesort merupakan kandidat terbaik untuk menjamin stabilitas komputasi [14]. Penelitian ini tidak hanya mengimplementasikan penggabungan Mergesort dan Elgamal, tetapi juga melakukan analisis komparasi metrik secara empiris antara Mergesort dan Quicksort. Pengujian waktu eksekusi (*execution time*) dirancang pada lingkungan tingkat rendah untuk mendapatkan presisi tinggi, guna membuktikan bahwa algoritma Mergesort menawarkan stabilitas komputasi yang lebih superior dan andal dalam mendukung infrastruktur kriptografi Elgamal.

2. METODE

Penelitian ini menggunakan pendekatan kuantitatif eksperimental dengan metode analisis komparatif. Fokus utama eksperimen ini adalah membandingkan secara langsung performa algoritma Mergesort dan Quicksort ketika diimplementasikan sebagai metode pengurutan (*sorting*) dalam fase pembangkitan kunci kriptografi Elgamal. Eksperimen dirancang untuk menguji kedua algoritma pada lingkungan isolasi komputasi yang sama guna menghindari bias interferensi sistem.

Untuk mendapatkan tingkat presisi metrik yang tinggi, simulasi pengurutan dan pencatatan waktu eksekusi (*execution time*) diprogram menggunakan bahasa C++ dengan memanfaatkan pustaka standar <chrono> dalam satuan ukur mikrodetik (*microseconds*). Pengujian dilakukan secara berulang dengan tiga variasi ukuran dataset, yaitu $n = 1.000, 10.000,$ dan 100.000 elemen bilangan bulat acak. Setiap ukuran data diuji melalui dua kondisi:

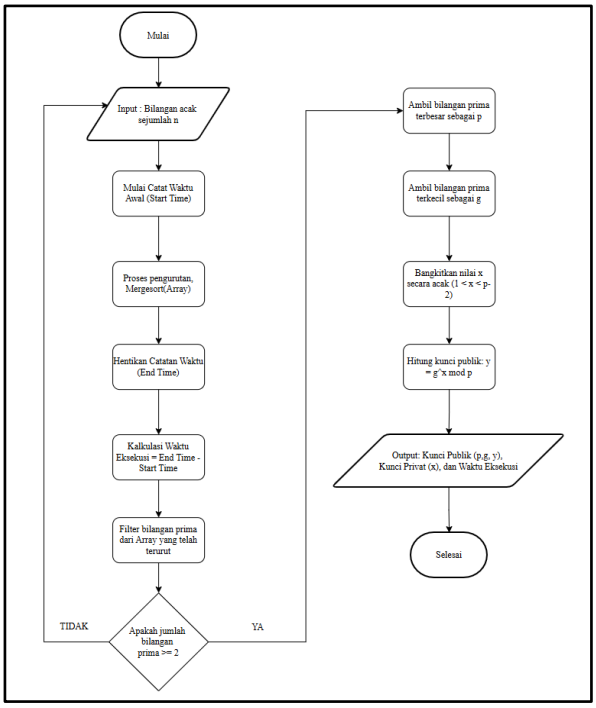
1. Kondisi Data Acak (*Average Case*): Himpunan data dibangkitkan secara acak murni tanpa pola tertentu untuk melihat performa harian dari kedua algoritma.
2. Kondisi Data Terurut Terbalik (*Worst Case*): Himpunan data disiapkan dalam urutan mengecil (dari nilai terbesar ke terkecil) untuk memicu kondisi terburuk asimetris pada Quicksort dan membuktikan stabilitas Mergesort.

Setiap skenario dieksekusi sebanyak 5 kali percobaan (*trials*) dan nilai rata-rata (*mean*) dari kelima percobaan tersebut digunakan sebagai data akhir untuk meminimalisasi bias interferensi sistem operasi.

Proses modifikasi pembangkitan kunci Elgamal menggunakan Mergesort dirancang mengikuti lima tahapan komputasi yang diilustrasikan pada Gambar 1.

1. Pembangkitan Kumpulan Bilangan (*Data Generation*): Sistem membangkitkan n buah bilangan bulat secara acak.
2. Pengurutan (*Sorting*): Algoritma Mergesort memproses himpunan bilangan acak tersebut sehingga tersusun secara *ascending* (dari kecil ke besar).

3. Penyaringan Keprimaan (*Primality Filtering*): Himpunan data yang telah terurut akan dievaluasi untuk mengekstrak bilangan prima yang memenuhi syarat parameter Elgamal. Proses penyaringan ini diimplementasikan menggunakan metode *Optimized Trial Division*. Pada metode ini, setiap elemen dalam himpunan akan diuji keterbagiannya secara inkremental, dimulai dari bilangan 2 hingga batas akar kuadrat dari bilangan tersebut ($\sqrt{n}$). Pendekatan matematis ini dipilih karena memiliki kompleksitas ruang yang minimal namun terbukti sangat efisien untuk memvalidasi kandidat kunci publik (p, g) dan kunci privat (x) pada sistem dengan kapabilitas memori terbatas.
4. Seleksi Parameter Kunci: Bilangan prima terbesar dari himpunan terurut digunakan sebagai modulus p. Untuk menentukan generator g, sistem menjalankan validasi *primitive root* berdasarkan teorema grup siklik. Grup perkalian modulo p yang dibentuk oleh bilangan prima p bersifat siklik, artinya terdapat setidaknya satu bilangan g yang mampu membangkitkan seluruh elemen $\{1, 2, \ldots, p-1\}$ melalui perpangkatan berulang modulo p, bilangan inilah yang disebut *primitive root*. Secara formal, g adalah *primitive root* dari p jika orde g dalam grup tersebut sama dengan $\varphi(p) = p - 1$. Secara komputasional, hal ini setara dengan syarat bahwa untuk setiap faktor prima q dari $(p-1)$, berlaku $g^{((p-1)/q)} \bmod p \neq 1$. Proses ini bekerja dalam tiga langkah: (1) faktorisasi $(p-1)$ untuk memperoleh seluruh faktor prima uniknya, (2) iterasi kandidat g dari himpunan bilangan prima yang tersedia mulai dari nilai terkecil, dan (3) pengujian tiap kandidat terhadap seluruh faktor prima tersebut, kandidat pertama yang lolos seluruh pengujian ditetapkan sebagai g yang valid.
5. Kalkulasi Kunci Publik: Sistem menghitung parameter kunci publik y menggunakan fungsi eksponensial modular dengan persamaan $y = g^x \bmod p$. Kunci publik akhirnya terdiri dari triplet (p, g, y), sedangkan kunci privat adalah x.



Gambar 1. Flowchart Pembangkitan Kunci Elgamal dengan Mergesort

Untuk memastikan penelitian ini dapat direplikasi, logika utama kedua algoritma direpresentasikan dalam bentuk pseudocode berikut. Mergesort diimplementasikan melalui dua fungsi utama:

```
Fungsi MERGESORT(arr, left, right):
  Jika left < right:
    mid = (left + right) / 2
    MERGESORT(arr, left, mid)
    MERGESORT(arr, mid + 1, right)
    MERGE(arr, left, mid, right)

Fungsi MERGE(arr, left, mid,
right):
  Hitung ukuran sub-array kiri (n1)
dan kanan (n2)
  Salin elemen ke sub-array
sementara L[n1] dan R[n2]
  Selama elemen di L dan R masih
ada:
    Bandingkan elemen L dan R

   Masukkan elemen terkecil kembali
ke dalam arr
```

Sedangkan Quicksort diimplementasikan dengan menentukan elemen paling akhir (*right-most*) sebagai poros (*pivot*), lalu mempartisi elemen lain ke sisi kiri (lebih kecil) atau kanan (lebih besar):

```
Fungsi QUICKSORT(arr, low, high):
  Jika low < high:
    pi = PARTITION(arr, low, high)
    QUICKSORT(arr, low, pi - 1)
    QUICKSORT(arr, pi + 1, high)

Fungsi PARTITION(arr, low, high):
  pivot = arr[high]
  i = low - 1
  Untuk j dari low hingga high - 1:
    Jika arr[j] < pivot:
      i = i + 1
      Tukar elemen arr[i] dengan
arr[j]
  Tukar elemen arr[i + 1] dengan
arr[high] (pivot)

  Kembalikan posisi pivot (i + 1)
```

Penilaian performa antar algoritma didasarkan pada dua parameter analitis. Parameter pertama adalah kompleksitas waktu (*execution time*), yaitu selisih waktu antara dimulainya fungsi pengurutan hingga seluruh elemen array selesai diurutkan. Parameter kedua adalah kompleksitas ruang (*space complexity*), berupa analisis teoritis terhadap alokasi memori tambahan yang dibutuhkan oleh kedua algoritma, di mana Quicksort beroperasi secara in-place $O(log\ n)$ sedangkan Mergesort membutuhkan memori alokasi sementara sebesar $O(n)$. Seluruh pengujian dilakukan pada lingkungan komputasi yang terisolasi dengan spesifikasi perangkat keras yang konstan, yaitu prosesor Intel Core i5-12450HX @ 2.40GHz, memori utama (RAM) 20 GB DDR5, dan media penyimpanan 512 GB NVMe SSD. Adapun lingkungan perangkat lunak yang digunakan adalah sistem operasi Windows 11 64-bit dengan proses kompilasi menggunakan compiler GCC MinGW-w64 (MSYS2) versi 13.2.0 melalui IDE Visual Studio Code.

3. HASIL DAN PEMBAHASAN

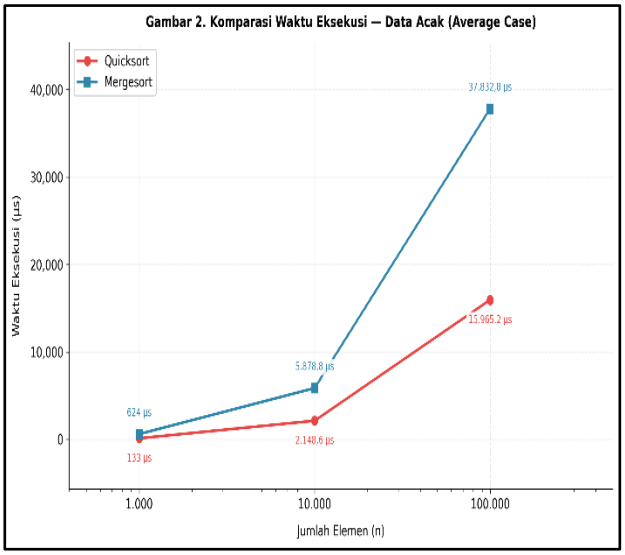
3.1. Hasil Pengujian Metrik Waktu Eksekusi

Pengujian performa algoritma dilakukan pada lingkungan komputasi yang terisolasi dengan menggunakan parameter jumlah himpunan data (n) sebesar 1.000, 10.000, dan 100.000 elemen bilangan bulat acak. Setiap ukuran data diuji melalui dua skenario, yaitu kondisi himpunan data acak (*average case*) dan kondisi himpunan data terurut terbalik (*worst case*). Untuk meminimalisasi bias interferensi dari sistem operasi dan mendapatkan data yang presisi, setiap skenario pengujian dieksekusi sebanyak 5 kali percobaan (*trials*). Nilai waktu eksekusi yang disajikan pada Tabel 1 merupakan nilai rata-rata (*mean*) dari kelima percobaan tersebut.

Tabel 1. Komparasi Waktu Eksekusi (Mikrodetik / µs)

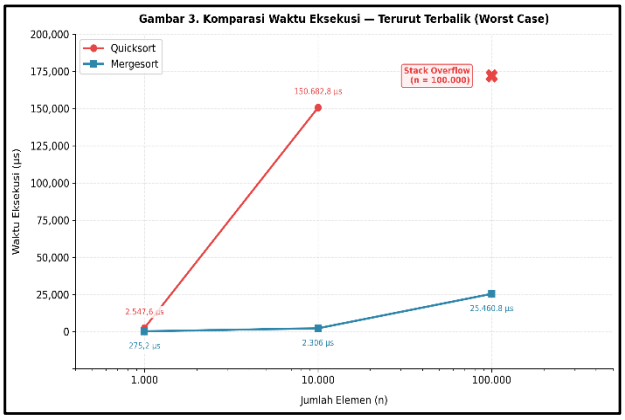
n	QS Acak	MS Acak	QS Worst Case	MS Worst Case
1.000	133	624	2547,6	275,2
10.000	2.148,6	5.878,8	150.682,8	2.306
100.000	15.965,2	37.832,8	> 5.000.000	25.460,8

Untuk memberikan gambaran visual yang lebih komprehensif mengenai tren skalabilitas waktu eksekusi pada kondisi rata-rata, data dari Tabel 1 untuk skenario data acak direpresentasikan dalam bentuk grafik garis pada Gambar 2 berikut.



Gambar 2. Grafik Komparasi Waktu Eksekusi (Data Acak)

Berdasarkan Gambar 2, terlihat tren linier yang stabil untuk kedua algoritma, meskipun Quicksort menunjukkan waktu pemrosesan yang lebih cepat secara konstan. Namun, tren ini berubah secara signifikan ketika data diuji dalam kondisi *worst case*. Gambar 3 mengilustrasikan perbandingan waktu eksekusi antara kedua algoritma pada himpunan data terurut terbalik.



Gambar 3. Grafik Komparasi Waktu Eksekusi (Terurut Terbalik / Worst Case)

Gambar 3 secara jelas menunjukkan kerentanan Quicksort pada skenario *worst case*, ditandai dengan lonjakan tajam waktu eksekusi, sementara garis Mergesort tetap landai. Anomali ini menegaskan urgensi penggunaan algoritma yang stabil seperti Mergesort dalam proses kritis seperti pembangkitan kunci kriptografi.

3.2. Pembahasan Kinerja pada Kondisi Rata-rata (*Average Case*)

Berdasarkan metrik pada pengujian himpunan data acak, Quicksort secara konsisten menunjukkan waktu eksekusi yang lebih superior dibandingkan Mergesort. Fenomena ini sejalan dengan teori analisis algoritma. Meskipun kedua algoritma memiliki kompleksitas waktu teoretis $O(n\,log\,n)$ pada kondisi rata-rata, Quicksort memiliki keunggulan arsitektural berupa operasi yang bersifat *in-place* (tidak membutuhkan array tambahan). Hal ini membuat Quicksort sangat efisien dalam memanfaatkan *cache locality* pada prosesor (CPU), di mana elemen yang ditukar berada pada blok memori yang berdekatan [9], [10]. Sebaliknya, Mergesort mengalami *overhead* (beban waktu tambahan) karena algoritma ini mengonsumsi memori dinamis sebesar $O(n)$ untuk mengalokasikan dan menyalin elemen ke sub-array sementara (L dan R) pada setiap fase penggabungan (*merge*).

3.3. Pembahasan Kinerja pada Kondisi Terburuk (*Worst Case*)

Pada kondisi terburuk, terdapat kemungkinan performa modifikasi ElGamal tidak selalu unggul atas algoritma standar untuk semua jenis data masukan [15]. Kelemahan arsitektural Quicksort terekspos secara empiris ketika dihadapkan pada skenario data terurut terbalik (*worst case*). Pada populasi data $n = 10.000$, waktu eksekusi Quicksort melonjak secara eksponensial. Hal ini terjadi karena pemilihan poros (*pivot*) yang selalu jatuh pada nilai ekstrem (terbesar atau terkecil) menyebabkan partisi array menjadi sangat tidak seimbang (satu bagian berisi $n-1$ elemen, bagian lain kosong). Ketidakseimbangan ini memaksa kompleksitas waktu memburuk menjadi $O(n^2)$ [11]. Bahkan, pada pengujian dengan $n = 100.000$, Quicksort mengalami kelumpuhan sistem berupa *Stack Overflow* karena kedalaman rekursi yang melebihi batas memori tumpukan (*stack memory*) sistem operasi. Di sisi lain, Mergesort terbukti memiliki stabilitas komputasi yang sangat konsisten [14], [16]. Karena proses pembelahan array pada Mergesort selalu membagi tepat di tengah tanpa bergantung pada nilai elemen, waktu eksekusinya tetap konstan di kisaran $O(n\,log\,n)$ dan berhasil menyelesaikan pengurutan 100.000 data tanpa kendala memori.

3.4. Hasil Pembangkitan Parameter Kunci Elgamal

Tujuan akhir dari pengurutan himpunan bilangan dalam penelitian ini adalah mengekstraksi parameter kunci kriptografi. Setelah Mergesort mengurutkan himpunan bilangan acak, sistem menyaring bilangan komposit sehingga tersisa himpunan bilangan prima terurut. Bilangan prima terbesar dari himpunan ini diambil sebagai modulus p. Generator g ditentukan melalui validasi *primitive root*: kandidat dari himpunan prima diuji satu per satu menggunakan kriteria $g^{((p-1)/q)}\ mod\ p \neq 1$ untuk setiap faktor prima q dari $(p-1)$, dan kandidat pertama yang memenuhi semua syarat ditetapkan sebagai g. Berbeda dengan pendekatan pada penelitian sebelumnya yang menetapkan parameter p dan akar primitif α secara langsung tanpa mekanisme seleksi algoritmik [17], penelitian ini memperoleh kedua parameter tersebut dari himpunan acak yang telah diurutkan dan diverifikasi keprimaannya secara sistematis melalui Mergesort. Kunci privat x kemudian dibangkitkan secara acak dengan batasan $1 < x < p-2$ dan kunci publik y dihitung menggunakan eksponensiasi modular $y = g^x\ mod\ p$. Hasilnya menunjukkan bahwa Mergesort mempertahankan integritas data sehingga validasi *primitive root* dan

pembangkitan kunci Elgamal dapat diselesaikan dengan benar.

3.5. Validasi *End-to-End* Kriptografi

Tujuan esensial dari pengurutan kumpulan bilangan dalam penelitian ini adalah untuk memastikan bahwa sistem dapat mengekstraksi parameter kunci yang valid secara matematis untuk digunakan pada proses kriptografi. Untuk membuktikan hal tersebut, dilakukan pengujian *end-to-end* terhadap kunci yang diekstrak dari himpunan data yang telah diurutkan secara stabil oleh algoritma Mergesort.

Setelah proses penyaringan keprimaan dilakukan, sistem menetapkan bilangan prima terbesar sebagai nilai modulus p. Nilai generator g kemudian ditentukan melalui prosedur validasi *primitive root*: dari hasil pengujian pada n = 10.000 elemen, sistem memperoleh p = 99.829 dengan nilai $p-1$ = 99.828 yang memiliki faktor prima $\{2, 3, 47, 59\}$. Iterasi kandidat menghasilkan g = 859 sebagai primitive root yang valid, ditunjukkan oleh: $g^{((p-1)/2)}\ mod\ p$ = 99.828, $g^{((p-1)/3)}\ mod\ p$ = 5.713, $g^{((p-1)/47)}\ mod\ p$ = 16.717, dan $g^{((p-1)/59)}\ mod\ p$ = 30.724, keempatnya bukan 1, sehingga syarat terpenuhi. Dengan parameter tersebut, kunci publik yang dihasilkan adalah (p, g, y) = (99.829, 859, 70.503) dan kunci privat x = 13.466. Dengan membangkitkan nilai acak spesifik sebagai kunci privat (x), sistem memproses fungsi eksponensiasi modular untuk menghasilkan kunci publik (y).

Pengujian fungsionalitas kunci diimplementasikan dengan menyimulasikan proses enkripsi pada sebuah pesan uji (*plaintext* = 65). Menggunakan kunci publik yang telah dibangkitkan, pesan tersebut diubah menjadi *ciphertext* yang terdiri dari sepasang nilai (c_1, c_2). Selanjutnya, proses dekripsi dilakukan menggunakan kunci privat untuk mengembalikan pesan ke bentuk aslinya. Hasil eksekusi komputasi menunjukkan bahwa nilai dekripsi sama persis dengan nilai *plaintext* awal. Keberhasilan pengujian fungsionalitas ini secara empiris membuktikan bahwa implementasi algoritma Mergesort pada fase pembangkitan kunci menjamin integritas data dengan tingkat keandalan tinggi, sehingga parameter kunci yang dibangkitkan sepenuhnya valid dan siap diimplementasikan pada infrastruktur pengamanan dokumen berskala besar.

3.6. Implikasi Keamanan Terhadap Sistem Kriptografi Dokumen

Dalam implementasi pengamanan dokumen berskala besar, parameter stabilitas komputasi jauh lebih berharga dibandingkan sekadar kecepatan. Upaya peningkatan fleksibilitas dan keamanan algoritma ElGamal juga dibuktikan secara matematis oleh Zega dan Yulianto [18] melalui generalisasi struktur kunci publik ElGamal yang terbukti efektif mencegah pembengkakan ukuran *ciphertext*. Sejalan dengan hal tersebut, fakta empiris bahwa algoritma Quicksort dapat memicu *Stack Overflow* pada data asimetris menjadikannya sangat rentan terhadap jenis serangan *Algorithmic Complexity Attack* (kerentanan di mana peretas sengaja menyuntikkan data *worst-case* untuk melumpuhkan server). Oleh karena itu, substitusi menggunakan Mergesort memberikan perlindungan keamanan berlapis: tidak hanya dokumen aman karena enkripsi Elgamal, tetapi infrastruktur sistem (*backend*) juga terlindungi dari risiko *Denial of Service* (DoS) karena garansi waktu eksekusi Mergesort yang stabil. Perbedaan mendasar antara kajian terdahulu yang memodifikasi Elgamal pada ranah matematis [12][13] terletak pada ancaman yang diatasi. Penelitian-penelitian tersebut memperkuat kompleksitas algoritma diskrit pada fase enkripsi, namun tidak mempertimbangkan stabilitas komputasi pada fase pembangkitan kunci. Sehingga, penelitian ini membuktikan bahwa ancaman terhadap sistem kriptografi tidak hanya datang dari sisi matematis, tetapi juga dari sisi algoritmik pada infrastruktur pendukungnya. Meskipun demikian, penelitian ini memiliki keterbatasan, yaitu pengujian performa algoritma baru dieksekusi pada ruang lingkup (n) maksimal 100.000 elemen dengan parameter perangkat keras tertentu. Kinerja pada lingkungan sistem terdistribusi nyata (*real-world server*) atau dengan ukuran data yang jauh lebih masif masih memerlukan investigasi lebih lanjut.

4. PENUTUP

4.1. Kesimpulan

Berdasarkan hasil pengujian dan analisis komparasi komputasi yang telah dilakukan, dapat disimpulkan bahwa algoritma Mergesort merupakan alternatif yang lebih andal dibandingkan Quicksort untuk

diimplementasikan pada fase pengurutan dalam pembangkitan kunci kriptografi Elgamal. Meskipun Quicksort menunjukkan waktu eksekusi yang lebih cepat pada kondisi himpunan data acak (*average case*), algoritma tersebut terbukti memiliki kerentanan arsitektural yang fatal berupa lonjakan waktu eksekusi hingga potensi kelumpuhan memori (*stack overflow*) pada kondisi data terurut terbalik (*worst case*). Sebaliknya, Mergesort berhasil memberikan garansi stabilitas komputasi yang optimal dengan mempertahankan kompleksitas waktu komputasi konstan pada rentang $O(n \log n)$ untuk seluruh skenario pengujian. Penggunaan memori ekstra pada Mergesort merupakan pertukaran (*trade-off*) komputasi yang sangat sepadan demi memastikan sistem keamanan kriptografi tetap stabil dan terhindar dari bottleneck saat menghadapi anomali penyebaran data ekstrem. Sebagai rekomendasi untuk penelitian selanjutnya, implementasi algoritma pengurutan *hybrid* atau *Randomized* Quicksort dapat dikaji untuk mengoptimalkan efisiensi waktu pada kondisi rata-rata tanpa harus mengorbankan stabilitas keamanan pada skenario terburuk.

4.2. Saran

Meskipun tujuan dari penelitian ini telah tercapai, masih ada beberapa hal yang bisa dikembangkan lagi untuk penelitian kedepannya. Oleh karena itu, penulis telah merumuskan saran yang dapat dilakukan untuk penelitian selanjutnya:

1. Penelitian selanjutnya disarankan untuk mengkaji implementasi algoritma pengurutan *hybrid* pada fase pembangkitan kunci kriptografi Elgamal.
2. Pengembangan lain yang sangat potensial untuk dieksplorasi adalah penggunaan varian *Randomized Quicksort* sebagai alternatif pendamping.
3. Langkah pengujian terhadap metode alternatif tersebut sebaiknya difokuskan untuk mengoptimalkan efisiensi waktu komputasi secara maksimal pada kondisi data rata-rata (*average case*).
4. Optimalisasi kecepatan tersebut harus dipastikan dapat berjalan tanpa mengorbankan jaminan stabilitas keamanan sistem ketika dihadapkan pada skenario anomali atau terburuk (*worst case*).
5. Melalui eksplorasi lanjutan ini, infrastruktur keamanan data di masa depan diharapkan mampu mencapai keseimbangan performa yang ideal antara kecepatan eksekusi komputasi dan efisiensi penggunaan memori.

5. DAFTAR PUSTAKA

[1] V. H. Zulian and P. Purwanto, "Implementasi Tanda Tangan Digital (Digital Signature) Menggunakan Algoritme Elgamal pada Dokumen di Balai Pendidikan dan Pelatihan Penerbangan (BP3) Curug Berbasis Web," in *Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, Jakarta, Indonesia, 2022, pp. 386–393. [Online]. Available: https://senafti.budiluhur.ac.id/index.php/senafti/index

[2] F. Husaini, A. M. H. Pardede, and I. Gultom, "Penerapan Enkripsi Menggunakan Metode Elgamal guna Meningkatkan Keamanan Data Text dan Gambar," *JUKI: Jurnal Komputer dan Informatika*, vol. 4, no. 1, pp. 67–73, 2022.

[3] R. Sadikin, *Kriptografi untuk Keamanan Jaringan*. Yogyakarta: C.V Andi Offset, 2012.

[4] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson, 2017.

[5] Z. Arif and A. Nurokhman, "Analisis Perbandingan Algoritma Kriptografi Simetris Dan Asimetris Dalam Meningkatkan Keamanan Sistem Informasi," *Jurnal Teknologi Sistem Informasi*, vol. 4, no. 2, pp. 394–405, 2023, doi: 10.35957/jtsi.v4i2.6077.

[6] S. Sabitha and B. V Nair, "Survey on Asymmetric Key Cryptographic Algorithms," *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 7, no. 2, pp. 404–408, 2020, doi: 10.32628/ijsrset207292.

[7] R. K. Lubis, A. M. H. Pardede, and H. Khair, "Digital Signature Security

Analysis By Applying The Emal Algorithm And The Idea Method," *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 3, no. 1, pp. 373–382, 2023, [Online]. Available: https://ioinformatic.org/

[8] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–481, 1985, doi: 10.1109/TIT.1985.1057074.

[9] R. Sedgewick, "Implementing quicksort programs," *Commun. ACM*, vol. 21, no. 10, pp. 847–856, 1978, doi: 10.1145/359619.359631.

[10] I. Ali, S. Mulla, and A. Datar, "Performance Comparison between Merge and Quick Sort Algorithms in Data Structure," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, pp. 192–197, 2018, doi: 10.14569/IJACSA.2018.091127.

[11] A. Fahreza and M. H. Suhartono, "Quick sort and merge sort performance comparison in the flutter framework," *Journal of Informatics and Science Media*, vol. 1, no. 1, pp. 1–5, 2024.

[12] Maxrizal, S. Irawadi, and S. Sujono, "Discrete Logarithmic Improvement for ElGamal Cryptosystem Using Matrix Concepts," in *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, IEEE, 2020, pp. 1–5. doi: 10.1109/CITSM50537.2020.9268832.

[13] R. Ranasinghe and P. Athukorala, "A generalization of the ElGamal public-key cryptosystem," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 25, pp. 1–9, May 2021, doi: 10.1080/09720529.2020.1857902.

[14] J. Kleinberg and É. Tardos, *Algorithm Design*. Pearson, 2005.

[15] A. Thakkar and R. Gor, "Cryptographic Method to Enhance the Data Security using ElGamal Algorithm and Sumudu Transform," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 11, no. 7, pp. 853–861, 2023, doi: 10.22214/ijraset.2023.54750.

[16] O. E. Taiwo, A. E. Ezugwu, O. N. Oyelade, and M. S. Almutairi, "Comparative Study of Two Divide and Conquer Sorting Algorithms: Quicksort and Mergesort," *Procedia Comput. Sci.*, vol. 171, pp. 2532–2540, 2020, doi: 10.1016/j.procs.2020.04.274.

[17] S. N. Fadlilah, Turmudi, and M. Khudzaifah, "Penggabungan Algoritma Hill Cipher dan ElGamal untuk Mengamankan Pesan teks," Jurnal Riset Mahasiswa Matematika (JRMM), vol. 1, no. 5, pp. 230–235, 2022, doi: 10.18860/jrmm.v1i5.14496.

[18] I. Zega and B. D. Yulianto, "Enhancing the Encryption Capabilities of the Generalization of the ElGamal Algorithm for Document Security," *Journal of Applied Informatics and Computing (JAIC)*, vol. 9, no. 4, pp. 1266–1271, 2025, [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC