# XML *Signature*

Salma Desenta[1], Antonius Santoso[2], dan Moh. Farid Taufiqurrohman [3]

*Departemen Teknik Informatika*
*Institut Teknologi Bandung*
*Jalan Ganesha 10 Bandung 40132*

E-mail : *if12004@students.if.itb.ac.id*[1], *if12012@students.if.itb.ac.id*[2],
*if12029@students.if.itb.ac.id*[3]

**Abstrak**

Data XML saat ini telah secara luas digunakan sebagai salah satu standar untuk format pertukaran data. Format XML ini juga dapat digunakan sebagai *digital signature*. Tanda tangan XML (XML *signatures*) yang dibahas pada makalah ini mengacu pada standar yang dibuat oleh W3C. Standar W3C menjelaskan proses penandatanganan, verifikasi tanda tangan dan struktur dokumen XML *signature*. Makalah ini juga membahas pemrosesan XML menjadi bentuk yang paling sederhana (*Canonical* XML) sehingga untuk dua dokumen XML yang memiliki konteks sama namun sintaks yang berbeda, dapat dikenali sebagai dokumen yang identik.

***Kata kunci**: XML, XML signatures*

## 1. Pendahuluan

Data dalam bentuk XML saat ini telah digunakan secara luas dalam dunia teknologi informasi. Peranan XML dalam transaksi bisnis pun cukup besar. Hal ini dikarenakan fitur yang diberikan XML, antara lain data yang tersimpan secara terstruktur, semantik yang dapat didefinisikan sesuai kebutuhan, berbasis teks, dan *web-ready*. Seiring dengan perkembangan *web-services* sebagai *middleware* untuk integrasi aplikasi antar perusahaan, XML digunakan pula sebagai format data yang dipertukarkan melalui *web services* tersebut.

Teknologi keamanan saat ini dirasakan kurang mencukupi untuk mengamankan transaksi bisnis yang dilakukan melalui *web*, khususnya untuk mengamankan data rahasia yang perlu dipertukarkan pada transaksi tersebut. Sebagai contoh, *Secure Sockets Layer* (SSL) hanya mengamankan data ketika data dikirimkan dari *browser* ke *web server* dan sebaliknya. Ketika data berada pada masing-masing *site*, data dapat diambil maupun diubah oleh pihak lain.

Selain memperhatikan aspek kerahasiaan data, perlu diperhatikan pula aspek keamanan lain seperti otentikasi, integritas data, dan anti-penyangkalan (*non-repudiation*). Untuk mengimplementasikan aspek keamanan tersebut, dibutuhkan suatu mekanisme yang dikenal dengan nama *digital signature*.

Pada makalah ini akan dibahas mengenai standar XML *signatures* yang ditetapkan oleh W3C. XML *signatures* merupakan dokumen XML yang berisi informasi mengenai tanda tangan digital. Tanda tangan digital dapat dilakukan terhadap dokumen dengan tipe apapun, termasuk dokumen

XML. XML *signatures* dapat ditambahkan pada dokumen XML yang ditandatangani ataupun dapat berupa sebuah dokumen XML tersendiri.

Secara garis besar, struktur XML *signatures* adalah sebagaimana (dimana "?" menandakan nol atau satu kemunculan, "+" menandakan satu atau lebih kemunculan, dan "*" menandakan nol atau lebih kemunculan) ditampilkan pada **Kode XML 1** [3].

Salah satu keuntungan penggunaan standar XML *signature* adalah dapat dilakukannya penandatanganan sebuah dokumen XML oleh lebih dari satu pihak. Pihak tertentu hanya akan menandatangani elemen XML yang menjadi tanggung jawabnya.

## 2. Langkah-langkah Penyusunan XML *Signature*

Secara garis besar, langkah-langkah pembubuhan XML *signature* adalah sebagai berikut [2]:

### 2.1 Tentukan dokumen yang akan ditandatangani

Informasi mengenai dokumen yang akan ditandatangani akan disediakan dalam bentuk *Uniform Resource Identifier* (URI) [2]

- http://www.abccompany.com/index.html : mengacu sebuah halaman HTML pada web.
- http://www.abccompany.com/logo.gif : mengacu sebuah citra GIF pada web.
- http://www.abccompany.com/xml/po.xml : mengacu sebuah file XML pada web.
- http://www.abccompany.com/xml/po.xml #sender1 : mengacu suatu elemen spesifik pada sebuah file XML pada web.

### 2.2 Lakukan perhitungan nilai *digest* untuk setiap dokumen

Pada XML *signatures*, setiap dokumen yang diacu (*reference*) dispesifikasikan dengan penggunaan elemen `<Reference>`. Nilai *digest* untuk *reference* tersebut dispesifikasikan dengan penggunaan elemen `<DigestValue>` sebagai elemen *child* dari elemen `<Reference>`. Elemen `<DigestMethod>` digunakan untuk mengidentifikasikan algoritma yang digunakan untuk menghitung nilai *digest*. Sebagai contoh, perhatikan potongan XML *signature* pada **Kode XML 2**.

### 2.3 Satukan elemen *reference*

Satukan elemen `<Reference>` (beserta nilai digest masing-masing) dalam elemen `<SignedInfo>` seperti contoh pada **Kode XML 3**.

Perhatikan bahwa potongan kode pada **Kode XML 3** merupakan perluasan dari kode pada **Kode XML 2**. Elemen `<CanonicalizationMethod>` memberikan informasi mengenai algoritma yang digunakan untuk proses kanonikalisasi. Proses kanonikalisasi dibahas lebih lanjut pada upa bab selanjutnya. Elemen `<SignatureMethod>` mengidentifikasikan algoritma yang digunakan untuk mendapatkan nilai *signature*.

### 2.4 Penandatanganan

Hitung nilai *digest* dari elemen `<SignedInfo>`, tandatangani nilai *digest* tersebut dan simpan nilai *signature* pada elemen `<SignatureValue>` seperti terlihat pada potongan kode pada **Kode XML 4**.

### 2.5 Tambahkan Informasi Kunci

Letakkan informasi tambahan tentang kunci pada elemen `<KeyInfo>`. Informasi tambahan ini berupa kunci publik yang dapat digunakan untuk proses verifikasi. Salah satu contoh informasi kunci yang digunakan adalah X509 seperti tercantum pada **Kode XML 5**.

### 2.6 Tutup XML *Signature* dengan elemen *Signature*

Letakkan elemen `<SignedInfo>`, `<SignatureValue>`, dan `<KeyInfo>` dalam elemen `<Signature>`. Kode XML *Signature* secara keseluruhan yang merupakan hasil dari penerapan langkah-langkah dari 1 hingga 6 dapat dilihat pada **Kode XML 6**.

## 3. Proses Verifikasi XML *Signature*

Secara garis besar, proses verifikasi XML signature adalah sebagai berikut [3)]:

1. *Canonicalize* elemen `<SignedInfo>` dengan menggunakan metode pada `<CanonicalizationMethod>`. Hitung ulang nilai *digest* untuk setiap dokumen yang terdapat dalam elemen `<SignedInfo>` dan bandingkan dengan nilai *digest* yang terdapat pada elemen `<DigestValue>` di dalam elemen `<Reference>` yang bersesuaian.
2. Verifikasi *signature* pada elemen `<SignedInfo>`. Untuk melakukan hal tersebut, hitung ulang nilai *digest* dari elemen `<SignedInfo>` (dengan menggunakan algoritma *digest* yang dispesifikasikan pada elemen `<SignatureMethod>`) dan gunakan kunci publik yang disediakan untuk melakukan verifikasi bahwa nilai elemen `<SignatureValue>` bersesuaian dengan nilai *digest* pada elemen `<SignedInfo>`.

Perhatikan agar proses *canonicalization* yang digunakan tidak akan menimbulkan efek yang tidak diinginkan seperti penulisan ulang URI.

## 4. Sintaks XML *Signature*

Pada bagian ini akan dijelaskan sintaks detail dari fitur *signature* utama. Sintaks akan didefinisikan melalui DTD dan skema XML.

### 4.1 Elemen `Signature`

Elemen **`Signature`** merupakan akar dari suatu XML *signature*. Dalam implementasinya harus dihasilkan skema XML elemen **`Signature`** yang valid yang dispesifikasikan dalam **Skema XML 1**.

### 4.2 Elemen `SignatureValue`

Elemen **`SignatureValue`** mengandung nilai aktual dari *digital signature*. Elemen ini selalu di-*encode* dengan menggunakan base64. Skema elemen **`SignatureValue`** dapat dilihat pada **Skema XML 2**.

### 4.3 Elemen `SignedInfo`

Struktur dari **`SignedInfo`** mencakup algoritma *canonicalization*, suatu algoritma *signature*, dan satu atau lebih referensi. Elemen **`SignedInfo`** dapat mengandung atribut ID opsional sehingga dapat direferensi oleh *signature* atau objek lain. Skema elemen **`SignedInfo`** dapat dilihat pada **Skema XML 3**.

#### 4.3.1 Elemen `CanonicalizationMethod`

**`CanonicalizationMethod`** merupakan elemen yang dibutuhkan untuk menspesifikasikan algoritma yang digunakan dalam proses *canonicalization* sebelum melakukan operasi penandatanganan. Skema elemen **`CanonicalizationMethod`** dapat dilihat pada **Skema XML 4**.

### 4.3.2 Elemen `SignatureMethod`

**`SignatureMethod`** merupakan elemen yang dibutuhkan untuk menspesifikasian algoritma yang digunakan untuk menghasilkan dan memvalidasi *signature*. Algoritma ini mengidentifikasi semua fungsi kriptografi yang terdapat di operasi *signature* (*hashing*, algoritma kunci publik, MAC, *padding*, dan sebagainya). Skema elemen **`SignatureMethod`** dapat dilihat pada **Skema XML 5**.

### 4.3.3 Elemen `Reference`

Elemen **`Reference`** merupakan elemen yang dapat muncul lebih dari satu kali. Elemen ini menspesifikasikan algoritma *digest* dan nilai *digest* serta (opsional) *identifier* dari objek yang ditanda tangan, tipe tanda tangan, dan *transform* yang harus dilakukan sebelum *digesting*. Identifikasi (URI) dan *transforn* mendekripsikan bagaimana konten yang telah di-*digest* dibuat. Skema elemen **`Reference`** dapat dilihat pada **Skema XML 6**.

### 4.4.4 Elemen `Transform`

Elemen **`Transform`** mengandung *list* terurut yang mendeksripsikan bagaimana penandatangan memperoleh objek data yang di-*digest*,. Keluaran dari setiap **`Transform`** akan menjadi masukan dari **`Transform`** berikutnya. Masukan dari **`Transform`** pertama berupa hasil deferensi dari atribut URI dari elemen **`Reference`**. Skema elemen **`Transform`** dapat dilihat pada **Skema XML 7**.

### 4.4.5 Elemen `DigestMethod`

**`DigestMethod`** merupakan elemen yang dibutuhkan untuk mengidentifikasi algoritma *digest* yang akan diterapkan ke objek yang ditandatangani. Elemen ini menggunakan skema seperti dispesifikasikan pada **Skema XML 8**.

### 4.4.6 Elemen `DigestValue`

**`DigestValue`** merupakan elemen yang mengandung nilai *digest* yang di-*encode*. Digest selalu di-*encode* dengan menggunakan base64. Skema elemen **`DigestValue`** dapat dilihat pada **Skema XML 9**.

### 4.4 Elemen `KeyInfo`

**`KeyInfo`** merupakan elemen opsional yang memungkinkan penerima untuk memperoleh kunci yang dibutuhkan untuk memvalidasi *signature*. **`KeyInfo`** mungkin mengandung kunci, nama, sertifikat, dan informasi manajemen kunci publik lainnya. Skema elemen **`SignedInfo`** dapat dilihat pada **Skema XML 10**.

### 4.5 Elemen `Object`

**`Object`** merupakan elemen opsional yang dapat muncul satu atau lebih. Elemen ini mungkin mengandung data apapun. Elemen **`Object`** ini dapat mengandung atribut tipe MIME, ID, dan encoding.

Atribut **`Encoding`** mungkin digunakan untuk menyediakan URI yang mengidentifikasi metode untuk men-*encode* objek tersebut.

Atribut **`MimeType`** merupakan atribut tambahan yang mendeskripsikan data yang

terdapat pada **`Object`**. Atribut ini berisi nilai *string* yang didefinisikan oleh MIME.

Atribut **`Id`** umumnya direferensi dari **`Reference`** di **`SignedInfo`**. Elemen ini umumnya digunakan pada *enveloping signature* dimana objek yang ditanda tangan dimasukkan ke dalam elemen *signature* . Skema elemen SignedInfo dapat dilihat pada **Skema XML 11**.

## 5. Canonical XML

Dokumen XML memiliki sintaks yang tidak terlalu mengikat jika dibandingkan dengan *format* data pada basis data. Hal ini memungkinkan dokumen-dokumen XML, yang ekuivalen dalam konteks aplikasi tertentu, memiliki banyak variasi leksikal dan representasi fisik yang berbeda. Perbedaan tersebut antara lain dalam [1]:

- struktur entitas dokumen,
- urutan atribut suatu elemen,
- *character encoding*,
- jumlah spasi antara nama elemen dan atribut-atributnya.

Sebagai contoh, bandingkan dokumen XML pada Listing 1 dan Listing 2 berikut

**Listing 1**

```
<doc>
  <a a1=″1″ a2=″2″> 123 </a>
</doc>
```

**Listing 2**

```
<?xml version=″1.0″
encoding=″UTF-8″?>
<doc>
  <a
    a2=″2″  a1=″1″
  >123</a>
<doc>
```

Kedua dokumen tersebut memiliki informasi yang sama, yang ada pada dalam elemen <a>. Akan tetapi, dokumen pada Listing 2 memiliki deklarasi XML (`<?xml version=″1.0″ encoding=″UTF-8″?>`) yang tidak terdapat pada Listing 1. Selain itu, keduanya memiliki perbedaan jumlah spasi, format penulisan, dan urutan atribut pada elemen `<a>`.

Fleksibilitas leksikal ini menyebabkan beberapa masalah, antara lain dalam *regression testing*, *byte-by-byte comparison*, dan *digital signatures* [1, 3]. Misalnya, Listing 1 dikirim melalui sistem pengiriman yang menyebabkan Listing 1 berubah menyerupai Listing 2. Akibatnya, hash sederhana atau *digital signature* dari kedua listing tersebut tidak akan sama. Padahal, kedua dokumen tersebut memiliki informasi yang sama.

W3C mendefinisikan ***canonical XML*** yang merupakan bentuk leksikal normal untuk dokumen XML. *Canonical XML* ini menghilangkan semua variasi yang diizinkan dan menerapkan aturan yang ketat untuk menghasilkan perbandingan *byte-by-byte* yang konsisten [4]. Proses pembentukannya disebut dengan ***canonicalization*** (populer dengan singkatan “c14n”).

Berdasarkan spesifikasi yang diberikan oleh W3C, pembentukan dokumen XML *canonic* memiliki aturan sebagai berikut [3]:

- Dokumen di-*encode* dalam UTF-8.
- *Line break* dinormalisasikan menjadi “#xA” pada input sebelum diparsing.
- Nilai atribut dinormalisasikan, seolah-olah dengan *validating processor*.
- Karakter dan *parsed entity references* diganti.
- Bagian CDATA diganti dengan isi karakternya.
- Deklarasi XML dan *Document Type Declaration* (DTD) dihilangkan karena

dokumen sudah dalam UTF-8 sehingga konversi tidak diperlukan lagi.
- Elemen-elemen kosong dikonversi menjadi pasangan *start-end tag*.
- *Whitespace* (spasi, tab, enter) diluar elemen dokumen dan antara *start* dan *end-tags* dinormalisasikan.
- Penanda nilai atribut diubah menjadi *quotation marks* (*double quotes*).
- Karakter-karakter khusus dalam nilai atribut dan isi dokumen diganti dengan referensi karakter.
- Deklarasi *superfluous namespace* dihilangkan dari setiap elemen.
- Atribut *default* ditambahkan ke setiap elemen.
- Urutan leksikografi disesuaikan dengan deklarasi *namespace* dan atribut dari setiap elemen.

Dengan menerapkan aturan *canonicalization*, struktur XML pada Listing 1 dan Listing 2 akan menjadi seperti pada Listing 3. Struktur yang baru ini membuat Listing 1 dan Listing 2 memiliki perbandingan *byte*-by-*byte* dan nilai hash yang sama.

**Listing 3**

```
<doc>
      <a a1=″1″ a2=″2″>123</a>
</doc>
```

## 6. Algoritma

Algoritma yang digunakan dalam *XML Digital Signature* diidentifikasi oleh URI yang muncul sebagai atribut elemen. Algoritma tersebut memiliki parameter yang eksplisit maupun implisit. Paramater eksplisit muncul sebagai elemen isi dalam elemen *algorithm role* yang bersangkutan. Parameter ini memiliki nama elemen deskriptif yang biasanya spesifik terhadap algoritma tertentu Sedangkan paramater implisit tidak ditulis oleh algoritma yang memilikinya, misalnya **`SignatureMethod`** yang secara impisit mempunya dua parameter, yaitu *keying info* dan keluaran yang diperoleh dari **`CanonicalizationMethod.`**

Beberapa algoritma yang dispesifikasikan dalam implementasi adalah [3)]:

- ***Digest :***

Hanya satu algoritma yang didefinisikan di sini, yaitu SHA-1. Akan tetapi, tidak tertutup kemungkinan pemakaian algoritma digest yang baru yang lebih kuat. SHA-1 tidak memiliki parameter eksplisit. Identifier yang digunakan adalah

```
<SignatureMethod
Algorithm=″http://www.w3.org/2000
/09/xmldsig#sha1″>
```

- ***Encoding :***

Dalam XML digital signature, digunakan encoding base64. Identifier yang digunakan adalah

```
<SignatureMethod
Algorithm=″http://www.w3.org/2000
/09/xmldsig#base64″>
```

- ***MAC :***

Algoritma MAC membutuhkan dua parameter, yaitu *keying material* yang ditentukan dari `KeyInfo` dan *octet stream output* dari `CanonicalizationMethod`. Identifier yang digunakan adalah

```
<SignatureMethod
Algorithm=″http://www.w3.org/2000
/09/xmldsig#hmac-sha1″>
```

- ***Signature :***

Algoritma yang digunakan adalah DSA dengan SHA1. Algoritma dalam *signature*

dan MAC memiliki sintaks yang identik, tetapi *signature* menggunakan algoritma kunci publik. Identifier yang digunakan adalah

```
<SignatureMethod
Algorithm="http://www.w3.org/2000
/09/xmldsig#dsa-sha1">
```

Selain menggunakan DSA, signature juga dapat menggunakan RSA. Identifier yang digunakan jika memakai RSA adalah

```
<SignatureMethod
Algorithm="http://www.w3.org/2000
/09/xmldsig#rsa-sha1">
```

- ***Canonicalization :***

Berbagai macam algoritma *canonicalization* memerlukan konversi ke UTF-8. Algoritma ini dapat menghilangkan *comment* maupun dengan menghilangkannya. Identifier yang digunakan dengan menghilangkan *comment*

```
<SignatureMethod
Algorithm="http://www.w3.org/TR/2
001/REC-xml-c14n-20010315">
```

Identifier dengan mempertahankan *comment*

```
<SignatureMethod
Algorithm="http://www.w3.org/TR/2
001/REC-xml-c14n-
20010315#WithComments">
```

- ***Transform :***

Untuk transformasi dari dokumen XML menjadi format lain, dapat digunakan beberapa macam algoritma. Transformasi dengan XSLT (optional) dapat menghasilkan output HTML. Identifier yang digunakan adalah

```
<SignatureMethod
Algorithm="http://www.w3.org/TR/1
999/REC-xslt-19991116">
```

Selain XSLT, dapat digunakan XPath filtering. Tujuan utama dari transformasi ini adalah untuk meyakinkan bahwa hanya perubahan yang didefinisikan secara spesifik terhadap dokumen XML masukan yang diizinkan setelah signature ditambahkan. Identifier yang digunakan adalah

```
<SignatureMethod
Algorithm="http://www.w3.org/TR/1
999/REC-xpath-19991116">
```

Disamping kedua algoritma transformasi tersebut, dapat digunakan *Enveloped Signature Transform*. Transformasi ini (misal **T**) menghapus seluruh elemen *signature* yang mengandung **T** dari perhitungan digest dalam elemen `Reference` yang mengandung **T**.

```
<SignatureMethod
Algorithm="http://www.w3.org/2000
/09/xmldsig#enveloped-isgnature">
```

## 7. Kesimpulan

XML *signatures* dapat digunakan sebagai salah satu alternatif dalam mengautentikasi dokumen digital, baik dokumen XML maupun dokumen bertipe lain. XML *signatures* memiliki struktur (elemen-elemen) yang setiap elemennya terdefinisi secara jelas (seperti dokumen yang akan ditandatangani, algoritma *signature* yang digunakan, serta algoritma *canonicalization* yang dipakai) dan memiliki makna tertentu. Dalam proses penandatanganan dan verifikasi XML *signatures*, diperlukan proses *canonicalization* untuk menjamin dokumen yang identik (mengandung informasi yang

sama) memiliki tanda tangan dan perbandingan *byte-by-byte* yang sama pula. Dari sisi algoritma yang digunakan, tidak menutup kemungkinan penggunaan algoritma lain di luar spesifikasi yang ada selama algoritma tersebut lebih baik.

## 8. Kode XML dan Skema XML

**Kode XML 1. Struktur XML *Signature***

```
<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
      (<Transforms>)?
      <DigestMethod>
      <DigestValue>
    </Reference>)+
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

**Kode XML 2. Contoh XML *Signature***

```
<Reference URI="http://www.abccompany.com/news/2000/03_27_00.htm">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#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/xmldsig#sha1"/>
  <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
</Reference>
```

**Kode XML 3 Penyatuan Elemen <Reference>**

```
<SignedInfo Id="foobar">
  <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"
/>
  <Reference URI="http://www.abccompany.com/news/2000/03_27_00.htm" />
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#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/xmldsig#sha1"/>
    <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
  </Reference>
```

```
</SignedInfo>
```

**Kode XML 4. Penyimpanan Nilai *Signature***

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

**Kode XML 5. Contoh Informasi Kunci**

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

**Kode XML 6. Kode XML *Signature* Lengkap**

```
<?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/xmldsig#dsa-
sha1" />
    <Reference URI="http://www.abccompany.com/news/2000/03_27_00.htm">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#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/xmldsig#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>
</Signature>
```

**Skema XML 1 Skema Elemen Signature**

```
Schema Definition:

<element name="Signature" type="ds:SignatureType"/>
  <complexType name="SignatureType">
    <sequence>
```

```
      <element ref="ds:SignedInfo"/>
      <element ref="ds:SignatureValue"/>
      <element ref="ds:KeyInfo" minOccurs="0"/>
      <element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

DTD:

<!ELEMENT Signature (SignedInfo, SignatureValue, KeyInfo?, Object*) >
<!ATTLIST Signature
  xmlns   CDATA   #FIXED 'http://www.w3.org/2000/09/xmldsig#'
  Id      ID  #IMPLIED >
```

**Skema XML 2 Skema Elemen SignatureValue**

```
Schema Definition:

<element name="SignatureValue" type="ds:SignatureValueType"/>
  <complexType name="SignatureValueType">
    <simpleContent>
      <extension base="base64Binary">
        <attribute name="Id" type="ID" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

DTD:

<!ELEMENT SignatureValue (#PCDATA) >
<!ATTLIST SignatureValue Id ID #IMPLIED >
```

**Skema XML 3 Skema Elemen SignedInfo**

```
Schema Definition:

<element name="SignedInfo" type="ds:SignedInfoType"/>
  <complexType name="SignedInfoType">
    <sequence>
      <element ref="ds:CanonicalizationMethod"/>
      <element ref="ds:SignatureMethod"/>
      <element ref="ds:Reference" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

DTD:

<!ELEMENT SignedInfo
```

```
  (CanonicalizationMethod, SignatureMethod,  Reference+) >
<!ATTLIST SignedInfo Id ID #IMPLIED >
```

**Skema XML 4 Skema Elemen CanonicalizationMethod**

```
Schema Definition:

<element name="CanonicalizationMethod"
type="ds:CanonicalizationMethodType"/>
  <complexType name="CanonicalizationMethodType" mixed="true">
    <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
      <!-- (0,unbounded) elements from (1,1) namespace -->
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

DTD:

<!ELEMENT CanonicalizationMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST CanonicalizationMethod Algorithm CDATA #REQUIRED >
```

**Skema XML 5 Skema Elemen SignatureMethod**

```
Schema Definition:

<element name="SignatureMethod" type="ds:SignatureMethodType"/>
  <complexType name="SignatureMethodType" mixed="true">
    <sequence>
      <element name="HMACOutputLength" minOccurs="0"
        type="ds:HMACOutputLengthType"/>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      <!-- (0,unbounded) elements from (1,1) external namespace -->
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

DTD:

<!ELEMENT SignatureMethod (#PCDATA|HMACOutputLength %Method.ANY;)* >
<!ATTLIST SignatureMethod Algorithm CDATA #REQUIRED >
```

**Skema XML 6 Skema Elemen Reference**

```
Schema Definition:

<element name="Reference" type="ds:ReferenceType"/>
  <complexType name="ReferenceType">
    <sequence>
      <element ref="ds:Transforms" minOccurs="0"/>
      <element ref="ds:DigestMethod"/>
```

```
      <element ref="ds:DigestValue"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
    <attribute name="URI" type="anyURI" use="optional"/>
    <attribute name="Type" type="anyURI" use="optional"/>
  </complexType>

DTD:

<!ELEMENT Reference (Transforms?, DigestMethod, DigestValue) >
<!ATTLIST Reference
    Id    ID     #IMPLIED
    URI   CDATA  #IMPLIED
    Type  CDATA  #IMPLIED >
```

**Skema XML 7 Skema Elemen Transform**

```
Schema Definition:

<element name="Transforms" type="ds:TransformsType"/>
  <complexType name="TransformsType">
    <sequence>
      <element ref="ds:Transform" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <element name="Transform" type="ds:TransformType"/>
  <complexType name="TransformType" mixed="true">
    <choice minOccurs="0" maxOccurs="unbounded">
      <any namespace="##other" processContents="lax"/>
      <!-- (1,1) elements from (0,unbounded) namespaces -->
      <element name="XPath" type="string"/>
    </choice>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

DTD:

<!ELEMENT Transforms (Transform+)>

<!ELEMENT Transform (#PCDATA|XPath %Transform.ANY;)* >
<!ATTLIST Transform Algorithm    CDATA    #REQUIRED >

<!ELEMENT XPath (#PCDATA) >
```

**Skema XML 8 Skema Elemen DigestMethod**

```
Schema Definition:

<element name="DigestMethod" type="ds:DigestMethodType"/>
```

```
<complexType name="DigestMethodType" mixed="true">
  <sequence>
    <any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

DTD:

<!ELEMENT DigestMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST DigestMethod Algorithm CDATA #REQUIRED >
```

**Skema XML 9 Skema Elemen DigestValue**

```
Schema Definition:

<element name="DigestValue" type="ds:DigestValueType"/>
<simpleType name="DigestValueType">
  <restriction base="base64Binary"/>
</simpleType>

DTD:

<!ELEMENT DigestValue  (#PCDATA)  >
<!-- base64 encoded digest value -->
```

**Skema XML 10 Skema Elemen KeyInfo**

```
Schema Definition:

   <element name="KeyInfo" type="ds:KeyInfoType"/>
   <complexType name="KeyInfoType" mixed="true">
     <choice maxOccurs="unbounded">
       <element ref="ds:KeyName"/>
       <element ref="ds:KeyValue"/>
       <element ref="ds:RetrievalMethod"/>
       <element ref="ds:X509Data"/>
       <element ref="ds:PGPData"/>
       <element ref="ds:SPKIData"/>
       <element ref="ds:MgmtData"/>
       <any processContents="lax" namespace="##other"/>
       <!-- (1,1) elements from (0,unbounded) namespaces -->
     </choice>
     <attribute name="Id" type="ID" use="optional"/>
   </complexType>
   DTD:

   <!ELEMENT KeyInfo (#PCDATA|KeyName|KeyValue|RetrievalMethod|
               X509Data|PGPData|SPKIData|MgmtData %KeyInfo.ANY;)* >
```

```
   <!ATTLIST KeyInfo
    Id  ID   #IMPLIED >
```

**Skema XML 11 Skema Elemen Object**

```
Schema Definition:

<element name="Object" type="ds:ObjectType"/>
<complexType name="ObjectType" mixed="true">
  <sequence minOccurs="0" maxOccurs="unbounded">
    <any namespace="##any" processContents="lax"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
  <attribute name="MimeType" type="string" use="optional"/>
  <attribute name="Encoding" type="anyURI" use="optional"/>
</complexType>

DTD:

<!ELEMENT Object
  (#PCDATA|Signature|SignatureProperties|Manifest %Object.ANY;)* >
<!ATTLIST Object
  Id          ID      #IMPLIED
  MimeType    CDATA   #IMPLIED
  Encoding    CDATA   #IMPLIED >
```

## Daftar Pustaka


[1] IBM, *Introducing XML Canonical Form : Making XML Suitable for regression testing, digital signatures, and more*. http://www-128.ibm.com/developerworks/xml/library/x-c14n/Introducing XML canonical form.htm, diambil tanggal 4 Januari 2005 pukul 16:15

[2] Simon, Ed et. al. *An Introduction to XML Digital Signatures.* http://www.xml.com/pub/a/2001/08/08/xmldsig.html, diambil tanggal 3 Januari 2005 pukul 16:00

[3] W3C Recommendation, *XML – Signature Syntax and Processing.* http://www.w3.org/TR/xmldsig-core/, diambil tanggal 3 Januari 2005 pukul 16:00

[4] W3C Recommendation, *Canonical XML Version 1.0*, http://www.w3.org/TR/2001/REC-xml-c14n-20010315/REC-xml-c14n-20010315.htm, diambil tanggal 4 Januari 2005 pukul 16:00