Analisis Perbandingan Performa Algoritma Asimetris Elgamal Dan Merkle-Hellman-Knapsack Pada Pesan Teks

Nurika Rahmadani[1], Hamdani Hamdani[2*], Zainal Arifin[3], Refi Riduan Achmad[4]
[1,2,3]Program Studi Informatika, Fakultas Teknik, Universitas Mulawarman, Samarinda
[4]Program Studi Teknik Informatika, Universitas Nahdlatul Ulama Kalimantan Timur, Samarinda
e-mail: [1]nurikarahmadani@gmail.com, [2]hamdani@unmul.ac.id, [3]zainal.arifin@unmul.ac.id,*
[4]refiriduanachmad@unukaltim.ac.id

Abstrak

Perkembangan algoritma kriptografi ditandai dengan beragamnya jenis-jenis algoritma kriptografi yang dapat digunakan dalam enkripsi dan dekripsi, seperti algoritma Elgamal dan Merkle-Hellman-Knapsack. Algoritma Elgamal didasarkan pada logaritma diskrit, sedangkan algoritma Merkle-Hellman-Knapsack didasarkan pada kurva aljabar eliptik.Penelitian ini bertujuan untuk mengetahui perbandingan performa algoritma Elgamal dan Merkle-Hellman-Knapsack dalam melakukan enkripsi dan dekripsi pesan teks. Analisis perbandingan dalam penelitian ini dilakukan dengan mengukur beberapa parameter, yaitu penggunaan memori Resident Set Size (RSS), penggunaan Central Processing Unit (CPU), ukuran ciphertext hasil enkripsi, dan waktu eksekusi. Penelitian dilakukan menggunakan 11 data plaintext dengan ukuran beragam. Hasil pengujian menggunakan ukuran plaintext 10.000 bytes pada metode Elgamal sebesar 51.114 bytes, sedangkan menggunakan metode Merkle Hellman Knapsack hanya sebesar 34.982 bytes. Sementara itu, plaintext 100.000 bytes menggunakan metode Elgamal menghasilkan ciphertext sebesar 511.216 bytes dan Merkle Hellman Knapsack menghasilkan ciphertext sebesar 348.138 bytes. Hasil penelitian menunjukkan bahwa kedua algoritma dapat digunakan untuk mengamankan pesan teks, namun algoritma Elgamal lebih sesuai digunakan untuk sistem dengan keterbatasan memori dan CPU.

Kata kunci*— Kriptografi, Perbandingan, Elgamal, Merkle-Hellman-Knapsack, Pesan Teks*

1. PENDAHULUAN

Keamanan pesan merupakan hal penting dalam komunikasi. Keamanan pesan telah menjadi perhatian sejak lama, bahkan sejak zaman pemerintahan Yunani menggunakan alat penyandian pesan yang disebut Scytale [1]. Berawal dari itu, sistem keamanan pesan semakin berkembang seiring dengan perkembangan zaman. Di zaman modern seperti saat ini banyak tercipta algoritma dan metode matematika yang digunakan untuk mengamankan pesan atau biasa disebut kriptografi. Salah satu metode yang digunakan untuk melindungi data sensitif dari akses yang tidak sah adalah kriptografi. [2]. Kriptografi ini dilakukan agar informasi dalam sebuah pesan tidak dicuri dan disalahgunakan [3].

Melindungi data atau pesan yang dikirim melalui media apapun seperti komputer, jaringan komputer, atau komunikasi elektronik adalah tujuan utama kriptografi [4]. Kriptografi memiliki dua proses yang sangat penting yaitu proses enkripsi dan deskripsi. Enkripsi merupakan kegiatan mengubah data awal atau disebut juga *plaintext* ke dalam kriptografi menjadi sebuah kode-kode acak yang tidak dapat dibaca yang dapat disebut *ciphertext* [5]. Kriptografi memiliki berbagai jenis algoritma dalam melakukan enkripsi dan dekripsi. Dengan keberagaman algoritma kriptografi ini, maka dibandingkan dua metode algoritma asimetris yaitu algoritma Elgamal dan Merkle-Hellman-Knapsack. Algoritma Elgamal adalah bagian dari gagasan kunci publik, juga dikenal sebagai kriptografi kunci publik. Algoritma ini memiliki beberapa proses penting, termasuk pembuatan kunci, enkripsi pesan, dan dekripsi pesan. Salah satu contohnya adalah algoritma cipher blok, yang mengenkripsi blok teks biasa untuk menghasilkan blok ciphertext, yang kemudian di dekripsi. Informasi pesan yang sesuai dengan urutan akan dihasilkan sebagai hasil dari proses dekripsi. [6].

Algoritma Merkle-Hellman-Knapsack merupakan bagian dari kriptografi kunci publik dimana untuk melakukan proses enkripsi dan dekripsi, algoritma ini menggunakan kunci yang berbeda. Algoritma merupakan bagian dari algortima *Nondeterministic Polynomial* (NP-*complete*) yang tidak dapat dipecahkan dalam waktu orde polynomial. Terdapat 3 tahapan yang dilakukan dalam algoritma ini, yaitu proses pembentukan kunci, proses enkripsi, dan proses dekripsi [7].

Penelitian ditujukan untuk mengetahui perbandingan performa antara algoritma Elgamal dan Merkle-Hellman-Knapsack dalam enkripsi dan dekripsi pesan teks. Karena kedua algoritma ini memiliki perbedaan dalam penerapannya. Algoritma Elgamal didasarkan pada logaritma diskrit, sedangkan algoritma Merkle-Hellman-Knapsack didasarkan pada kurva aljabar eliptik [8]. Dari perbedaan tersebut, dicari perbandingan performa antara algoritma Elgamal dan Merkle-Hellman-Knapsack. Performa Elgamal dan Merkle-Hellman-Knapsack dilihat berdasarkan beberapa parameter, yaitu pengukuran memori *Resident Set Size* (RSS), penggunaan *Central Processing Unit* (CPU), ukuran *ciphertext*, dan waktu eksekusi dari masing-masing algoritma dalam enkripsi dan dekripsi pesan teks.

Penelitian tentang perbandingan algoritma kriptografi telah dilakukan dalam berbagai kasus dengan algoritma berbeda. Studi pertama menganalisis kemampuan kriptografi Rivest Shamir Adleman (RSA) dan Elgamal dalam enkripsi dan dekripsi pesan. Tujuan dari penelitian ini adalah untuk melihat bagaimana keduanya bekerja. Penelitian menemukan bahwa algoritma RSA lebih cepat untuk menyelesaikan proses enkripsi dan dekripsi daripada Elgamal. Sementara itu, algoritma Elgamal memiliki hasil kalkulasi peak memory yang lebih baik daripada algoritma RSA[6]. Penelitian kedua adalah implementasi hibrid dari sistem kripto twofish, AES, elgamal dan RSA. Tujuan penelitian ini adalah membandingkan algoritma kriptografi dengan menekankan pada kinerja yang lebih baik, kecepatan maksimum suatu algoritma, pengecekan efektivitas dan perbandingan dengan algoritma lain. Hasilnya algoritma Twofish menjadi algoritma tercepat dibandingkan algoritma lainnya [9].

Penelitian ketiga adalah analisis kecepatan algoritma Merkle-Hellman dan RSA dalam rancang bangun aplikasi penyandian pesan teks. Penelitian ini menemukan bahwa algoritma Merkle-Hellman membutuhkan waktu delapan kali lebih lama untuk memproses teks plain dan cipher. [10]. Penelitian keempat mempunyai tujuan untuk mengukur dan membandingkan kecepatan enkripsi dan dekripsi algoritma Cipher dan ROT13 dalam proses pengamanan data. Pengujian dengan plaintext yang sama menemukan bahwa algoritma Cipher membutuhkan 8 milisecond untuk enkripsi dan 5 milisecond untuk dekripsi, sedangkan algoritma ROT13 membutuhkan 5 milisecond untuk enkripsi dan 5 milisecond untuk dekripsi [11].

2. METODE PENELITIAN

Desain Penelitian

Penelitian ini melakukan perbandingan antara dua algoritma kriptografi, yaitu algoritma Elgamal dan Merkle-Hellman-Knapsack. Penelitian dilakukan secara kuantitatif dengan eksperimen untuk mengukur performa berdasarkan beberapa parameter yang telah ditentukan. Penelitian ini merupakan studi eksperimen komparatif yang membandingkan performa dua algoritma kriptografi asimetris. Pengukuran dilakukan menggunakan aplikasi pengukuran performa enkripsi dan dekripsi yang berjalan di *desktop* yang dirancang khusus untuk proses enkripsi dan dekripsi. Sistem yang digunakan dalam pengukuran ini memiliki spesifikasi prosesor Intel Core i3-7020U (2.30 GHz), RAM 4 GB DDR4, sistem operasi Windows 10.

Parameter pengukuran yang digunakan dalam penelitian ini adalah penggunaan memori *Resident Set Size* (RSS), penggunaan *Central Processing Unit* (CPU), ukuran *ciphertext* hasil enkripsi, dan waktu eksekusi. Data yang digunakan adalah data pesan teks yang berisi karakter *American Standard Code for Information Interchange* (ASCII) *printable characters* dengan kode karakter 32 sampai 127 yang mencakup spasi, angka, huruf besar, huruf kecil, dan simbol-simbol umum.

Prosedur Penelitian

1. Persiapan Data:
   Membuat 11 jenis *plaintext* dengan ukuran bertahap, yaitu *plaintext* dengan ukuran kapasitas 100 *bytes*, 10000 *bytes*, 20000 *bytes*, 40000 *bytes*, 50000 *bytes*, 60000 *bytes*, 70000 *bytes*, 80000 *bytes*, 90000 *bytes*, dan 100000 *bytes*.
2. Proses Enkripsi dan Dekripsi:
   Menggunakan algoritma Elgamal dan Merkle-Hellman-Knapsack ke aplikasi uji secara terpisah untuk setiap *plaintext*. Kemudian mencatat parameter performa selama proses berlangsung.
3. Pengumpulan Data:
   Data yang dihasilkan meliputi penggunaan RSS *Memory*, CPU, waktu eksekusi, dan ukuran *ciphertext*.
4. Analisis Data:
   Membandingkan performa kedua algoritma menggunakan visualisasi grafik dan analisis pola hasil pengukuran.

Proses Enkripsi dan Dekripsi Elgamal

Enkripsi dan dekripsi dimulai dengan proses pembentukan kunci. Pembentukan kunci algoritma Elgamal dilakukan dengan memilih bilangan prima p dan dua buah bilangan acak g dan x, dengan syarat bahwa nilai g dan x lebih kecil dari p yang memenuhi Persamaan 1.

$$Y = g^x \bmod p \quad (1)$$

di mana :

p = bilangan prima dengan syarat $p > 255$
g = bilangan acak dengan syarat $g < p$
x = bilangan acak dengan syarat $x < p$
y = hasil perhitungan dari $y = g^x \bmod p$ yang akan menjadi *Public Key*

Dari persamaan tersebut nilai $y, g,$ dan p merupakan pasangan kunci publik, sedangkan x, p merupakan pasangan *private key* [12].

Proses enkripsi algoritma Elgamal dilakukan menggunakan *public key* (p, g, y). Langkah-langkah dalam melakukan enkripsi pesan adalah sebagai berikut:

1. Potong *plaintext* menjadi blok-blok $m_1, m_2, \ldots$.
2. Ubah nilai blok pesan ke dalam nilai ASCII.
3. Pilih bilangan acak k, dengan syarat $1 \leq k \leq p-2$. Nilai k digunakan untuk menghitung nilai a dan b.
4. Setiap blok m dienkripsi dengan Persamaan 2 dan Persamaan 3.

   $$Nilai\ a\ (Gamma) = g^k \bmod p \quad (2)$$
   $$Nilai\ b\ (Delta) = y^k . m \bmod p \quad (3)$$

   di mana :
   m = blok *plaintext*
   k = bilangan acak dengan syarat $1 \leq k \leq p-2$
   a = *ciphertext* ganjil untuk blok pesan m
   b = *ciphertext* genap untuk blok pesan m
5. Pasangan a dan b adalah *chipertext* untuk blok pesan m. *Ciphertext* diperoleh dengan menyusun masing-masing nilai a dan b dengan urutan $a_1, b_1, a_2, b_2, \ldots, a_n, a_n$

Proses dekripsi algoritma Elgamal menggunakan *private key* x dan p untuk mendekripsi a dan b menjadi *plaintext* (m). Langkah-langkah dalam melakukan dekripsi *ciphertext* adalah sebagai berikut:

1. Penentuan nilai Gama dan Delta adalah nilai Gamma (a) diperoleh dari *chipertext* dengan urutan ganjil, sedangkan Delta (b) diperoleh urutan genap.
2. Hitung *plaintext* m dengan Persamaan 4.

   $$m = b \,.\, a^{(p-1-x)} \bmod p \quad (4)$$

di mana:
m = *plaintext*
a = *chipertext* ganjil untuk blok pesan m
b = *chipertext* genap untuk blok pesan m
p = bilangan prima dengan syarat $p > 255$
x = bilangan acak dengan syarat $x < p$

3. Ubah nilai m yang didapat ke dalam nilai ASCII.
4. Susun *plaintext* dengan urutan $m_1, m_2, \dots, m_n$.

Proses Enkripsi dan Dekripsi Merkle-Hellman-Knapsack

Enkripsi dan dekripsi dimulai dengan proses pembentukan kunci. Proses pembentukan kunci algoritma Merkle-Hellman-Knapsack dilakukan dengan menentukan barisan *superincreasing* s_i sebagai *private key* dan menggunakan bilangan prima m dan n sebagai salah satu variabel perhitungan *public key*. Di mana bilangan prima m dan n harus lebih besar dari elemen terakhir pada deretan s_i dan n relatif prima dengan m. *Public Key* dihitung dengan Persamaan 5 [7]:

$$Kp_i = s_i \times n \bmod m \quad (5)$$

di mana:
Kp = kunci publik
s_i = elemen dalam barisan *superincreasing private key*
m = bilangan prima yang harus lebih besar dari elemen terakhir s_i
n = bilangan prima yang relatif prima dengan m

Proses enkripsi algoritma Merkle-Hellman-Knapsack dilakukan menggunakan *public key* Kp. Langkah-langkah dalam melakukan enkripsi pesan adalah sebagai berikut:

1. Pesan atau *plaintext* dibagi kedalam beberapa *blok bit* yang panjangnya sama dengan panjang elemen kunci publik.
2. Hitung nilai *ciphertext* menggunakan Persamaan 6.
$$c = b_1 \times Kp_1 + b_2 \times Kp_2 + \dots + b_n \times Kp_n \quad (6)$$
di mana:
c = *ciphertext*
b = bit dari *plaintext*
Kp = *Public Key*

Proses dekripsi algoritma Merkle-Hellman-Knapsack dilakukan menggunakan *private key* s_i. Langkah-langkah dalam melakukan dekripsi *ciphertext* adalah sebagai berikut:

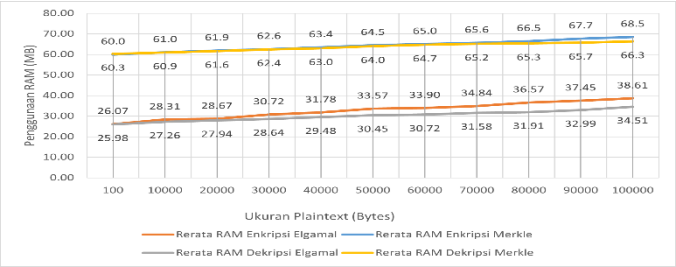
1. Kunci *private* digunakan untuk melakukan proses dekripsi.
2. Tentukan *plaintext* menggunakan Persamaan 7.
$$P_i = c \times n^{-1} \bmod m \quad (7)$$
Di mana n^{-1} nilai modular invers dari m. Nilainya adalah banyak urutan yang diperlukan agar hasil $n \bmod m = 1$. Jadi dilakukan percobaan menginputkan nilai x = (1, 2, 3, 4, …, dst) kedalam Persamaan 8.
$$x = (n \times x) \bmod m \quad (8)$$
di mana:
P_i = *plaintext*
c = *ciphertext*
$n - 1$ = nilai invers dari m d
m = bilangan prima yang harus lebih besar dari elemen terakhir s_i
x = banyak perhitungan hingga n mod $m = 1$
3. Kurangkan nilai Pi dengan nilai elemen s_i yang besarnya mendekati nilai Pi untuk menghitung nilai biner Pi. Prosedur ini diulang sampai hasil = 0 atau 1.
4. Gabungkan nilai yang dihasilkan dengan syarat jika hasilnya adalah 0, nilai biner dimasukkan, dan jika hasilnya adalah satu (1), maka nilai tidak dimasukkan.

3. HASIL DAN PEMBAHASAN

Penelitian ini menggunakan 11 data *plaintext*. Teks yang digunakan dalam penelitan tersusun dari *American Standard Code for Information Interchange* (ASCII) *printable characters* dengan kode karakter 32 sampai 127. *Plaintext* yang digunakan memiliki ukuran yang beragam, mulai dari 100 *Bytes* hingga 100000 *Bytes*.

Tabel 1. *Plaintext* yang Digunakan

No	***Plaintext* yang Digunakan**	**Ukuran (*Bytes*)**
1	Di zaman modern ini, teknologi telah menjadi bagian yang tak terpisahkan dari kehidupan sehari-hari.	100
2	Di dalam sejarah Eropa, Abad Pertengahan adalah kurun waktu yang bermula sekitar tahun 500 dan berakhir pada tahun 1500 tarikh Masehi … wilayah kekaisaran.	10000
3	Tata Surya[a] adalah kumpulan benda langit yang terdiri atas sebuah bintang yang disebut Matahari dan semua objek yang terikat oleh gaya gravitasinya … spiral yang terjadi.	20000
4	Letnan Kolonel Imam Syafei (27 September 1918 9 September 1982), sering dieja Imam Sjafe'i, Imam Sapi'ie, Imam Syafi'ie, Imam Sjafei, dan populer dengan sebutan Bang Pi'ie … jabatan perdana menteri.	30000
5	Partai Komunis Indonesia (PKI) adalah sebuah partai komunis di Hindia Belanda, kemudian Indonesia … bergabung dengan PKI.	40000
6	Hindia Belanda atau Hindia Timur Belanda (bahasa Belanda: Nederlands(ch)-Indi) (bahasa Inggris: Dutch East Indies) adalah sebuah daerah pendudukan Belanda … 500000 kuli diangkut.	50000
7	(Malaysia (Jawi:) adalah sebuah negara federal[11] yang terdiri dari tiga belas negeri (negara bagian) dan tiga wilayah federal di Asia Tenggara dengan luas 330.803km persegi … Malaya Britania hingga pembubarannya.	60000
8	Indonesia, dengan nama resmi Republik Indonesia,[a] adalah sebuah negara kepulauan di Asia Tenggara yang dilintasi garis khatulistiwa dan berada di antara daratan benua Asia dan Oseania … Susilo Bambang Yudhoyono.	70000
9	Amerika Serikat[h] (AS) adalah negara republik konstitusional federal yang terdiri dari lima puluh negara bagian dan sebuah distrik federal … tertinggi di dunia.	80000
10	Rusia, dengan nama resmi Federasi Rusia (bahasa Rusia: , translit. Rossyskaya Federtsiya), adalah sebuah negara federasi yang bersistem semi-presidensial … *Transport Statistics Database*.	90000
11	Perang Dunia II atau Perang Dunia Kedua (biasa disingkat menjadi PDII atau PD2) adalah sebuah perang global yang berlangsung mulai tahun 1939 sampai 1945 … *the devil's*.	100000

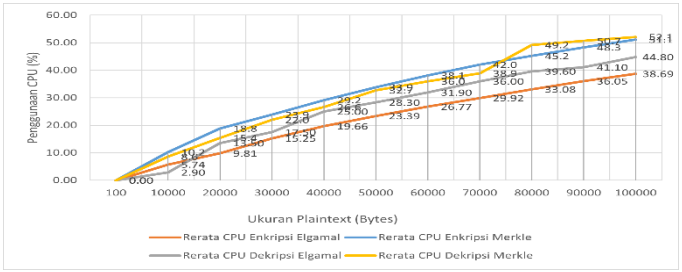


Gambar 1 Grafik Perbandingan Penggunaan RSS Memory

Dari 11 data penelitian tersebut dilakukan 11 kali pengujian untuk masing-masing algoritma Elgamal dan Merkle-Hellman-Knapsack. Setelah diujikan, diperoleh data yang digambarkan dengan 4 grafik perbandingan performa.

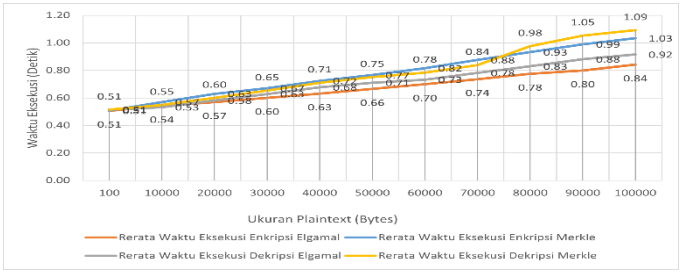
Dari Gambar 1 tersebut terlihat bahwa baik pada proses enkripsi maupun dekripsi, alokasi memori proses enkripsi dan dekripsi meningkat secara bertahap seiring dengan bertambahnya ukuran data. Pada proses dekripsi, pola serupa terlihat. Merkle-Hellman-Knapsack (garis kuning) tetap memerlukan lebih banyak memori, mulai dari 60 MB hingga mencapai 68,5 MB, sementara Elgamal (garis abu-abu) lebih efisien, dengan alokasi memori proses enkripsi dan dekripsi yang meningkat dari 25,98 MB hingga 38,61 MB. Sehingga secara keseluruhan, algoritma Elgamal membutuhkan lebih sedikit memori dibandingkan algoritma Merkle-Hellman-Knapsack dalam

enkripsi dan dekripsi pesan teks, sehingga Elgamal dapat lebih sesuai untuk sistem dengan keterbatasan sumber daya memori.



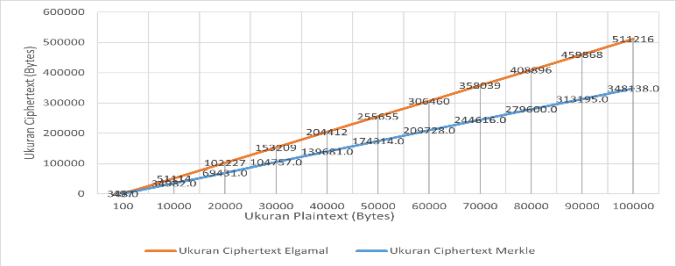
Gambar 2 Grafik Perbandingan Penggunaan CPU

Dari Gambar 2 tersebut terlihat pola yang menunjukkan peningkatan penggunaan CPU proses enkripsi dan dekripsi seiring bertambahnya ukuran *plaintext*. Pada proses enkripsi, algoritma Merkle cenderung menggunakan CPU lebih tinggi dibandingkan dengan Elgamal, terutama saat ukuran *plaintext* mulai melebihi 40.000 *bytes*, dengan penggunaan CPU proses enkripsi dan dekripsi yang mencapai 52,1% pada ukuran 100.000 *bytes*. Sebaliknya, Elgamal memiliki penggunaan CPU proses enkripsi dan dekripsi yang lebih rendah selama enkripsi, dengan rata-rata 38,69% pada ukuran *plaintext* yang sama.Pada proses dekripsi, pola yang serupa juga terlihat, namun perbedaan penggunaan CPU proses enkripsi dan dekripsi antara kedua algoritma tidak sebesar saat enkripsi. Algoritma Merkle tetap menunjukkan penggunaan CPU proses enkripsi dan dekripsi yang lebih tinggi dibandingkan Elgamal, dengan penggunaan yang mencapai 50,27% untuk Merkle dan 44,80% untuk Elgamal pada ukuran *plaintext* 100.000 *bytes*. Secara keseluruhan, Elgamal memerlukan lebih sedikit sumber daya CPU untuk memproses data berukuran besar dibandingkan dengan Merkle-Hellman-Knapsack, sehingga Elgamal lebih cocok digunakan untuk sistem dengan keterbatasan sumber daya CPU.



Gambar 3 Grafik Perbandingan Waktu Eksekusi

Gambar 3 menunjukan bahwa kedua algoritma mengalami peningkatan waktu eksekusi seiring bertambahnya ukuran *plaintext*. Pada proses enkripsi, algoritma Merkle membutuhkan waktu eksekusi yang lebih lama dibandingkan Elgamal. Hal ini terlihat pada ukuran *plaintext* 100.000 *bytes*, di mana Merkle membutuhkan waktu 1,09 detik, sedangkan Elgamal hanya memerlukan 0,92 detik. Demikian pula pada proses dekripsi, algoritma Merkle kembali menunjukkan waktu eksekusi yang lebih tinggi dibandingkan Elgamal. Pada ukuran *plaintext* 100.000 *bytes*, Merkle memerlukan waktu 1,03 detik, sementara Elgamal hanya membutuhkan 0,84 detik. Jadi secara keseluruhan, Merkle-Hellman-Knapsack memerlukan waktu eksekusi lebih lama dibandingkan dengan Elgamal, sehingga Elgamal lebih cocok untuk aplikasi yang membutuhkan performa tinggi dan waktu pemrosesan cepat.



Gambar 4 Grafik Perbandingan Ukuran Ciphertext

Gambar 4 menampilkan grafik hasil perbandingan ukuran *ciphertext* yang dihasilkan oleh algoritma Elgamal dan Merkle Hellman Knapsack terhadap berbagai ukuran *plaintext*. Terlihat bahwa ukuran *ciphertext* meningkat seiring dengan peningkatan ukuran *plaintext* pada kedua algoritma. Namun, algoritma Elgamal secara konsisten menghasilkan *ciphertext* yang lebih besar dibandingkan dengan Merkle Hellman Knapsack untuk setiap ukuran *plaintext* yang diuji. Pada ukuran *plaintext* 10.000 *bytes*, ukuran *ciphertext* Elgamal sebesar 51.114 *bytes*, sedangkan Merkle Hellman Knapsack hanya sebesar 34.982 *bytes*. Perbedaan ini semakin terlihat jelas saat ukuran *plaintext* semakin besar. Selanjutnya, ukuran *plaintext* 100.000 *bytes*, Elgamal menghasilkan *ciphertext* sebesar 511.216 *bytes*, sementara Merkle Hellman Knapsack menghasilkan *ciphertext* sebesar 348.138 *bytes*. Jadi secara keseluruhan, Merkle-Hellman-Knapsack lebih efisien dalam penggunaan ruang penyimpanan dibandingkan Elgamal, terutama untuk pesan dengan ukuran yang besar. Hal ini dapat menjadi pertimbangan penting dalam aplikasi dengan keterbatasan kapasitas penyimpanan atau bandwidth.

4. KESIMPULAN

Aplikasi pengukuran performa enkripsi dan dekripsi Elgamal dan Merkle-Hellman-Knapsack dapat digunakan untuk mengamankan pesan teks.

Aplikasi pengukuran performa enkripsi dan dekripsi dapat melakukan pengukuran performa proses enkripsi dan dekripsi algoritma Elgamal dan Merkle-Hellman-Knapsack, sehingga dapat menghasilkan data performa berupa kebutuhan RSS *memory*, kebutuhan CPU, waktu eksekusi, dan ukuran data *ciphertext*.

Algoritma kriptografi yang lebih sesuai untuk digunakan pada sistem dengan keterbatasan sumber daya memori dan CPU adalah algoritma Elgamal. Hal ini ditunjukkan oleh ukuran kapasitas penggunaan RSS *memory* dan persentase penggunaan CPU yang lebih kecil, serta waktu eksekusi yang lebih singkat pada algoritma Elgamal dibandingkan dengan algoritma Merkle-Hellman Knapsack dalam proses enkripsi dan dekripsi pesan teks.

5. SARAN

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran penelitian untuk selanjutnya antara lain:

1. Menambah jenis *input* yang dapat diproses seperti *input file* dokumen .txt dan csv.
2. Menambah jumlah maksimal karakter *plaintext* yang dapat dimasukkan dan diproses dalam enkripsi dan dekripsi.

DAFTAR PUSTAKA

[1] M. L. M. Diepenbroek, “Myths and Histories of the Spartan scytale,” University of Bristol, 2020.

[2] V. Pongsitammu, A. R. Y. Simatupang, D. Annura, Y. Sari, Dachi, and D. R. Harries, “Keamanan Kriptosistem Modern Berdasarkan Algoritma Kriptografi Kunci Publik,” *Siteba*, vol. 2, no. 1, p. 2023, 2023, [Online]. Available: https://journal.iteba.ac.id/index.php/jurnalsiteba/indexSITEBA

[3] Ardi, *Android & Kriptografi Algoritma Rivest Code 6*. Medan, Indonesia: CV Sentosa Deli Mandiri, 2020.

[4] D. Widyawan and I. Imelda, "Pengamanan File Menggunakan Kriptografi Dengan Metode Aes-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi," *Skanika*, vol. 4, no. 1, pp. 15–22, 2021, doi: 10.36080/skanika.v4i1.2216.

[5] A. Muhammad, "Modifikasi Huruf Katakana Pesan Menggunakan Metode Advance Vigenere Cipher," *J. Sci. Appl. Informatics*, vol. 3, no. 3, pp. 156–162, 2020, [Online]. Available: https://doi.org/10.36085/jsai.v3i3.1027

[6] A. M. Fajrin, J. R. Benedict, and H. J. Kusuma, "Analisis Performa dari Algoritma Kriptografi RSA dan ElGamal dalam Enkripsi dan Dekripsi Pesan," *J. Ris. Sist. Inf. Dan Tek. Inform.*, vol. 8, no. 1, pp. 91–98, 2023, [Online]. Available: https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

[7] R. Rahmadani, H. D. Hutahaean, and R. D. Sari, "Implementasi Algoritma Merkle-Hellman Knapsack dalam Penyandian Record Database," *MEANS (Media Inf. Anal. dan Sist.*, vol. 5, no. 2, pp. 162–165, 2020, [Online]. Available: http://dx.doi.org/10.54367/means.v5i2.983

[8] V. K. Pachghare, *Chryptography and Information Security, Third Edition*. Delhi, India: PHI Learning Pvt, 2019. [Online]. Available: https://www.google.co.id/books/edition/CRYPTOGRAPHY_AND_INFORMATION_SECURITY_TH/k-rKDwAAQBAJ?hl=id&gbpv=0

[9] E. Jintcharadze and M. Lavich, "Hybrid Implementation of Twofish, AES, ElGamal and RSA Cryptosystems," in *IEEE East-West Design and Test Symposium*, Institute of Electrical and Electronics Engineers Inc, 2020. [Online]. Available: https://doi.org/10.1109/EWDTS50664.2020.9224901

[10] K. E. Widiyasari, "Perbandingan Algoritma RSA dan Merkle-Hellman Dalam Rancang Bangun Aplikasi Penyandian Pesan Teks," Universitas Pembangunan Panca Budi Medan, 2022. [Online]. Available: https://eprints.pancabudi.ac.id/id/eprint/2651/

[11] D. Kurniawan, N. Mayasari, and W. Fitriani, "Perbandingan Algoritma Cipher Dengan Algoritma ROT13 Pada Proses Pengamanan Data," *J. Darma Agung*, vol. 30, no. 1, pp. 534–541, 2022.

[12] F. Alfiah, R. Sudarji, and D. T. Al Fatah, "Aplikasi Kriptografi Dengan Menggunakan Algoritma Elgamal Berbasis Java Desktop Pada Pt. Wahana Indo Trada Nissan Jatake," *ADI Bisnis Digit. Interdisiplin J.*, vol. 1, no. 1, pp. 22–34, 2020, doi: 10.34306/abdi.v1i1.114.