Implementasi dan Analisis Perbandingan *Schmidt-Samoa Cryptosystem* dengan RSA, ElGamal, dan *Rabin Cryptosystem*

Elvina 13517079[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13517079@std.stei.itb.ac.id

***Abstract*—Telah banyak algoritma enkripsi dan dekripsi dalam kriptografi, baik kriptografi kunci publik maupun kuncil privat. Salah satu cara pembangkitan kunci dalam algoritma kriptografi kunci publik adalah dengan memanfaatkan pemfaktoran bilangan. *Schmidt-Samoa Cryptosystem* merupakan algoritma kriptografi kunci publik yang memanfaatkan pemfaktoran bilangan. Algoritma ini dinyatakan sebagai algoritma yang lebih baik daripada RSA jika dilihat dari sisi matematis.**

Keywords*— *Schmidt-Samoa Cryptosystem*, kriptografi, kriptografi kunci publik, RSA, ElGamal, *Rabin Cryptosystem

I. PENDAHULUAN

Algoritma kriptografi pada umumnya terbagi menjadi dua macam, yaitu kriptografi kunci privat dan kriptografi kunci publik. Berbeda dengan algoritma kriptografi kunci privat yang seluruh kuncinya bersifat rahasia, algoritma kriptografi kunci publik tidak bergantung pada kerahasiaan kunci dalam menjaga kerahasiaan pesan. Algoritma kriptografi kunci publik memanfaatkan persoalan-persoalan integer klasik sebagai cara pembangkitan kunci, enkripsi, dan dekripsi. Persoalan-persoalan integer klasik yang paling umum digunakan adalah persoalan pemfaktoran, logaritma diskrit, dan *Elliptic Curve Discrete Logarithm Problem*.

Banyak algoritma umum yang memanfaatkan pemfaktoran sebagai metode pembangkitan kunci yang digunakan, seperti RSA, *Rabin Cryptosystem*, dan *Schmidt-Samoa Cryptosystem*. Pemfaktoran bilangan integer mengandalkan besarnya bilangan yang digunakan sebagai dasar kompleksitas dan kesulitan bagi kriptanalisis untuk memecahkan nilai kunci dan isi pesan. Selain itu pemfaktoran, persoalan yang akan dibahas juga adalah persoalan logaritma diskrit. Contoh algoritma yang memanfaatkan persoalan ini adalah algoritma ElGamal.

RSA, *Rabin Cryptosystem*, dan *Schmidt-Samoa Cryptosystem* secara berturut-turut ditemukan pada waktu yang berbeda-beda. RSA merupakan salah satu algoritma kunci publik yang paling tua dan paling terkenal, sehingga banyak yang mengaplikasikannya dan mengembangkannya. Salah satu usulan algoritma kunci publik lain yang memanfaatkan pemfaktoran dan dibuat setelah RSA (tahun 1976) adalah *Rabin Cryptosystem* (tahun 1979). Algoritma RSA dinyatakan aman karena sampai saat ini, belum diketahui cara penyelesaian masalah pemfaktoran yang efisien. Sama halnya dengan *Rabin Cryptosystem*. Algoritma *Rabin Cryptosystem* juga dinyatakan sebagai algoritma yang aman selama belum ada algoritma yang efisien untuk menyelesaikan persoalan pemfaktoran bilangan integer. Namun, dari sisi keamanan dan efisiensi, *Rabin Cryptosystem* terbukti lebih baik karena *Rabin Cryptosystem* menggunakan operasi yang lebih efisien yang akan dijelaskan di bab selanjutnya.

Kemudian, pada tahun 2005, diajukan algoritma kriptografi kunci publik yang merupakan hasil pengembangan dari algoritma *Rabin Cryptosystem* yang dinamakan *Schmidt-Samoa Cryptosystem*. Algoritma ini menggunakan operasi yang dapat menghilangkan kekurangan-kekurangan yang dimiliki oleh *Rabin Cryptosystem*. Pada makalah ini, akan dilakukan implementasi dan alanisis mengenai keamanan dari algoritma *Schmidt-Samoa Cryptosystem*, apakah benar algoritma ini lebih baik dan efisien daripada RSA dan *Rabin Cryptosystem*, serta bagaimana perbandingannya dengan algoritma persoalan logaritma diskrit seperti ElGamal.

II. DASAR TEORI

A. Algoritma RSA dan ElGamal

RSA merupakan dalah satu algoritma kriptografi kunci publik yang memanfaatkan persoalan pemfaktoran bilangan integer. Pemfaktoran bilangan merupakan persoalan sederhana yang masih sulit untuk diselesaikan, karena semakin besar suatu bilangan, semakin banyak kemungkinan faktor dari bilangan tersebut, sehingga semakin sulit juga dilakukan penyelesaian persoalan.

Terdapat beberapa properti yang digunakan dalam algoritma RSA, yaitu :

- p dan q bilangan prima (rahasia)
- $n = p * q$ (tidak rahasia)
- $\varphi(n) = (p - 1)(q - 1)$ (rahasia)
- $e,\ PBB(e, \varphi(n)) = 1$ (tidak rahasia)
- $d = e^{-1} mod(\varphi(n))$ (rahasia)
- $plaintext\ (m)$ (rahasia)
- $ciphertext\ (c)$ (tidak rahasia)

Properti-properti tersebut digunakan untuk melakukan pembangkitan kunci, enkripsi, dan dekripsi dengan algoritma RSA. Kunci yang dihasilkan dari properti-properti tersebut dapat dikelompokan menjadi dua, yaitu :

- Kunci publik = (e, n)
- Kunci privat = (d, n)

Enkripsi dengan menggunakan RSA dapat dilakukan dengan formula berikut :

$$ciphertext = m^e mod(n)$$

Dekripsi dengan menggunakan RSA dapat dilakukan dengan formula berikut :

$$plaintext = c^d mod(n)$$

Berbeda dengan RSA yang memanfaatkan persoalan pemfaktoran, ElGamal merupakan algoritma yang memanfaatkan persoalan logaritma diskrit. Untuk melakukan pembangkitan kunci, enkripsi, dan dekripsi dengan menggunakan ElGamal, digunakan beberapa properti sebagai berikut :

- Bilangan prima p (tidak rahasia)
- $g,\ g < p$ (tidak rahasia)
- $x,\ x < p$ (rahasia, kunci privat)
- $y = g^x mod(p)$ (tidak rahasia, kunci publik)
- $plaintext\ (m)$ (rahasia)
- $ciphertext\ (a, b)$ (tidak rahasia)

Dari properti-properti tersebut, dapat dihasilkan dua kelompok kunci, yaitu :

- Kunci publik = (y, g, p)
- Kunci privat = (x, p)

Enkripsi dengan menggunakan ElGamal membagi *ciphertext* menjadi dua bagian, sehingga panjang *ciphertext*-nya, yang setiap bagiannya dapat dihitung dengan formula berikut :

$$a = g^x mod(p)$$
$$b = y^k m\ mod(p)$$

Dekripsi dengan menggunakan ElGamal dapat dilakukan dengan formula berikut :

$$plaintext = b(a^x)^{-1} mod(p)$$

B. Rabin Cryptosystem

Sama seperti RSA, *Rabin Cryptosystem* menggunakan persoalan pemfaktoran dalam proses pembangkitan kunci, enkripsi, dan dekripsi. *Rabin Cryptosystem* memiliki beberapa properti seperti RSA dan ElGamal, yaitu :

- Bilangan prima $p, p \equiv 3\ mod(4)$ (rahasia)
- Bilangan prima $q, q \equiv 3\ mod(4)$ (rahasia)
- $n = p * q$ (tidak rahasia)
- $plaintext\ (m)$ (rahasia)
- $ciphertext\ (c)$ (tidak rahasia)

Dari properti-properti tersebut, didapat dua kelompok kunci, yaitu :

- Kunci publik = n
- Kunci privat = p, q

Enkripsi dengan menggunakan ElGamal dapat dilakukan dengan formula berikut :

$$ciphertext = m^2 mod(n)$$

Dekripsi dengan menggunakan ElGamal dapat dilakukan dengan formula berikut :

- Menghitung $m_p = c^{\frac{1}{4}(p+1)} mod(p)$ dan $m_q = c^{\frac{1}{4}(q+1)} mod(q)$
- Menggunakan *extended euclidean* untuk mencari nilai y_p dan y_q sehingga $y_p * p + y_q * q = 1$
- Menghitung beberapa nilai kemungkinan plaintext :

$$r_1 = (y_p * p * m_q + y_q * q * m_p)\ mod(n)$$
$$r_2 = n - r_1$$
$$r_3 = (y_p * p * m_q - y_q * q * m_p)\ mod(n)$$
$$r_4 = n - r_3$$

Dari keempat nilai tersebut, salah satu di antaranya adalah *plaintext* yang sesungguhnya. *Plaintext* yang sesungguhnya tidak dapat ditentukan menggunakan algoritma atau formula lain, perlu diketahui petunjuk lebih lanjut seperti konteks pesan atau pentunjuk langsung dari pengirim pesan.

C. Schmidt-Samoa Cryptosystem

Algoritma ini merupakan usulan algoritma baru yang terisnpirasi dari algoritma RSA dan *Rabin Cryptosystem*. Sama seperti RSA dan *Rabin Cryptosystem*, *Schmidt-Samoa Cryptosystem* memanfaatkan persoalan pemfaktoran untuk proses pembangkitan kunci, enkripsi, dan dekripsinya. Namun, berbeda dengan RSA dan *Rabin Cryptosystem* yang menggunakan pemfaktoran bilangan $n = p * q$, *Schmidt-Samoa Cryptosystem* memanfaatkan pemfaktoran bilangan $n = p^2 q$. Untuk lebih detailnya, berikut ini merupakan properti-properti yang digunakan pada algoritma Schmist-Samoa :

- Bilangan prima p dan q (rahasia)
- $n = p^2 q$ (tidak rahasia)
- $d = n^{-1} mod\ lcm(p - 1, q - 1)$ (rahasia)
- $plaintext\ (m)$ (rahasia)
- $ciphertext\ (c)$ (tidak rahasia)

Dari properti-properti tersebut, dapat dihasilkan dua kelompok kunci, yaitu :

- Kunci publik = n
- Kunci privat = d

Enkripsi dengan menggunakan ElGamal dapat dilakukan dengan formula berikut :

$$ciphertext = m^n mod(n)$$

Dekripsi dengan menggunakan ElGamal dapat dilakukan dengan formula berikut :

$$plaintext = c^d mod(p * q)$$

III. IMPLEMENTASI

Pada bab sebelumnya, telah dipaparkan mengenai formula-formula yang digunakan dalam proses pembangkitan kunci, enkripsi, dan dekripsi dengan menggunakan beberapa algoritma kriptografi kunci publik yang menjadi pokok bahan bahasan pada makalah ini. Selanjutnya, akan dilakukan implementasi secara langsung dengan menggunakan bahasa *python* untuk proses pembangkitan kunci, enkripsi, dan dekripsi.

A. Algoritma RSA dan ElGamal

Implementasi dari proses pembangkitan kunci pada algoritma RSA dapat dilihat pada Tabel 1.

Tabel 1. Implementasi pembangkitan kunci RSA

```
def generate_keys(p, q, e):
    n = p * q
    phi = (p-1) * (q-1)
    k = 1
    d = (1 + k*phi) / e
    while not float.is_integer(d):
        k += 1
        d = (1 + k*phi) / e
    d = round(d)
    return [e, n], [d, n]
```

Implementasi dari enkripsi dan dekripsi dengan algoritma RSA dapat dilihat pada Tabel 2.

Tabel 2. Implementasi enkripsi dan dekripsi algoritma RSA

```
def encrypt(plaintext, n, e):
    ciphertext = []
    for block in plaintext:
        ciphertext.append(pow(block, e, n))
    return ciphertext

def decrypt(ciphertext, n, d):
    plaintext = []
    for block in ciphertext:
        plaintext.append(pow(block, d, n))
    return plaintext
```

Implementasi dari proses pembangkitan kunci pada algoritma ElGamal dapat dilihat pada Tabel 3.

Tabel 3. Implementasi pembangkitan kunci ElGamal

```
def generate_keys(g, x, p):
    if (rsa.is_prime(p)) and (g < p) and (1
    <= x) and (x <= p-2):
      return [(pow(g, x, p)), g, p], [x, p]
```

Implementasi dari enkripsi dan dekripsi dengan algoritma ElGamal dapat dilihat pada Tabel 4.

Tabel 4. Implementasi enkripsi dan dekripsi algoritma ElGamal

```
def encrypt(plaintext, y, g, p, k):
    block_ciphertext = []
    for blok in plaintext:
        a = pow(g, k, p)
        b = pow(y, k) * blok % p
        block_ciphertext.append([a, b])
    ciphertext = block_ciphertext
    return ciphertext

def decrypt(ciphertext, x, p):
    plaintext = []
    for blok in ciphertext:
        a = blok[0]
        b = blok[1]
        a2 = pow(a, (p-1-x), p)
        m = b * a2 % p
        plaintext.append(m)
    return plaintext
```

B. Rabin Cryptosystem

Implementasi dari proses pembangkitan kunci pada *Rabin Cryptosystem* dapat dilihat pada Tabel 5.

Tabel 5. Implementasi pembangkitan kunci *Rabin Cryptosystem*

```
def generate_keys(p, q):
    n = p * q
    return n, [p, q]
```

Implementasi dari enkripsi dan dekripsi dengan algoritma *Rabin Cryptosystem* dapat dilihat pada Tabel 6.

Tabel 6. Implementasi enkripsi dan dekripsi algoritma *Rabin Cryptosystem*

```
def encrypt(plaintext, n):
    ciphertext = []
    for block in plaintext:
        ciphertext.append(pow(block, 2, n))
    return ciphertext

def decrypt(ciphertext, p, q):
    r1, r2, r3, r4 = [], [], [], []

    for block in ciphertext:
        mp = pow(c, 0.25*(p+1), p)
        mq = pow(c, 0.25*(q+1), q)

        yp, yq = gcd_extended(p, q)

        r1.append((yp*p*mq + yq*q*mp) % n)
        r2.append(n-r1)
        r3.append((yp*p*mq + yq*q*mp) % n)
        r4.append(n-r3)

    plaintext = [r1, r2, r3, r4]
    return plaintext
```

C. Schmidt-Samoa Cryptosystem

Implementasi dari proses pembangkitan kunci pada *Schmidt-Samoa Cryptosystem* dapat dilihat pada Tabel 7.

Tabel 7. Implementasi pembangkitan kunci *Schmidt-Samoa Cryptosystem*

```
def lcm(a, b):
    return abs(a*b) // math.gcd(a, b)

def generate_keys(p, q):
    n = pow(p, 2) * q
    d = pow(n, -1, lcm(p-1, q-1))
  return n, d
```

Implementasi dari enkripsi dan dekripsi dengan algoritma *Schmidt-Samoa Cryptosystem* dapat dilihat pada Tabel 8.

Tabel 8. Implementasi enkripsi dan dekripsi algoritma *Schmidt-Samoa Cryptosystem*

```
  def encrypt(plaintext, n):
      ciphertext = []
      for block in plaintext:
          ciphertext.append(pow(block, n, n))
      return ciphertext

  def decrypt(ciphertext, p, q, d):
      plaintext = []
      for block in ciphertext:
          plaintext.append(pow(block,      d,
p*q))
      return plaintext
```

IV. Analisis Perbandingan

Pada bab-bab sebelumnya, telah dijelaskan mengenai formula-formula yang digunakan dan implementasi langsung pada algoritma RSA, ElGamal, *Rabin Cryptosystem*, dan *Schmidt-Samoa Cryptosystem*. Keempat algoritma tersebut memiliki formula yang berbeda dalam mengoperasikan pembangkitan kunci, enkripsi, dan dekripsi.

A. Overview

Telah diketahui bahwa algoritma RSA termasuk dalam algoritma yang lambat dalam melakukan proses enkripsi dan dekripsi. Pada tahap implementasi, banyak operasi yang dilakukan yang dapat memakan banyak waktu, terutama pada proses perpangkatan modulo yang juga terdapat pada algoritma ElGamal, *Rabin Cryptosystem*, dan *Schmidt-Samoa Cryptosystem*. Operasi tersebut dapat menyebabkan waktu yang dibutuhkan dalam proses pembangkitan kunci, enkripsi, dan dekripsi di algoritma-algoritma yang dibahas menjadi semakin meningkat secara eksponensial berdasarkan besarnya nilai kunci yang digunakan. Hal tersebut membuktikan pernyataan yang menyatakan bahwa algoritma kunci publik memiliki waktu komputasi yang lebih lambat jika dibandingkan dengan algoritma kunci privat.

B. Pembangkitan Kunci

Berdasarkan apa yang telah dituliskan pada bab II bagian A, RSA memiliki beberapa properti seperti $p, q, n, e, \varphi(n)$, dan d. RSA merupakan algoritma yang menggunakan properti paling banyak di antara algoritma-algoritma lain yang dibahas pada makalah ini. Hal ini dapat berakibat pada aplikasi penggunaan algoritma RSA. Untuk melakukan perhitungan dan pertukaran kunci, terdapat banyak properti yang diperlukan, sehingga proses perhitungan dan pertukaran menjadi kurang efisien. Hal ini juga dapat terlihat pada proses pembangkitan kunci pada algoritma RSA. Terdapat banyak parameter dan operasi yang diperlukan untuk membangkitkan pasangan kunci publik dan kunci privat.

Pada algoritma ElGamal, proses pembangkitan kunci menggunakan banyak operasi kondisional. Selain itu, digunakan juga operasi perpangkatan modulo yang relatif memakan waktu lebih lama daripada operasi perkalian biasa seperti `n = p * q`. Walaupun memiliki jumlah properti yang lebih sedikit daripada RSA, waktu komputasi yang dibutuhkan algoritma ElGamal tidak lebih baik daripada algoritma RSA.

Dari dua pernyataan di atas mengenai proses pembangkitan dan pertukaran kunci pada algoritma RSA dan ElGamal, dapat disimpulkan bahwa algoritma *Rabin Cryptosystem* lebih baik dalam melakukan proses pembangkitan kunci. *Rabin Cryptosystem* hanya menggunakan operasi perkalian untuk mendapatkan pasangan kunci publik dan privat, serta hanya memiliki dua properti (p dan q) yang dibutuhkan untuk mendapatkan nilai n. Dengan jumlah properti yang sedikit, *Rabin Cryptosystem* dapat melakukan proses enkripsi dan dekripsi dengan baik.

Sementara itu, untuk proses pembangkitan kunci pada *Schmidt-Samoa Cryptosystem*, diperlukan waktu yang lebih lama karena terdapat operasi perpangkatan dan pencarian nilai kelipatan persekutuan terkecil (*least common multiplication* / LCM). Dari pernyataan tersebut, dapat disimpulkan bahwa algoritma *Schmidt-Samoa Cryptosystem* memiliki waktu komputasi yang lebih buruk daripada *Rabin Cryptosystem*, walaupun memiliki properti pertukaran kunci yang lebih sedikit, karena baik kunci publik maupun kunci privat, *Schmidt-Samoa Cryptosystem* hanya mempertukarkan masing-masing satu nilai kunci.

C. Enkripsi

Pada proses enkripsi, berdasarkan apa yang telah dilakukan pada tahap implementasi, dapat dilihat bahwa struktur kode yang ditulis pada RSA, *Rabin Cryptosystem*, dan *Schmidt-Samoa Cryptosystem* cukup mirip satu sama lain. Ketiga algoritma yang memanfaatkan persoalan pemfakoran untuk enkripsi tersebut akan menghasilkan nilai *ciphertext* yang kurang lebih memiliki panjang yang sama dengan *plaintext*-nya. Namun, pada algoritma ElGamal, proses enkripsi akan menghasilkan nilai *ciphertext* yang memiliki panjang dua kali lipat dari panjang *plaintext*, karena *ciphertext* yang dihasilkan merupakan gabungan dari dua properti yang terdapat pada algoritma ElGamal. Karena memiliki dua properti yang perlu dihitung, algoritma ElGamal juga memiliki waktu komputasi yang lebih lama daripada algoritma RSA dan *Rabin Cryptosystem*.

Berdasarkan hasil uji coba melakukan *running* hasil implementasi, perpangkatan merupakan operasi yang cukup lama pada proses komputasi. RSA dan *Rabin Cryptosystem* sama-sama menggunakan nilai yang berbeda untuk perpangkatan dan perhitunagn modulo. Nilai n yang terdapat pada ketiga algoritma tersebut merupakan nilai yang cukup besar, karena didapat dari hasil perkalian dua bilangan prima yang disarakan memiliki nilai yang besar untuk meningkatkan keamanan. Pada RSA, digunakan nilai e dan n untuk menghitung nilai *ciphertext* $m^e mod(n)$, sementara untuk *Rabin Cryptosystem* digunakan nilai 2 dan n untuk menghitung nilai *ciphertext* $m^2 mod(n)$. Nilai e merupakan nilai yang lebih kecil daripada nilai n, sehingga dapat menghemat waktu komputasi yang dibutuhkan. *Rabin Cryptosystem* menyederhanakannya dengan langsung menggunakan satu nilai mutlak, yaitu 2. Dengan demikian, waktu komputasi pada *Rabin Cryptosystem* menjadi lebih sedikit jika dibandingkan dengan RSA.

Namun, *Schmidt-Samoa Cryptosystem* menggunakan nilai n yang sama dalam melakukan proses permangkatan dan perhitngan modulo dalam menghitung nilai *ciphertext* $m^n mod(n)$. Selain itu, nilai n yang digunakan pada *Schmidt-Samoa Cryptosystem* dapat dikatakan jauh lebih besar dibandingkan nilai n yang digunakan pada RSA dan *Rabin Cryptosystem*, karena RSA dan *Rabin Cryptosystem* sama-sama menggunakan operasi perkalian $n = p * q$, sementara *Schmidt-Samoa Cryptosystem* memangkatkan terlebih dahulu nilai p dengan 2 sehingga perhitungan nilai n pada algoritma tersebut menjadi $n = p^2 q$.

Usulan perhitungan nilai n yang demikian dapat meningkatkan keamanan dengan meningkatkan jumlah kemungkinan hasil pemfaktoran yang mungkin. Semakin besar nilai-nilai yang digunakan, semakin sulit juga bagi kriptanalis untuk mencari nilai kunci dan *plaintext* yang sesuai. Namun, nilai n yang sangat besar dapat meningkatkan waktu komputasi secara eksponensial. Oleh sebab itu, karena nilai n yang besar digunakan dalam operasi perpangkatan pada proses enkripsi, waktu komputasi enkripsi pada *Schmidt-Samoa Cryptosystem* juga lebih lambat dibandingkan dengan RSA dan *Rabin Cryptosystem*.

D. Dekripsi

Proses dekripsi yang dilakukan pada *Rabin Cryptosystem* tergolong unik jika dibandingkan dengan proses dekripsi pada RSA dan ElGamal. Pada RSA dan ElGamal, *plaintext* didapatkan hanya dari perhitungan perpangkatan modulo pada setiap blok *ciphertext*, yang kemudian akan digabungkan untuk menjadi *plaintext* yang utuh. Namun, pada *Rabin Cryptosystem*, didapatkan empat kemungkinan *plaintext* dari haisl dekripsi.

Hasil yang didapatkan dari proses dekripsi *Rabin Cryptosystem* memiliki sisi positif dan sisi negatif. Keuntungan dari diperolehnya empat kemungkinan nilai *plaintext* adalah peningkatan pada keamanan, walau tidak signifikan karena kemungkinan besar, tiga dari empat nilai yang dihasilkan tidak akan memiliki makna ketika diubah ke dalam bentuk alfabet. Oleh karena itu, kekurangan dari hasil dekripsi pada *Rabin Cryptosystem* jauh menutupi keuntungannya. Dengan didapatnya empat nilai yang berbeda, akan semakin banyak juga memori yang perlu dialokasikan untuk menampung hasil dekripsi. Selain itu, komputasi yang perlu dilakukan juga menjadi lebih banyak, karena perlu dilakukan perhitungan empat nilai. Apabila secara kebetulan nilai yang didapat sedikit mirip atau memiliki konteks yang berkaitan dengan isi pesan sesungguhnya, dapat ditimbulkan ambiguitas terhadap isi pesan yang sesunguhnya. Bisa jadi, penerima pesan gagal menangkap isi pesan yang sesungguhnya karena salah memilih satu di antara empat kemungkinan *plaintext*.

Schmidt-Samoa Cryptosystem memiliki cara dekripsi yang misip dengan RSA, yaitu cukup dengan melakukan perhitungan modulo pada setiap blok pesan. Selain itu, karena jumlah properti yang digunakan lebih sedikit daripada RSA, proses pertukaran kunci yang dilakukan untuk melakukan dekripsi juga jadi lebih mudah dan efisien. Berbeda dengan proses enkripsinya, proses dekripsi pada *Schmidt-Samoa Cryptosystem* menggunakan nilai yang besarnya kurang lebih mirip dengan nilai yang digunakan pada RSA. Oleh karena itu, dari sisi waktu komputasi, *Schmidt-Samoa Cryptosystem* kurang lebih sama cepatnya dengan dekripsi pada RSA.

V. KESIMPULAN

Setelah melakukan implementasi terhadap beberapa algoritma kriptografi kunci publik RSA, ElGamal, *Rabin Crptosystem*, dan *Schmidt-Samoa Cryptosystem*, proses pembuatan implementasi dapat dilakukan dengan cukup mudah. *Schmidt-Samoa Cryptosystem* dapat diimplementasikan dengan mudah seperti RSA, sehingga proses implementasinya juga tidak memakan waktu lama.

Dari sisi keamanan, *Schhmidt-Samoa Cryptosystem* tidak memiliki keunggulan yang signifikan jika dibandingkan dengan *Rabin Cryptosystem*. Kedua algoritma tersebut sama-sama memiliki kerahasiaan yang sangat tergantung erat dengan pemfaktoran nilai n yang didapat dari perkalian bilangan-bilangan prima.

Jika dibandingkan dengan algoritma RSA dan *Rabin Cryptosystem*, proses enkripsi pada algoritma *Schmidt-Samoa* memiliki waktu komputasi yang lebih lama karena menggunakan nilai n yang jauh lebih besar, serta perhitungan perpangkatan bilangan dengan menggunakan n. Demikian juga dengan proses pembangkitan kuncinya, *Schmidt-Samoa Cryptosystem* memiliki waktu komputasi yang lebih lama jika dibandingkan dengan *Rabin Cryptosystem.* Pada proses dekripsi, struktur kode dari implementasi *Schmidt-Samoa* sama dengan RSA, sehingga dapat dikatakan bahwa *Schmidt-Samoa* memiliki kecepatan yang sama dengan RSA pada dekripsi, dan lebih cepat dan mengurangi ketidak pastian hasil dekripsi jika dibandingkan dengan Rabin *Cryptosystem*.

VI. ACKNOWLEDGMENT

Pertama-tama, saya mengucapkan syukur yang sebesar-besarnya kepada Tuhan Yang Maha Esa yang telah memberi segala berkat dan kesempatan bagi saya sehingga dapat menyelesaikan makalah ini dengan sebaik-baiknya. Berikutnya, saya juga mengucapkan terima kasih kepada dosen pengajar mata kuliah Kriptografi saya, Bapak Rinaldi, yang telah memberikan bekal ilmu yang cukup serta memberikan kesempatan bagi para muridnya, termasuk saya, untuk mengaplikasikan ilmu yang telah diajarkan ke dalam tugas-tugas yang telah diberikan.

Saya juga mengucapkan terima kasih kepada pihak-pihak lain yang telah senantiasa mendukung saya dalam proses pengerjaan makalah seperti keluarga dan teman-teman yang selalu siap memberikan dukungan mental dan para penulis jurnal yang saya baca yang telah bersedia untuk membagikan ilmunya untuk diakses secara bebas, sehingga saya dapat menyelesaikan makalah ini. Semoga dengan segala dukungan dan bantuan yang telah saya peroleh, makalah ini dapat diterima dengan baik dan berguna bagi pembacanya.

REFERENCES

[1] K. *Schmidt-Samoa Cryptosystem, A New Rabin Cryptosystem-type Trapdoor Permutation Equivalent to Factoring and Its Applications*. Darmstadt, Germany : Technische Universit at Darmstadt.

[2] M.C. *Rabin Cryptosystem, MIT-LCS-TR-212*: *Digitalized Signatures and Public-Key Functions as Interactable as Factorization*.

[3] R. Munir, " Kriptografi Kunci-Publik Bahan Kuliah IF4020 Kriptografi," Bandung, 2020.

[4] R. Munir, "Algoritma ElGamal Bahan Kuliah IF4020 Kriptografi," Bandung, 2020.

[5] R. Munir, "Algoritma RSA Bahan Kuliah IF4020 Kriptografi," Bandung, 2020.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 21 Desember 2020

Elvina 13517079