Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 91-98
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Analisis Performa dari Algoritma Kriptografi RSA dan ElGamal dalam Enkripsi dan Dekripsi Pesan

Ahmad Miftah Fajrin[1] , Jeremy Richard Benedict[2], Henri Jayanata Kusuma[3]
[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Surabaya, Indonesia
Email: [1]ahmadmiftah@staff.ubaya.ac.id, [2]yerechen@gmail.com, [3]hjayanata@gmail.com

Abstract

Cryptography is one of the most important things in computer science because it is related to data security or message security. Cryptographic algorithms are used to secure a message, the most popular of which are the Rivest-Shamir-Adleman (RSA) and ElGamal algorithms. ElGamal employs a prime number modulo while RSA employs the factorial of two large integer numbers. The workings of these two algorithms are distinct, and each has distinct advantages. Even though the two algorithms use different processes, they can both encrypt and decrypt a message. There are speed factors of execution time and peak memory when running the encryption and decryption process. One of these two factors is hardware, which determines the environment that will be used. According to the findings of this study, the RSA algorithm has a faster execution time than ElGamal. In terms of peak memory, both algorithms get the result similarly, with the same peak memory results shown..

Keywords*: Cryptography, RSA, ElGamal, Execution time, Peak Memory*

Abstrak

Kriptografi dalam ilmu komputasi adalah salah satu topik yang penting karena terkait dengan keamanan data atau pesan. Dalam mengamankan suatu pesan, tersedia algoritma kriptografi dan yang popular yaitu Rivest-Shamir-Adleman (RSA) dan ElGamal. RSA menggunakan factorial dua angka integer yang besar sedangkan ElGamal menggunakan modulo bilangan prima. Cara kerja dua algoritma ini berbeda dan tentu mempunyai kelebihan masing-masing. Dua algoritma dengan proses yang berbeda tersebut tetap dapat digunakan untuk proses enkripsi dan dekripsi suatu pesan. Ketika menjalankan proses enkripsi dan dekripsi, terdapat faktor kecepatan waktu eksekusi dan peak memory. Dua faktor itu sangat penting dan menentukan environment yang akan digunakan salah satunya hardware. Hasil dari penelitian ini didapatkan algoritma RSA mempunyai kecepatan waktu eksekusi yang lebih cepat daripada ElGamal. Sedangkan untuk peak memory, kedua algoritma mempunyai performa yang sama dengan ditunjukkan hasil peak memory yang sama.

Kata Kunci*: Kriptografi, RSA, ElGamal, Waktu Eksekusi, Peak Memory*

1. PENDAHULUAN

Salah satu mekanisme keamanan yang ada di komputer yang banyak digunakan saat ini adalah metode kriptografi [1]. Kata kriptografi adalah gabungan kata *kryptós* yaitu menyembunyikan dan *graphein* yaitu menulis. Menulis dengan menyembunyikan informasi adalah inti dari kriptografi. Dengan menggunakan kriprografi, suatu informasi seperti datan dan pesan dapat terjadi keamanan dan

kerahasiaanya. Pada kriptografi, enkripsi adalah salah satu hal esensial yang penting dalam mengamankan sebuah pesan. Enkripsi pada sebuah pesan dapat mencegah orang yang tidak berkepentingan untuk membaca pesan rahasia [2]. Jika pesan rahasia dapat dibaca oleh semua pihak, akan menjadi masalah tersendiri yang dapat dimanfaatkan oleh berbagai pihak salah satunya *attacker*. Banyak penelitian yang sudah dilakukan untuk mengamankan sebuah pesan menggunakan algoritma kriptografi [3] [4].

Algoritma kriptografi untuk mengamankan pesan yaitu Rivest-Shamir-Adleman (RSA) dan ElGamal. Algoritma RSA dan ElGamal dibangun berdasarkan ide dari algoritma *asymmetric cryptography* yaitu Diffie-Hellman-Merkle. Konsep *asymmetric* adalah setiap anggota mempunyai kuncinya sendiri. Algoritma RSA adalah algoritma yang paling banyak digunakan karena kekuatannya pada bilangan faktorial dari dua bilangan prima yang besar [5]. Namun RSA mempunyai kebutuhan waktu yang lama daripada algoritma yang berdasarkan konsep *symmetric cryptography* [6]. Disisi lain, algoritma ElGamal adalah termasuk algoritma a*symmetric* yang menawarkan kemudahan dalam hal mekanisme proses enkripsi [7]. Algortima ElGamal dibangun untuk proses perhitungan secara diskrit pada masalah *finite field* [8]. Meskipun algoritma RSA dan ElGamal adalah algoritma *asymmetric*, terdapat perbedaan cara proses skema. Pada algoritma RSA, mempunyai kekuatan untuk menemukan factorial dari dua angka interger yang besar sedangkan ElGamal untuk menemukan *discrete logs* modulo dari bilangan prima [9].

Pada penelitian ini, dua algoritma *symmetric* yaitu RSA dan ElGamal akan dianilisis performanya. Performa RSA dan ElGamal akan dilihat berdasarkan pada tingkat penggunaan *peak memory* dan kecepatan waktu. Penggunaan *peak memory* sangat penting agar tidak terjadi *stuck* atau kehabisan *memory* ketika menggunakan algoritma kriptografi. Kecepatan waktu juga akan dihitung pada analisis ini agar dapat dilihat dari segi efisiensi penggunaan waktu pada eksekusi proses enkripsi.

2. METODOLOGI PENELITIAN

Dalam penelitian ini, akan dilakukan uji coba performa antara algoritma RSA dan ElGamal dalam melakukan proses enkripsi. Teori dan *pseudocode* dari algoritma akan juga dijelaskan.

2.1. Rivest-Shamir-Adleman (RSA)

RSA memiliki dua kunci yaitu *public key* dan *private key* [10]. Algoritma RSA adalah salah satu algoritma dengan skema *Public Key Cryptography*. Keamanan proses *encrypt* dan *decrypt* pada algoritma RSA terletak di modulus *n* yang difaktorkan dengan sangat besar. Cara kerja algoritma RSA untuk mengenkripsi pesan yaitu seperti berikut, Alice mengirim kunci publik (*public key*) miliknya (*n* & *e*) kepada Bob, kemudian merahasiakan kunci privat (*private key*) miliknya. Setelah itu, Bob menuruskan pesan *M* ke Alice. Ketika melakukan enkripsi pada pesan *M*, Bob melakukan *split* pesan M menjadi beberapa $m < n$, kemudian Bob menghitung *c*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 91-98
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

sebagai *ciphertext.* Setelah mendapatkan hasil penghitungan enkripsi dari *c* tersebut, Bob mengirimkan hasil tersebut kepada Alice. Untuk *pseudocode* dari RSA dapat dilihat pada Gambar 1.

```
def gcd(a, b):
 r = 0
 while 1:
  r = a % b
  if r == 0:
   return b
  a = b
  b = r
def isPrime(num):
 if num > 1:
  for i in range(2,num):
   if num % i == 0:
    return False
  else:
   return True
 else:
  return False
def RSA_encrypt(character, public, n):
 return (character**public) % n
def RSA_decrypt(character, private, n):
 return (character**private) % n
def Alphabet(num):
 if num > 53:
  return num % 53
 else:
  return num
def RSA(RSAinput):
 p1 = 17
 q1 = 11

 if isPrime(p1):
  if isPrime(q1):
   prime = isPrime(q1)
  else:
   print('q is not prime\n')
 else:
  print('p is not prime\n')
```

```
 n1 = p1 * q1
 fn = (p1-1)*(q1-1)
 x = 1
 helper = False
 while x < 6:
  d = (x * fn) + 1
  i1 = 2
  while i1 < d:
   if d % i1 == 0:
    b1 = d / i1
    print("the public key is
(%i,%i) and the private key is
(%i,%i,%i)\n"
%(n1,i1,p1,q1,b1))
    helper = True
    break
   else:
    i1+=1
  else:
   x+=1
  if helper == True:
   break

encryptor = {..}
decryptor = {..}
msg = RSAinput
print()
encrypt = ""; decrypt = ""
num = []
encrypted = []
decrypted = []
arrhelp = []
# Convert to number
for char in msg:
 num.append(encryptor[char])
 length = len(num)
```

```
 # Encrypt
 for x in range(length):
  arrhelp.append(RSA_encry
pt(num[x],i1,n1))
   num[x] =
Alphabet(RSA_encrypt(num
[x],i1,n1))
  for number in num:
   encrypted.append(decrypt
or[number])
  for char in encrypted:
   encrypt += char
  print("Encrypted result = ",
encrypt, "\n")

  # Decrypt
  for y in range(length):
   num[y] =
RSA_decrypt(arrhelp[y],int(
b1),n1)

  # Convert to character
  for number in num:

  decrypted.append(decryptor
[number])

  # Print Decrypted message
  for char in decrypted:
   decrypt += char
  print("Decrypted result = ",
decrypt)
```

Gambar 1. *Pseudocode* RSA

Terdapat fungsi utama pada Gambar 1 yaitu *def gcd, def isPrime, def RSA_encrypt* dan *def Alphabet. Def gcd* digunakan untuk menentukan nilai *gcd* yang ada. *Def isPrime* berfungsi mengetahui bilangan yang ada adalah bilangan prima. *Def RSA_encrypt* digunakan untuk melakukan enkripsi RSA, sedangkan *def RSA_decrypt* digunakan untuk melakukan dekripsi RSA. *Def Alphabet* mengubah angka diatas 53 menjadi angka di bawah 53 agar dapat di dekripsi oleh *decryptor*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 91-98
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

2.2. ElGamal

Algoritma ElGamal juga termasuk dalam konsep kunci *public* atau *Public Key Cryptography*, sama seperti dengan algoritma RSA. Pada umumnya, algoritma ElGamal dapat melakukan *digital signature* akan tetapi dapat digunakan juga sebagai cara untuk enkripsi dan dekripsi informasi, baik pesan maupun data [11]. Algoritma ini didasarkan pada grup *Zp**. ElGamal mempunyai proses penting yaitu *generate key*, *encryption* pesan, dan *decryption pesan*. Algoritma ElGamal termasuk algoritma *cipher block*, yaitu proses mengenkripsi *block-block plaintext* yang menghasilkan *block-block ciphertext* kemudian di dekripsi. Hasil dari dekripsi akan dijadikan menjadi informasi pesan yang sesuai dengan urutan. *Pseudocode ElGamal* dapat dilihat pada Gambar 2

```
def El_Gamal(EGinput):
  import random
  from math import pow

  a = random.randint(2, 10)

  def gcd(a, b):
    if a < b:
      return gcd(b, a)
    elif a % b == 0:
      return b;
    else:
      return gcd(b, a % b)

  # Generating large random
numbers
  def gen_key(q):

    key = random.randint(pow(10,
20), q)
    while gcd(q, key) != 1:
      key =
random.randint(pow(10, 20), q)

    return key

# Modular exponentiation
  def power(a, b, c):
    x = 1
    y = a

    while b > 0:
      if b % 2 != 0:
        x = (x * y) % c;
      y = (y * y) % c
      b = int(b / 2)

    return x % c
```

```
  # =Asymmetric encryption
  def encrypt(msg, q, h, g):

    en_msg = []

    k = gen_key(q)# Private key
for sender
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
      en_msg.append(msg[i])

    print("g^k used : ", p)
    print("g^ak used : ", s)
    for i in range(0, len(en_msg)):
      en_msg[i] = s *
ord(en_msg[i])

    return en_msg, p

  def decrypt(en_msg, p, key, q):

    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
      dr_msg.append(chr(int(en_
msg[i]/h)))

    return dr_msg

  # Driver code
  def main():
    msg = EGinput

    q = random.randint(pow(10,
20), pow(10, 50))
    g = random.randint(2, q)
```

```
key = gen_key(q)# Private
key for receiver
    h = power(g, key, q)
    print("g used : ", g)
    print("g^a used : ", h)

    en_msg, p =
encrypt(msg, q, h, g)
    dr_msg =
decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr  msg)
    print("Decrypted
Message :", dmsg);


  if __name__ ==
'  main  ':
    main()
```

Gambar 2. *Pseudocode* ElGamal.

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 91-98
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Tiap bagian *code* pada Gambar 2 memiliki fungsi dan tujuannya masing-masing. Fungsi dan tujuan tersebut, antara lain :

a) *def El_Gamal* kami buat agar semua code enkripsi dan dekripsi El-Gamal dapat dijalankan pada cell terakhir bersamaan dengan algoritma RSA.
b) Melakukan *random* angka dari 2 hingga 10.
c) *def gcd* digunakan untuk menghitung dan menentukan *gcd* yang ada.
d) *def gen_key* digunakan untuk menentukan kunci yang akan dipakai, serta menentukan jumlah angka *random* yang besar.
e) *def power* digunakan untuk melakukan *modular exponentiation*
f) *def encrypt* digunakan ketika enkripsi pesan.
g) *def decrypt* digunakan ketika dekripsi pesan.
h) *def main* merupakan fungsi akhir yang digunakan untuk menjalankan seluruh *code* yang ada.

3. HASIL DAN PEMBAHASAN

Perbandingan kecepatan waktu dan *peak memory* akan dihitung untuk algoritma RSA dan ElGamal. Perbandingan yang dilakukan akan menentukan algoritma terbaik diantara dua algoritma tersebut.

3.1. Kecepatan Waktu

Untuk menghitung kecepatan waktu pada RSA, akan diuji coba sebuah pesan dan sesuai dengan pseudocode pada Gambar 3.

```
AlgorithmInput = input("type your message here: ")
print()
print("RSA:")
%time RSA(AlgorithmInput)
print()
```

Gambar 3. *Pseudocode* Input Message for RSA

Input pesan pada RSA akan dimasukkan pada saat awal eksekusi proses enkripsi. Ketika input pesan sudah dimasukkan maka akan keluar kecepatan waktu yang diperlukan untuk proses enkripsi dan dilanjutkan dengan dekripsi pada algoritma RSA. Begitu juga untuk algoritma ElGamal, proses memasukkan input dapat dilihat pada Gambar 4.

```
AlgorithmInput = input("type your message here: ")
print()
print("El-Gamal:")
%time El_Gamal(AlgorithmInput)
print()
```

Gambar 4. *Pseudocode* Input Message for ElGamal.

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 91-98
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Untuk ElGamal pada proses perhitungan kecepatan waktu caranya sama dengan RSA. Perbedaannya hanya pada algoritma yang digunakan. Untuk uji coba, inputan yang akan digunakan berupa pesan yaitu : "Uvuvwevwevwe Onyetemevwe Ngubwemubwem Osas". Hasil dari algoritma RSA dan ElGamal dapat dilihat pada Gambar 5.

```
RSA :

the public key is (187,7) and the private key is (17,11,23)
Encrypted result =  hQWQHiQHiQHiMiImiliMiQHiMGzWLHiMWLHiMMiKaK
Decrypted result =  Uvuvwevwevwe Onyetemevwe Ngubwemubwem Osas
CPU time: user 3.93 ms, sys: 1.03 ms, total: 4.96 ms
Wall time: 8.77 ms
--------------------------------------------------------------------------------

El-Gamal :

g used :  177718463775223549916360342602819989455200926429l5
g^a used :  14334559464031835237598407925682008797939369021547
g^k used :  35739677525885922274322458182799414140214032533075
g^ak used :  78451746945318436041785350781634269204945301802 93
Decrypted Message : Uvuvwevwevwe Onyetemevwe Ngubwemubwem Osas
CPU time: user 6.36 ms, sys: 930 µs, total: 7.29 ms
Wall time: 13.3 ms
```

Gambar 5. Hasil Kecepatan Waktu dari Algoritma RSA dan ElGamal

Hasil pada Gambar 5 menunjukkan bahwa algoritma RSA mempunyai *CPU time* dengan total 4.96 ms sedangkan ElGamal mempunyai *CPU time* dengan total 7.29 ms. Ini menunjukkan bahwa algoritma RSA lebih cepat untuk eksekusi proses enkripsi dan dekripsi dari pada ElGamal.

3.2. *Peak Memory*

Untuk menghitung *peak memory* dari algoritma RSA dan ElGamal, *pseudocode* dapat dilihat pada Gambar 6.

```
AlgorithmInput = input("type your message here: ")
print()
print("RSA:")
%memit RSA(AlgorithmInput)

print()
print("El-Gamal:")
%memit El_Gamal(AlgorithmInput)
print()
```

Gambar 6. *Pseudocode* Kalkulasi *Peak Memory*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 91-98
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Algoritma RSA dan ElGamal akan dimasukkan sebuah input pesan yaitu "Uvuvwevwevwe Onyetemevwe Ngubwemubwem Osas". Enkripsi dan dekripsi dilakukan secara bertahap yang akhirnya akan mendapat kalkulasi *peak memory.* Untuk hasil kalkulasi dapat dilihat pada Gambar 7.

```
RSA :

the public key is (187,7) and the private key is (17,11,23)
Encrypted result =  hQWQHiQHiQHiMiImiliMiQHiMGzWLHiMWLHiMMiKaK
Decrypted result =  Uvuvwevwevwe Onyetemevwe Ngubwemubwem Osas
Peak Memory: 184.16 MiB, increment: 0.00 MiB
------------------------------------------------------------------------

El-Gamal :

g used :  177718463775223549916360342602819989455200926429l5
g^a used :  14334559464031835237598407925682008797939369021547
g^k used :  35739677525885922274322458182799414140214032533075
g^ak used :  78451746945318436041785350781634269204945301802l3
Decrypted Message : Uvuvwevwevwe Onyetemevwe Ngubwemubwem Osas
Peak Memory: 184.16 MiB, increment: 0.00 MiB
```

Gambar 7. Hasil kalkulasi *Peak Memory.*

Terlihat pada hasil kalkulasi *peak memory,* Algoritma RSA dan ElGamal memiliki hasil yang sama. Ini terlihat pada Gambar 7 yang mendapatkan *peak memory* sebesar 184.16 MiB

4. SIMPULAN

Dari kedua algoritma yang telah kami bandingkan, yaitu algoritma RSA dan algoritma ElGamal. Dapat disimpulkan bahwa kedua algoritma tersebut merupakan algoritma yang dapat digunakan untuk melakukan enkripsi dan dekripsi pesan dengan tepat, serta keduanya memiliki ciri khas dan keunggulannya masing-masing. Namun, berdasarkan uji coba yang telah dilakukan, algoritma RSA memiliki keunggulan yaitu dapat melakukan enkripsi dan dekripsi pesan dengan kecepatan waktu komputasi yang lebih cepat apabila dibandingkan dengan algoritma ElGamal. Kecepatan waktu komputasi tersebut diambil dari kecepatan waktu *CPU Times* dan *Wall Time* yang telah dilakukan. Kedua algoritma ini memiliki kesamaan dari hasil uji coba kedua kami, yaitu perbandingan besar pemakaian *memory*. Hal ini dikarenakan berdasarkan hasil uji yang dilakukan, jumlah *peak memory* dan *increment* dari kedua algoritma ini menghasilkan angka yang sama. Namun hasil dari pemakaian *memory* (*peak memory* dan *increment*) tersebut juga bisa berbeda. Maka dapat disimpulkan bahwa algoritma RSA memiliki keunggulan yang lebih menguntungkan apabila dibandingkan dengan algoritma ElGamal.

DAFTAR PUSTAKA

[1] R. Verma and J. Dhiman, "Implementation of Improved Cryptography Algorithm," *International Journal of Information Technology and Computer Science*, vol. 14, no. 2, pp. 45–53, Apr. 2022, doi: 10.5815/ijitcs.2022.02.04.

[2] F. Mallouli, A. Hellal, N. Sharief Saeed, and F. Abdulraheem Alzahrani, "A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms," in *Proceedings - 6th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2019 and 5th IEEE International Conference on Edge Computing and Scalable Cloud, EdgeCom 2019*, Jun. 2019, pp. 173–176. doi: 10.1109/CSCloud/EdgeCom.2019.00022.

[3] Y. Ma, C. Li, and B. Ou, "Cryptanalysis of an image block encryption algorithm based on chaotic maps," *Journal of Information Security and Applications*, vol. 54, Oct. 2020, doi: 10.1016/j.jisa.2020.102566.

[4] R. Adi Putra and E. R. Prasetyo, "Android-Based Text Message Encryption and Decryption Application Using the Advanced Encryption Standard Algorithm," *Jurnal Media Computer Science*, vol. 2, no. 1, pp. 57–62, 2023.

[5] D. Rachmawati and M. A. Budiman, "On Using the First Variant of Dependent RSA Encryption Scheme to Secure Text: A Tutorial," in *Journal of Physics: Conference Series*, Jun. 2020, vol. 1542, no. 1. doi: 10.1088/1742-6596/1542/1/012024.

[6] A. Hamza and B. Kumar, "A Review Paper on DES, AES, RSA Encryption Standards," in *Proceedings of the 2020 9th International Conference on System Modeling and Advancement in Research Trends, SMART 2020*, Dec. 2020, pp. 333–338. doi: 10.1109/SMART50582.2020.9336800.

[7] D. Sow, L. Robert, and P. Lafourcade, "Linear Generalized ElGamal Encryption Scheme," *Cryptology ePrint Archive*, 2020.

[8] A. K. Koundinya and G. SK, "Two-Layer Encryption based on Paillier and ElGamal Cryptosystem for Privacy Violation," *International Journal of Wireless and Microwave Technologies*, vol. 11, no. 3, pp. 9–15, Jun. 2021, doi: 10.5815/ijwmt.2021.03.02.

[9] O. A. Imran, S. F. Yousif, I. S. Hameed, W. N. Al-Din Abed, and A. T. Hammid, "Implementation of El-Gamal algorithm for speech signals encryption and decryption," in *Procedia Computer Science*, 2020, vol. 167, pp. 1028–1037. doi: 10.1016/j.procs.2020.03.402.

[10] W. Susilo, J. Tonien, and G. Yang, "Divide and capture: An improved cryptanalysis of the encryption standard algorithm RSA," *Comput Stand Interfaces*, vol. 74, Feb. 2021, doi: 10.1016/j.csi.2020.103470.

[11] S. F. Yousif, A. J. Abboud, and H. Y. Radhi, "Robust Image Encryption with Scanning Technology, the El-Gamal Algorithm and Chaos Theory," *IEEE Access*, vol. 8, pp. 155184–155209, 2020, doi: 10.1109/ACCESS.2020.3019216.