Pemakaian Kriptografi Kunci Publik dengan Algoritma RSA untuk Keamanan Data XML

Budi Hartono[1], Jazi Eko Istiyanto [2]
1 Fakultas Teknik Universitas Stikubank
Jl. Tri Lomba Juang No.1 Mugas, Semarang 50241 – Indonesia
email: pakbudi@yahoo.com
2 Fakultas MIPA Universitas Gadjah Mada
Sekip Utara, Bulak Sumur, Yogyakarta – Indonesia
email: jazi@ugm.ac.id

Ringkasan

Format penyimpanan data menggunakan XML (*Extensible Markup Language)* memiliki kelebihan yaitu struktur penyimpanan data yang fleksibel. Aturan penggunaannya telah distandarkan oleh *world wide web consortium* (W3C). Data XML dapat diolah lebih lanjut agar menghasilkan informasi. Selanjutnya untuk mengatur siapa saja yang berhak memperoleh informasi tersebut maka digunakan sistem enkripsi.

Teknik enkripsi yang digunakan adalah kriptografi kunci publik dengan algoritma RSA. Algoritma ini menghasilkan sepasang kunci yaitu kunci publik (e,n) yang bersifat umum untuk proses enkripsi dan kunci privat (d,n) yang bersifat rahasia untuk proses dekripsi. Kedua kunci ini diaplikasikan pada bagian tertentu dari data XML. Pengguna yang berhak memiliki kunci yang dapat digunakan untuk mendekripsi data sehingga memperoleh informasi yang sesuai dengan hak aksesnya.

Penelitian ini mengerjakan proses merancang, membuat program, dan menganalisa pemakaian algoritma RSA untuk keamanan data XML. Hasil implementasi berupa *prototype* program komputer yang dibuat menggunakan bahasa C.

Kata kunci

Data XML, Kunci Publik, Kunci Privat, Enkripsi, Dekripsi, Algoritma RSA.

1. Pendahuluan

Konsep penyimpanan data menggunakan format XML (*Extensible Markup Language)* berbeda dengan penyimpanan secara *record* dan *field,* yaitu di dalam bentuk struktur penyimpanan data yang lebih fleksibel[1]. Format penyimpanan data menggunakan XML mengikuti aturan yang telah dibuat oleh suatu organisasi internasional yaitu *world wide web consortium* atau W3C[2].

Data XML dapat diolah untuk menghasilkan informasi. Penyajian informasi dapat ditujukan kepada banyak pihak dimana setiap pihak dapat memperoleh atau mengakses bagian informasi yang berbeda. Untuk mengatur siapa saja yang berhak dan siapa yang tidak berhak mengakses informasi tersebut maka digunakan sistem enkripsi pada data XML.

Teknik enkripsi yang dipilih pada penelitian ini yaitu kriptografi kunci publik dengan algoritma RSA. Nama algoritma ini diambil dari nama belakang penemunya yaitu Ronald Rivest, Adi Shamir dan Leonard Adleman. Sepasang kunci yang dihasilkan dari algoritma RSA digunakan untuk proses enkripsi dan dekripsi pada bagian tertentu dari dokumen XML. Pengguna yang berhak memiliki kunci yang dapat digunakan untuk mendekripsi data sehingga memperoleh informasi sesuai dengan hak aksesnya.

Perumusan masalah yang muncul untuk diteliti adalah: membuat contoh data XML, membuat pasangan kunci publik menggunakan algoritma RSA, dan menganalisa proses enkripsi dan dekripsi pada data XML. Tujuan penelitian ini adalah membuat dan mengaplikasikan data XML yang telah dienkripsi menggunakan kunci publik dengan algoritma RSA. Hasil implementasi berupa dua buah *prototype* program komputer yaitu FINDKEY dan CRYPTO yang dibuat menggunakan bahasa pemrograman C.

2. Data XML

XML merupakan bagian dari *Structured Generalized Markup Language* (SGML). XML dirancang untuk kemudahan implementasi dan untuk interoperabilitas dengan SGML dan HTML. XML adalah bahasa markup yang dirancang untuk penyampaian informasi melalui *World Wide Web*. Setiap elemen XML memiliki sepasang *tag*-awal dan *tag*-akhir[3].

Sebuah data XML terdiri dari dua bagian utama yaitu prolog dan elemen dokumen atau root elemen. Bagian prolog berisi:

a. Deklarasi yang menyatakan dokumen XML dengan nomor versinya.
b. Sebuah komentar didalam <!-- dan -->.
c. Dua buah komponen opsional, yaitu:
   1. Sebuah deklarasi tipe dokumen, yang mendefinisikan tipe dan struktur dokumen.
   2. Satu atau lebih instruksi pemrosesan yang memberikan informasi untuk mengaitkan sebuah *style sheet ke* dokumen XML.

Sedangkan bagian elemen dokumen berisi sebuah elemen tunggal root yang dapat berisi elemen-elemen tersarang lain sehingga membentuk hierarki pohon. Elemen-elemen ini menandakan struktur logika dokumen XML dan digunakan untuk menyimpan data[4][5]. Pada gambar 1 dapat dilihat contoh data XML yang akan diproses lebih lanjut.

```
<?xml version="1.0"?>
<!-- Name File : sim.xml -->
<!-- Keterangan: File data simpanan
deposito -->
<?xml-stylesheet type="text/xsl"
href="xslsim.xsl"?>
<SIMPANAN>
  <DATABANK>
    <NAMABANK>BANK
SEJAHTERA</NAMABANK>
    <PRODUK>DEPOSITO RUPIAH</PRODUK>
    <CABANG>SEMARANG TIMUR</CABANG>
    <ALAMATCABANG>JL.ARTERI TIMUR
      103-105</ALAMATCABANG>
    <KOTACABANG>SEMARANG</KOTACABANG>
    <TANGGAL>1 MARET 2002</TANGGAL>
  </DATABANK>
  <DATANASABAH>
    <NAMANASABAH>DRS. SLAMET
      SUPRAPTO</NAMANASABAH>

<NOMORBILYET>050210123456</NOMORBILYE
T>

<NOREKENING>050230747424</NOREKENING>
    <NOMINALRP>25000000</NOMINALRP>
    <BUNGAPERTAHUN>13</BUNGAPERTAHUN>
    <ALAMAT>PERUM. ASRI PERMAI,
JL.KEBUN
      BUNGA RAYA IV/51</ALAMAT>
    <KOTA>PEDURUNGAN -
SEMARANG</KOTA>
    <PERIODEBULAN>6</PERIODEBULAN>
    <TGLMASUK>10 JANUARI
2002</TGLMASUK>
    <TGLJATUHTEMPO>10 JULI 2002
      </TGLJATUHTEMPO>
  </DATANASABAH>
  <DATANASABAH>
    <NAMANASABAH>GUNAWAN
      WIBISONO</NAMANASABAH>

<NOMORBILYET>050210332660</NOMORBILYE
```

Gambar 1. File sim.xml

Menampilkan dokumen XML pada *browser* dilakukan menggunakan pengurai XML yang dimiliki oleh *browser* tersebut. Pada gambar 2 dapat dilihat contoh tampilan data dari file sim.xml menggunakan *browser* Microsoft Internet Explorer versi 5. Data tersebut digunakan untuk mencatat simpanan deposito berjangka.

Gambar 2. Tampilan pada *browser* data sim.xml

3. Teori Bilangan

Teori bilangan yang digunakan pada algoritma RSA adalah: modulo, modulo kongruen, pembagi persekutuan terbesar, bilangan prima relatif, modulo invers, fungsi totient Euler, dan bilangan prima.

3.1 Modulo dan modulo kongruen

Operasi aritmatika modulo ditulis dengan notasi a mod n. Modulo akan menghasilkan bilangan bulat yang merupakan sisa dari pembagian suatu bilangan. Pada a mod n menghasilkan kemungkinan bilangan antara 0 sampai dengan n-1[6]. Modulo kongruen n (*congruent modulo*) terjadi apabila (a mod n) = (b mod n), ditulis dengan notasi a ≡ b mod n. *Contoh:*

a. 23 mod 12= **11**, karena 23 mod 12 = 1.12 + **11**
b. 11 mod 12 = **11**, karena 11 mod 12 = 0.12 + **11**
c. 36 mod 6= **0**, karena 36 mod 6 = 6.6 + **0**
d. 23 ≡ 11 mod 12 = **11**
e. –43 mod 9 = **2** → -5.9 + **2**

3.2 Pembagi persekutuan terbesar dan bilangan prima relatif

Pembagi persekutuan terbesar (*Greatest Common Divisor*) ditulis dengan notasi gcd(a,b)=n. Contoh pembagi persekutuan terbesar dari 60 dan 24 adalah gcd(60,24)=12. Pada gcd(a,b)=n, a dan b disebut bilangan prima relatif apabila gcd(a,b)=1. Contoh gcd(15,28)=1, maka bilangan 15 dan 28 adalah bilangan prima relatif[7].

Untuk mencari gcd(a,b) dengan nilai a dan b yang cukup besar dapat menggunakan algoritma Euclid. Cara kerja algoritma ini menggunakan perhitungan modulo, yaitu:

$r_1 = a \bmod b$
$r_2 = b \bmod r_1$
$r_3 = r_1 \bmod r_2$
$r_4 = r_2 \bmod r_3$
$r_i = r_{i-2} \bmod r_{i-1}$

Sampai dengan $r_i=0$, maka gcd(a,b) adalah r_{i-1}.

Contoh:
gcd(1812,1572) adalah:
240=1812 mod 1572
132=1572 mod 240
108=240 mod 132
24=132 mod 108
12=108 mod 24
0=24 mod 12, maka gcd(1812,1572)= **12**

3.3 Modulo invers

Modulo invers ditulis dengan notasi a^{-1} mod n atau (a.x) mod n = 1, yaitu mencari nilai x yang sesuai[8].

Contoh:
a. 79^{-1} mod 3220=1019, karena (79 . 1019) mod 3220 =1
b. 2^{-1} mod 14, tidak mempunyai solusi.

Apabila a dan n bukan bilangan prima relatif maka modulo invers tidak mempunyai solusi. Untuk nilai n bilangan prima maka setiap bilangan dari 1 sampai dengan n-1 adalah bilangan prima relatif terhadap n dan memiliki sebuah modulo invers pada bilangan diantara 1 sampai dengan n-1.

Algoritma Euclid ekstended digunakan untuk mencari a^{-1} mod n atau modulo invers (a,n). Cara kerja algoritma ini adalah sebagai berikut:

1. Pada saat i=0 dan i=1, berilah harga awal untuk g,u,v yaitu: $g_0=n$ $u_0=1$ $v_0=0$ $g_1=a$ $u_1=0$ $v_1=1$
2. Pada saat i=2 dan seterusnya, lakukan perhitungan berulang berikut ini:

$y_i = g_{i-2} \text{ div } g_{i-1}$
$u_i = u_{i-2} - y_i.u_{i-1}$
$v_i = v_{i-2} - y_i.v_{i-1}$
$g_i = u_i.n + v_i.a$ atau $g_{i-2} \bmod g_{i-1}$, sampai dengan $g_i=0$, maka:

- hasil a^{-1} mod n: v_{i-1}, jika v_{i-1} bernilai positip
- hasil a^{-1} mod n: $n+v_{i-1}$, jika v_{i-1} bernilai negatip

Contoh: modulo invers (299,323)

i	y	g	u	v
0	-	323	1	0
1	-	299	0	1
2	1	24	1	-1
3	12	11	–12	13
4	2	2	25	-27
5	5	1	–137	**148**
6	2	0	299	-323

Maka 299^{-1} mod 323 adalah **148**, karena (**148** * 299) mod 323 = 1

3.4 Fungsi Totient Euler

Fungsi Totient Euler dari n ditulis dengan notasi $\phi(n)$, yaitu bilangan bulat positif yang kurang dari n dan juga bilangan prima relatif terhadap n[9]. Pada pemakaian algoritma RSA, n berupa bilangan prima. Untuk sebuah bilangan prima p maka: $\phi(p)=(p-1)$.

Jika diketahui dua buah bilangan prima p,q, dimana n=pq, maka: $\phi(n)= \phi(pq)= \phi(p)X\ \phi(q)=(p-1)X(q-1)$

3.5 Bilangan prima

Bilangan prima n adalah bilangan bulat yang lebih besar dari 1 dan hanya dapat difaktorkan oleh bilangan 1 dan bilangan n itu sendiri. Demi alasan keamanan maka algoritma RSA untuk aplikasi di dunia industri menggunakan bilangan prima yang cukup besar yaitu 100 digit atau lebih[6].

4. Kriptografi kunci publik

Kriptografi bertujuan untuk mengamankan data atau menjaga kerahasiaan informasi. Proses enkripsi (*encrypt*) adalah mengubah data biasa (*plaintext*) menjadi data sandi (*ciphertext*), sedangkan proses dekripsi (*decrypt*) adalah sebaliknya. Bentuk *Ciphertext* diperlukan pada saat proses menyimpan atau mengirim data[10].

Kriptografi kunci publik bersifat asimetris, yaitu kunci yang digunakan untuk proses enkripsi dan dekripsi adalah berbeda namun bersifat berpasangan [9]. Pasangan kunci ini dibuat oleh penerima data. Proses enkripsi dilakukan pengirim menggunakan kunci publik (*public key*), sedangkan proses dekripsi dilakukan penerima menggunakan kunci privat (*private key*). Kunci publik boleh dimiliki siapapun yang ingin mengirimkan data, sedangkan kunci privat tetap dirahasiakan oleh pembuatnya yang tidak lain adalah pihak penerima data. Beberapa contoh algoritma jenis ini adalah: ElGamal, RSA, dan Elliptic Curve Cryptosystems (ECC).

4.1 Algoritma RSA

Cara kerja algoritma ini adalah pada *enkripsi* menggunakan rumus $C = M_a^e$ *mod n*, dan pada dekripsi menggunakan *rumus* $M_b = C^d$ mod n. M merupakan *plaintext* dan C sebagai *ciphertext*, yang disusun dari blok data bilangan bulat dengan nilai [1 sampai dengan *n*-1], karena proses enkripsi dan dekripsi merupakan hasil perhitungan *modulo*-n. Proses yang benar akan menghasilkan $M_b = M_a$, untuk selanjutnya M_a dan M_b disebut M.

Aturan yang harus dipenuhi agar algoritma di atas dapat berfungsi dengan baik, yaitu:

a. Nilai e dan n diketahui oleh pihak pengirim, menjadi kunci publik = {e,n}
b. Nilai d dan n hanya diketahui oleh penerima, menjadi kunci privat = {d,n}

Nilai n, e, dan d, diperoleh dengan cara[9]:

a. Memilih dua bilangan prima p,q, dimana ($p \neq q$)
b. $n = p.q$ dan $m = \phi(n)=(p-1)X(q-1)$
c. Memilih e, e = [gcd(m,e)=1 dimana $1 < e < m$]
d. $d \equiv e^{-1}$ *mod m*

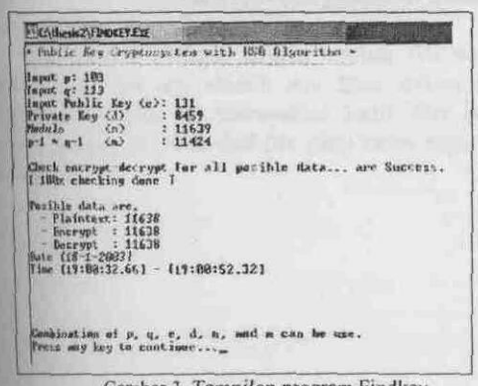
Nilai p, q, m, dan d, hanya diketahui oleh pengirim.

5. Prototype Program

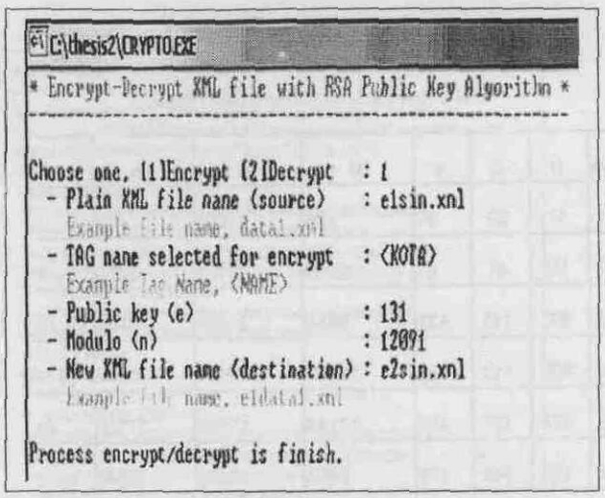
5.1 Program Findkey dan Crypto

Tampilan program Findkey dan Crypto dapat dilihat pada gambar 3 dan gambar 4. Program Findkey menghasilkan pasangan kunci publik dan privat berdasarkan nilai input p, q, dan e. Program ini melakukan pengujian terhadap seluruh kemungkinan *plaintext* yang dapat diproses (*code breaking*) yaitu bilangan integer 1 sampai dengan n-1. Berikut ini adalah contoh enkripsi dan dekripsi, diketahui:

p=47; q=71; e=79; m=(p-1).(q-1)=46.70 = 3220;
n=p.q = 47*71=3337; $d = 79^{-1}$ mod 3220 = 1019;
maka: *Plaintext* = 688, *Ciphertext* = 688^{79} mod 3337 = 1570. Proses dekripsi: 1570^{1019} mod 3337 = 688.



Gambar 3. Tampilan program Findkey



Gambar 4. Tampilan program Crypto

Program Crypto berguna untuk proses enkripsi dan dekripsi isi *tag* yang dipilih dari file XML. Hasil dari program ini berupa file XML dengan data terenkripsi pada *tag* tertentu. Proses enkripsi dapat dilakukan kembali untuk mengenkripsi data pada *tag* yang lain menggunakan pasangan kunci yang baru.

5.2 Kemungkinan Data yang Dapat Diproses

Jumlah kemungkinan data yang dapat diproses pada penelitian ini berasal dari bilangan prima (p,q), sebesar 3 digit, yaitu p = 2..997 dan q = 2.. 991. Maksimum nilai n = p.q = 988027, sehingga nilai data yang mungkin adalah 1..(n-1) = 1..988026. Rentangan nilai e diperoleh dari, m = (p-1).(q-1) = 996.990=986040 maka e = 1 < e < m = 2..986039.

Dari penjelasan di atas maka jumlah kemungkinan data yang dapat dienkripsi adalah data bernilai 1 sampai dengan 988026 (lebar 6 digit). Nilai data maksimum dengan lebar 6 digit adalah 999999, apabila nilai *plaintext* yang dienkripsi antara 988027 sampai dengan 999999 maka kesalahan akan terjadi pada proses dekripsi. Proses dekripsi tidak dapat mengembalikan *ciphertext* sama dengan *plaintext*. Hal ini diakibatkan oleh pengerjaan modulo n pada proses enkripsi dan dekripsi, dimana hasilnya tidak akan pernah $\geq$ n.

5.3 Kombinasi kunci publik dan kunci privat

Pada tabel 1 dapat dilihat pasangan kunci publik dan kunci privat yang dihasilkan dari program Findkey. Kolom t adalah waktu dalam detik yang mencatat lamanya proses simulasi *code breaking* menggunakan komputer prosesor *Intel Pentium III*-800 MHz dengan RAM 128 Mb.

Tabel 1. Pasangan kunci publik dan kunci privat

No.	p	q	e	d	n	m	t
1	17	29	37	109	493	448	0.11
2	59	41	67	2043	2419	2320	1.1
3	103	113	131	8459	11639	11424	19.66
4	107	113	131	11147	12091	11872	21.6
5	127	137	109	14149	17399	17136	43
6	127	149	109	2053	18923	18648	16.5
7	157	173	257	4385	27161	26832	26
8	157	173	509	17765	27161	26832	104
9	467	223	5501	99089	104141	103452	3021
10	503	607	103	200839	305321	304212	5521
11	1009	1021	4001	42401	1030189	1028160	5713
12	4597	2179	6143	6916583	10016863	10010088	-

Data dari tabel 1 baris 12, pada kolom t tidak tercatat karena keterbatasan kemampuan program Findkey untuk melakukan simulasi *code breaking*. Waktu yang dibutuhkan untuk simulasi *code breaking* dipengaruhi oleh nilai p, q, dan e, yaitu semakin besar nilainya maka semakin lama waktu yang dibutuhkan.

5.4 Konversi data

Pengubahan bentuk data atau konversi data perlu dilakukan pada saat proses enkripsi dan dekripsi. Data pada tag XML adalah berbentuk sekumpulan karakter sedangkan algoritma RSA dapat bekerja pada data dalam bentuk numerik. Proses konversi dapat dilakukan dengan cara mengubah data karakter menjadi bentuk kode ASCII (*American Standard Code for Information Interchange*).

Besarnya data numerik yang dapat dienkripsi dan didekripsi terpengaruh oleh besar kecilnya nilai p dan q. Seperti telah diketahui nilai p dan q digunakan untuk membentuk nilai mod(n). Semakin besar nilai p dan q maka semakin besar nilai mod(n), hal ini berakibat pada semakin besar data numerik yang dapat dienkripsi dan didekripsi. Dengan nilai p dan q yang besar yaitu 100 digit atau lebih maka proses *code breaking* menjadi sangat lama untuk dilakukan.

6. Hasil uji coba

6.1 Mengecek hasil enkripsi dekripsi

Hasil proses enkripsi dan dekripsi file XML adalah benar apabila:

a. Kunci publik dan kunci privat yang digunakan harus merupakan pasangannya.
b. Proses enkripsi dikerjakan dengan benar.
c. Isi data XML yang terenkripsi tidak terjadi perubahan atau kerusakan data.
d. Proses dekripsi dikerjakan dengan benar.

Sebagai alternatif untuk mengurangi tingkat kesalahan maka pada file data XML yang terenkripsi dapat ditambahkan informasi berikut, yaitu: kunci publik yang digunakan, besarnya ukuran file XML sebelum dan sesudah dienkripsi.

6.2 Usulan Aplikasi

Hasil penelitian ini dapat dijadikan usulan untuk pembuatan program aplikasi yang lebih baik dan interaktif. Program aplikasi tersebut berfungsi untuk proses distribusi data XML yang telah dienkripsi. Contoh: sebuah perusahaan memiliki empat bagian dan masing-masing mencatat datanya di dalam format data XML. Terdapat 10 jenis *tag* XML, dimana masing-masing bagian membutuhkan data dari *tag* yang berbeda sesuai dengan bagiannya. Isi *tag* data disimpan oleh pusat data dalam keadaan terenkripsi, hal ini perlu dilakukan dengan pertimbangan pada keamanan isi data. Teknik enkripsi dilakukan menggunakan kunci publik. Setiap *tag* dienkripsi menggunakan kunci publik yang berbeda. Kunci privat yang menjadi pasangannya hanya dimiliki oleh setiap bagian yang memang berhak untuk mengakses data. Pembagian nama *tag* dan bagian yang memerlukannya dapat dilihat pada tabel 2.

Tabel 2. Distribusi data XML dan bagian yang memerlukan

DATA YANG DISIMPAN OLEH PUSAT DATA		
Nama Tag (Plaintext)	Jumlah Tag (Ciphertext)	Bagian yang Membutuhkan
tag a	2	II,III
tag b	2	I,II
tag c	2	I,II
tag d	2	I,II
tag e	3	I,II,IV
tag f	3	I,II,IV
tag g	1	IV
tag h	1	IV
tag i	2	III,IV
tag j	1	III

NAMA BAGIAN DAN DATA TAG YANG BOLEH DIAKSES			
BAGIAN I	BAGIAN II	BAGIAN III	BAGIAN IV
tag b	tag a	tag a	tag e
tag c	tag b	tag i	tag f
tag d	tag c	tag j	tag g
tag e	tag d		tag h
tag f	tag e		tag i
	tag f		

Keterangan:

a. Data tag a sampai dengan tag j didistribusikan dalam bentuk *ciphertext*.

b. Setiap bagian memiliki sepasang kunci privat dan kunci publik.

c. Kunci publik digunakan oleh bagian penyimpan data untuk mengenkripsi tag.

d. Kunci privat disimpan masing-masing bagian.

Keuntungan dari sistem ini adalah proses distribusi data menjadi lebih aman karena data dalam keadaan terenkripsi dan tanggung jawab kerahasiaan data ada pada masing-masing bagian. Setiap bagian hanya dapat mengakses data yang ditujukan bagi bagiannya, dengan cara melakukan proses dekripsi menggunakan kunci privat yang dimilikinya.

Pada gambar 5 dapat dilihat contoh file simpanan (sim.xml) dengan *tag alamat dan kota* dalam kondisi terenkripsi. Isi file ini merupakan hasil dari program Crypto. Bentuk *plaintext* dari file yang sama dapat dilihat pada gambar 1.


```
<?xml version="1.0"?>
<!-- Name File : e2sim.xml -->
<!-- Keterangan: File data simpanan deposito dengan isi tag -->
<!--        <ALAMAT> dan <KOTA> dalam kondisi terenkripsi -->
<?xml-stylesheet type="text/xsl" href="xslsim.xsl"?>
<SIMPANAN>
 <DATABANK>
  <NAMABANK>BANK SEJAHTERA</NAMABANK>
  <PRODUK>DEPOSITO RUPIAH</PRODUK>
  <CABANG>SEMARANG TIMUR</CABANG>
  <ALAMATCABANG>JL.ARTERI TIMUR 103-105
   </ALAMATCABANG>
  <KOTACABANG>SEMARANG</KOTACABANG>
  <TANGGAL>1 MARET 2002</TANGGAL>
 </DATABANK>
 <DATANASABAH>
  <NAMANASABAH>DRS. SLAMET SUPRAPTO
   </NAMANASABAH>
  <NOMORBILYET>050210123456</NOMORBILYET>
  <NOREKENING>050230747424</NOREKENING>
  <NOMINALRP>25000000</NOMINALRP>
  <BUNGAPERTAHUN>13</BUNGAPERTAHUN>
  <ALAMAT>ˆ…]P,ŽJSaT‰7+y;†K+Mx‰5{?c‰h?€4,g+Il˜Og8o†?q
   e?‰□,˜Y-Š+Pl+D^5T…il7j‹1^2†Šfh5u†x,.;‡P];E</ALAMAT>
  <KOTA>ˆ…]k\-†…SKŒ,,‰],g+6Y‡t?k}†k]Rl,g+AN-Š+…b
   </KOTA>
  <PERIODEBULAN>6</PERIODEBULAN>
  <TGLMASUK>10 JANUARI 2002</TGLMASUK>
  <TGLJATUHTEMPO>10 JULI 2002</TGLJATUHTEMPO>
 </DATANASABAH>
 <DATANASABAH>
  <NAMANASABAH>GUNAWAN
WIBISONO</NAMANASABAH>
  <NOMORBILYET>050210332660</NOMORBILYET>
  <NOREKENING>050230122500</NOREKENING>
  <NOMINALRP>50000000</NOMINALRP>
  <BUNGAPERTAHUN>14</BUNGAPERTAHUN>
  <ALAMAT>‡Fq.□…Ph2†‰O]MUŠVICxŽT]2†‡@gAyŒz…9_+a
   {3VŽ‹S□[,~IE}Š[]WY‡@g:?˜2?FJ†^…Hg□^h5T</ALAMAT>
  <KOTA>†V…Up‰<?2gŽ}h‰}</KOTA>
  <PERIODEBULAN>12</PERIODEBULAN>
  <TGLMASUK>15 JANUARI 2002</TGLMASUK>
  <TGLJATUHTEMPO>15 JANUARI 2003</TGLJATUHTEMPO>
 </DATANASABAH>
```


Gambar 5. File sim.xml dengan *tag* alamat dan kota dalam kondisi terenkripsi

7. kesimpulan

Dari hasil penelitian yang telah dilakukan diperoleh kesimpulan, yaitu:

a. Data XML dapat dienkripsi menggunakan kriptografi kunci publik dengan algoritma RSA. Enkripsi dapat dilakukan untuk nama tag tertentu yang dipilih.

b. Program Findkey dapat digunakan untuk mencari pasangan kunci publik dan privat, dengan nilai p dan q sebesar 2, 3, atau 4.

c. Program Crypto dapat melakukan fungsi enkripsi pada data karakter setelah dikonversi terlebih dahulu menjadi blok data numerik. Panjang satu blok data numerik sebesar 4 digit.

d. Waktu yang dibutuhkan untuk simulasi *code breaking* oleh program Findkey dipengaruhi oleh nilai p, q, dan e. Semakin besar nilai p, q, dan e maka semakin lama waktu yang dibutuhkan.

e. Hasil penelitian ini dapat dijadikan usulan untuk pembuatan program aplikasi yang berfungsi untuk proses distribusi data XML yang telah dienkripsi.

Referensi

[1] Airi Salminen, Frank Wm. Tompa, 1998, *Requirements for XML Document Database Systems*, Department of Computer Science and Information Systems University of Jyvaskyla Finland and Department of Computer Science University of Waterloo Canada.

[2] W3C Recommendation 6 October 2000, "Extensible Markup Language (XML) 1.0", Second Edition,*http://www.w3.org/tr/rec-xml*.

[3] Pontus Norman, 1999, *a Study of eXtensible Markup Language (XML)*, theses master of science degree in computer science, Department of Teleinformatics, Royal Institute of Technology, Sweeden.

[4] Michael J.Y., alih bahasa: Imam Mustaqim,2001, *XML Step by Step*. Elex Media Komputindo, Jakarta.

[5] Murdoch Mactaggart, 2001, "An Introduction to XML Encryption and XML Signature", IBM Developer Works Enabling XMLSecurity, http://www-106.ibm.com/developerworks/library/s-xmlsec.html

[6] RSA Laboratories 2000, "RSAES-OAEP Encryption Scheme, Algorithm specification and Supporting documentation", RSA Security Inc. Bedford, MA 01730 USA. http://www.rsalabs.com

[7] Bruce Schneier, 1996, *Applied Cryptography, Protocols, Algorithms, and Source Code in C*. John Wiley & sons, Inc.

[8] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, 1997, *Handbook of Applied Cryptography*. CRC Press LLC, Florida, USA.

[9] William Stallings, 1999, *Cryptography and Network Security Principles and Practice second edition*. Prentice Hall, New Jersey, USA.

[10] Bruce Schneier, 1999, "Security in the Real World: How to Evaluate Security Technology", *Computer Security Journal, Volume XV, Number 4, 1999*.

Budi Hartono, Pendidikan: S1 Teknik Informatika Universitas Kristen Duta Wacana Yogyakarta (1997), S2 Ilmu Komputer Universitas Gadjah Mada Yogyakarta (2003). Pekerjaan mengajar di Teknik Elektronika dan Teknik Informatika, Universitas Stikubank Semarang. Minat Penelitian: Kriptografi, Kompresi Data, dan Kecerdasan Buatan

Jazi Eko Istiyanto, memperoleh gelar sarjana ilmu fisika (1986) dari Universitas Gadjah Mada, Yogyakarta, Indonesia. Kemudian ia mengikuti studi lanjut di Inggris dan memperoleh *Postgraduate Diploma in Computer Programming and Microprocessor Applications* (1987), *M.Sc in Computer Science* (1988), dan *Ph.D in Electronic Systems Engineering* (1995), semuanya dari University of Essex. Kini ia Ketua Program Studi Elektronika dan Instrumentasi pada Jurusan Fisika, dan mengajar pada Program Magister Ilmu Komputer serta Program Magister Ilmu Fisika, Universitas Gadjah Mada. Minat risetnya: keamanan informasi, layanan web, dan sistem tertanam (*embedded systems*). Dr. Jazi adalah anggota Himpunan Fisika Indonesia dan Persatuan Insinyur Indonesia.