

Keamanan Web Service

Arya Adriansyah¹, Wahyudi Arifandi², Narenda Wicaksono³

Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

E-mail : if12003@students.if.itb.ac.id¹, if12022@students.if.itb.ac.id²,
narenda@gmail.com³

Abstrak

Web services menjadi sangat populer di *enterprise* karena kemampuannya dalam mengintegrasikan aplikasi-aplikasi yang berbeda platform dan mampu memperbaiki kelemahan dari *middleware* konvensional. Web services digunakan oleh *enterprise* misalnya untuk mengintegrasikan dan mengotomasi proses bisnis, *supply chain*, dan *customer relationship*. Saat sebuah *enterprise* ingin mengintegrasikan sistem bisnis dengan partnernya menggunakan internet, maka informasi yang dialirkan harus dipastikan dalam keadaan aman. Oleh karena itu keamanan menjadi isu yang sangat penting dalam implementasi web services. Dalam paper ini akan dijelaskan mengenai spesifikasi dari keamanan web services (*WS-Security*) dan bagaimana spesifikasi tersebut menanggulangi ancaman terhadap keamanan web services.

Kata kunci : Web services, keamanan, WS-Security

1. Pendahuluan

Saat ini web services menjadi sangat populer di *enterprise* karena kemampuannya dalam mengintegrasikan aplikasi-aplikasi yang berbeda platform [7]. Web Services adalah sebuah komponen layanan aplikasi yang dapat diakses melalui protokol terbuka [5, LIU04] yang memanfaatkan Web melalui Simple Object Access Protocol (SOAP) dengan bahasa Web Services Description Language (WSDL) dan teregistrasi dalam Universal Discovery Description and Integration (UDDI) [10].



Gambar 1 - Sebuah web services

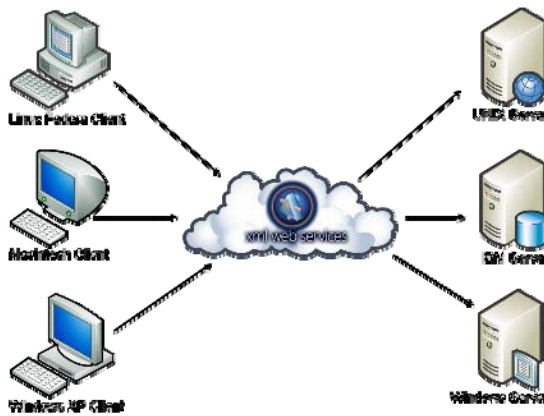
Web services mendukung komunikasi antar aplikasi dan integrasi aplikasi dengan menggunakan XML dan Web [6, 7, 10]. XML (*eXtensible Markup Language*) adalah sebuah standar untuk mendefinisikan data dalam format yang sederhana dan fleksibel.

Mengapa web services menjadi sangat populer saat ini? Jawabannya adalah karena web services mampu mengintegrasikan aplikasi yang berbeda platform secara lebih sederhana dan mampu memperbaiki kelemahan dari *middleware* konvensional.

Web services adalah komponen yang independen terhadap platform ataupun bahasa. Web services menggunakan web protokol (HTTP) yang sangat mendukung heterogenitas dan interoperabilitas serta

memudahkan integrasi. Selain itu web services mendukung koneksi *loosely coupled*, sehingga sebuah perubahan pada satu aplikasi tidak akan memaksa perubahan pada aplikasi yang lain. Sebuah web services memiliki *interface* berupa web API (*Application Programming Interface*) yang dapat dipanggil oleh suatu aplikasi untuk mengakses aplikasi yang mengimplementasikan layanan web services.

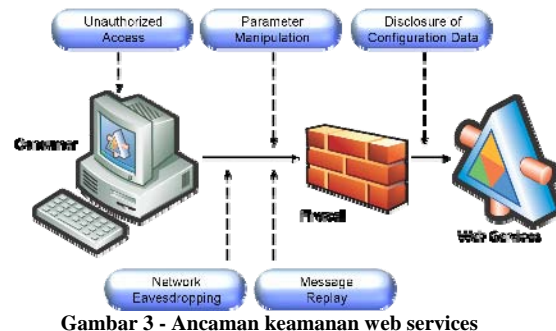
Interoperabilitas adalah prioritas utama sebuah enterprise dalam Enterprise Application Integration (EAI) dan Business-to-Business Integration (B2Bi). EAI dan B2Bi adalah permasalahan yang dihadapi oleh enterprise untuk mengintegrasikan berbagai macam aplikasi yang sudah ada. Web services dapat diimplementasikan sebagai komponen yang mendukung komunikasi antar aplikasi dalam *enterprise*.



Gambar 2 Heterogenoitas dan interoperabilitas web services

Web services saat ini semakin banyak digunakan oleh *enterprise* untuk memudahkan akses pada produknya, meningkatkan layanan ke konsumen dan ke *business partner* melalui internet atau *corporate extranet* [8]. Sebagai contoh

mengintegrasikan dan mengotomasikan proses bisnis, *supply chain*, dan *customer relationship*. Saat sebuah *enterprise* ingin mengintegrasikan sistem bisnisnya dengan partnernya menggunakan internet, informasi yang dialirkan harus dipastikan dalam kondisi aman [14]. Oleh karena itu keamanan menjadi isu yang sangat penting untuk keperluan tersebut. Dalam beberapa kasus, layanan keamanan berbasis Operating System dan Internet Information Services (IIS) dapat diandalkan. Akan tetapi lingkungan implementasi yang heterogen selalu menimbulkan celah-celah ancaman keamanan baru. Ancaman terhadap keamanan web services antara lain *unauthorized access*, *parameter manipulation*, *network eavesdropping*, *disclosure of configuration data*, dan *message replay*.



Gambar 3 - Ancaman keamanan web services

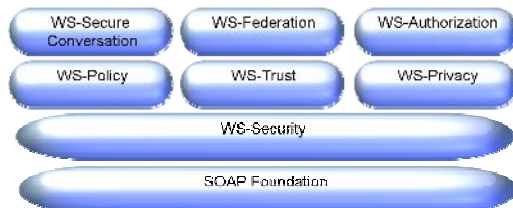
Jika nilai informasi yang dibawa oleh web services tinggi, maka web services tersebut harus diamankan dari ancaman-ancaman diatas.

2. Web Services Security Specification

Saat web services menggunakan secure connection seperti Secure Sockets Layer (SSL) atau Transport Layer Security (TLS) membangun koneksi *end to end* yang aman tidak akan sulit [9]. Akan tetapi jika terdapat perantara pada komunikasi

tersebut, maka dibutuhkan protokol yang memiliki tingkat keamanan yang tinggi.

Berikut adalah spesifikasi dari keamanan web services. Spesifikasi berikut termasuk message security model (*WS-Security*) yang memberikan dasar bagi spesifikasi yang lain [13]. Kemudian di atasnya terdapat lapisan khusus yang menangani masalah policy, yaitu *WS-Policy*, *WS-Trust* dan *WS-Privacy*. *WS-Policy* mendeskripsikan kemampuan dan batasan dari keamanan, *WS-Trust* mendeskripsikan framework yang digunakan, dan *WS-Privacy* mendeskripsikan model privasi untuk web services. Lapisan tersebut memberikan fondasi untuk keamanan *WS-Security*.



Gambar 4 - Spesifikasi web services security

Diatas lapisan policy terdapat lapisan yang memberikan fondasi, yaitu *WS-SecureConversation*, *WS-Federation*, dan *WS-Authorization*. *WS-SecureConversation* yang mengelola dan mengautentifikasi pesan, *WS-Federation* yang mengelola relasi dan lingkungan yang heterogen, dan *WS-Authorization* yang mengelola otorisasi data.

3. Web Services Security

3.1 Pendahuluan

Web Services Security (WS-Security) merupakan model pengamanan pesan SOAP yang menjadi dasar bagi *web services security specification* lainnya. *WS-*

Security berurusan dengan integritas pesan dan kerahasiaan isi pesan SOAP. Selain itu, *WS-Security* juga mengatur cara menyisipkan *security token* dalam pesan SOAP dalam bentuk plain teks maupun dalam bentuk biner (seperti sertifikat X.509). *WS-Security* didesain sefleksibel mungkin terhadap tipe *security token* yang dapat disisipkan. *WS-Security* menyediakan keamanan pada pesan SOAP tanpa mempedulikan bagaimana pesan tersebut disalurkan ke penerima.

WS-Security dibangun berdasarkan teknologi-teknologi yang sudah ada sebelumnya. Dua teknologi yang menjadi pondasi utama *WS-Security* adalah *XML Signature* dan *XML Encryption*. *XML Signature* berperan dalam menjaga integritas pesan SOAP, sedangkan *XML Encryption* lebih berperan dalam menjaga kerahasiaan isi pesannya. *XML Signature* dapat juga dikombinasikan dengan *Security Token*.

3.2 Menjaga Integritas SOAP dengan *XML Signature* dan *Security Token*

Integritas pesan SOAP berarti penerima pesan SOAP dapat memastikan bahwa pesan yang diterimanya benar-benar berasal dari pihak tertentu tanpa ada perubahan sedikitpun pada isi pesannya. Dalam *WS-Security*, integritas pesan SOAP dijaga menggunakan *XML Signature* yang terkadang dilengkapi juga dengan *security token*.

XML Signature merupakan implementasi *digital signature* dalam XML. *XML Signature* digunakan dalam *WS-Security*, sebab pesan SOAP pada dasarnya merupakan sebuah dokumen XML. Prinsip yang digunakan dalam membuat *XML Signature* secara umum

sama dengan prinsip pembuatan *digital signature*.

Proses untuk membangkitkan *XML Signature* dibagi menjadi dua komponen, yakni pembangkitan *reference* dan pembangkitan *signature*:

1. Pembangkitan *reference*

Proses ini terdiri atas proses iterasi terhadap seluruh objek data yang akan diberi *signature* dan menghitung nilai *hash*. Sebelum pemrosesan, lokasi objek data yang akan diproses dan algoritma yang digunakan dalam menghitung nilai *hash* didefinisikan dalam tag *Reference* yang terletak dalam dokumen XML. Setelah pemrosesan, tag *Reference* ditambahkan dengan elemen *DigestValue* yang berisi nilai *message digest* yang telah dihitung sebelumnya.

```
<Reference
URI="http://www.foo.com/securePage.html"
>

  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xml
dsig#sha1" />

  <DigestValue>60NvZvtdTB+7UnlLp/H24
</DigestValue>

</Reference>
```

Kode 1. Contoh tag *Reference*

2. Pembangkitan *signature*

Proses ini baru dilakukan setelah elemen *Reference* diciptakan. Proses ini dimulai dengan pembentukan elemen *SignedInfo*. *SignedInfo* berisi *method* yang akan digunakan dalam mengubah bentuk XML menjadi bentuk kanonik harus ditentukan terlebih dahulu. *Method* ini disimpan dalam tag *CanonicalizationMethod*. Selain itu, *SignedInfo* juga mengandung subelemen *SignatureMethod*. Atribut subelemen ini menentukan lokasi algoritma

kunci publik-privat mana yang akan digunakan dalam pembangkitan *signature*. Elemen *reference* yang sudah dibangkitkan sebelumnya juga dimasukkan kedalam subelemen *SignedInfo*.

```
<SignedInfo>
  <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2000/WD-
xml-c14n-20001011" />

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

  <Reference
URI="http://www.foo.com/securePage.html"
>

    <DigestMethod
Algorithm="http://www.w3.org/2000/09/xml
dsig#sha1" />

    <DigestValue>60NvZvtdTB+7UnlLp/H24
</DigestValue>

  </Reference>
</SignedInfo>
```

Kode 2. Contoh tag *SignedInfo* dalam *XML Signature*

Setelah elemen *SignedInfo* terbentuk, elemen tersebut akan diubah menjadi bentuk kanonik menggunakan algoritma yang ditunjukkan *CanonicalizationMethod*. Bentuk kanonik elemen tersebut lalu di-*hash* dengan fungsi yang ditunjukkan subelemen *DigestInfo* pada subelemen *Reference*. Setelah itu, dihitung nilai *signature* menggunakan algoritma yang ditunjukkan subelemen *SignatureMethod* dengan masukan nilai hasil *hash* tersebut.

```
<SignatureValue>
hTHQJyd3C6ww/OJz07P4bMOgjqBdznSUOsCh6P+0
MpF69w2tln/PFLdx/EP4/VKX
</SignatureValue>
```

Kode 3. Contoh tag *SignatureValue* dalam *XML Signature*

Setelah didapat nilai *signature*, bungkus *SignedInfo* dan *SignatureValue* dalam satu elemen *Signature*. Elemen inilah yang disebut dengan *XML Signature*.

Untuk melakukan validasi terhadap *XML signature*, dilakukan langkah-langkah sebagai berikut:

1. Validasi *reference*

Validasi ini berguna untuk memastikan bahwa objek yang ditunjukkan oleh elemen *Reference* tidak berubah. Pertama-tama, dilakukan perubahan elemen *SignedInfo* menjadi bentuk kanonik dengan algoritma yang terdapat dalam subelemen *CanonicalizationMethod*. Setelah itu, ambil seluruh data yang ditunjukkan oleh elemen *Reference*. Untuk setiap data, lakukan proses *hash* dengan algoritma yang ditunjukkan *DigestMethod* dan dibandingkan hasilnya dengan nilai elemen *DigestValue*. Jika terdapat perbedaan, berarti proses validasi gagal.

2. Validasi *signature*

Proses ini baru dijalankan bila validasi *reference* berhasil dilakukan. Tujuan validasi ini selain memastikan ulang tidak terjadi perubahan pada isi pesan juga untuk memastikan bahwa pesan benar-benar berasal dari pengirim tertentu. Elemen *SignedInfo* diubah menjadi bentuk kanonik lalu di-*hash* dengan algoritma yang terdapat pada elemen *DigestMethod*. Hasil *hash* dibandingkan dengan hasil dekripsi *SignedInfo* menggunakan algoritma *SignatureMethod* dan kunci publik yang disediakan (biasanya dalam tag *KeyInfo* yang menjadi subelemen *Signature*). Jika cocok, proses validasi berhasil.

Dalam praktiknya, penerapan *XML Signature* dalam *WS-Security* sering memanfaatkan *security token*. *Security token* adalah segala artifak keamanan yang disisipkan dalam pesan *SOAP*. *Security token* biasa digunakan untuk keperluan

otentikasi atau otorisasi. Tipe *security token* yang ada diantaranya sertifikat X.509, asersi SAML, token Kerberos, token XML, dan token XrML.

Security token dalam pesan *SOAP* digunakan untuk menyimpan bagian kunci publik yang selanjutnya digunakan untuk mengecek keabsahan *digital signature* yang terdapat di dalam pesan *SOAP*. Pada sertifikat X.509 yang dimasukkan kedalam pesan *SOAP*, *security token* dimasukkan dalam tag *BinarySecurityToken* yang menjadi subelemen dari elemen *Security* (Lihat Kode 3).

```
<S:Envelope>
  <S:Header>
    <wsse:Security>
      <wsse:BinarySecurityToken
        ValueType="wsse:X509v3"
        EncodingType="wsse:Base64Binary"
        wsu:Id="X509Token">
        F1gEZzCRF1EgILBAGIQEmtJZc0rqrKh5i
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
            Algorithm="
            http://www.w3.org/2001/10/xml-exc-c14n#"
          />
          <ds:SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xml
            dsig#rsa-sha1" />
          <ds:Reference URI="#body">
            <ds:Transforms>
              <ds:Transform
                Algorithm="
                http://www.w3.org/2001/10/xml-exc-c14n#"
              />
            </ds:Transforms>
            <ds:DigestMethod
              Algorithm="http://www.w3.org/2000/09/xml
              dsig#sha1" />
            <ds:DigestValue>EULddytSo1
            </ds:DigestValue>
            </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>
            XLdER8=ErToEb11/vXcmZNNjPOV...
          </ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference
                URI="#X509Token" />
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
```

```

    </ds:Signature>
  </wsse:Security>
</S:Header>
<S:Body wsu:Id="body">
  <StatusRequest
xmlns="http://www.myCompany.com/Order">
  <OrderNumber>1234</OrderNumber>
  </StatusRequest>
</S:Body>
</S:Envelope>

```

Kode 4. Contoh pesan SOAP lengkap yang menggunakan XML Signature dan Security Token

Pesan SOAP yang dikirim dapat divalidasi integritasnya menggunakan metode yang sama dengan metode validasi *XML Signature*. *Signature* diletakkan pada bagian *header* SOAP dan dibuat oleh pihak yang memberi *response Web Services*. Elemen yang diproses dalam *XML Signature* tidak hanya berupa *body* dari SOAP, tapi juga sasmpai ke *header*. Ini untuk menjaga supaya bagian *header* pun tidak mengalami perubahan.

3.3 Kerahasiaan Pesan SOAP dengan XML Encryption

XML Encryption merupakan teknologi yang fleksibel untuk mengenkripsi seluruh atau sebagian dokumen XML. *XML Encryption* akan membungkus elemen yang dienkripsi menggunakan tag XML penanda. Dalam tag XML penanda tersebut terdapat kunci enkripsi, petunjuk metode yang digunakan unuk melakukan enkripsi, chiperteks, dan properti tambahan lainnya. Tag XML yang menjadi penanda tersebut biasanya berupa tag *EncryptedData*. Skema XML tersebut dapat dilihat pada Kode 5.

```

<EncryptedData Id? Type? MimeType?
Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey?>
    <AgreementMethod?>
    <ds:KeyName?>
    <ds:RetrievalMethod?>
    <ds:*?>

```

```

</ds:KeyInfo?>
<CipherData>
  <CipherValue?>
  <CipherReference URI??>?
</CipherData>
<EncryptionProperties?>
</EncryptedData>

```

Kode 4. Skema XML Encryption

Elemen utama dari *EncryptedData* adalah elemen *EncryptionMethod* dan *CipherData*. Elemen

EncryptionMethod berisi pointer ke algoritma yang digunakan dalam enkripsi, sementara elemen *CipherData* berisi chiperteks (menggunakan tag *CipherValue*) atau pointer ke informasi chiperteks (menggunakan tag *CipherReference URI*).

Untuk mendeskripsikan elemen yang dienkripsi, terdapat informasi *Type*, *Encoding*, dan *MimeType*. *Type* adalah penanda plainteks apa yang terenkripsi. Isi *Type* berbeda berdasarkan masuk tidaknya tag XML dalam proses enkripsi. Meskipun tag *Type* bersifat opsional, pada praktiknya keberadaan tag ini sangat dibutuhkan dalam proses dekripsi. *Encoding* dan *MimeType* adalah elemen opsional.

Elemen *KeyInfo* menyediakan informasi mengenai kunci enkripsi sebagaimana fungsi elemen *KeyInfo* dalam *XML Signature* dan digunakan dalam kunci simetri. *KeyInfo* pada *XML Encryption* berisi informasi mengenai *secret key* (bukan *public key* seperti dalam *XML Signature*), sehingga kunci tidak disisipkan tanpa pengamanan tambahan. Pengamanan tambahan yang dimaksudkan berupa enkripsi pada kunci dengan algoritma tertentu yang secara opsional dapat dituliskan dalam tag *RetrievalMethod*. Elemen yang harus ada dalam tag *KeyInfo* hanyalah *EncryptedKey*.

XML Encryption diimplementasikan dalam pesan SOAP berupa tag `EncryptedKey` dan `EncryptedData`. Tag `EncryptedKey` diletakkan sebagai subelemen tag `Security` pada bagian *header* SOAP. Sementara `EncryptedData` dapat diletakkan dalam *header* maupun dalam *body* SOAP. Untuk plaintext berupa data dengan tipe selain teks, tag `EncryptedData` diletakkan dalam *header* SOAP. Berikut ini contoh *XML Encryption* dalam pesan SOAP:

```
<S:Envelope>
  <S:Header>
    <wsse:Security>
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod
Algorithm="..." />
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
EncodingType="wsse:Base64Binary"
Value="F2JfLa0GXsq..." />
          </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:ReferenceList>
        <xenc:DataReference
URI="#body" />
      </xenc:ReferenceList>
    </wsse:Security>
  </S:Header>
  <S:Body>
    <xenc:EncryptedData Id="body">
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Kode 5. Contoh *XML Encryption* dalam SOAP

4. Web Services Policy (WS-Policy)

WS-Policy menyediakan cara bagi penyedia *web services* untuk

mendeskripsikan kebutuhan dan kemampuan yang harus dimiliki suatu *client* untuk mengakses *web services* yang disediakan. Kebutuhan akan *WS-Policy* dipicu dari kebutuhan keamanan. Sebagai contoh, untuk mengetahui apakah suatu *web services* membutuhkan masukan kunci privat dari *client*, suatu *client* dapat melihat *WS-Policy* yang dimiliki *web services* tersebut.

Dalam kaitannya dengan *WS-Security*, beberapa hal yang menjadi peranan *WS-Policy* yaitu mendeskripsikan kepada *client*:

1. Perlu tidaknya masukan *security token* dengan tipe tertentu
2. Perlu tidaknya pesan didekripsi dan dimana lokasi kunci
3. Ada tidaknya mekanisme terkait *signature*

Saat ini, belum ada spesifikasi yang baku mengenai *WS-Policy*. Sebagian pihak mencoba mengimplementasikan *WS-Policy* sebagai bagian tambahan dalam WSDL. Namun, banyaknya aspek yang terkait dengan *WS-Policy* (*WS-Policy* bisa dibuat berbeda untuk setiap pesan yang ada dalam satu buah *web services*) menyebabkan *WS-Policy* akan menjadi begitu besar dan dinamis sehingga kemungkinan akan berdiri sendiri sebagai pengganti WSDL yang spesifikasinya sudah ada saat ini.

5. Web Services Trust (WS-Trust)

WS-Trust mendeskripsikan model untuk membentuk kepercayaan dalam hubungan langsung maupun tidak langsung (melibatkan pihak ketiga). Spesifikasi *WS-Trust* akan menggambarkan bagaimana kepercayaan (*trust*) langsung pada pihak tertentu dapat digunakan sebagai dasar kepercayaan bagi pihak lain. Model sekuriti

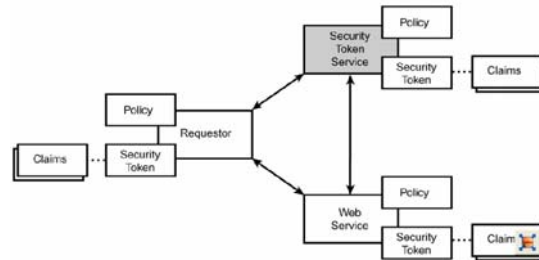
yang didefinisikan dalam WS-Trust didasarkan pada sekumpulan data yang disediakan oleh *request* yang masuk (sebagai contoh yaitu nama, kunci, dan *permission*). Jika suatu *request* sampai tanpa data tersebut, *service* akan mengabaikan atau menolak *request* tersebut. WS-Trust ditujukan supaya pihak yang menggunakan *web services* benar-benar pihak yang mempunyai kewenangan.

Model WS-Trust sangat mirip dengan model *Public Key Infrastructure* (PKI). Pihak yang melakukan *request* (selanjutnya disebut *requestor*) dapat membuktikan bahwa ia dapat dipercaya dengan menyisipkan *security token* milik pihak lain yang terpercaya dan *signature* yang membuktikan bahwa *security token* benar-benar dimiliki oleh pihak yang terpercaya. Jika *requestor* tidak memiliki token yang dibutuhkan, ia dapat menghubungi *Security Token Service* untuk mendapatkan token yang dimaksud.

Pada sisi pemberi *web services*, pemeriksaan kewenangan *requestor* dilakukan dengan langkah-langkah sebagai berikut:

1. Verifikasi bahwa isi *security token* yang dimasukkan cocok dengan *WS-policy* tertentu yang disediakan
2. Verifikasi bahwa *security token* yang disediakan oleh *requestor* sesuai dengan *signature*.
3. Lakukan verifikasi terhadap pihak yang memberikan *security token* pada *requestor*. Proses ini bisa dilakukan dengan melakukan pemeriksaan terhadap *security token* pihak pemberi. Mekanisme verifikasi serupa dengan mekanisme PKI. Proses verifikasi dapat dilanjutkan sampai pada level *root* atau sampai pihak yang dipercaya sebagai pemberi *security token*.

Jika seluruh verifikasi ini berhasil, barulah *request* diproses oleh pemberi *service*. Sebagai catatan, pemberi *security token* bisa saja merupakan layanan *web services* lain, sehingga proses verifikasi dapat dilakukan sepenuhnya oleh mesin tanpa intervensi manusia.



Gambar 5 Model WS-Trust untuk menyediakan broker trust Security Token Service yang mendukung WS-Security

6. Web Services Privacy

Siapa pun yang membuat atau memanfaatkan sebuah komponen web services harus mendeklarasikan kebijakan privasinya [13]. Untuk mendefinisikan kebutuhan privasi, terlebih dahulu harus didefinisikan tipe data yang terlibat [16]. Kebijakan privasi dideklarasikan pada *WS-Privacy*. *WS-Privacy* bertujuan untuk meyakinkan privasi dengan merekomendasikan kebijakan *access-control* pada atribut informasi [12].

Dengan menggunakan kombinasi dari *WS-Policy*, *WS-Security*, and *WS-Trust* kita dapat mendeklarasikan kebijakan privasi kita. Spesifikasi ini akan menjelaskan pemodelan tentang penyertaan *privacy language* dalam deskripsi *WS-Policy* dan bagaimana *WS-Security* dapat digunakan untuk mengasosiasikan klaim pada sebuah pesan. Akhirnya spesifikasi ini akan mendeskripsikan bagaimana mekanisme *WS-Trust* dapat digunakan untuk

mengevaluasi klaim privasi untuk kedua pihak yang berhubungan dengan menggunakan web services.

7. Web Services Secure Conversation

WS-Secure Conversation mendefinisikan *security context* diantara dua pihak yang berorganisasi berdasarkan atas *shared secret* seperti *symmetric encryption*. Pihak yang berkomunikasi saling berbagi (shared) sebuah *security context* di dalam suatu sesi. *Session key* diturunkan dari *shared secret* dan digunakan untuk mendekripsi setiap pesan pada komunikasi tersebut. *Security context* direpresentasikan sebagai sebuah tipe *security token* yang baru

Beberapa mekanisme untuk autentikasi dan *confidentiality* pesan membutuhkan komputasi yang kompleks. Banyak juga teknik-teknik enkripsi yang menuntut adanya *resource* pemrosesan yang besar. *Costs* ini pada umumnya tidak dapat dihindari ketika pesan diamankan secara individual. Namun bagaimanapun saat dua buah *web services* saling bertukar informasi informasi dalam jumlah yang banyak, pendekatan yang lebih efisien dan *robust* untuk *confidentiality* pesan daripada yang terdefinisi pada *WS-Security* tersedia (*Security Context Token* atau SCT).

Ada tiga cara dalam melakukan *establishing* suatu *security context* antara dua pihak dalam sebuah komunikasi yang aman, yaitu:

1. Sebuah *security token service* dapat menciptakan komunikasi tersebut, dan kemudian melakukan inisiasi pihak yang terlibat dalam komunikasi tersebut.
2. Salah satu dari pihak yang berkomunikasi menciptakan *security*

context dan memprogasinya melalui sebuah pesan kepada pihak lain yang akan berkomunikasi dengan pihaknya.

3. *Security context* diciptakan selama negosiasi dan pertukaran pesan dilakukan.

Web service memilih pendekatan yang paling sesuai dengan kebutuhannya. *Security context* bisa diperbaharui apabila memang diperlukan. Sebuah contoh yang mengharuskan *update* terhadap sebuah *security context* adalah kebutuhan untuk memperpanjang *context's expiration time*.

Sebuah *security context token* terdiri atas sebuah *shared secret* yang digunakan untuk menandatangani dan / atau melakukan enkripsi pesan. Saat menggunakan *shared secret*, pihak-pihak yang berkomunikasi dapat memilih untuk menggunakan pola penurunan kunci yang berbeda. Sebagai contoh. Empat buah kunci dapat diturunkan sehingga dua pihak yang berkomunikasi dapat menandatangani dan melakukan enkripsi dengan menggunakan kunci yang berbeda

Untuk menjaga bahwa kunci yang digunakan tidak *out of date* dan untuk mempertahankan level keamanan, diperlukan penggunaan suatu urutan penurunan kunci. *Securing sessions* menggunakan pendekatan ini.

Spesifikasi *WS-Secure Conversation* mendefinisikan mekanisme untuk mengindikasikan pemilihan *derivation* yang akan digunakan terhadap pesan yang diberikan. Masing-masing algoritma *derivation* diidentifikasi melalui URI.

8. Web Services Federation

Federation adalah koleksi domain yang dipercaya untuk membuat *mutual pair-wise trust*. Level kepercayaan bisa jadi berbeda-beda, namun secara tipikal akan menginkutsertakan *autentikasi* dan *authorisasi*.

WS-Federation mendeskripsikan sebuah model untuk mengintegrasikan mekanisme pengamanan yang tidak kompatibel, atau mekanisme yang bersesuaian yang dikembangkan pada domain yang berbeda.

Sebagai contoh, apabila setiap rekan bisnis IBM mengimplementasikan identitas infrastruktur berbasis PKI, atau jika satu diantara rekannya tersebut mengimplementasikan sistem *karberos*, spesifikasi *WS-Federation* akan menawarkan *roadmap* untuk menaplikasikan teknologi *web service* untuk menghubungkan sistem-sistem tersebut.

Application security membutuhkan mekanisme tambahan sebagaimana yang telah diterangkan sebelumnya. Identitas, sebagai contohnya adalah valid pada sebuah domain yang dipercaya. Agar layanan pada domain terpercaya yang berbeda dapat melakukan validasi identitas, mekanisme yang bersesuaian diperlukan. *WS-Federation* mendefinisikan mekanisme untuk mengaktifkan identitas, atribut, autentikasi, dan *authorisasi information sharing* pada berbagai domain yang dipercaya.

Dengan menggunakan mekanisme ini, *multiple security domains* dapat melakukan federasi dengan mengizinkan *brokering trust of identities*, atribut, dan autentikasi diantara beberapa *web service* yang berpartisipasi.

Spesifikasi ini melakukan *extend* model *WS-Trust* agar mengizinkan atribut dan *pseudonyms* diintegrasikan kedalam mekanisme *token issuance*, dihasilkan di

dalam sebuah mekanisme *multi-domain identity mapping*. Mekanisme ini mendukung *single sign on*, *sign out* dan *pseudonyms*, dan mendeskripsikan aturan spesialisasi *services*, untuk atribut dan *pseudonyms*.

Variasi *requirement* yang besar dapat dialamatkan terhadap identitas federasi. Contoh yang bersesuaian dengan hal ini adalah seorang *employee* dengan *employer*. Dalam kasus ini, *jane* dari perusahaan melakukan pembelian dari toko OfficeSupplyStore.com, perusahaan dan toko tersebut telah memiliki kontrak pembelian. Karena identitas *Jane* berasosiasi dengan perusahaan A, ia bisa melakukan *authorisasi* pembelian berdasarkan kontrak yang telah dibuatnya.

Contoh lainnya adalah melakukan *mapping single person* ke *multiple pseudonyms*. Joe dikenali sebagai pekerja dan memiliki identitas joe@company.com. Dia juga bisa memiliki identitas lain, seperti joe_bloggs@hotmail.com dan josephb@cornell.edu. Melalui *identity federation*, sistem dapat menentukan bahwa setiap identitas tersebut adalah milik Joe.

9. Web Services Authorization dan Autentication

Web Services authorization (*WS-Authorization*) membahas mengenai cara mengatur data otorisasi dan *policy* dalam otorisasi. Secara lebih spesifik, *WS-Authorization* mengatur cara memperoleh *security token* dan mengatur bagaimana *security token* ini diinterpretasikan di penyedia *web services*. Otorisasi erat kaitannya dengan otentikasi, sehingga muncul juga spesifikasi untuk otentikasi *web services*.

Cara otentikasi *web services* dapat dibagi kedalam tiga buah skema besar, yaitu: Otentikasi pada level *platform*, Otentikasi pada level pesan, dan Otentikasi pada level aplikasi. Saat ini fitur otentikasi pada level *platform* yang lengkap baru dimiliki oleh .NET dengan adanya *Windows Authentication*.

Setelah otentikasi, baru dapat dilakukan otorisasi. Saat ini otorisasi *web service* dapat diterapkan dengan tiga metode utama yaitu: *Programmctic Authorization*, *Web service endpoint Authorization*, dan *Web method Authorization*.

Programmatic Authorization berarti proses pengecekan otorisasi akan dilakukan secara *hardcoded* oleh pemrogram. *Web service endpoint Authorization* melakukan otorisasi berdasarkan lokasi *request*. Sedangkan *Web method Authorization* membedakan *service* yang bisa digunakan oleh pihak yang melakukan *request*. Ketiga

metode ini sudah didukung pada *platform* .NET.

10. Kesimpulan

Meskipun *Webservice* performansinya besar kemungkinan akan menggantikan media *middleware* lainnya seperti CORBA dan RMI, dari segi *security web service* masih belum matang. Terdapat perbedaan implementasi keamanan *web service* pada berbagai *platform* sehingga masih diperlukan suatu standar baku dalam mengimplementasikan *security* pada *web service*. Tanpa adanya standar baku, interoperabilitas *web service* akan terbatas pada satu *platform* saja.

Kendati pun demikian, implementasi *security web service* dalam satu *platform* sudah cukup menjanjikan untuk saat ini. Terutama *platform* .NET yang dimiliki Microsoft.

DAFTAR REFERENSI

- [1] Bilal Siddiqui, *Web Services Security*, <http://webservices.xml.com/pub/a/ws/2003/03/04/security.html?page=2>, diakses 29 Desember 2005 pukul 14.15
- [2] Della-Libera, Geovanni dkk, *Security in a Web Services World: a Proposed Architecture and Roadmap*, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp>, diakses tanggal 29 Desember 2005 pukul 14.00
- [3] Hartman, Bret dkk, *Mastering Web Services Security*, Wiley Publishing Inc, 2003.
- [4] Jothy Rosenberg dan David L. Remy, *Securing Web Services with WS-Security*, Sams Publishing, 2004.
- [5] Skonnard, Aaron (2002). *XML Files: The Birth of Web Services*. October 2002 issue of MSDN Magazine. Microsoft Corporation.
- [6] Oellermann, William L. (2001). *Architecting Web Services*. Apress: New York USA.
- [7] Manes, Anne T. (2003). *Web Services: A Manager Guide*. Pearson Education, Inc: Boston USA.
- [8] Microsoft (2004). *Building Secure Web Services*. Pattern and Practices Module. Microsoft Corporation.
- [9] Luis Felipe Cabrera, Christopher Kurt, Don Box (2004). An Introduction to the Web Services Architecture and Its Specifications. Web Services Technical Articles. Microsoft Corporation.
- [10] Wolter, Roger (2001). *XML Web Services Basics*. Web Services Technical Articles. Microsoft Corporation.
- [11] Bob Atkinson, etc (2002). Web Services Security (WS-Security). A Joint Paper from Microsoft, IBM dan VeriSign.
- [12] Liberty Alliance (2003). *Liberty Alliance & WS-Federation: A Comparative Overview*. Liberty Alliance Project White Paper.
- [13] Giovanni D. Libera, etc (2002). Security in a Web Services World: A Proposed Architecture and Roadmap. A Joint White Paper from IBM Corporation and Microsoft Corporation.
- [14] Snell, James (2002). *Securing Web Services*. IBM Corporation.
- [15] Donald F. Ferguson, etc (2003). *Secure, Reliable, Transacted Web Services: Architecture and Composition*. A Joint Paper from Microsoft dan IBM.
- [16] AlAzzawe, Abdul (2001). *IBM Video Central for e-Business- Web Services Privacy, Security and Accuracy*. IBM Corporation.