STUDI KEAMANAN PADA ENKRIPSI XML DAN XML *DIGITAL SIGNATURE*

Andrew Pratomo Budianto – NIM : 13505046

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : andrewpratomo@gmail.com

Abstrak

Makalah ini membahas tentang studi mengenai *Extensible Markup Language* (XML) dan metoda yang digunakan untuk mengamankan data yang terkandung di dalamnya. Menjelaskan algoritma – algoritma yang sering digunakan untuk mengenkripsi data XML, proses membuat dan memverifikasi XML *digital signature*, dan bagaimana gabungan dari enkripsi dan XML *digital signature* yang salah dapat mengakibatkan enkripsi XML lebih lemah dari yang seharusnya.

***Kata kunci**:* XML, *Extensible Markup Language*, *digital signature*, enkripsi, dekripsi.

1. Pendahuluan

Perkembangan teknologi yang begitu pesat akhir – akhir ini ternyata masih belum cukup untuk mengamankan transaksi bisnis yang terjadi dalam *Web*. Mekanisme pengamanan yang diberikan oleh *browser* tercanggih pada saat ini pun, hanya dapat digunakan untuk transaksi bisnis berskala kecil, dan tidak menyediakan keamanan dan fleksibilitas yang dibutuhkan untuk melindungi transaksi komersil berskala tinggi dan pertukaran data sensitif di seputar kegiatan tersebut.

Sebagai contoh, *Secure Sockets Layer* (SSL) menyediakan layanan pertukaran data sensitif antara *browser* dan *Web server*. Namun, ketika data berada di dalam *server* tersebut, seluruh data berada dalam keadaan tak terproteksi. Pada kenyataannya, SSL melindungi data mulai pada titik data tersebut dikirim, yang sebenarnya titik tersebut cukup jarang menerima serangan. Jika seorang *hacker* harus memilih antara: menyadap paket IP yang lewat untuk menemukan satu nomor kartu kredit seseorang, atau membobol basis data yang menampung ribuan nomor kartu kredit, tentunya pilihan kedua lebih diminati. Masalah ini tentunya dapat dicegah apabila data tersebut berada dalam keadaan terenkripsi tidak hanya ketika data tersebut dikirim, namun ketika berada dalam server publik.

Sama pentingnya dengan melindungi kerahasiaan pesan bisnis yaitu memastikan bahwa pesan tersebut asli, untuk itu diperlukan mekanisme yang mampu mengetahui siapa pengirim sebuah data, apakah data tersebut dimodifikasi di tengah jalan, dan apakah pengirim dapat menyangkal pernah mengirim data tersebut. Dengan kata lain, fasilitas fungsional yang sudah ada seputar keamanan teknologi internet (SSL dan *username/password*) tidak cukup tanpa mekanisme tambahan. Sebuah mekanisme mengatasi permasalahan ini yang umum digunakan untuk mengamankan transaksi bisnis adalah menggunakan *digital certificates* yang memungkinkan enkripsi dan *digital signing* pada data yang dipertukarkan.

2. *Extensible Markup Language*

Hampir seluruh dokumen memiliki struktur tertentu. XML merupakan bahasa markup untuk dokumen yang mengandung informasi terstruktur. Informasi terstruktur tersebut mengandung isi (kata – kata, gambar, dll) dan keterangan mengenai isi yang terkandung di dalamnya.

Bahasa markup, dengan kata lain adalah mekanisme untuk mengidentifikasi struktur dalam dokumen. Spesifikasi XML mendefinisikan standar untuk menambahkan markup ke dalam dokumen.

Sebuah fitur yang membuat XML begitu diminati untuk data transaksi bisnis (data terstruktur yang kaya semantik, berbasis teks,

siap untuk digunakan di *Web*) adalah bahwa XML memungkinkan sebuah aplikasi mengenkripsi dan membubuhkan *digital signature* pada data XML. Sebagai contoh, banyak skenario yang membutuhkan data XML dipindah tangankan secara bergilir, dan membutuhkan *digital signature* pada sub bagian tertentu dalam data XML untuk menandakan bahwa pihak yang menbubuhkan digital *signature* menyetujui sub bagian tersebut.

Dua buah mekanisme baru yang memanfaatkan kelebihan dari data XML adalah XML *signature* dan enkripsi XML. Keduanya sedang menempuh proses standardisasi. XML *signature* dibuat berdasar kerjasama antara World Wide Web Consortium (W3C) dan Internet Engineering Task Force (IETF), sedangkan enkripsi XML adalah murni hasil dari W3C.

3. Enkripsi XML

Enkripsi XML menyediakan layanan keamanan *end-to-end* untuk aplikasi yang membutuhkan pertukaran data terstruktur yang aman. XML itu sendiri merupakan teknologi yang paling populer untuk menstrukturkan data, dan oleh karena itu enkripsi berbasis XML merupakan cara natural untuk menangani kebutuhan yang kompleks dalam pertukaran data.

Saat ini, *Transport Layer Security* (TLS) merupakan standar *de facto* untuk keamanan berkomunikasi di atas internet. TLS merupakan protokol keamanan *end-to-end* yang mengikuti jejak *Secure Socket Layer* (SSL).

Dengan enkripsi XML, baik data yang aman dan tidak aman dapat dipertukarkan dalam dokumen yang sama. Sebagai contoh, sebuah aplikasi *chat* yang mengandung sejumlah ruang *chat* dengan sejumlah orang di tiap – tiap ruang. Data dengan enkripsi XML dapat dipertukarkan dengan lawan *chatting* sehingga data yang diintensikan untuk sebuah ruang tidak terlihat oleh ruang – ruang yang lain.

Di bawah ini sebuah contoh pertukaran data XML sederhana, dengan data seperti di bawah ini:

```
<purchaseOrder>
   <Order>
      <Item>book</Item>
      <Id>123-958-74598</Id>
      <Quantity>12</Quantity>
   </Order>
   <Payment>
      <CardId>
         123654-8988889-996874
      </CardId>
      <CardName>visa</CardName>
      <ValidDate>12-10-2004</ValidDate>
   </Payment>
</purchaseOrder>
```

Data di atas mengandung detail dari buku yang ingin di beli. Sebagai tambahan, data tersebut juga mengandung informasi kartu kredit untuk pembayaran. Secara natural, kita akan menggunakan komunikasi dengan pengamanan khusus untuk data sensitif ini. Salah satu pilihan adalah menggunakan SSL, yang akan mengamankan seluruh proses komunikasi. Alternatif lain adalah menggunakan enkripsi XML. Seperti yang telah dijelaskan sebelumnya, enkripsi XML bukan alternatif dari SSL/TLS. Jika sebuah aplikasi membutuhkan seluruh proses komunikasi diamankan, maka SSL lebih disarankan. Di sisi lain, enkripsi XML merupakan pilihan terbaik jika aplikasi membutuhkan kombinasi dari komunikasi yang aman dan tidak (yang berarti sebagian data akan dipertukarkan secara aman dan sisanya dikirim begitu saja).

3.1 Mengenkripsi seluruh dokumen dengan enkripsi XML

Apabila data pada dokumen XML di atas dienkripsi secara keseluruhan, maka hasil yang di dapat adalah:

```
<?xml version='1.0' ?>
<EncryptedData
xmlns='http://www.w3.org/2001/04/xmlenc#'
Type='http://www.isi.edu/in-
notes/iana/assignments/media-
types/text/xml'>
   <CipherData>
      <CipherValue>
         A23B45C56
      </CipherValue>
   </CipherData>
</EncryptedData>
```

Perhatikan tag `<CipherData>` dan `<CipherValue>`. Data yang sebenarnya berada dalam tag `<CipherValue>`. Seluruh elemen `CipherData` berada di dalam elemen `EncryptedData`. Elemen `EncryptedData` mengandung *namespace* XML yang digunakan dalam proses enkripsi. Sebagai contoh, data asli sebelum enkripsi adalah XML dan tipe definisi resmi oleh Internet Assigned Numbers Authority (IANA) untuk XML adalah *http://www.isi.edu/in-*

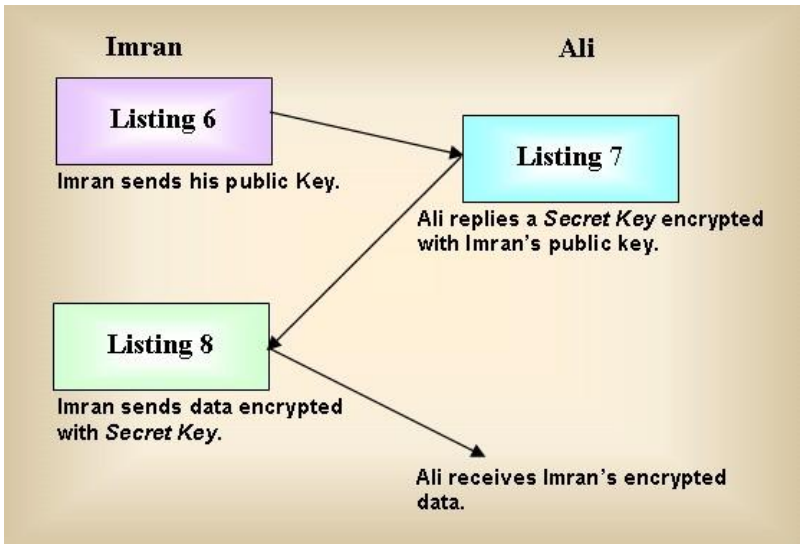
notes/iana/assignments/media-types/text/xml. Alamat tersebut muncul sebagai nilai dari atribut `Type`. Enkripsi XML menggunakan definisi tipe dari IANA untuk beberapa format data terkenal, seperti RTF, PDF, dan JPG. Apabila seseorang memiliki tipe data spesial untuk aplikasi tertentu, maka mereka dapat menspesifikasikan tipe data tersebut dalam atribut `Type` atau elemen `EncryptedData`. Atribut yang lain seperti xmlns, menspesifikasikan *namespace* enkripsi XML yang digunakan untuk mengenkripsi data XML.

3.2 Mekanisme enkripsi XML

Enkripsi XML tidak dapat dilakukan tanpa adanya sebuah kunci. Dengan enkripsi XML, seluruh masalah yang berhubungan dengan kunci terbagi menjadi dua bagian:

- Pertukaran kunci (enkripsi asimetris)
- Menggunakan kunci yang dipertukarkan sebelumnya (enkripsi simetris)

Dengan cara di atas, seseorang dapat mempertukarkan kunci dan menggunakannya setelah itu.



Gambar 1 Enkripsi XML

3.2.1 Kunci asimetris untuk mempertukarkan kunci

Pihak pertama mengirimkan kunci publiknya ke pihak kedua. Pihak kedua menggunakan kunci publik ini untuk mengenkripsi kunci rahasianya. Pertukaran data ini terlihat seperti di bawah ini:

```
<?xml version='1.0' ?>
<SecureCommunicationDemonstration>
   <EncryptedKey CarriedKeyName="Muhammad
Imran"
xmlns='http://www.w3.org/2001/04/xmlenc#'
>
      <ds:KeyInfo
xmlns:ds='http://www.w3.org/2000/09/xmlds
ig#'>
      <ds:KeyValue>
1asd25fsdf2dfdsfsdfds2f1sd23
      </ds:KeyValue>
      </ds:KeyInfo>
   </EncryptedKey>
</SecureCommunicationDemonstration>
```

Berdasar gambar 1, maka kita coba mengandaikan Imran sebagai pihak pertama dan Ali sebagai pihak ke dua, berkomunikasi satu sama lain. Imran memulai pertukaran kunci publik dan mengirim kunci publik tersebut ke dalam elemen bernama `KeyValue`. Atribut `CarriedKeyName` merepresentasikan nama kunci yang sedang ditranspor. Perhatikan bahwa akar elemen dari struktur ini adalah `EncryptedKey`, yang mengandung `ds:KeyInfo` dan `ds:KeyValue`. `ds:KeyInfo` dan `ds:KeyValue` merupakan bagian dari XML *Digital Signature* (`ds:`). Enkripsi XML bergantung seluruhnya pada spesifikasi XML *Digital Signature* untuk pertukaran kunci. Oleh karena itu, baik `<ds:EncryptedKey>` dan `<ds:KeyValue>` juga merupakan bagian dari spesifikasi XML *Digital Signature.*

3.2.2 Menggunakan kunci yang telah dipertukarkan

Sebelumnya, kita telah menukarkan kunci. Sekarang, kunci tersebut akan kita gunakan untuk mengenkripsi data. Kita mengasumsikan bahwa Imran mengirim pesan XML sebagai balasan dari kiriman Ali. Imran akan mendekripsi kunci tersebut dengan kunci privat miliknya. Imran akan dapat mengenkripsi data yang dia mau untuk dikirim kepada Ali menggunakan kunci rahasia tersebut dan meletakkannya di dalam elemen `CipherValue`.

Elemen `CipherData` dapat muncul dalam elemen `EncryptedData` atau `EncryptedKey`. Kita menggunakan elemen `CipherData` untuk merujuk pada salah satu dari data terenkripsi (ketika muncul dalam elemen `EncryptedData`) atau kunci terenkripsi (ketika muncul dalam elemen `EncryptedKey`).

Selain itu kita juga dapat merujuk pada *encrypted data* dan *encrypted keys* eksternal. Hal ini berarti *encrypted data* atau kunci yang sebenarnya akan diberikan di tempat lain (mungkin suatu tempat lain dalam internet) dan tidak di dalam file enkripsi XML. Pada kasus ini, kita akan menggunakan `CipherReference`, bukan `CipherValue` di dalam `CipherData`. Kita akan merujuk pada *encrypted data* yang sebenarnya melalui URI.

Kode di bawah akan mengilustrasikan variasi dalam merujuk pada data XML eksternal:

```
<?xml version='1.0' ?>
<EncryptedData  ID="Enc-Data"
xmlns='http://www.w3.org/2001/04/xmlenc#'
   <CipherReference
URI="http://www.waxsys.com/EncFile.xml" >
      <Transforms
xmlns:ds="http://www.w3.org/2000/09/xmlds
ig#" >
         <ds:Transform
          Algorithm="http://www.w3.org/TR
/1999/REC-xpath-19991116">
            <wax:XPath
xmlns:wax="http://www.waxsys.com/xpathNS"
>

PruchaseOrder/EncryptedData [@Id="Imran-
Enc-Data"]
            </wax:XPath>
         </ds:Transform>
      </Transforms>
   </CipherReference>
</EncryptedData>
```

Pada kode di atas kita mereferensi hanya bagian tertentu dari file eksternal yang ditunjuk oleh URI. Terdapat elemen anak `Transforms` di dalam elemen `CipherReference`. Elemen `Transform` Tersebut mengandung sejumlah elemen `Transform`, tiap elemen `Transform` tersebut mengandung sebuah elemen `XPath`. Elemen `XPath` tersebut menspesifikasikan sebuah ekspresi `XPath` yang merujuk pada node tertentu dari dokumen XML eksternal.

4. XML *Signature*

XML *Signatures* adalah *digital signatures* yang didesain untuk digunakan dalam transaksi XML. Standar yang ada mendefinisikan skema untuk mencatat hasil dari operasi digital *signature* diaplikasikan pada banyak jenis data (umumnya data XML). Sama seperti data *digital signatures* yang bukan XML (misalnya PKCS), XML *signatures* menambahkan autentifikasi, integritas data, dan mendukung *non-repudiation* pada data yang telah dibubuhkan *digital signatures*. Namun, bagaimanapun tidak seperti standar *digital signature* untuk data bukan XML, XML *signatures* telah didesain untuk memanfaatkan internet dan struktur XML itu sendiri.

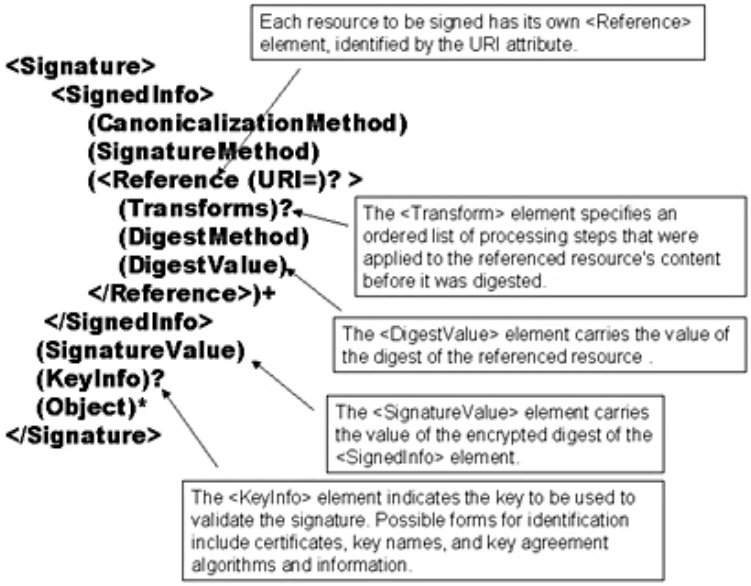
Salah satu kelebihan XML *Signature* yang paling fundamental adalah kemampuan untuk membubuhkan *digital signature* hanya pada bagian tertentu dari pohon XML, bukan pada keseluruhan dokumen. Hal ini akan relevan ketika sebuah dokumen XML memiliki sejarah yang panjang, dimana komponen yang berbeda dibuat pada waktu yang berbeda oleh berbagai pihak yang berbeda, tiap pembubuhan *digital signature* hanya pada elemen yang relevan terhadap *digital signature* tersebut. Fleksibilitas ini juga akan menjadi sangat penting pada situasi dimana diperlukan untuk memastikan integritas dari bagian tertentu dari dokumen XML, sementara membuka kemungkinan untuk bagian lain dari XML diubah. Bayangkan, apabila sebuah formulir XML dikirim pada pengguna untuk dilengkapi. Apabila *digital signature* yang diberikan berada pada keseluruhan data XML, seluruh perubahan yang dibubuhkan oleh pengguna akan membuat *digital signature* dari data XML tersebut tidak valid.

Sebuah XML *signature* dapat menandai lebih dari satu tipe *resource*. Sebagai contoh, sebuah XML *signature* mungkin menandatangani data HTML, data *binary* (JPG), data XML itu sendiri, dan bagian tertentu dari *file* XML.

Validitas *signature* membutuhkan bahwa data objek yang dibubuhi *digital signature* dapat diakses. XML *signature* itu sendiri hanya akan mengindikasikan lokasi dari objek yang dibubuhi *digital signature*. Acuan ini dapat:

- Diacu oleh URI di dalam XML *digital signature*
- Berada dalam *resource* yang sama dengan XML *signature*.
- Diselipkan dalam XML *signature*.
- Memiliki XML *signature* yang diselipkan dalam dirinya sendiri

Komponen dari XML *signature* adalah seperti di bawah ini:



Gambar 2 Komponen XML Signature

4.1 Mekanisme XML *Signature*

Di bawah ini akan dituturkan mekanisme XML *Signature* secara singkat.

4.1.1 Menentukan *resource* yang akan dibubuhi *digital signature*

Proses ini akan mengidentifikasi *resource* melalui URI. Sebagai contoh:

- "*http://www.abccompany.com/index.html*" akan merujuk pada halaman HTML pada *Web*.
- "*http://www.abccompany.com/logo.gif*" akan merujuk pada gambar GIF pada *Web*.
- "*http://www.abccompany.com/xml/po.xml*" akan merujuk pada file XML pada *Web*.
- "*http://www.abccompany.com/xml/po.xml#sender1*" akan merujuk pada elemen tertentu dalam file XML pada *Web*.

4.1.2 Menghitung *digest* pada tiap *resource*

Pada XML *signature*, tiap *resource* yang direferensi dispesifikasikan melalui elemen `<Reference>` dan *digest* yang dikandungnya (dihitung pada *resource* yang diidentifikasi, bukan pada elemen `<Reference>` itu sendiri) diletakkan dalam elemen `<DigestValue>`.

```
<Reference
URI="http://www.abccompany.com/news/2000/
03_27_00.htm">
   <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1" />

<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=
</DigestValue>
   </Reference>
   <Reference
  URI="http://www.w3.org/TR/2000/WD-
xmldsig-core-20000228/signature-
example.xml">
      <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"/>

<DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=
</DigestValue>
   </DigestMethod>
</Reference>
```

4.1.3 Mengambil elemen *Reference*

Elemen `<Reference>` dan digest yang terasosiasi di dalam `<SignedInfo>` seperti pada kode di bawah ini:

```
<SignedInfo Id="foobar">
   <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
      <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#dsa-sha1" />
         <Reference
URI="http://www.abccompany.com/news/2000/
03_27_00.htm">
            <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1" />
```

```
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=
</DigestValue>
        </Reference>
        <Reference
    URI="http://www.w3.org/TR/2000/WD-
xmldsig-core-20000228/signature-
example.xml">
           <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"/>

<DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=
</DigestValue>
        </Reference>
</SignedInfo>
```

Pada kode di atas, elemen `<CanonicalizationMethod>` mengindikasikan bahwa algoritma digunakan untuk mengkanonisasikan elemen `<SignedInfo>`. Aliran data yang berada dalam XML *information set* mungkin memiliki representasi tekstual yang berbeda, misalnya bagaimana menangani *whitespace*. Untuk menghindari hasil verifikasi yang tidak akurat, XML *information set* harus dikanonisasikan sebelum mengekstrak representasi bitnya untuk proses pembubuhan *signature*. Elemen `<SignatureMethod>` mengidentifikasi algoritma yang digunakan untuk menghasilkan nilai *signature*.

4.1.4 Pembubuhan *ignature*

Menghitung digest pada elemen `<SignedInfo>`, beri tanda pada digest tersebut, lalu bubuhkan nilai signature pada elemen `<SignatureValue>`.

```
<SignatureValue>MC0E LE=</SignatureValue>
```

4.1.5 Menambahkan informasi kunci

Jika informasi kunci ingin ditambahkan, maka diletakkan pada elemen <KeyInfo>. Pada elemen tersebut, informasi kunci mengandung sertifikat X.509 untuk pengirim, yang akan mengandung kunci publik yang dibutuhkan untuk memverifikasi proses pembubuhan *signature*.

```
<KeyInfo>
 <X509Data>
  <X509SubjectName>CN=Ed Simon,O=XMLSec
Inc.,ST=OTTAWA,C=CA</X509SubjectName>

<X509Certificate>MIID5jCCA0+gA...lVN</X50
9Certificate>
 </X509Data>
</KeyInfo>
```

4.1.6 Penutup dalam elemen *signature*.

Letakkan elemen `<SignedInfo>`, `<SignatureValue>`, dan `<KeyInfo>` ke dalam elemen `<Signature>`. Elemen `<Signature>` menandakan XML *signature*. Hasil akhir:

```
<?xml version="1.0" encoding="UTF-8"?>
   <Signature
xmlns="http://www.w3.org/2000/09/xmldsig#
">
      <SignedInfo Id="foobar">
         <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
         <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#dsa-sha1" />
         <Reference
URI="http://www.abccompany.com/news/2000/
03_27_00.htm">
            <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1" />

<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=
</DigestValue>
         </Reference>
         <Reference
URI="http://www.w3.org/TR/2000/WD-
xmldsig-core-20000228/signature-
example.xml">
            <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"/>

<DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=
</DigestValue>
         </Reference>
      </SignedInfo>

<SignatureValue>MC0E~LE=</SignatureValue>

<KeyInfo>
   <X509Data>
      <X509SubjectName>CN=Ed
Simon,O=XMLSec
Inc.,ST=OTTAWA,C=CA</X509SubjectName>
      <X509Certificate>
         MIID5jCCA0+gA...lVN
      </X509Certificate>
   </X509Data>
</KeyInfo>
```

5. Kesimpulan

Penggunaan enkripsi dan *digital signature* pada suatu bagian dari dokumen XML yang kurang tepat dapat mengakibatkan proses dekripsi dan verifikasi *signature* menjadi sulit. Umumnya, ketika memverifikasi *signature*, seseorang harus tau apakah *signature* dihitung pada data setelah proses enkripsi atau sebelum proses enkripsi dilakukan.

Hal lain, namun sangat penting adalah *cryptographic vulnerabilities* ketika

menggabungkan *digital signature* dan enkripsi pada sebuah elemen XML. Hal Finney menyarankan untuk mengenkripsi data yang telah dibubuhi *signature*, sementara meninggalkan *digital signature* dalam keadaan bersih, yang memungkinkan terjadinya serangan *plain text guessing*. Kelemahan ini dapat ditanggulangin menggunakan *secure hashes* dan *nonces* pada data yang diproses.

Menurut W3C, interaksi enkripsi dan pembubuhan *signature* sebenarnya merupakan masalah skala aplikasi, namun, W3C menyarankan:

- Ketika mengenkripsi data, *digest* atau *signature* untuk data tersebut sebaiknya dienkripsi. Hal ini mencegah terjadinya masalah pertama yang memungkinkan hanya *signature* tersebut yang dapat divalidasi. Hal ini juga mencegah terjadinya serangan *plain text guessing.*
- Gunakan teknik "decrypt-except" *signature transform.* Teknik ini bekerja dengan cara: ketika proses *signature transform*, apabila ditemukan *decrypt transform*, maka seluruh data terenkripsi akan didekripsi, terkecuali data yang yang tidak direferensikan.

DAFTAR PUSTAKA

[1] Blair Dillaway, Ed Simon, Takeshi Imamura. (2002). XML Encryption Sytax and Processing. *http://www.w3.org/TR/xmlenc-core*. Tanggal akses: 30 Maret 2009 pukul 20:00.

[2] Carlisle Adams, Ed Simon, Paul Madsen. (2001). An Introduction to XML Digital Signatures. *http://www.xml.com/pub/a/2001/08/08/xmldsig.html*. Tanggal Akses: 21 Maret 2009 pukul 20:00.

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

[4] Siddiqui, Bilal. (2002). Exploring XML Enryption. *http://www.ibm.com/developerworks/xml/library/x-encrypt*. Tanggal akses: 21 Maret 2009 pukul 19:00.

[5] Walsh, Norman. (1998). A Technical Introduction to XML. *http://www.xml.com/pub/a/98/10/guide0.html*. Tanggal akses: 30 Maret 2009 pukul 20:00.