Penerapan Kriptografi Kunci Publik untuk Pengiriman Data User saat Melakukan Top Up

Muhammad Ziad Rahmatullah - 13518032
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13518032@std.stei.itb.ac.id

***Abstract*—Algortima RSA, Elgamal, dan Paillier merupakan algoritma kunci publik yang terkenal sebagai algoritma yang digunakan untuk mengenkripsi pesan yang cukup kuat karena memiliki properti yang membuat kriptanalisis tidak tahu cara untuk melakukan dekripsi cipherteks secara efisien. Algoritma ini dapat dimanfaatkan dalam hal kemanan dalam bertransaksi. Salah satu transaksi yang sering terjadi pada para *gamer* adalah top up mata uang pada sebuah game. Untuk melakukan top up ada tahap perpindahan data user yang mana rentan terhadap *cyberattack*.**

Keywords—Algoritma Kunci Publik; RSA; enkripsi; dekripsi, keamanan akun

I. PENDAHULUAN

Genshin Impact merupakan sebuah game action RPG yang dirilis pada tahun 2020 oleh miHoYo. Pada game ini terdapat 3 mata uang yang digunakan yaitu mora, primogems, dan genesis crystal. Mora merupakan mata uang yang bisa didapatkan dengan melakukan penjelajahan atau *farming* di dalam game. Primogems bisa didapatkan dengan penjelajahan, quest, dan event. Sedangkan genesis crystal bisa didapat hanya dengan melakukan top up. Ada berbagai macam jenis top up, yaitu top up genesis crystal, battle pass dan blessing. Dengan melakukan top up battle pass, pemain dapat mengambil hadiah jika mereka sudah melaksanakan beberapa quest yang membuat level dari battle pass meningkat, hadiah tersebut bisa berupa equipment atau material untuk meningkatkan karakter. Blessing merupakan jenis top up yang membuat pemain mendapatkan 90 primogems setiap hari nya dengan melakukan login pada hari tersebut. Jangka waktu dari blessing ini dihitung perbulan.

Pada saat ini, untuk melakukan top up genesis crystal dan battle pass bisa dilakukan dengan hanya memberikan informasi UID pemain. Banyak penyedia jasa top up seperti codashop yang sudah menyediakan hal tersebut. Namun untuk melakukan top up blessing, pemain diharuskan melakukan top up dalam game dengan menggunakan kartu kredit. Seiring berjalannya waktu, bermunculan penyedia jasa top up blessing dengan harga yang lebih murah, mereka memanfaatkan perbedaan nilai mata uang rupiah dengan dollar, sehingga mereka bisa mendapatkan harga murah dengan berpindah server. Banyak dari pemain tertarik untuk menggunakan jasa tersebut. Namun dengan adanya aturan untuk melakukan top up blessing di dalam game, mempersulit terjadinya transaksi. Pemain mungkin masih belum percaya terhadap penyedia jasa, sehingga pemain sungkan untuk memberikan username dan passwordnya kepada mereka. Pengiriman username dan password ini pun dilakukan para penyedia jasa pada chat media sosial seperti whatsapp, line, instagram, dan lain lain. Jalur pengiriman ini kurang aman mengingat banyaknya kasus kejahatan yang terjadi seperti penyadapan. Penyadapan bisa dilakukan saat pemain memberikan username dan password mereka atau saat memberikan *secondary verifiaction*.

Untuk mengatasi hal ini, dapat dilakukan algoritma kunci-publik sehingga data user yang diberikan tidak terekspos oleh para penyadap. Para penyadap mungkin hanya melihat pesan yang sudah terenkripsi saat melakukan penyerangan.

II. DASAR TEORI

A. *Kriptografi*

Kriptografi merupakan kakas yang sangat penting di dalam keamanan informasi. Kata *cryptography* berasal dari bahasa Yunani, *cryptos (secret)* dan *graphein (writing)*. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentifikasi. Pada zaman dahulu, sebelum ditemukannya komputer, proses enkripsi dan dekripsi masih menggunakan pensil dan kertas, sehingga proses enkripsi dan dekripsi yang digunakan masih sederhana, misalkan menggunakan kunci enkripsi dan kunci dekripsi yang sama dan membutuhkan operasi yang tidak banyak pula, seperti hanya digunakan proses rotasi, permutasi, dan/atau substitusi. Sedangkan pada jaman setelah komputer ditemukan, prinsip menjaga keamanan pesan dapat dilakukan dengan menggunakan prinsip dan konsep matematika yang rumit karena perhitungan tersebut dapat dilakukan oleh komputer.

B. *Algoritma RSA*

Algoritma RSA merupakan salah satu algoritma kriptografi kunci asimetri. Kunci asimetri berarti kunci yang digunakan pengirim dan penerima akan berbeda. Terdapat kunci publik dan juga kunci privat. Kunci publik digunakan untuk melakukan proses enkripsi, sedangkan kunci privat digunakan untuk melakukan proses dekripsi. Keamanan dari algoritma RSA terletak pada sulitnya memfaktorkan bilangan bulat yang besar menjadi faktor-faktor prima. Properti pada algoritma RSA adalah sebagai berikut :

1. p dan q bilangan prima (rahasia)
2. $n = p \cdot q$ (tidak rahasia)
3. $\Phi(n) = (p - 1)(q - 1)$ (rahasia)
4. e kunci enkripsi, Syarat: PBB(e, Φ (n)) = 1, PBB = Pembagi bersama terbesar = gcd (tidak rahasia)
5. d kunci dekripsi, d dihitung dari $d \equiv e^{-1} \bmod (\Phi(n))$ (rahasia)
6. m plainteks (rahasia)
7. c chiperteks (tidak rahasia)

Terdapat penurunan rumus RSA yang berawal dari Teorema Euler yaitu $a^{\Phi(n)} \equiv 1 \pmod{n}$ dengan syarat sebagai berikut :

1. a harus relatif prima terhadap n
2. $\Phi(n)$ = Toitent Euler = fungsi yang menentukan berapa banyak dari bilangan-bilangan 1, 2, 3, ..., n yang relatif prima terhadap n, Contoh: $\Phi(20) = 8$, sebab 8 buah yang relatif prima dengan 20, yaitu 1, 3, 7, 9, 11, 13, 17, 19
3. Jika $n = pq$ adalah bilangan komposit dengan p dan q prima, maka $\Phi(n) = \Phi(p)\,\Phi(q) = (p - 1)(q - 1)$

Dengan penurunan rumus sebagai berikut :

$a^{\Phi(n)} \equiv 1 \pmod{n}$, pangkatkan kedua ruas dengan k

$a^{k\Phi(n)} \equiv 1^k \pmod{n}$

$a^{k\Phi(n)} \equiv 1 \pmod{n}$, ganti a dengan m

$m^{k\Phi(n)} \equiv 1 \pmod{n}$, kalikan kedua ruas dengan m

$m^{k\Phi(n)+1} \equiv m \pmod{n}$

misalkan e dan d dipilih sedemikian sehingga

$e \cdot d \equiv 1 \pmod{\Phi(n)}$ atau $e \cdot d = k\Phi(n) + 1$

maka, $m^{k\Phi(n)+1} \equiv m \pmod{n}$

$m^{e \cdot d} \equiv m \pmod{n}$

$(m^e)^d \equiv m \pmod{n}$, sehingga :

Enkripsi : $E_e(m) = c = m^e \bmod n$

Dekripsi : $D_d(c) = m = c^d \bmod n$

Prosedur pembangkitan kunci adalah sebagai berikut :

1. Pilih dua bilangan prima, p dan q
2. Hitung $n = pq$
3. Hitung $\Phi(n) = (p - 1)(q - 1)$
4. Pilih sebuah bilangan bulat e sebagai kunci publik, e harus relatif prima terhadap $\Phi(n)$
5. Hitung kunci dekripsi, d, dengan persamaan,

   $ed \equiv 1 \pmod{\Phi(n)}$ atau $d \equiv e^{-1} \bmod(\Phi(n))$

Hasil dari algoritma diatas adalah :

- Kunci publik dengan pasangan (e, n)
- Kunci privat dengan pasangan (d, n)

Proses Enkripsi adalah sebagai berikut :

1. Nyatakan pesan menjadi blok-blok plainteks: m_1, m_2, m_3, ... (syarat: $0 \le m_i < n - 1$)
2. Hitung blok chiperteks c_i untuk blok plainteks m_i menggunakan kunci publik e dengan persamaan,

   $c_i = m_i^e \bmod n$

Proses Dekripsi adalah sebagai berikut :

1. Misalkan blok-blok cipherteks adalah c_1, c_2, c_3, ...
2. Hitung kembali blok plainteks m_i dari blok cipher teks c_i menggunakan kunci privat d dengan persamaan,

   $m_i = c_i^d \bmod n$

Keamanan dari algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan bulat n faktor-faktor prima (p dan q), yang dalam hal ini $n = p \times q$. jika seorang kriptanalisis dapat memfaktorkan bilangan n b menjadi p dan q, maka $\Phi(n) = (p - 1)(q - 1)$ dapat dihitung. Selanjutnya, karena kunci enkripsi e diumukan (tidak rahasia), maka kunci dekripsi d bisa dihitung dari kekongruenan $ed \equiv 1 \pmod{\Phi(n)}$. Penemu algoritma RSA menyarankan nilai p dan q panjangnya lebih dari 100 digit. Dengan demikian hasil kali $n = p \times q$ akan berukuran lebih dari 200 digit. Usaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 miliar tahun, sendangkan untuk bilangan 500 digit membutuhkan waktu 10^{25} tahun (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik). Algoritma pemfaktoran saat ini memiliki kompleksitas, $O(\exp(((64/9)b(\log(b)^2))^{1/3}))$ untuk bilangan bulat n sepanjang b-bit. Hingga saat ini belum ditemukan algoritma pemfaktoran bilangan bulat besar dalam waktu polinomial. Fakta inilah yang membuat algoritma RSA dianggap masih aman untuk saat ini. Semakin panjang bilangan bulatnya, maka semakin lama waktu yang dibutuhkan untuk memfaktorkannya.

Secara umum dapat disimpulkan bahwa RSA hanya aman jika n cukup besar. Jika panjang n hanya 256 bit atau kurang, ia dapat difaktorkan dalam beberapa jam saja dengan sebuah komputer *PC* dan program yang tersedia secara bebas. Jika panjang n adalah 512 bit atau kurang, ia dapat difaktorkan dengan beberapa ratus komputer. Kelemahan dari algoritma RSA adalah lebih lambat daripada algoritma kriptografi kunci simetri seperti *DES* dan *AES*. Dalam prakteknya, *RSA* tidak digunakan untuk mengenkripsi pesan, tetapi mengenkripsi kunci simetri (kunci sesi) dengan kunci publik penerima pesan.

C. Algoritma Elgamal

Algoritma Elgamal dibuat oleh Taher Elgamar pada tahun 1985. Algoritma ini pertama kali dikemukakan di dalam makalah berjudul "*A public key cryptosystem and a signature scheme based on discrete logarithms*". Kemanan algoritma ini terletak pada sulitnya menghitung logaritma diskrit. Masalah yang ada pada logaritma diskrit adalah jika p adalah bilangan prima dan g dan y adalah sembarang bilangan bulat, maka akan

didapat x sedemikian sehingga $g^x \equiv y \pmod p$. Untuk properti pada algortima Elgamal adalah sebagai berikut :

1. p Bilangan prima (tidak rahasia)
2. g dengan ($g < p$), Bilangan acak (tidak rahasia)
3. x dengan ($x < p$), Bilangan acak (rahasia)
4. $y = g^x \bmod p$ (tidak rahasia)
5. m plainteks (rahasia)
6. a dan b, chiperteks (tidak rahasia)

Prosedur Pembangkitan kunci untuk algoritma elgamal adalah sebagai berikut :

1. Pilih sembarang bilangan prima p (p dapat di-*share* di antara anggota kelompok
2. Pilih 2 buah bilangan acak, g dan x, dengan syarat $g < p$ dan $1 \le x \le p - 2$
3. Hitung $y = g^x \bmod p$

Hasil algoritma diatas adalah :

- Kunci publik: tripel (y, g, p)
- Kunci privat: pasangan (x, p)

Prosedur Enkripsi adalah sebagai berikut :

1. Susun plaintesks menjadi blok $m_1, m_2, \ldots,$ (nilai setiap blok di dalam selang $[0, p - 1]$
2. Pilih bilangan acak k, yang dalam hal ini $1 \le k \le p - 2$
3. Setiap blok m dienkripsi dengan rumus

   $a = g^k \bmod p$

   $b = y^x m \bmod p$

   pasangan a dan b adalah chiperteks untuk blok pesan m. Jadi, ukuran chiperteks menjadi dua kali ukuran plainteksnya.

Prosedur Dekripsi adalah sebagai berikut :

1. Gunakan kunci privat x untuk menghitung

   $(a^x)^{-1} = a^{p-1-x} \bmod p$

2. Hitung plainteks m dengan persamaan

   $m = b/a^x \bmod p = b(a^x)^{-1} \bmod p$

D. Algoritma Paillier

Algoritma Paillier termasuk kedalam algoritma kriptografi kunci-publik. Algoritma ini dikembangkan oleh Pascal Paillier pada tahun 1999. Keamanan algoritma ini didasarkan pada sulitnya memecahkan persoalan residu ke-n (*composite residuosity problem*). Persoalan residu ke-n adalah sebagai berikut: Diberikan bilangan komposit n dan bilangan bulat z. Bilangan z dikatakan residu ke-n module n^2 jika terdapat sebuah nilai y sedemikian sehingga $z \equiv y^n \pmod{n^2}$. Properti dari algoritma paillier adalah sebagai berikut :

1. p dan q bilangan prima (rahasia)
2. $n = p \cdot q$ (tidak rahasia)
3. $\lambda = lcm(p - 1, q - 1)$ (rahasia)
4. $g,\ g < n^2$ (tidak rahasia)
5. $\mu = (\mathrm{L}(g^\lambda \bmod n^2))^{-1} \bmod n$ (rahasia)

Prosedur pembangkitan kunci untuk algoritma Paillier adalah sebagai berikut :

1. Pilih 2 buah bilangan prima p dan q yang memenuhi syarat $gcd(pq, (p - 1)(q - 1)) = 1$
2. Hitung $n = p \cdot q$ dan $\lambda = lcm(p - 1, q - 1)$
3. Pilih sembarang bilangan bulat g dengan $g < n^2$
4. Hitung $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$, dengan fungsi $L(x) = (x - 1)/n$

   Hasil algoritma diatas adalah :

- Kunci publik pasangan (g, n)
- Kunci privavt pasangan (λ, μ)

Prosedur enkripsi adalah sebagai berikut :

1. Misalkan m adalah pesan yang akan dienkripsi, dengan syarat $0 \le m < n$
2. Pilih bilangan bulat acak r dengan syarat $0 \le r < n$ dan $gcd(r, n) = 1$
3. Hitung chiperteks dari m dengan persamaan

   $c = g^m \cdot r^n \bmod n^2$. Dalam hal ini, c adalah residu ke-n dalam modulus n^2, dilambangkan dengan $[c]_g$

Prosedur dekripsi adalah sebagai berikut :

1. Misalkan c cipherteks yang akan didekripsi
2. Hitung plainteks dari c dengan persamaan

   $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

III. PENJELASAN PROGRAM

Pada program ini, dibuat implementasi dari algoritma RSA, Elgamal dan Paillier yang sudah dijelasakan sebelumnya. Beberapa fungsi untuk membantu dalam implementasi algoritma kunci publik adalah sebagai berikut dalam bahasa python :

```
def gcd( a, b );
def modinv(a, b, n);
def modexp( base, exp, modulus );
def lcm(a, b);
def encode(text);
def decode(text);
```

Fungsi gcd dapat mengeluarkan nilai gcd dari 2 buah bilangan, fungsi ini digunakan untuk memberi syarat pada nilai $gcd(a, b)$ yang ingin mengeluarkan nilai 1. Fungsi modinv merupakan fungsi untuk mencari modular inverse untuk

mencari nilai x pada $x \equiv a^{-b} \bmod n$. Fungsi modexp berfungsi untuk nilai dari modular x pada $x \equiv a^{b} \bmod n$. Fungsi lcm berguna untuk menemukan nilai *lcm* dari bilangan *a* dan b. Fungsi encode berguna untuk mengubah plaintext menjadi nilai ordinal. Fungsi decode memiliki fungsi kebalikannya dari decode.

A. Algoritma RSA

Kode program untuk generate key pada algoritma RSA adalah sebagai berikut :

```
def generateKeyR(p, q, e):
  n = p*q
  phi = (p-1) * (q-1)
  if (gcd(phi, e) != 1):
    return TypeError
  d = modinv(e,1, phi)[0]
    return {
      "public": (e, n),
      "private": (d, n)
    }
```

Fungsi generateKeyR akan meminta input nilai *p, q,* dan *e,* lalu akan menjalankan prosedur pembangkitan kunci algoritma RSA sehingga fungsi ini akan memiliki output berupa nilai publik key berupa (*e*, *n*) dan private key berupa (*d*, *n*). Kode program untuk enkripsi dan dekripsi algoritma RSA adalah sebagai berikut :

```
def encryptR(publicKey, plainText):
  e, n = publicKey
  encoded = encode(plainText)
  result = [modexp(char, e, n) for char in
            encoded]
  return result

def decryptR(privateKey, cipherText):
  d, n = privateKey
  result= [str(modexp(int(char),int(d),n))
           for char in cipherText]
  result = decode(result)
  return ''.join(result)
```

Fungsi encryptR adalah fungsi untuk tahap enkripsi pada algoritma RSA, fungsi ini akan mengeluarkan hasil enkripsi dari plaintext berupa angka. Fungsi decryptR adalah fungsi untuk tahap deksripsi pada algoritma RSA, fungsi ini akan mendekripsi ciphertext yang diinput menjadi plaintext.

B. Algoritma Elgamal

Kode program untuk generate key pada algoritma Elgamal adalah sebagai berikut :

```
def generateKeyG(p , g, x):
  y = modexp(g, x, p)
  return {
    'publicKey': (p,g,y),
    'privateKey': (p,x)
  }
```

Fungsi generateKeyG akan meminta input nilai *p, g,* dan *x,* lalu akan menjalankan prosedur pembangkitan kunci algoritma Elgamal sehingga fungsi ini akan memiliki output berupa nilai publik key berupa (*p*, *g, y*) dan private key berupa (*p*, *x*). Kode program untuk enkripsi dan dekripsi algoritma Elgamal adalah sebagai berikut :

```
def encryptG(publicKey, sPlaintext):
  m = encode(sPlaintext)
  p , g, y = publicKey
  cipher_pairs = []
  for i in m:
    k = random.randint( 0, p )
    a = modexp( g, k, p)
    b = (i*modexp( y, k, p)) % p
    cipher_pairs.append( [a, b] )
    encryptedStr = ""
  for pair in cipher_pairs:
    encryptedStr += str(pair[0]) + ' ' +
      str(pair[1]) + ' '
    return encryptedStr

def decryptG(privateKey, cipher):
  plaintext = []
  p, x = privateKey
  cipherArray = cipher.split()
  for i in range(0, len(cipherArray), 2):
    a = int(cipherArray[i])
    b = int(cipherArray[i+1])
    s = modexp( a, x, p )
    plain = (b*modexp( s, p-2, p)) % p
    plaintext.append( plain )
    decryptedText = decode(plaintext)
  return decryptedText
```

Fungsi encryptG adalah fungsi untuk tahap enkripsi pada algoritma Elgamal, fungsi ini akan mengeluarkan hasil enkripsi dari plaintext berupa angka. Fungsi decryptG adalah fungsi untuk tahap deksripsi pada algoritma Elgamal, fungsi ini akan mendekripsi ciphertext yang diinput menjadi plaintext.

C. Algoritma Paillier

Kode program untuk generate key pada algoritma Paillier adalah sebagai berikut :

```
def generateKeyP(p, q):
  n = p*q
  l = lcm(p-1, q-1)
  g = random.randint( 0, pow(n,2)-1)
  u = modinv( lFunction( modexp(g,int(l),
      pow(n,2) ) ,n) ,1,  n)[0]
  return{
    'publicKey' : (g,n),
    'privateKey' : (l, u,n)
  }
```

Fungsi generateKeyR akan meminta input nilai *p* dan *q,* lalu akan menjalankan prosedur pembangkitan kunci algoritma Paillier sehingga fungsi ini akan memiliki output berupa nilai publik key berupa (*g*, *n*) dan private key berupa (*l*, *u, n*). Kode

program untuk enkripsi dan dekripsi algoritma RSA adalah sebagai berikut

```
def encryptP(publicKey, plainText):
  encoded = encode(plainText)
  g, n = publicKey
  r = n
  n2 = pow(n,2)
  while(gcd(r,n) != 1):
    r = random.randint(0, n-1)
  result = [modexp(modexp(g, char, n2)*
           modexp(r,n,n2),1,n2) for char
           in encoded]
  return result

def decryptP(privateKey, cipherText):
  l, u, n = privateKey
  n2 = pow(n,2)
  lFunct= [lFunction(modexp(char, int(l),
          n2),n) for char in cipherText]
  result= [modexp(int(char)*int(u),1, n)
          for char in lFunct]
  result = decode(result)
  return result
```

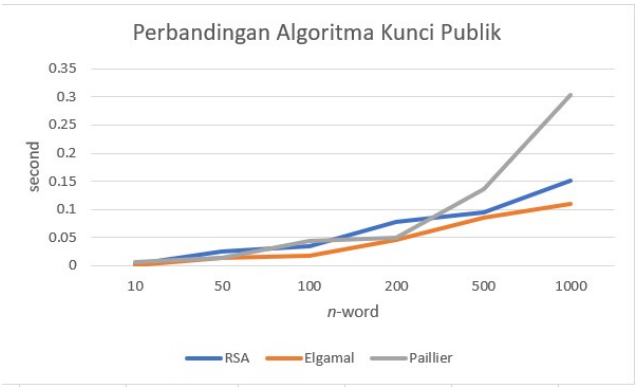
Fungsi encryptP adalah fungsi untuk tahap enkripsi pada algoritma Paillier, fungsi ini akan mengeluarkan hasil enkripsi dari plaintext berupa angka. Fungsi decryptP adalah fungsi untuk tahap deksripsi pada algoritma Paillier, fungsi ini akan mendekripsi ciphertext yang diinput menjadi plaintext

IV. Analisis

Untuk menentukan algoritma kunci publik mana yang akan digunakan untuk diimplementasikan pada keamanan penyedia jasa top up, dilakukan perbandingan kecepatan enkripsi dan dekripsi pada program yang telah dibuat. Perbandingan ini dilakukan dengan panjang properti pembangkitan kunci yang sama yaitu sebanyak 5 digit. Device dan teks yang akan dienkripsi sudah disamakan untuk ke 3 algoritma. Dilakukan enkripsi dan dekripsi pada panjang teks yang berbeda dengan selang 10, 50, 100, 200, 500, dan 100 kata. Waktu dihitung dalam bentuk second. Waktu ini dihitung mulai dari saat fungsi encrypt(R, G, P) atau fungsi decrypt(R, G, P) dijalankan, dan akan berhenti saat fungsi tersebut selesai. Berikut hasil dari perbandingan tersebut :

Algoritma	Action	Word					
		10	50	100	200	500	1000
RSA	Enkripsi	0.0009971	0.0122044	0.0180054	0.0389757	0.0479412	0.0761075
	Dekripsi	0.0009832	0.0121304	0.0172384	0.0393948	0.0473830	0.0752830
	Total	0.0019803	0.0243348	0.0352438	0.0783705	0.0953242	0.1513905
Elgamal	Enkripsi	0.0000138	0.0069129	0.0090308	0.0229380	0.0429652	0.0580077
	Dekripsi	0.0000121	0.0066322	0.0091924	0.0238390	0.0419430	0.0523210
	Total	0.0000259	0.0135451	0.0182232	0.0467770	0.0849082	0.1103287
Paillier	Enkripsi	0.0029912	0.0068634	0.0210257	0.0239735	0.0690241	0.1446145
	Dekripsi	0.0028383	0.0068430	0.0238382	0.0248310	0.0672320	0.1583830
	Total	0.0058295	0.0137064	0.0448639	0.0488045	0.1362561	0.3029975

Dari beberapa Algoritma kunci publik yang ada, algoritma Elgamal memiliki kecepatan dalam enkripsi paling cepat diantara algoritma lainnya. Berikut diagram yang lebih jelas untuk mengetahui perbedaan ke 3 algoritma tersebut :



Dilihat dari hasil perbandingan yang sudah dilakukan, algoritma Elgamal memiliki waktu eksekusi yang paling cepat. Sedankan algoritma Paillier memiliki waktu eksekusi paling lambat. Oleh karena itu, algoritma kunci publik yang akan digunakan dalam menyelesaikan masalah keamanan data user saat melakukan top up adalah algoritma Elgamal.

V. Implementasi

Transaksi top up merupakan sebuah transaksi yang memiliki celah dalam keamanan. Transaksi ini dilakukan dalam media digital. Tentunya masih ada para penyedia jasa top up yang masih menggunakan media sosial untuk melakukan perpindahan informasi. Untuk itu, algoritma kunci publik sangat berguna untuk menjaga kerahasiaan pertukaran akun. Salah satu transaksi yang ramai pada saat ini adalah top up blessing pada sebuah gim genshin impact. Para penyedia jasa perlu login ke akun player untuk melakukan pengisian, karena mihoyo belum menyediakan top up blessing melalui UID pemain. Biasanya para penyedia jasa akan mengirim pesan sebagai berikut untuk meminta data user.

FORMAT GENSHIN IMPACT
Jika Login dengan Facebook/Twitter/Google , Link-an Username dan Password ke MIHOYO dahulu. Caranya ikuti : http://bit.ly/akunmihoy0 🙏🙏

SERVER : ĶĤÜŠÙŞ ÄŠÍÁ
1. Username atau Email :
2. Password MIHOYO :
3. Nickname :
4. AR :
5. Item yang dibeli :

ISI FORMAT DENGAN BENAR YA🙏🙏
sesuai antrian mohon bersabar, tolong standby di room chat ini 1-60 menit, admin akan ketik angka 1 utk minta kode dari e-mail, dan kirim code ke sini >< (Ada Update terbaru 2FA)

Jika user meninput data mereka pada format tersebut tanpa di enkripsi terlebih dahulu, maka akun merekan mungkin tidak akan aman. Mungkin saja ada pihak yang tidak bertanggung jawab yang melihat handphone user atau handphone dari penyedia jasa, sehingga data user ini akan terambil alih oleh penyadap tersebut. Untuk itu, penyedia jasa harus menerapkan algoritma kunci publik untuk menjaga kemanan data. Penyedia

jasa harus memiliki kunci publik dan kunci privat tersendiri untuk menggunakan algoritma ini. Seperti yang sudah dianalisis sebelumnya, algoritma kunci publik yang akan digunakan adalah algoritma Elgamal. Untuk implementasi algoritma Elgamal adalah sebagai berikut

1. Pembangkitan Kunci, pertama-tama penyedia jasa akan membangkitkan kunci terlebih dahulu dengan program yang sudah dibuat, untuk membuat publik key dan private key mereka. Salah satu contohnya adalah sebagai berikut :

```
  P : 83761
  G : 74831
  X : 68457
{'publicKey': (83761, 74831, 76704),
 'privateKey': (83761, 68457)}
```

2. Setelah mendapatkan kunci publik dan kunci privat, penyedia jasa akan melampirkan kunci publik kedalam pesan mereka yang dikirimkan pada pelanggan.
3. Pelanggan akan melakukan enkripsi terlebih dahulu dengan kunci publik yang diberikan oleh penyedia jasa sehingga tampilannya adalah sebagai berikut :

FORMAT GENSHIN IMPACT
Jika Login dengan Facebook/Twitter/Google , Link-an Username dan Password ke MIHOYO dahulu. Caranya ikuti : http://bit.ly/akunmihoy0 🙏🙏
Gunakan kunci publik Elgamal (83761, 74831, 76704) untuk mengenkripsi username dan password anda

SERVER : ĶĤÜŠÙŞ ÄŠÍÁ
1. Username atau Email : 5271557185594455851412812728926083779330158913885531557533075545213207358347103203784045772232723491154976583282778803227815429562132022589822612. Password MIHOYO : 2885366258600337961044952980673610787761678989766800225715496103013024483173172387601675053353013. Nickname : Pelanggan
4. AR : 60
5. Item yang dibeli : Blessing 3 bulan

ISI FORMAT DENGAN BENAR YA🙏🙏
sesuai antrian mohon bersabar, tolong standby di room chat ini 1-60 menit, admin akan ketik angka 1 utk minta kode dari e-mail, dan kirim code ke sini >< (Ada Update terbaru 2FA)

4. Setelah itu penyedia jasa akan mendekripsikan username dan password pelanggan untuk melakukan top up.

```
Username : Pelanggan_Setia
Password : TukangSate
```

Dengan adanya enkripsi pada data username dan password pelanggan, perpindahan data akan lebih aman. Tingkat kejahatan yang terjadi bisa berkurang, dan membuat penyedia jasa tidak mengalami kerugian jika terjadi sesuatu. Untuk melakukan peretasan dibutuhkan waktu yang cukup lama. Sedangkan untuk melakukan top up hanya diperlukan kurang lebih 5 menit saja, sehingga waktu yang dibutuhkan oleh para peretas kurang. Pelanggan mungkin sudah mengganti passwordnya sebelum pesan terdekripsi oleh peretas.

VI. KESIMPULAN

Algoritma kunci publik merupakan algoritma yang sulit dipecahkan karena kunci yang dimiliki oleh pengirim pesan dan penerima pesan berbeda. Dengan algoritma ini, tidak diperlukan juga pertukaran kunci yang dilakukan secara rahasia. Dari beberapa algoritma kunci publik yang ada yaitu RSA, Elgamal, dan Paillier, ketiganya memiliki kelebihan dan kekurangan masing-masing, Namun untuk masalah waktu eksekusi Algoritma Elgamal memiliki waktu paling singkat. Untuk itu pada penggunaan algoritma kunci publik yang diimplementasikan pada perpindahan data user saat melakukan top up digunakanlah algoritma Elgamal. Hal ini akan membuat transaksi saat melakukan top up, utamanya saat pertukaran data user akan lebih aman. Orang yang tidak sengaja melihat pesan kita dengan penyedia jasa, tidak akan mengetahui data yang sudah dienkripsi, bahkan jika orang tersebut berniat melakukan kejahatan, waktu yang dibutuhkan cukup lama untuk mendekripsi pesan tersebut. Username dan Password yang akan dienkripsi juga tidak mungkin melebihi 25 karakter, sehingga penggunaan algoritma kunci publik tidak memakan banyak waktu. Selain itu, keamanan algoritma kunci publik lebih tinggi dari algoritma kunci simetris.

REFERENCES

[1] Munir, Rinaldi. 2021. "Kriptografi Klasik". Bandung: ITB.
[2] Munir, Rinaldi. 2021. "Algoritma RSA". Bandung: ITB.
[3] Munir, Rinaldi. 2021. "Algoritma Elgamal. Bandung: ITB.
[4] Munir, Rinaldi. 2021. "Enkripsi homomorfik". Bandung: ITB.

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, 26 April 2021

Muhammad Ziad Rahmatullah
13518032