Perbandingan Tanda Tangan Digital RSA dan DSA Serta Implementasinya untuk Antisipasi Pembajakan Perangkat Lunak

Mohamad Ray Rizaldy – NIM : 13505073

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15073@students.if.itb.ac.id

Abstrak

Makalah ini membahas penggunaan tanda tangan digital untuk pengantisipasian pembajakan perangkat lunak. Salah satu cara dalam pengantisipasian pembajakan perangkat lunak adalah dengan pemberian lisensi aplikasi kepada pemilik aplikasi. Lisensi ini diperiksa sedemikian rupa sehingga hanya pemegang resmi dari perangkat lunak yang bisa menggunakan aplikasi. Selama ini cara yang diterapkan adalah melalui internet. Namun kelemahannya adalah tidak semua pengguna aplikasi terhubung dengan internet. Jika pengecekan dilakukan di komputer pengguna, perlu dibuat supaya lisensi aplikasi tidak dapat dibongkar. Salah satu caranya adalah dengan menggunakan tanda tangan digital.

Ada beberapa metode yang digunakan dalam penandatanganan secara digital. Dua metode yang umum digunakan adalah penandantangan digital dengan RSA (*Rivest-Shamir-Adleman*) dan DSA (*Digital Signature Algorithm*). Dari dua metode ini akan dipilih metode mana yang paling cocok untuk lisensi aplikasi.

Dengan menggunakan lisensi aplikasi yang berformat dokumen XML, pihak pembuat ataupun distributor perangkat lunak bisa mengatur hanya pemegang resmilah yang bisa menggunakan perangkat lunak.

Kata kunci : *tanda tangan digital, lisensi aplikasi, XML*

1. Pendahuluan

Sejak dulu untuk memeriksa keabsahan sebuah dokumen cetak, orang-orang menggunakan tanda tangan. Kini setelah memasuki era tanpa kertas (paperless era), menggunakan cara yang mirip, diperkenalkanlah teknologi tanda tangan digital.

Prinsip yang sama juga bisa diterapkan untuk memeriksa keabsahan sebuah perangkat lunak. Selama ini, banyak pengembang aplikasi yang dihantui oleh pembajakan hasil kekayaan intelektual mereka. Biasanya untuk mengurangi pembajakan ini dilakukan pemberian lisensi aplikasi. Namun sayangnya cara yang ada untuk memeriksa lisensi cenderung mahal dan sulit diimplementasi. Sebagai contoh, salah satu cara adalah pemeriksaan lisensi melalui internet mungkin mudah diimplementasi, namun di sisi lain tidak semua pengguna aplikasi terhubung melalui internet.

Cara kedua yang bisa dilakukan adalah dengan pengecekan di sisi komputer pengguna aplikasi (klien). Namun jelas cara ini sangat mudah diserang. Untuk itulah prinsip tanda tangan digital diterapkan, untuk menangkis serangan di sisi klien tersebut. Selanjutnya makalah ini akan membahas bagaimana tanda tangan digital diimplementasikan untuk pengecekan lisensi aplikasi.

2. Fungsi Hash

Fungsi *hash* atau sering juga disebut sebagai *cryptographic cheksum* merupakan fungsi yang mentransformasikan masukan string dengan panjang sembarang menjadi sebuah string lain dengan panjang yang tetap. Hasil keluaran dari fungsi ini umumnya berukuran jauh lebih kecil dari masukan awalnya. Keluaran dari fungsi *hash* disebut dengan nilai *hash* atau *message digest*.

Jika dituliskan dalam notasi matematis akan jadi seperti:

$$MD = H(M)$$

Oleh fungsi *hash* sebuah string yang berukuran apapun diubah menjadi *message digest* yang berukuran tetap (128-512 bit).

Adapun sifat-sifat yang dimiliki oleh fungsi *hash* adalah sebagai berikut :

- Fungsi H dapat diterapkan pada blok data yang berukuran berapa saja.
- Nilai hash yang dihasilkan memiliki panjang yang tetap.
- Untuk setiap h yang diberikan, tidak mungkin menemukan suatu x sedemikian sehingga H(x)=h. Fungsi H tidak dapat mengembalikan nilai *hash* menjadi masukan awal.
- Untuk setiap x yang diberikan, tidak mungkin mencari pasangan $x \neq y$ sedemikian sehingga H(x)=H(y).

Ada dua jenis fungsi *hash* yang sering digunakan hingga sekarang. Fungsi *hash* yang pertama adalah MD5. Fungsi MD5 dibuat oleh Ron Rivest pada tahun 1994. Nilai *hash* yang dihasilkan oleh MD5 berukuran 128 bit. Fungsi *hash* yang lain adalah SHA (*Secure Hash Algorithm*) dikembangkan oleh NIST sebagai spesifikasi SHS (*Secure Hash Standard*). SHA sendiri memiliki banyak varian. Salah satunya adalah varian SHA-1. Varian SHA-1 ini menghasilkan nilai *hash* yang berukuran 160 bit.

Dalam makalah ini, untuk mempermudah, fungsi *hash* yang akan dipakai adalah algortima SHA-1.

3. Tanda Tangan Digital

Sebagai sebuah alat pemeriksa keabsahan sebuah tanda tangan secara garis besar memiliki karakteristik sebagai berikut:

- Merupakan bukti yang otentik.
- Tidak dapat dilupakan.
- Tidak dapat dipindahtangankan.
- Tidak dapat disangkal.
- Dokumen yang telah ditandatangani tidak dapat berubah.

Fungsi tanda tangan pada dokumen kertas diterapkan untuk otentikasi pada data digital seperti pesan yang dikirim melalui saluran komunikasi dan dokumen elektronis yang disimpan di dalam memori komputer. Tanda tangan pada data digital ini dinamakan tanda tangan digital (*digital signature*).

Perlu ditekankan bahwa tanda tangan digital bukanlah tanda tangan manual (pada dokumen cetak) yang didigitasi dengan alat pemindai. Tanda tangan digital sebenarnya adalah suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan. Oleh karena itu tanda tangan digital seorang akan berbeda pada dua buah dokumen atau pesan yang berbeda.

Pembubuhan tanda tangan digital pada sebuah dokumen, umumnya menggunakan *hash*. Dokumen yang hendak dikirim diubah terlebih dahulu menjadi bentuk yang ringkas yang disebut *message digest* MD.

Selanjutnya, *message digest* MD dienkripsi dengan algoritma kriptografi kunci-publik menggunakan kunci rahasia (SK) milik penandatangan atau pengirim dokumen. Hasil dari enkripsi inilah yang kita sebut sebagai tanda tangan digital (*digital signature)* S.

Dokumen M dan tanda tangan digital S kemudian dikirim melalui saluran komunikasi. Dalam hal ini, dokumen M telah ditandatangani oleh pengirim dengan tanda tangan S. Sebuah tanda tangan digital bisa dilekatkan (*embedded*) pada sebuah dokumen atau bisa juga disimpan di dokumen yang terpisah.

Setelah sampai di penerima, dokumen diverifikasi untuk dibuktikan keabsahannya. Adapun caranya adalah dengan mendekripsi tanda tangan digital S dengan kunci publik PK milik pengirim menjadi *message digest* MD kembali. Jika *message digest* hasil dekripsi cocok berarti dokumen yang diterima valid atau terbukti keabsahannya.

Dalam penandatanganan secara digital ini ada dua standar yang umum digunakan ketiga standar tersebut adalah :

- RSA (Rivest-Shamir-Adleman)
- DSA (Digital Signature Algorithm)

3.1 Tanda Tangan Digital Menggunakan RSA

RSA adalah salah satu algoritma kriptografi dengan kunci publik. Saat ini RSA bisa dikatakan sebagai algoritma kunci publik paling populer. Nama RSA sendiri diambil dari singkatan nama-nama penemu algoritmanya, yaitu Ron Rivest, Adi Shamir dan Leonard Adleman.

Dalam melakukan enkripsi dan dekripsi secara umum algoritma RSA memiliki besaran-besaran sebagai berikut :

- p dan q, bilangan prima rahasia
- $n = p \cdot q$, tidak rahasia

- $\Phi(n) = (p - 1) \cdot (q - 1)$, rahasia
- SK, kunci privat rahasia
- PK, kunci publik tidak rahasia.
- M, pesan yang dienkripsi.

Kunci publik PK dalam RSA merupakan hasil pembangkitan bilangan secara acak dari pencarian bilangan yang relatif prima terhadap Φ(n). Sedangkan kunci privat SK dibangkitkan dengan menggunakan persamaan

$$PK.SK = 1 \left(mod\ \Phi(n)\right)$$

Dalam enkripsi biasa pesan dienkripsi dengan kunci publik baru didekripsi dengan kunci privat. Namun pada praktek tanda tangan digital digunakan sebaliknya.

Secara ringkas, pembubuhan tangan digital dengan RSA adalah sebagai berikut:

1. Pengirim menghitung nilai *hash* dari pesan *M* yang akan dikirim, misalkan nilai *hash* dari M adalah MD.
2. Pengirim mengenkripsi MD dengan kunci privatnya menggunakan persamaan enkripsi *RSA*.

Sedangkan untuk verifikasi tanda tangan digital :

1. Penerima menghitung nilai *hash* dari pesan *M* yang akan dikirim, misalkan nilai *hash* dari *M* adalah MD'.
2. Penerima melakukan dekripsi terhadap tanda-tangan *S* dengan kunci publik si pengirim.
3. Penerima membandingkan MD dengan MD'. Jika MD = MD' maka dokumen beserta tanda-tangan digitalnya adalah otentik. Jika sebaliknya, berarti dokumen atau pengirimnya dinyatakan tidak valid.

3.2 Tanda Tangan Digital Menggunakan DSA

DSA (*Digital Signature Algorithm*) merupakan pengembangan dari algoritma ElGamal. Pada DSA, algoritma signature dan verifikasi berbeda.

Untuk penandatanganan sebuah dokumen, DSA memiliki parameter sebagai berikut :

- p, bilangan prima dengan panjang L bit. L adalah kelipatan 64 dan $512 \leq L \leq 1024$. p tidak dirahasiakan.
- q, bilangan prima 160 bit. q tidak dirahasiakan.
- $g = h^{(p-1)/q} \bmod p$, $h < (p - 1)$. Parameter g tidak dirahasiakan.
- x, kunci privat. x bilangan bulat < q
- $y = g^x \bmod p$, kunci publik.
- M, pesan yang akan ditanda tangani.

Untuk membangkitkan kunci pada DSA, langkahnya adalah sebagai berikut :

1. Pilih bilangan p dan q yang prima, dimana $(p - 1) \bmod q = 0$.
2. Hitung $g = h^{(p-1)/q} \bmod p$. h merupakan nilai dalam *range* $1 < h < (p - 1)$ serta harus memenuhi syarat $h^{(p-1)/q} \bmod p > 1$.
3. Acak kunci privat x dengan syarat x < q.
4. Hitung kunci publik y, dimana $y = g^x \bmod p$.

Setelah kunci dibangkitkan, langkah untuk pembubuhan tanda tangan digitalnya adalah sebagai berikut :

1. Ubah dokumen M menjadi nilai *hash* MD.
2. Tentukan bilangan acak k, dimana k < q.
3. Hitung nilai r dan s sebagai tanda tangan.
   $r = (g^k\ mod\ p) mod\ q$ dan
   $s = \left(k^{-1}\ (MD + x * r)\right) mod\ q$

Adapun untuk verifikasi dokumen dan tanda tangan digital dilakukan langkah-langkah sebagai berikut :

1. Hitung nilai *hash* MD dari dokumen M.
2. Hitung w, dimana $w = s^{-1}\ mod\ q$.
3. Hitung u_1 dan u_2, dimana
   $u_1 = (MD * w) mod\ q$ dan
   $u_2 = (r * w) mod\ q$.
4. Hitung v, $v = \left((g^{u_1} * y^{u_2}) mod\ p\right) mod\ q$.
5. Jika v = r artinya dokumen beserta tanda tangan digitalnya valid.

3.3 Perbandingan DSA dan RSA

Perbandingan kecepatan DSA dan RSA dengan menggunakan kunci berukuran 512 bit diperlihatkan pada tabel di bawah :

Jenis	Penandatanganan (dalam 100 ms)	Verifikasi (dalam 100 ms)
RSA	915	160
DSA	634	988

Dari tabel tersebut bisa disimpulkan:

- Penandatanganan secara digital dengan DSA lebih cepat dibandingkan RSA.
- Verifikasi dengan RSA memiliki kecepatan yang jauh lebih cepat dibanding DSA.

Jika dilihat dari ukuran tanda tangan, RSA unggul. Ukuran RSA lebih unggul karena ukuran tanda tangan pada RSA adalah sebesar *message digest* dari dokumen. Sedangkan ukuran tanda tangan pada DSA adalah dua kali lipatnya.

Dari aspek-aspek tersebut yang paling cocok dipakai untuk lisensi aplikasi adalah RSA. Karena seperti dijelaskan sebelumnya, cara penandatanganan digital adalah untuk

mempermudah pengecekan lisensi aplikasi, dimana ukuran dan kecepatan verifikasi yang diunggulkan.

4. XML

XML adalah singkatan dari eXtensible Markup Language. XML digunakan pada dokumen yang mengandung informasi terstruktur. Sebuah bahasa markup sebenarnya adalah mekanisme untuk mengindentifikasi struktur dalam sebuah dokumen.

Adapun bentuk dokumen XML contohnya adalah sebagai berikut:

```
<book>
  <title>Night Fall</title>
  <author>Demille, Nelson</author>
  <publisher>Warner</publisher>
  <price>$26.95</price>
  <contentType>Fiction</contentType>
  <format>Hardback</format>
  <isbn>0446576638</isbn>
</book>
```

4.1 Tanda Tangan XML

Sebuah tanda tangan XML merupakan objek kriptografi yang kompleks. Sintak untuk tanda tangan digital telah didefinisikan oleh organisasi W3C secara resmi.

Tanda tangan digital dengan XML ini yang dianggap sebagai solusi untuk verifikasi lisensi aplikasi.

Dalam penandatangan digital dengan XML ada hal yang perlu diperhatikan yaitu spasi dan enter (*white spaces*). Hal ini menjadi penting karena penambahan satu *white space* saja bisa membuat *message digest* yang dihasilkan akan berbeda. Oleh karena itu, maka sebelum ditandatangani sebuah dokumen XML akan mengalami proses kanonikalasi. Proses ini akan membuang semua *white space* dalam dokumen. Setelah mengalami proses kanonikalasi, baru dihitung nilai *hash*nya dan dienkripsi menjadi tanda tangan digital.

Tanda tangan XML bisa dipisah (*detached*) atau disatukan di dalam dokumen (*enveloped*).

Berdasarkan skema yang ditetapkan oleh organisasi W3C, dokumen XML yang digunakan untuk tanda tangan digital terdiri dari:

- Informasi tanda tangan
- Tanda tangan digital
- Informasi kunci

Berikut contoh tanda tangan digital XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
   <Reference URI="http://www.w3.org/TR/xml-stylesheet">
     <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
     <DigestValue>60NvZvtdTB+7UnlLp/H24p7h4bs=</DigestValue>
   </Reference>
  </SignedInfo>
  <SignatureValue>
juS5RhJ884qoFR8flVXd/rbrSDVGn40CapgB7qeQiT+rr0NekEQ6BHhUA8dT3+BC
TBUQI0dBjlml9lwzENXvS83zRECjzXbMRTUtVZiPZG2pqKPnL2YU3A9645UCjTXU
+jgFumv7k78hieAGDzNci+PQ9KRmm//icT7JaYztgt4=
  </SignatureValue>
  <KeyInfo>
    <KeyValue>
      <RSAKeyValue>
        <Modulus>
uCiukpgOaOmrq1fPUTH3CAXxuFmPjsmS4jnTKxrv0w1JKcXtJ2M3akaV1d/karvJ
lmeao20jNy9r+/vKwibjM77F+3bIkeMEGmAIUnFciJkR+ihO7b4cTuYmEi8xHtu4
iMn6GODBoEzqFQYdd8p4vrZBsvs44nTrS8qyyhba648=
        </Modulus>
        <Exponent>AQAB</Exponent>
      </RSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

4.2 Lisensi Aplikasi Dengan XML

Saat pertama kali seseorang membeli sebuah perangkat lunak secara resmi, pihak distributor perangkat lunak bisa meminta informasi komputer pembeli. Informasi ini adalah informasi yang unik misalnya seperti *MAC address, HD Serial Number* atau *SID* komputer.

Dari informasi tersebut, distributor membuatkan sebuah dokumen XML berisi informasi tadi ditambah waktu kadaluarsa. Dokumen kemudian ditanda-tangani dengan kunci privat milik distributor. Waktu kadaluarsa menjadi penting karena biasanya lisensi perangkat lunak hanya berlaku dalam jangka waktu tertentu.

Di dalam dokumen disertakan kunci publik untuk verifikasi di komputer pembeli. Saat verifikasi akan dicek apakah informasi komputernya sesuai dengan yang ada pada dokumen XML. Jika hasil verifikasi valid berarti komputer benar adalah komputer pemegang resmi lisensi aplikasi tersebut dan belum melewati masa kadaluarsa.

Jika hasil verifikasi tidak valid, dibuat mekanisme sehingga perangkat lunak tidak dapat digunakan. Dengan begitu perangkat lunak akan aman dari pembajakan.

5. Implementasi

5.1 Lingkungan Implementasi

Spesifikasi komputer yang digunakan dalam melakukan implementasi ini sebagai berikut:

- *Prosesor Intel Core Duo 1.66GHz*
- *1.5 GB DDR2 RAM*
- Sistem Operasi *Windows XP SP2*
- *.NET Framework 3.0*
- *Microsoft Visual Studio 2008*
- Bahasa pemrograman *C#.NET*

5.2 Batasan Implementasi

Dalam implementasi ini, masalah akan dibatasi dengan batasan-batasan sebagai berikut :

- Implementasi berbasis *command prompt*.
- Implementasi menunjukan pembuatan dan verifikasi lisensi. Tidak ada mekanisme pengecekan ID komputer dan kadaluarsa.
- Implementasi menggunakan fungsi bawaan C#.NET untuk penandatangan XML dengan RSA dan SHA-1.
- Implementasi tidak menyertakan mekanisme perusakan perangkat lunak jika terjadi ketidakvalidan dalam verifikasi.

5.3 Pembangkitan Lisensi

```
class BuildLicense{
[STAThread]
static int Main(string[] args){
  // lisensi berupa dokumen XML
  XmlDocument XmlDoc = new XmlDocument();
  XmlDoc.Load(args[0]);
  SignedXml sxml = new SignedXml(XmlDoc);

  CspParameters parms = new CspParameters(1);
  parms.Flags=
    CspProviderFlags.UseMachineKeyStore;
  //  membuat kunci PIHAK DISTRIBUTOR
  parms.KeyContainerName = "Ganesha IT Solution";
  parms.KeyNumber = 2;
  RSACryptoServiceProvider csp =
   new RSACryptoServiceProvider(parms);
  // assign csp sebagai kunci
  sxml.SigningKey = csp;
  //metode kanonikalasi
  sxml.SignedInfo.CanonicalizationMethod =
   SignedXml.XmlDsigCanonicalizationUrl;

  // Pembuatan tanda tangan
  Reference r = new Reference("");
  r.AddTransform(new
    XmlDsigEnvelopedSignatureTransform(false));
  sxml.AddReference(r);
  sxml.ComputeSignature();
  XmlElement Sig = sxml.GetXml();

  //tambahkan kunci publik ke xml signature
  sxml.SigningKey=csp.ToXmlString(false);
  //tambahkan tanda tangan ke dokumen
  XmlDoc.DocumentElement.AppendChild(Sig);
}}
```

Di dalam kode di atas terdapat parameter `parms.KeyContainerName` yang berisi nama distributor. Harus diingat bahwa variabel ini bersifat *case sensitive*.

Selanjutnya `csp` akan membangkitkan pasangan kunci milik distributor untuk penandatanganan. Perlu diketahui bahwa nilai *hash*nya sudah dihitung secara otomatis saat deklarasi objek `sxml`. Setelah dibuat, tanda tangan dijadikan elemen anak (*appendChild*) dari xml awal.

5.4 Verifikasi Lisensi

```
class VerifyLicense{
[STAThread]
static int Main(string[] args){
  // lisensi berupa dokumen XML
  XmlDocument XmlDoc = new XmlDocument();
  XmlDoc.Load(args[0]);
  SignedXml sxml = new SignedXml(XmlDoc);

  // mengambil kunci publik yang ada di XML
  XmlNode xmlkey =
    XmlDoc.GetElementsByTagName("KeyValue");
  RSACryptoServiceProvider csp = new
    RSACryptoServiceProvider();
  csp.FromXmlString(xmlkey);

  XmlNode Signature =
    XmlDoc.GetElementsByTagName("Signature",
    SignedXml.XmlDsigNamespaceUrl)[0];
  sxml.LoadXml((XmlElement)Signature);

  // mengecek apakah lisensi benar atau salah
  if (sxml.CheckSignature(xmlkey)){
    Console.WriteLine("Aplikasi Resmi");
  }else{ Console.WriteLine("Aplikasi Bajakan");
}}}
```

Karena pada saat pembangkitan lisensi, kunci publik milik distributor telah digabungkan dalam XML, jadi saat verifikasi hanya perlu mengambil dari elemen XML yang mengandung kunci,yaitu pada `XmlDoc.GetElementsByTagName`.

Dengan fungsi bawaan yang ada pada C#.NET kunci diverifikasi. Jika hasilnya valid akan dituliskan "Aplikasi Resmi" pada konsol. Jika sebaliknya, konsol akan berisi tulisan "Aplikasi Bajakan".

6. Hasil Implementasi

Misalnya distributor yang dalam kode dituliskan sebagai "*Ganesha IT Solution*" ingin membangkitkan sebuah lisensi untuk komputer bernama *RAIECOMPUTER*. Maka distributor perlu membuat sebuah berkas XML berisi:

```
<license>
  <computerName>RAIECOMPUTER</computerName>
  <expires>2010-09-31T00:00:00</expires>
</license>
```

Simpan berkas tersebut dengan nama `contoh.xml`.

Distributor bisa membangkitkan lisensi dengan menjalankan program BuildLicense pada *command prompt*. Setelah berkas XML berada pada direktori yang sama dengan program BuildLicense.exe, ketik:

```
C:\>LISENSI\BuildLicense contoh.xml
```

Dokumen lisensi akan terbentuk dengan isi seperti di bawah ini:

```
<license>
  <computerName>RAIECOMPUTER</computerName>
  <expires>2010-09-31T00:00:00</expires>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
  <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <Reference URI="http://www.w3.org/TR/xml-stylesheet">
      <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>60NvZvtdTB+7UnlLp/H24p7h4bs=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
juS5RhJ884qoFR8flVXd/rbrSDVGn40CapgB7qeQiT+rr0NekEQ6BHhUA8dT3+BC
TBUQI0dBjlml9lwzENXvS83zRECjzXbMRTUtVZiPZG2poKPnL2YU3A9645UCjTXU
+jgFumv7k78hieAGDzNci+PQ9KRmm//icT7JaYztgt4=
    </SignatureValue>
    <KeyInfo>
      <KeyValue>
        <RSAKeyValue>
          <Modulus>
uCiukpgOaOmrq1fPUTH3CAXxuFmPjsmS4jnTKxrv0w1JKcXtJ2M3akaV1d/karvJ
lmeao20jNy9r+/vKwibjM77F+3bIkeMEGmAIUnFciJkR+ih07b4cTuYnEi8xHtu4
iMn6GODBoEzqFQYdd8p4vrZBsvs44nTrS8qyyhba648=
          </Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
<license>
```

Dokumen ini diserahkan kepada pengguna perangkat lunak beserta program verifikasinya dalam satu direktori. Selanjutnya untuk verifikasi lisensi, pengguna perangkat lunak harus menjalankan program VerifyLicense di *command prompt*. Ketik:

```
C:\>LISENSI\VerifyLicense contoh.xml
```

Jika dokumen XML – dimana terdapat informasi dan kunci publik distributor – belum mengalami perubahan maka lisensi valid, begitu juga sebaliknya.

7. Kesimpulan

i. Tanda tangan digital bisa digunakan sebagai salah satu cara untuk antisipasi pembajakan perangkat lunak, yaitu untuk pemberian lisensi aplikasi.
ii. Untuk lisensi aplikasi, tanda tangan digital yang ideal adalah menggunakan RSA. Hal ini karena RSA unggul dalam ukuran tanda tangan dan kecepatan verifikasi.
iii. Lisensi sebuah perangkat lunak bisa dibuat dalam format dokumen XML yang telah ditandatangani oleh distributor perangkat lunak.
iv. Distributor perlu memiliki informasi tentang komputer pengguna perangkat lunak untuk membuat lisensi XML. Informasi yang digunakan harus unik supaya tidak bisa digunakan di komputer lain (dibajak).
v. Untuk pengembangan lebih lanjut perlu dibuat mekanisme dimana pada saat verifikasi, informasi yang unik di komputer bisa dikenali dan waktu kadaluarsa bisa dicek. Jika hasil verifikasi valid, program akan melakukan perusakan terhadap perangkat lunak. Dengan begitu pembajakan terhadap perangkat lunak dapat diatasi.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Dykes, Lucinda. (2005). XML For Dummies 4th Edition. Wiley Publishing

[3] Dournaee, Blake. (2002). XML Security. McGraw-Hill.

[4] Neubia. (2003). RSA vs DSA Signatures. *http://neubia.com/archives/000191.html* Tanggal akses: 16 Mei 2009 pukul 13:10.

[6] W3C. (2008). XML Signature Syntax and Processing (Second Edition). *http://www.w3.org/TR/xmldsig-core/* Tanggal akses: 17 Mei 2009 pukul 20:04.

[7] Stewart, Heath. (2003). Using XML Signature for Application Licensing. *http://www.codeproject.com/KB/security/xmldsiglic.aspx* Tanggal akses: 17 Mei 2009 pukul 20:10.