Analisis dan Perbandingan Kecepatan Algoritma RSA dan Algoritma ElGamal

Nikolaus Indra - 13508039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18039@students.if.itb.ac.id

***Abstraksi*—Algoritma kriptografi kunci publik lebih diminati dibanding algoritma kriptografi kunci privat. Hal ini dikarenakan keamanan dalam menyimpan kunci jika dibandingkan dengan algoritma kriptografi kunci privat. Pada algoritma kriptografi kunci privat, pengirim pesan harus mengirimkan juga kunci dekripsi kepada penerima.**

Walaupun dari segi kecepatan memproses algoritma kriptografi kunci publik lebih lambat daripada algoritma kriptografi kunci privat, algoritma kriptografi kunci publik sudah banyak diimplementasikan. Contoh algoritma kriptografi yang menggunakan prinsip kunci publik adalah: Diffie-Helman, DSS, ElGamal, RSA, Cramer-Shoup, dan lain-lain. Pada makalah ini, penulis akan membandingkan algoritma RSA dan ElGamal dalam melakukan proses enkripsi dan dekripsi dari segi kecepatan memproses, keamanan, beserta perbandingan yang lainnya.

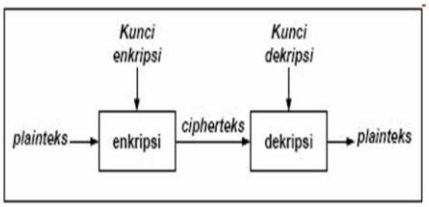
***Kata Kunci*—Algoritma kriptografi kunci privat, algoritma kriptografi kunci publik, algoritma RSA, algoritma ElGamal.**

I. PENDAHULUAN

Perkembangan teknologi informasi sekarang ini berkembang sangat cepat. Hal ini dapat dibuktikan dengan hampir semua orang pasti memiliki barang yang berkenaan dengan teknologi entah itu telepon genggam pintar (*smart phone*), komputer personal (*PC*), laptop, atau bahkan tablet yang sedang menjadi pusat perhatian di masyarakat luas. Tentu dengan adanya teknologi tersebut, orang-orang akan semakin bergantung pada teknologi yang disebutkan di atas. Pada umumnya mereka akan menyimpan semua data-data pribadinya atau data-data pekerjaan di dalam komputer atau laptop mereka. Sudah pasti data-data tersebut sangat penting bagi mereka. Mereka akan menyimpannya sedemikian rupa sehingga isi dari data-data tersebut tidak diketahui oleh pihak yang tidak diinginkan. Contohnya adalah data mengenai rencana yang akan dilakukan oleh sebuah perusahaan untuk setahun ke depan atau data mengenai suatu penelitian di bidang tekonologi komputasi yang belum dipublikasikan. Informasi seperti yang telah disebutkan merupakan sebuah informasi yang sangat berharga dan tidak ingin diketahui oleh sembarang orang.

Yang menjadi permasalahan adalah ketika ingin menyebarkan/mengirim informasi tersebut ke pihak lain yang membutuhkan. Permasalahannya adalah bagaimana mengirim informasi tersebut melalui saluran komunikasi yang aman. Saluran komunikasi yang diminati adalah internet. Namun media internet tidak dapat menjamin dengan pasti apakah data/informasi yang kita kirim dapat tetap terjaga kerahasiaannya atau tidak.

Oleh karena tidak adanya kepastian tentang keamanan suatu informasi yang disalurkan melalui media komunikasi maka diperlukan suatu metode untuk menjaga kerahasiaan informasi tersebut. Metode yang dapat digunakan adalah kriptografi. Kriptografi dilakukan untuk menyembunyikan konten dari suatu informasi dengan cara menyandikannya dengan sebuah kunci sehingga untuk membaca isi informasi tersebut dibutuhkan sebuah kunci.



Gambar 1. Skema Umum Kriptografi

Namun hanya dengan sebuah kunci saja dalam melakukan enkripsi/dekripsi dapat memungkinkan informasi yang dirahasiakan akan 'bocor'. Oleh karena itu muncul ide kriptografi kunci publik yang lebih dapat menjamin keamanan informasi. Terdapat banyak contoh algoritma kriptografi kunci public seperti: Diffie-Helman, DSS, ElGamal, RSA, Cramer-Shoup, dan lain-lain.

Pada makalah ini penulis akan mengimplementasi algoritma kriptografi kunci publik yang terkenal yaitu RSA dan ElGamal, lalu membandingkan antara kedua algoritma tersebut untuk mengetahui mana yang lebih baik di antara keduanya dari segi kecepatan memproses, keamanan, dan perbandingan lainnya.

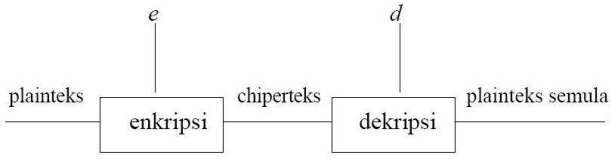
II. ALGORITMA KUNCI PUBLIK

Seperti yang dijelaskan di atas, untuk melakukan enkripsi dan dekripsi pesan dibutuhkan satu buah kunci yang sama. Sebagai contoh jika Alice memiliki sebuah informasi yang ingin disampaikan ke Bob, maka Alice membuat sebuah kunci untuk melakukan enkripsi

informasi menjadi sebuah cipher teks. Kemudian cipher teks tersebut dapat dikirimkan melalui media komunikasi mana saja tanpa peduli masalah keamanan informasi tersebut karena apabila ada yang berhasil mendapatkan cipher teks tersebut, hal itu akan sia-sia jika tidak memiliki kunci untuk membuat cipher teks tersebut dapat dibaca (di-dekripsi).

Yang menjadi masalah adalah bagaimana mengirimkan kunci yang telah dibuat oleh Alice untuk sampai dengan aman kepada Bob karena Bob tidak akan bisa mengerti konten dari informasi yang dikirimkan oleh Alice tanpa memilki kunci untuk mendekripsi cipher teks yang telah diterima. Untuk mengirim kunci dibutuhkan saluran komunikasi yang benar-benar aman. Ternyata metode menggunakan satu buah kunci dalam melakukan enkripsi dan dekripsi pesan tidaklah aman sepenuhnya. Kriptografi dengan satu buah kunci dinamakan dengan kriptografi kunci privat.

Karena kriptografi kunci privat tidak sepenuhnya aman, maka pada tahun 1976 dikembangkan metode yang lebih aman yaitu kriptografi kunci publik. Idenya adalah masing-masin pengirim pesan dan penerima pesan memilki sepasang kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi pesan sedangkan kunci privat digunakan untuk mendekripsi pesan. Keuntungannya adalah tidak diperlukan pengiriman kunci secara rahasia dan jumlah kunci dapat ditekan.



Gambar 2. Skema Kriptografi kunci publik (e = kunci publik, d = kunci privat).

Kriptografi kunci publik didasarkan pada fakta bahwa komputasi untuk enkripsi dan dekripsi pesan mudah dilakukan. Kemudian secara komputasi hampir tidak mungkin menurunkan kunci privat (d) bila diketahui kunci publik (e). Walaupun demikian ada kelemahan pada kriptografi kunci public yaitu proses enkripsi dan dekripsi data umumnya lebih lambat dibandingkan kriptografi kunci privat, ukuran cipher teks lebih besar daripada ukuran plain teks-nya, dan ukuran kunci pada kriptografi kunci publik lebih besar daripada kunci pada kriptografi kunci privat.

Beberapa algoritma kriptografi yang menggunakan prinsip kriptografi kunci publik adalah : Diffie-Helman, DSS, ElGamal, RSA, Cramer-Shoup, dan lain-lain.

III. RSA

Algoritma RSA merupakan algoritma kriptografi kunci publik yang paling terkenal dan paling banyak aplikasinya. Algortima RSA ini ditemukan oleh tiga peneliti dari MIT, yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976.

Parameter yang dibutuhkan oleh algoritma RSA :

1.	p dan q	(rahasia)
2.	$n = p \cdot q$	(tidak rahasia)
3.	$\phi(n) = (p-1)(q-1)$	(rahasia)
4.	e	(tidak rahasia)
5.	d	(rahasia)

Keterangan dan langkah untuk mendapatkan kunci publik dan kunci privat adalah sebagai berikut:

1. Pilih dua buah bilangan prima, p dan q yang bersifat rahasia.
2. Setelah mendapat p dan q, maka akan didapat nilai n = p.q
3. Selanjutnya hitung $\phi(n) = (p-1)(q-1)$.
4. Dari nilai $\phi(n)$, selanjutnya dicari nilai e untuk kunci publik. Nilai e harus relatif prima terhadap nilai $\phi(n)$.
5. Nilai d didapatkan dengan persamaan :
   $ed \equiv 1 \pmod{\phi(n)}$ atau $d \equiv e^{-1} \bmod (\phi(n))$

Setelah mendapat nilai-nilai p, q, n, e, dan d, maka selanjutnya dapat dilakukan enkripsi pesan.

Dalam melakukan proses enkripsi, plain teks terlebih dahulu dipecah menjadi blok-blok kecil. Setelah itu dilakukan perhitungan menggunakan rumus sebagai berikut:

$$c_i = p_i \wedge e \bmod n$$

Dengan e adalah kunci publik dan n = p.q

Sebaliknya untuk melakukan proses dekripsi digunakan rumus sebagai berikut:

$$p_i = c_i \wedge d \bmod n$$

Dengan d adalah kunci privat dan n = p.q

IV. ELGAMAL

Algoritma ElGamal merupakan algoritma kunci publik yang ditemukan oleh Taher Elgamal pada tahun 1985. Sama seperti algoritma RSA, algoritma ElGamal terdiri dari 3 proses utama, yaitu pembentukan kunci, proses enkripsi, dan proses dekripsi. Algoritma ElGamal ini juga membagi plain teks menjadi blok-blok kecil sebelum di enkripsi. Keamanan algoritma ini terletak pada sulitnya menghitung logaritma diskrit.

Persoalan logaritma diskrit adalah sebagai berikut: Terdapat bilangan prima p dan terdapat bilangan bulat g dan y. Maka carilah x sedemikian sehingga

$$g^x \equiv y \pmod p$$

Bilangan x disebut logaritma diskrit terhadap y dengan basis g ($x = \log g^y$).

Parameter yang dibutuhkan algoritma ElGamal:

1.	p	(tidak rahasia)
2.	g (g < p)	(tidak rahasia)
3.	x (x < p)	(rahasia)
4.	$y = g^x \bmod p$	(tidak rahasia)

Keterangan dan langkah untuk mendapatkan kunci publik dan privat adalah sebagai berikut:

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

Melalui langkah di atas maka akan didapat:
- kunci publik: tripel (y, g, p).
- kunci privat: pasangan (x, p).

Setelah mendapatkan kunci publik dan kunci privat, maka selanjutnya akan dilakukan enkripsi terhadap plain teks. Proses enkripsi yang dilakukan adalah sebagai berikut:

1. Pertama-tama plain teks dipecah-pecah menjadi blok yang lebih kecil dan disusun menjadi blok-blok $m_1, m_2, \ldots, m_n$.
2. Pilih bilangan acak k yang terletak pada nilai $1 \leq k \leq p-2$
3. Setiap blok m dienkripsi dengan rumus:
   $a = g^k \bmod p$
   $b = y^k m \bmod p$
   Pasangan a dan b adalah cipher teks untuk blok pesan m. Jadi ukuran cipher teks dua kali ukuran plain teksnya.

Selanjutnya akan dijelaskan proses dekripsi dari algoritma ElGamal:

1. Gunakan kunci privat x untuk menghitung
   $(a^x)^{-1} = a^{p-1-x} \bmod p$

2. Hitung plain teks m dengan persamaan:
   $M = b/a^x \bmod p = b(a^x)^{-1} \bmod p$

V. IMPLEMENTASI RSA

Penulis mengimplementasikan algoritma kriptografi RSA dengan menggunakan bahasa pemrograman C# dengan menggunakan framework .NET4 dengan bantuan IDE Visual Studio 2010.

Sesuai dengan dasar teori algoritma RSA yang telah dituliskan di atas, penulis akan mengimplementasikan sesuai aturan dan rumus-rumus tersebut. Penulis menggunakan tipe integer BigNum (BigInteger).

Program dapat melakukan generate key dengan sebelumnya menerima masukan nilai p, q, dan e sehingga menghasilkan nilai n (n = pq) dan d (kunci privat). Dengan adanya generate key ini, pengguna tidak perlu mencari-cari nilai d yang relatif sulit untuk dicari, karena kita harus mencari nilai d dengan persamaan:

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

Berikut potongan kode untuk melakukan generate key yang dibuat penulis:

```
//Generate Private Key
    public void GenerateKey()
    {
      n = p * q;
      long toitent = (p - 1) * (q - 1);
      bool b = false;

      for (int i = 1; !b; i++)
      {
        double N = (1.0 * 1 + i * toitent) / Enc;
        if (IsRoundNumber(N))
        {
          b = true;
          Dec = (long)N;
        }
      }
    }
```

Berikut potongan kode untuk prosedur enkripsi algoritma RSA yang dibuat penulis.

```
public string Encrypt(byte[] plain)
    {
      string result = "";
      //GenerateKey();
      string plainASCII = ByteToASCIIString(plain);
      string[] AS = StringToArrOfString(plainASCII,
n.ToString().Length - 1);
      string[] cipher = new string[AS.Length];

      if (AS[AS.Length - 1].Length < AS[0].Length)
      {
        for (int i = 0; i < (AS[0].Length - AS[AS.Length -
1].Length); i++)
        {
          AS[AS.Length - 1] = "0" + AS[AS.Length - 1];
        }
      }

      for (int i = 0; i < AS.Length; i++)
      {
        BigInteger a = Convert.ToInt64(AS[i], 10);
        BigInteger b = new BigInteger(a.modPow(Enc, n));
        long cip = Convert.ToInt64(b.ToString(),10);
        cipher[i] = cip.ToString();
      }

      for (int i = 0; i < cipher.Length; i++)
      {
        if (i != cipher.Length - 1)
        {
          result += cipher[i] + " ";
        }
        else
        {
          result += cipher[i];
        }
      }
      return result;
    }
```

Berikut potongan kode untuk prosedur dekripsi algoritma RSA yang dibuat penulis.

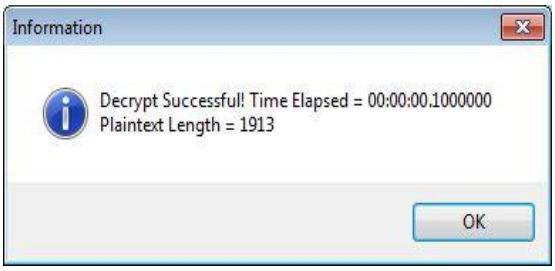
```
public byte[] Decrypt(string ciphertext)
    {
        string result = "";
        //GenerateKey();
        string[] AS = ciphertext.Split(' ');
        string[] plain = new string[AS.Length];
        for (int i = 0; i < AS.Length; i++)
        {
            BigInteger c = Convert.ToInt64(AS[i], 10);
            BigInteger b = new BigInteger(c.modPow(Dec,n));
            if (b.ToString().Length < n.ToString().Length - 1)
            {
               string temp = b.ToString();
               for (int j = 0; j < (n.ToString().Length - 1 -
b.ToString().Length); j++)
               {
                  temp = "0" + temp;
               }
               plain[i] = temp;
            }
            else
            {
               plain[i] = b.ToString();
            }
        }
        for (int i = 0; i < plain.Length; i++)
        {
           result += plain[i];
        }
        string[] rs = StringToArrOfString(result, 3);
        byte[] res = new byte[rs.Length];
        for (int i = 0; i < rs.Length; i++)
        {
           res[i] = (byte)(int.Parse(rs[i]));
        }
        return res;
    }
```

Sebagai data uji, penulis mencoba melakukan enkripsi file teks berukuran 1913 byte.

Berikut hasil yang didapat ketika melakukan enkripsi file plain teks yang bersangkutan (cipher text ditampilkan dalam desimal):

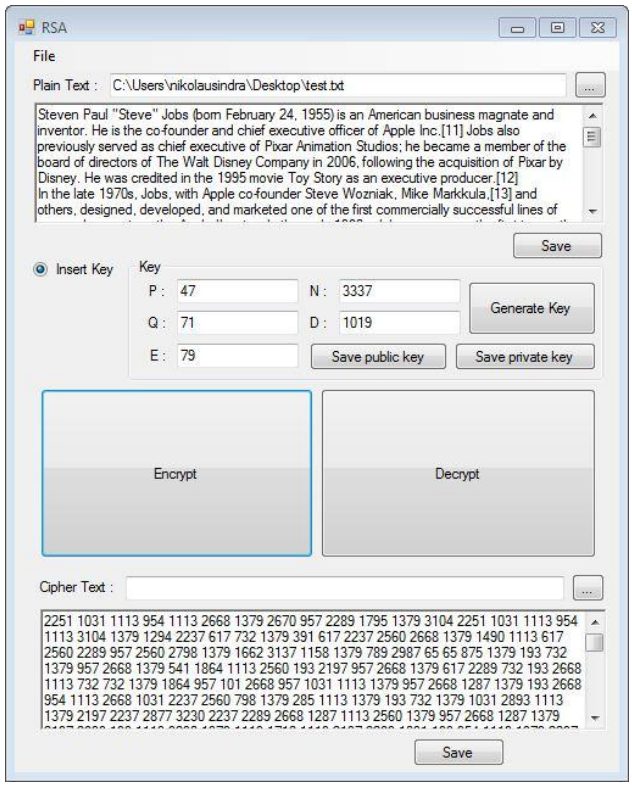
Gambar 3. Waktu dan ukuran file cipher teks hasil enkripsi

Berikut hasil yang didapat ketika melakukan dekripsi file cipher teks yang bersangkutan:



Gambar 5. Waktu dan ukuran file plain teks hasil dekripsi

Berikut tampilan antarmuka setelah melakukan enkripsi file tersebut:



Gambar 4. Tampilan antarmuka algoritma RSA

VI. IMPLEMENTASI ELGAMAL

Penulis mengimplementasikan algoritma kriptografi ElGamal dengan menggunakan bahasa pemrograman C# dengan menggunakan framework .NET4 dengan bantuan IDE Visual Studio 2010.

Sesuai dengan dasar teori algoritma ElGamal yang telah dituliskan di atas, penulis akan mengimplementasikan sesuai aturan dan rumus-rumus tersebut. Penulis menggunakan tipe integer BigNum (BigInteger).

Berikut potongan kode untuk prosedur enkripsi algoritma ElGamal yang dibuat penulis:

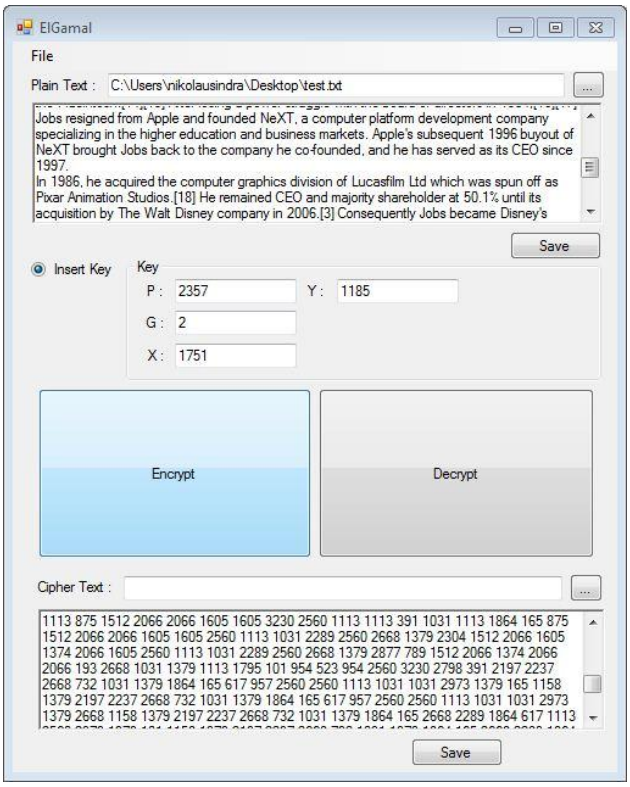
```
public BigInteger[] Encrypt(BigInteger biInput,
PublicKeyPacket pkpKey) {
        EG_Public_Key epkKey = new EG_Public_Key();
        epkKey.p = pkpKey.KeyMaterial[0];
        epkKey.g = pkpKey.KeyMaterial[1];
        epkKey.y = pkpKey.KeyMaterial[2];

        BigInteger k = new BigInteger();

        //random untuk enkripsi
        k = BigInteger.genRandom(epkKey.p.bitCount()-1);

        while (k > (epkKey.p-1)) {
        k = new BigInteger();
        k = BigInteger.genRandom(epkKey.p.bitCount()-1);
        }

        BigInteger B = epkKey.g.modPow(k, epkKey.p);
        BigInteger c = epkKey.y.modPow(k, epkKey.p);
        c = (biInput * c) % epkKey.p;
        //BigInteger c = (biInput * epkKey.y.modPow(k,
epkKey.p)) % epkKey.p;

        BigInteger[] biOutput = new BigInteger[2];

        biOutput[0] = B;
        biOutput[1] = c;

        return biOutput;

}
```

Berikut potongan kode untuk prosedur dekripsi algoritma ElGamal yang dibuat penulis:

```
public BigInteger Decrypt(BigInteger[] biInput,
SecretKeyPacket skpKey, string strPassphrase) {
        BigInteger[]biKeyMaterial=
skpKey.GetDecryptedKeyMaterial(strPassphrase);
        EG_Secret_Key eskKey = new EG_Secret_Key();
        eskKey.x = biKeyMaterial[0];
        eskKey.p = skpKey.PublicKey.KeyMaterial[0];
        eskKey.g = skpKey.PublicKey.KeyMaterial[1];
        eskKey.y = skpKey.PublicKey.KeyMaterial[2];

        if (biInput.Length != 2)
                throw new ArgumentException("biInput
is not an ElGamal encrypted Packet");

        BigInteger B = biInput[0];
        BigInteger c = biInput[1];

        BigInteger z = B.modPow(eskKey.x,
eskKey.p).modInverse(eskKey.p);

        BigInteger output = (z * c) % eskKey.p;

        return output;
    }
```

Untuk implmentasi algoritma ElGamal, penulis mengimplementasikan keluaran (cipher teks) nilai a dan b untuk ditampilkan dalam textbox cipher text secara bersebelahan.

Untuk file teks yang sama ketika digunakan pada pengujian algoritma RSA di atas, akan menghasilkan tampilan sebagai berikut (cipher text ditampilkan dalam desimal):

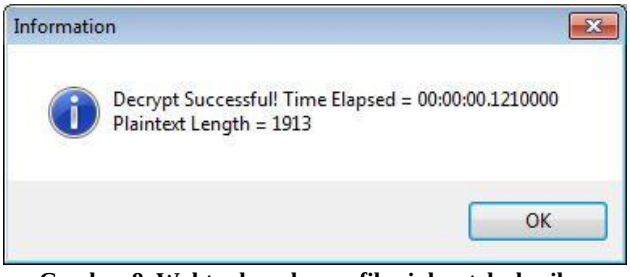


Gambar 6. Tampilan antarmuka algoritma ElGamal

Berikut hasil yang didapat ketika melakukan enkripsi file plain teks yang bersangkutan:

Gambar 7. Waktu dan ukuran file cipher teks hasil enkripsi

Berikut hasil yang didapat ketika melakukan dekripsi file cipher teks yang bersangkutan:



Gambar 8. Waktu dan ukuran file cipher teks hasil dekripsi

VII. Analisis Perbandingan Algoritma RSA dan Algoritma Elgamal

Setelah mengimplementasikan RSA dan ElGamal, penulis akan membandingkan kedua algoritma tersebut berdasarkan waktu proses enkripsi dan dekripsi dengan beberapa ukuran file. Perbandingan tersebut dapat dilihat pada tabel berikut ini:

a. Algortima RSA

No	**Waktu (s)**		**Ukuran File (byte)**	
	Enkripsi	Dekripsi	Plainteks	Cipherteks
1	0.136	0.100	1913	9035
2	0.334	0.296	5969	23876
3	0.894	0.802	13459	53836

Tabel 1. Perbandingan enkripsi/dekripsi file untuk algoritma RSA

b. Algoritma ElGamal

No	**Waktu (s)**		**Ukuran File (byte)**	
	Enkripsi	Dekripsi	Plainteks	Cipherteks
1	0.186	0.121	1913	18909
2	0.401	0.309	5969	53721
3	1.002	0.909	13459	121131

Tabel 2. Perbandingan enkripsi/dekripsi file untuk algoritma ElGamal

Pada tabel di atas, kita dapat mengetahui bahwa waktu proses algoritma RSA dalam hal enkripsi dan dekripsi lebih baik disbanding algoritma ElGamal.

Juga dari segi ukuran cipher teks, algoritma ElGamal memiliki ukuran yang lebih besar disbanding dengan ukuran cipher teks algoritma RSA. Hal ini disebabkan karena algoritma ElGamal memiliki pasangan cipher teks untuk setiap blok cipher, yaitu a dan b. Berbeda dengan algoritma RSA yang setiap blok cipher hanya memiliki satu nilai saja.

VII. Kesimpulan Dan Saran

Algoritma kriptografi kunci publik jauh lebih aman dibandingkan dengan algoritma kunci privat. Kelebihan kriptografi kunci publik yang lain adalah kita hanya perlu menjaga kunci privat, sedangkan kunci publik dapat dikirimkan bersamaan dengan cipher teks-nya.

Algoritma RSA dan algoritma ElGamal memiliki rumusan yang berbeda dalam melakukan enkripsi dan dekripsi.

Dalam segi kecepatan memproses informasi, algoritma RSA lebih cepat disbandingkan dengan algoritma ElGamal. Juga dalam besar ukuran file cipher teks, algoritma RSA menghasilkan ukuran untuk cipher teks yang lebih kecil dibandingkan dengan algoritma ElGamal.

Namun dalam hal keamanan, algoritma ElGamal akan lebih sulit dipecahkan dibandingkan dengan algoritma RSA, walaupun pada dasarnya untuk memecahkan algortima RSA pun sudah sangat sulit.

Sebagai saran, untuk kedua algoritma tersebut (RSA dan ElGamal), sebaiknya memilih nilai p dan q dengan panjang 100 digit sehingga nilai n (n = pq) akan memiliki panjang lebih dari 200 digit. Untuk algoritma ElGamal, pemilihan nilai p yang besar, sehingga nilai x dan g juga akan besar yang mengakibatkan kerahasiaan informasi akan semakin terjaga.

IX. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena atas rahmat-Nya, makalah ini dapat terselesaikan.

Penulis juga mengucapkan kepada Bapak Rinaldi Munir, selaku dosen Kriptografi semester genap tahun ajaran 2010/2011 atas bimbingan yang diberikan selama kuliah sehingga makalah ini dapat terselesaikan berkat materi kuliah yang disampaikan oleh Bapak Rinaldi Munir.

References

[1] http://en.wikipedia.org/wiki/ElGamal_encryption
[2] http://en.wikipedia.org/wiki/RSA
[3] http://www.di-mgt.com.au/rsa_alg.html
[4] http://ww3.algorithmdesign.net/full.html
[5] http://www.informatika.org/~rinaldi
[6] http://www.iusmentis.com/technology/encryption/elgamal/
[7] http://en.wikipedia.org/wiki/Taher_Elgamal
[8] http://www.quadibloc.com/crypto/pk0502.htm
[9] http://library.thinkquest.org/27158/concept2_4.html

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, 5 Mei 2011

Nikolaus Indra / 13508039