IMPLEMENTASI XML ENCRYPTION (XML Enc) MENGGUNAKAN JAVA

Tenia Wahyuningrum[1]
Program Studi D-III Teknik Telekomunikasi
Akademi Teknik Telkom Sandhy Putra Purwokerto
tenia@akatelsp.ac.id

ABSTRAK

Seiring dengan semakin luasnya penggunaan XML pada berbagai layanan di internet, yang penyebaran informasinya sebagian besar menggunakan infrastruktur jaringan umum, maka mulai muncul permasalahan mengenai kebutuhan akan keamanan data bagi informasi yang terkandung didalam sebuah dokumen XML. Salah satu caranya adalah dengan menggunakan teknologi XML Enc. Pada makalah ini akan dibahas mengenai cara menggunakan XML Enc menggunakan bahasa pemrograman java, khususnya menyandikan dokumen XML (enkripsi), dengan algoritma AES dan Tripple DES untuk menyandikan dokumen XML menggunakan kunci simetrik.

Kata kunci : XML, XML Enc, kunci simetrik

I. PENDAHULUAN

1.1 Latar Belakang

Data statistik jumlah pengguna internet di dunia sampai 31 Desember 2011 adalah 2.267.233.742. Bila dibandingkan dengan jumlah penduduk dunia 6.930.055.154, maka didapatkan prosentase sebesar 30,5%, dengan pertumbuhan internet pada tahun 2000-20011 mencapai angka 528,1%. Hal ini menunjukkan dalam jangka waktu 11 tahun, pengguna internet dunia telah naik lebih dari kali lipat 6,2 kali lipat[1].

Meningkatnya penggunaan internet antara lain disebabkan sifat *platform* yang terbuka (*open platform*) sehingga menghilangkan ketergantungan perusahaan pada sebuah vendor tertentu seperti jika menggunakan sistem yang tertutup (*proprietary systems*). *Open platform* juga mempermudah *interoperability* antar *vendor*. Selain alasan di atas, saat ini internet merupakan media yang paling ekonomis untuk digunakan sebagai basis sistem informasi. Perangkat lunak (*tools*) untuk menyediakan sistem informasi berbasis internet (dalam bentuk *server web*, ftp, gopher), membuat informasi (HTML *editor*), dan untuk mengakses informasi (*web browser*) telah banyak tersedia dan mudah didapat[2].

Teknologi internet merupakan kumpulan jaringan-jaringan *heterogen* yang saling terhubung dan memiliki karakter komputasi yang tersebar (*distributed system*), infrastruktur perangkat keras dan perangkat lunak (antara lain sistem operasi dan aplikasi) pada masing-masing jaringan lokal dapat beragam. Untuk itu dibutuhkan suatu perluasan standar baru, disamping TCP/IP, yang dapat mengadopsi perubahan kebutuhan, menggabungkan teknologi baru dengan teknologi yang sudah berjalan, dan dapat diterapkan secara modular untuk bagian-bagian yang diperlukan saja. Standar ini juga harus dapat bekerja sama dengan baik, bukan secara replikasi (dimana masing-masing *site*, misalnya menyimpan

duplikasi data yang sama), dan juga harus dapat cocok dengan teknologi baru untuk dapat membentuk sistem tersebar terbuka (*open distributed system*), penggabungan aplikasi, dan *content management*. Salah satu standar baru yang ditujukan untuk menjawab kebutuhan-kebutuhan tersebut diatas, adalah **eXtensible Markup Language (XML)**[3].

Seiring dengan semakin luasnya penggunaan XML pada berbagai layanan di internet, yang penyebaran informasinya sebagian besar menggunakan infrastruktur jaringan umum, maka mulai muncul permasalahan mengenai kebutuhan akan keamanan data bagi informasi yang terkandung didalam sebuah dokumen XML. Hal ini dikarenakan sebuah dokumen XML tersusun dari sekumpulan teks yang mudah dipahami pengguna atau program komputer. Berdasarkan kebutuhan tersebut, maka W3C (*World Wide Web Consortium*) berusaha mengembangkan beberapa spesifikasi tambahan untuk XML. Spesifikasi tersebut ditujukan kepada para pengguna untuk menggunakan fasilitas pengamanan data pada dokumen XML yang hendak didistribusikan. Sistem keamanan data yang terdapat pada spesifikasi XML tersebut dikenal dengan istilah ***XML Security***.

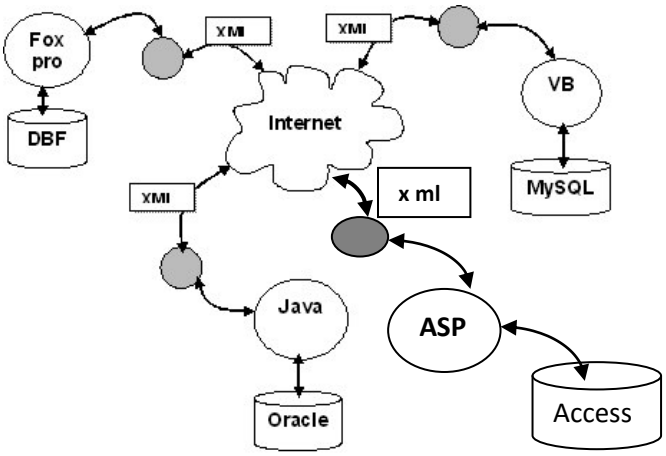
1.2 Tujuan

Tujuan penulisan untuk mengetahui proses penyandian dengan menggunakan XML *Encryption* untuk meningkatkan keamanan dokumen XML.

II. Tinjauan Pustaka

2.1 XML (eXtensible Markup Language)

XML singkatan dari eXtensible Markup Language, merupakan bahasa markup yang memiliki nilai lebih dibandingkan HTML. XML merupakan penyederhanaan dari SGML (*Standard Generalized Markup Language*) dan direkomendasikan oleh W3C pada 10 Febuari 1998. XML bukan merupakan pengganti HTML, namun merupakan pelengkap HTML. Masing-masing dikembangkan untuk tujuan yang berbeda. HTML digunakan untuk menampilkan informasi dan berfokus pada bagaimana informasi terlihat, XML mendeskripsikan susunan informasi dan berfokus pada informasi itu sendiri[4].



Gambar 1. Sistem dan Format data yang berbeda pada jaringan internet

Dalam sebuah jaringan internet, terdapat jaringan-jaringan kecil di dalamnya yang memiliki perbedaan sistem dan format data. Salah satu keunggulan XML adalah menyederhanakan pertukaran data dalam sistem dan format data yang berbeda-beda.

Jenis dokumen XML[5]

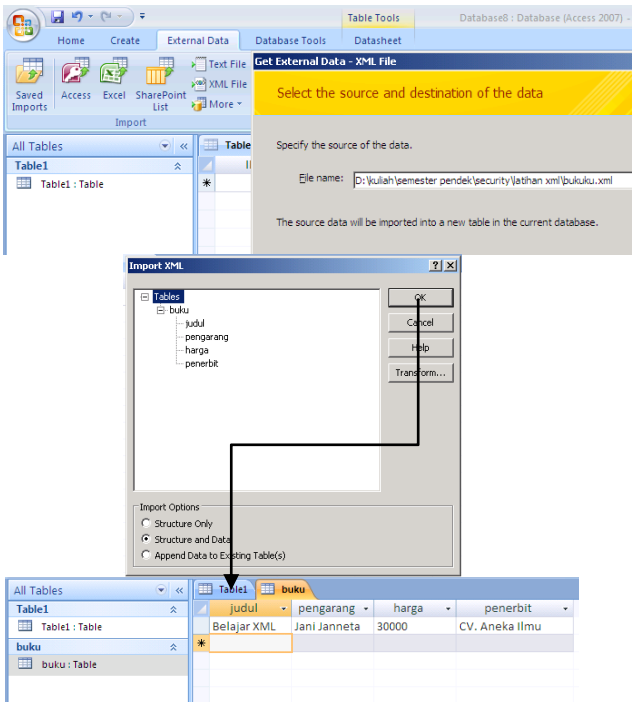
- **Invalid document**, tidak mengikuti aturan penulisan yang didefinisikan oleh spesifikasi XML. Jika seorang pengembang mendefinisikan aturan struktur suatu dokumen atau *schema*, dan dokumen tidak mengikuti aturan tersebut, maka dokumen tersebut juga invalid.
- **Valid document**, mengikuti aturan spesifikasi XML dan aturan DTD (*Document Type Definition*) baik secara internal maupun eksternal atau schema.
- **Well-formed document**, mengikuti aturan spesifikasi XML namun tidak memiliki DTD atau schema.

Contoh dokumen yang valid menggunakan DTD Internal

```
1  <?xml version="1.0"?>
2  <!DOCTYPE koleksibuku [
3  <!ELEMENT koleksibuku (buku)+>
4              <!ELEMENT              buku
(judul,pengarang,harga,penerbit*)>
5  <!ELEMENT judul (#PCDATA)>
6  <!ELEMENT pengarang (#PCDATA)>
7  <!ELEMENT harga (#PCDATA)>
8  <!ELEMENT penerbit (#PCDATA)> ]>
9
10 <koleksibuku>
11     <buku>
12           <judul>Belajar XML</judul>
13           <pengarang>Jani
Janneta</pengarang>
14           <harga>30000</harga>
15           <penerbit>CV.         Aneka
Ilmu</penerbit>
16     </buku>
17 </koleksibuku>
```

Kode 1. bukuku.xml dengan menggunakan DTD internal

Baris pertama dari contoh buku.xml adalah elemen deklarasi XML yang mendeskripsikan dokumen tersebut sebagai sebuah dokumen XML. Baris 2 sampai baris 8 merupakan *Document Type Definition* (DTD) yang menjelaskan elemen dari tabel buku. Elemen dari *code* diatas adalah judul, pengarang, harga dan penerbit. Isi dari judul adalah ” Belajar XML”, isi dari pengarang adalah ”Jani Janneta”, isi dari harga adalah ”30000”, isi dari penerbit adalah ”CV. Aneka Ilmu”. Dokumen XML tersebut dapat diimpor ke dalam format *database* menggunakan MS. Access (.mdb).



Gambar 2. Impor XML file ke dalam database MS. Access

Contoh dokumen yang valid menggunakan DTD Eksternal

```
<?xml version="1.0"?>
<!DOCTYPE koleksibuku SYSTEM
"note.dtd">
<koleksibuku><buku>
  <judul>Belajar XML</judul>
  <pengarang>Jani Janneta</pengarang>
  <harga>30000</harga>
```

```
  <penerbit>CV. Aneka Ilmu</penerbit>
</buku></koleksibuku>
```

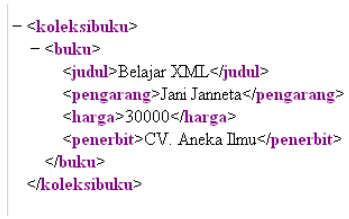
Kode 2. bukuku.xml dengan menggunakan DTD Eksternal

Dokumen bukuku.xml di atas merupakan contoh dokumen XML yang valid, karena dalam dokumen tersebut terdapat deklarasi <!DOCTYPE koleksibuku SYSTEM "note.dtd">, yang menyatakan bahwa struktur XML untuk koleksibuku harus mengikuti definisi DTD (*Document Type Definition*) yang tersimpan pada file note.dtd. Selain DTD, terdapat bahasa lain yang dapat digunakan untuk mendefinisikan struktur dokumen XML, antara lain XDR (*XML-data Reduced Language*) dan XSD (*XML Schema Definition*). Definisi struktur dan batasan terhadap struktur dokumen XML dapat diletakkan secara *internal* ataupun *eksternal*. Seperti pada contoh bukuku.xml di atas, definisi DTD didefinisikan secara eksternal. Berikut contoh definisi struktur untuk elemen root <koleksibuku> yang tersimpan dalam file note.dtd.

```
<!ELEMENT koleksibuku (buku)+>
<!ELEMENT buku
(judul,pengarang,harga,penerbit*)>
<!ELEMENT judul (#PCDATA)>
<!ELEMENT pengarang (#PCDATA)>
<!ELEMENT harga (#PCDATA)>
<!ELEMENT penerbit (#PCDATA)>
```

Kode 3. note.dtd

Jika dijalankan melalui *web browser*, akan terlihat seperti gambar 3.



Gambar 3. Dokumen XML setelah dijalankan melalui *web browser*

2.2 XML Security

Sebagai format untuk menukarkan informasi melalui internet, popularitas XML terus berkembang. Dan satu dari hal-hal penting yang berasosiasi dengan pertukaran informasi adalah keamanan. Tidak ada format pertukaran yang lengkap tanpa mekanisme untuk memastikan kemanan dan *reliability* dari informasi. Unit dasar untuk keamanan XML adalah *elemen*. *Encryption* lebih lanjut dapat menyaring spesifikasi tipe penyandian, apakah *elemen* atau *content*. *Element encryption* menyandikan seluruh *elemen*, termasuk atribut dan menimpanya dengan *EncryptedData element*. *Content encryption* hanya *element child nodes* yang disandikan, dan menimpanya dengan *EncryptedData element*[6].

Sebuah arsitektur keamanan dikembangkan dengan mendasari dirinya pada beberapa layanan keamanan berikut ini :

- Kerahasiaan (*confidentially*) memastikan hanya penerima yang berwenang yang dapat membaca bagian dari XML. Hal ini dapat dipastikan dengan melakukan penyandian XML. XML *Encryption* merupakan standar untuk melakukannya.

- Integritas (*integrity*) memastikan bawa XML tidak berubah selama dalam perjalanan dari sumber ke tujuan. Standar XML *Signature* mengizinkan pengirim untuk melampirkan *digital signature* pada *content*.
- Otentikasi (*authenticity*) adalah kemampuan untuk memastikan bahwa XML telah terkirim oleh orang yang menyatakan telah mengirimnya. XML *Signature* mengirim dengan *content help ensure* untuk memastikan pengirim. Penerima dapat melakukan validasi *digital signature* dengan kunci dari pengirim. Jika *digital signature* dinyatakan *valid*, kemudian identitas dikonfirmasi.
- Anti penyangkalan (*non repudiation*) digunakan untuk memastikan penerima tidak dapat menyangkal mengirim XML. XML *Signature* di *generate* dari *private key* pengirim dan ditambahkan pada XML untuk memastikan keaslian pengirim.

Standar XML *Security* ini memiliki beberapa standar inti yang dirancang untuk menyediakan teknologi yang kompatibel dengan XML untuk dapat memenuhi kebutuhan keamanan. Standar inti tersebut kemudian sebagian besar diterapkan untuk beberapa spesifikasi lain seperti *Web Services*, *Digital Rights Management - eXtensible Rights Markup Language 2.0 (XrML)*, *Privacy - Platform for Privacy Preferences (P3P)* dan *ebXML*.

Berikut standar-standar inti dari XML *Security*.

- XML *Digital Signature* untuk tanda tangan dan integritas,
- XML *Encryption* untuk kerahasiaan,
- Kemudian berdasar kedua standar tersebut terdapat standar lain sebagai komponen inti dari arsitektur XML *Security*, yaitu:
  - XML *Key Management* (XKMS) untuk manajemen kunci,
  - *Security Assertion Markup Language* (SAML) untuk pembuatan pernyataan otentikasi dan otorisasi, dan
  - XML *Access Control Markup Language* (XACML) untuk memulai aturan otorisasi.

Standard XML *Digital Signature* dan XML *Encryption* merupakan dua standar pokok dalam arsitektur XML *Security*, karena keduanya juga dipakai dalam komponen arsitektur XML *Security*. Dalam tulisan ini pembahasan hanya difokuskan pada XML *Encryption*.

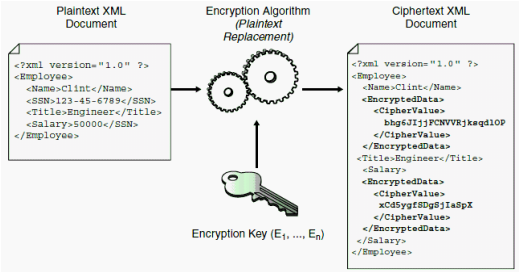
III. PEMBAHASAN XML ENCRYPTION (XML Enc)

3.1 Pengenalan XML *Encryption*

Inti dari XML *Encryption* adalah elemen <EncryptedData> yang didalamnya memuat seluruh informasi mengenai parameter-parameter yang digunakan dalam proses enkripsi. Dalam penggunaannya, elemen tersebut akan menggantikan simpul yang dienkripsi beserta seluruh simpul anak yang dimilikinya. Adapun sebuah proses enkripsi

dapat diterapkan pada beberapa macam simpul, antara lain[7]:

- Elemen beserta *tag* dari elemen tersebut
- Isi dari suatu elemen
- Seluruh isi dari sebuah dokumen XML



Gambar 4. Proses enkripsi pada dokumen XML

3.2 Syntax XML Encryption

Struktur elemen <EncryptedData> yang dikembangkan oleh W3C dalam spesifikasinya. Pada daftar tersebut dilengkapi dengan keterangan mengenai jumlah pemunculan yang diperbolehkan pada elemen tersebut. Beberapa tanda tersebut antara lain '?' (boleh tidak muncul atau hanya muncul satu kali), '+' (harus muncul minimal satu kali), dan '*' (boleh tidak muncul atau muncul beberapa kali).

```
<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. Syntax *XML Encryption*

Penjelasan masing-masing elemen

Elemen <EncryptedData> merupakan elemen teratas dari element XML Encryption. Di dalamnya terdapat informasi-informasi yang digunakan dalam proses enkripsi. Dalam dokumen XML yang terenkripsi, elemen ini akan menggantikan elemen atau pun isi elemen yang dienkripsi, sebaliknya pada proses dekripsi, elemen ini akan digantikan oleh elemen asli.

Sebuah Elemen <EncryptedType> memiliki beberapa attribute, antara lain:

- **Id**, berfungsi untuk memberikan sebuah tanda berupa teks untuk elemen tertentu. *Attribute* ini sering digunakan pada saat pengguna menggunakan proses *Super-Encryption* atau pun untuk menandai sebuah EncryptedKey agar mudah dikenali.
- **Type**, memberikan informasi mengenai tipe dari *plain text* yang telah dienkripsi. *Attribute* ini

  memiliki dua macam tipe *plain text*, yaitu '*element*' atau 'content'. Informasi ini dibutuhkan agar sistem dapat lebih mudah dalam mengembalikan data hasil proses dekripsi.

  Misal :

  `'http://www.w3.org/2001/04/XMLenc#Ele me nt',`

  `'http://www.w3.org/2001/04/XMLenc#Con tent'.`
- **MimeType**, data teks yang mendeskripsikan format data yang digunakan oleh *plain text*. Misal : 'image/png', 'text/XML'.

- **Encoding**, sebuah data berupa URI yang memberikan informasi mengenai metode encoding
  yang digunakan. Misal :
  'http://www.w3.org/2000/09/XMLdsig#base64'.

Elemen <EncryptionMethod> mendefinisikan algoritma yang digunakan pada proses enkripsi. Jika elemen ini tidak disertakan maka pengguna harus mengetahui algoritma yang dipakai. Nama algoritma yang digunakan disimpan pada *attribute* Algorithm yang nilainya bertipe URI. Misal :
'http://www.w3.org/2001/04/XMLenc#rsa-1_5' menunjuk bahwa algoritma yang digunakan adalah RSA-1.5.

Elemen <ds:KeyInfo> berasal dari spesifikasi XML *Digital Signature* yang berguna untuk menyimpan informasi mengenai kunci yang di gunakan untuk mengekripsi data yang disimpan pada bagian <CipherData>. Nama kunci yang digunakan disimpan pada elemen <ds:KeyName>.
Elemen <AgreementMethod> digunakan jika kunci perlu didistribusikan pada sebuah kelompok tertentu. Elemen ini menyimpan informasi mengenai algoritma yang digunakan serta kunci yang dibutuhkan untuk menciptakan kunci.

Elemen <CipherData> merupakan elemen pokok yang harus ada dalam sebuah elemen <EncryptedData>. Elemen ini berfungsi untuk menyimpan data yang sudah terenkripsi. Elemen ini memiliki dua macam simpul anak, yaitu *ciphervalue*, digunakan untuk menyimpan *cipher text* dalam bentuk teks *dan xenc:CipherReference*, digunakan jika *ciphertext* diletakan secara terpisah pada sebuah dokumen yang dapat diakses melalui alamat URI.

Elemen <EncryptionProperties> berfungsi untuk menyimpan informasi-informasi yang berhubungan dengan proses enkripsi, seperti : hari, tanggal, nomer seri *hardware* yang digunakan, dan sebagainya.

III.3 Mengelola XML *Encryption* menggunakan JAVA

Kriptografi dapat didefinisikan sebagai ilmu dan teknik untuk mengamankan data dengan menyandikan atau mentransformasikannya ke dalam format yang tidak dapat dikenali, kemudian mendekripsikannya kembali ke dalam format asal. Enkripsi lebih lanjut didefinisikan sebagai proses untuk membawa data (dikenal sebagai *cleartext* atau *plaintext*) dan mengubahnya menggunakan kunci kriptografi untuk menghasilkan *chipertext*, yang tidak dapat dikenali oleh pihak yang tidak berwenang. Dekripsi merupakan kebalikan dari enkripsi, didefinisikan sebagai proses untuk mengubah *chipertext* menjadi *cleartext* kembali[8].

Kriptografi Java dan Kunci Simetrik

JCE mendukung kunci simetrik melalui class API dan antarmuka dalam javax.cypto.*packages.

Dalam code Java dituliskan sebagai berikut.

```
 SecretKey key =
KeyGenerator.getInstance("DES").generate
Key();
```

Kemudian, gunakan Chiper class untuk membuat Chiper instance agar dapat melakukan enkripsi dan dekripsi data.

```
Cipher cipher =
Cipher.getInstance("DES");
```

Pada point ini, harus dipanggil metode init() untuk menspesifikasikan fungsi enkripsi dan dekripsi untuk menjalankannya. Terakhir, dipanggil metode doFinal() untuk melakukan fungsi kriptografi yang diinginkan. Kode menjelaskan inisialisasi Chiper instance untuk melakukan enkripsi byte array data menggunakan kunci simetrik sebelumnya.

Kriptografi Java dan Kunci Asimetrik

JCE mendukung kunci asimetrik melalui class API dan antarmuka yang ditemukan pada java.security.package. Dengan package ini, dapat di generate pasangan public/private key dengan ketentuan sebagai berikut.

1. Buat *instance* dari *KeyPairGenerator* untuk algoritma yang diinginkan.
2. Inisialisasi *instance KeyPairGenerator* dengan jumlah bit ukuran kunci yang diinginkan .
3. Panggil metode generateKeyPair() untuk melakukan generate pasangan key.

Dalam code Java dituliskan sebagai berikut.

```
   KeyPairGenerator generator =
KeyPairGenerator.getInstance("RSA");
   generator.initialize(1024);
   KeyPair keyPair =
generator.generateKeyPair();
```

Pemrograman XML *Encyption* menggunakan JAVA dan Apache XML Security

Class EncryptTool digunakan untuk membaca input dari sebuah file, kemudian melakukan enkripsi pada isi file, dan menyimpan file yang telah di enkripsi ke dalam *disk*. Tool menggunakan Apache XML framework untuk membuat dua kunci simetrik dengan tujuan mengenkripsi data file XML yang asli, dan mengenkripsi kunci menggunakan data file.

EncryptTool.java

Pertama, inisialisasi Apache XML framework, pada kasus ini, untuk memberi nilai default.

```
static
{ org.apache.xml.security.Init.init();}
```

Kemudian, gunakan metode di bawah untuk membaca dan melewatkan file XML yang akan dienkripsi. Metode membuat dan mengembalikan DOM *document*.

```
private static Document parseFile(String
fileName)throws Exception
{javax.xml.parsers.DocumentBuilderFactor
y dbf = javax.xml.parsers.
DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
javax.xml.parsers.DocumentBuilder db =
dbf.newDocumentBuilder();
Document document = db.parse(fileName);
return document; }
```

Generate key yang digunakan untuk enkripsi data, yaitu *Encryption Tool*. Kunci yang di generate perlu untuk disimpan dan mengangkut *data-encryption tool* dengan aman.

```
private static SecretKey
GenerateKeyEncryptionKey()
throws Exception      {
String jceAlgorithmName = "DESede";
KeyGenerator keyGenerator =
```

```
KeyGenerator.getInstance(jceAlgorithmNam
e);
SecretKey keyEncryptKey =
keyGenerator.generateKey();

         return keyEncryptKey;
      }
```

Untuk menyimpan *key-encryption key* gunakan metod menulis *key-encryption key* ke dalam file. *Decryption tool* akan mengembalikannya kemudian dan menggunakannya untuk mendekripsikan kembali data yang telah dienkripsi.

```
private static void storeKeyFile(Key
keyEncryptKey)
throws IOException
      {
byte[] keyBytes =
keyEncryptKey.getEncoded();
File keyEncryptKeyFile = new
File("keyEncryptKey");
FileOutputStream outStream =
new FileOutputStream(keyEncryptKeyFile);
outStream.write(keyBytes);
outStream.close();
System.out.println("Key encryption key
stored in: "
+keyEncryptKeyFile.toURL().toString());}
```

Kemudian, *generate data simetrik encryption key* menggunakan metode GenetateSymetricKey().

```
private static SecretKey
GenerateSymmetricKey()
throws Exception
{ String jceAlgorithmName = "AES";
KeyGenerator keyGenerator =
KeyGenerator.getInstance(jceAlgorithmNam
e); keyGenerator.init(128); return
keyGenerator.generateKey();      }
```

Terakhir, menulis dokumen yang telah dienkripsi ke dalam file.

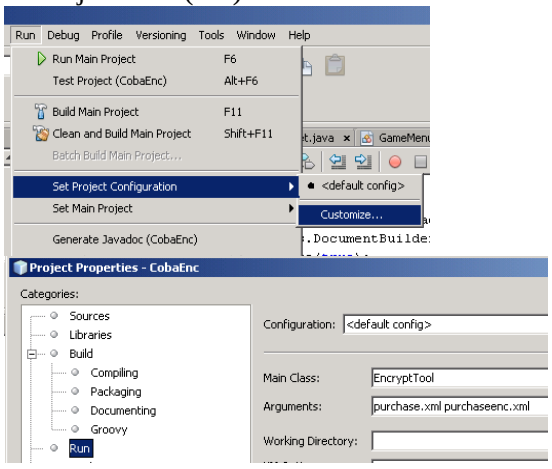
```
 private static void
writeEncryptedDocToFile(
Document doc, String fileName)
throws Exception
      {
File encryptionFile = new
File(fileName);
FileOutputStream outStream =
new FileOutputStream(encryptionFile);
TransformerFactory factory =
TransformerFactory.newInstance();
Transformer transformer =
factory.newTransformer();
transformer.setOutputProperty(
OutputKeys.OMIT_XML_DECLARATION, "no");
DOMSource source = new DOMSource(doc);
StreamResult result = new
StreamResult(outStream);
transformer.transform(source, result);
outStream.close();
System.out.println(
"Encrypted XML document written to: "
+ encryptionFile.toURL().toString());
      }
```

Contoh enkripsi seluruh isi dari dokumen XML

```
<?xml version='1.0'?>
<purchaseOrder>
       <Order>
              <Item>book</Item>
              <Id>123-958-74598</Id>
              <Quantity>12</Quantity>
       </Order>
       <Payment>
              <CardId>123654-8988889-
9996874</CardId>
              <CardName>visa</CardName>
              <ValidDate>12-10-
2004</ValidDate>
       </Payment>
</purchaseOrder>
```

Kode 5. purchase.xml

Program EncryptTool.java terlebih dahulu dikonfigurasi *project*, kemudian dikompilasi, dan dijalankan (run).



Gambar 5. *Setting* konfigurasi *project* EncryptTool.java

Pada arguments dituliskan nama dokumen xml (purchase.xml) yang akan di enkripsi kemudian nama dokumen xml (purchaseenc.xml) yang

akan dijadikan tempat menampung hasil enkripsi.

Contoh hasil eksekusi program EncryptTool.java dapat dilihat pada kode di bawah ini.

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseOrder>
   <xenc:EncryptedData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"

Type="http://www.w3.org/2001/04/xmlenc#C
ontent">
   <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xml
enc#aes128-
   cbc"
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"/>
<ds:KeyInfo
xmlns:ds="http://www.w3.org/2000/09/xmld
sig#">
<xenc:EncryptedKey
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">

<xenc:EncryptionMethodAlgorithm="http://
www.w3.org/2001/04/xmlenc#kw-
    tripledes"
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"/>
     <xenc:CipherData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
        <xenc:CipherValue
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">

66gFKmOVbYgHMYsQ6RB0JqAW/Vr2vpaO1tTnOQ3j
eWo=
        </xenc:CipherValue>
     </xenc:CipherData>
   </xenc:EncryptedKey>
</ds:KeyInfo>
   <xenc:CipherData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
      <xenc:CipherValue
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">

xjpkpO2yjZrI6VFZcJDkP0yeo1TRH0teYE+RBE7x
+IsM7Wsleab2cxXX8r1gRTJpAxNlWq

QtahhZnlM1dvphelljGTzUZGqnVSN5bYZ/iM8OTc
XE8xRacaWLO9lOJb+6ML2eTJnAwgp3d

sf/8ScZ96CIs/Kb3OUEIsrg0LZ5bXJ3iVkoFVCor
p5W3gukddIcaWuxhj0fd0gP1DeKS48J
```

```
k+F4oyQWuNEHYgRIcbkJjJvX4WsUoqPoe+iO2v2A
PsRtfdlCyLQ0YIFhb65pEbBsATPy5Jz

iz31PP0x5uDnBmdPHOLls37lJXQZssDaSDpSJ
      </xenc:CipherValue>
   </xenc:CipherData>
 </xenc:EncryptedData>
</purchaseOrder>
```

Kode 6. purchaseenc.xml

Contoh enkripsi elemen beserta tag dari dokumen XML

Pada contoh berikut, elemen dari tag <Payment> hingga </Payment> akan disandikan. Caranya sama dengan menyandikan keseluruhan dokumen, namun pada arguments setelah nama dokumen yang akan dijadikan tempat menyimpan hasil enkripsi (purchaseenc1.xml) ditambah kata Payment. Penulisan argument menjadi "purchase.xml purchaseenc1.xml Payment" (tanpa tanda petik).

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseOrder>
  <Order>
    <Item>book</Item>
    <Id>123-958-74598</Id>
    <Quantity>12</Quantity>
  </Order>
<Payment>
<xenc:EncryptedData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"
Type="http://www.w3.org/2001/04/xmlenc#C
ontent">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xml
enc#aes128-cbc"
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"/>
<ds:KeyInfo
xmlns:ds="http://www.w3.org/2000/09/xmld
sig#">
   <xenc:EncryptedKey
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
   <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xml
enc#kw-
    tripledes"
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"/>
```

```
   <xenc:CipherData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
     <xenc:CipherValue
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">

VF3VIX8LC69YVgAHUA7b8iX9e/+d84BXqihTCMPp
0Jc=
     </xenc:CipherValue>
   </xenc:CipherData>
   </xenc:EncryptedKey>
</ds:KeyInfo>

<xenc:CipherData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
<xenc:CipherValue
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">

Tlq0RDztvNvxk3zDlo7Oh9uk3lKK2Q+RR8MfwFt7
SvEb19eSLajt68ETaVPcp3J1zaskrnEmmNoB8Waa
S8OcnRymzO3fhTPL5Ny0vEvSaNYe1zI66zJn4B08
0UH6wnhyI0S+LhTH15OUI4GBSFCHFJ/f4r0yEY7v
tedy7E9pWMU=</xenc:CipherValue></xenc:Ci
pherData></xenc:EncryptedData>
</Payment>
</purchaseOrder>
```

Kode 7. purchaseenc1.xml

Contoh enkripsi isi dari sebuah elemen dokumen XML

Pada contoh berikut, isi dari elemen <CardId> akan disandikan. Caranya sama dengan menyandikan keseluruhan dokumen, namun pada arguments setelah nama dokumen yang akan dijadikan tempat menyimpan hasil enkripsi (purchaseenc2.xml) ditambah kata CardId. Penulisan argument menjadi "purchase.xml purchaseenc2.xml CardId" (tanpa tanda petik).

```
<?xml    version="1.0"    encoding="UTF-
8"?><purchaseOrder>
      <Order>
            <Item>book</Item>
            <Id>123-958-74598</Id>
            <Quantity>12</Quantity>
      </Order>
  <Payment>
    <CardId>
```

```
<xenc:EncryptedData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"
Type="http://www.w3.org/2001/04/xmlenc#C
ontent">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xml
enc#aes128-cbc"
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"/>
<ds:KeyInfo
xmlns:ds="http://www.w3.org/2000/09/xmld
sig#">
<xenc:EncryptedKey
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xml
enc#kw                          tripledes"
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#"/>

<xenc:CipherDataxmlns:xenc="http://www.w
3.org/2001/04/xmlenc#">
     <xenc:CipherValue
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
yQcXR7FpVgV4i5IEmxU/0NQa7Vv63fkffV4Oj4k1
6Xo=</xenc:CipherValue>
     </xenc:CipherData>
     </xenc:EncryptedKey>
     </ds:KeyInfo>
     <xenc:CipherData
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
     <xenc:CipherValue
xmlns:xenc="http://www.w3.org/2001/04/xm
lenc#">
NME1kc4JnTzi2QWj/t/cqaZRaoJBSmLhmtjBtBTg
4DkIV6zmsoAhvnU6tPOUtPS
     </xenc:CipherValue>
     </xenc:CipherData>
     </xenc:EncryptedData>
   </CardId>
<CardName>visa</CardName>
   <ValidDate>12-10-2004</ValidDate>
   </Payment>
</purchaseOrder>
```

Kode 8. purchaseenc2.xml

IV. KESIMPULAN

1. Standar keamanan dokumen XML antara lain menggunakan XML *Encryption*. Salah satu cara mengelola XML *Encryption* yaitu menggunakan bahasa pemrograman Java.
2. Dokumen XML dapat dienkripsi dengan 3 cara, yaitu elemen beserta *tag* dari

elemen tersebut, isi dari suatu elemen, dan seluruh isi dari sebuah dokumen XML.

3. Program EncryptTool.java menggunakan algoritma AES dan Tripple DES untuk menyandikan dokumen XML menggunakan kunci simetrik.

DAFTAR PUSTAKA

1. Internet Usage Statistics, http://www.internetworldstats.com/stats.htm. Diakses 12 Oktober 2012.
2. Rahardjo, Budi, Keamanan Sistem Informasi Berbasis Internet, http://budi.insan.co.id/courses/security/docs.html. Diakses tanggal 12 Oktober 2012.
3. Susanto, Budi, Pemrograman XML Security, http://budsus.files.wordpress.com/2007/08/xmlsecurity.pdf. Diakses tanggal 12 Oktober 2012.
4. Introduction to XML, http://www.w3schools.com/xml/xml_whatis.asp. Diakses tanggal 12 Oktober 2012.
5. Susanto, Budi, Pengantar XML, http://lecturer.ukdw.ac.id/budsus/webdb/xml.pdf. Diakses tanggal 12 Oktober 2012.
6. Verma, Manish. XML Security, Implements security layers, Part 1, https://www.ibm.com/developerworks/xml/library/x-seclay1. Diakses tanggal 12 Oktober 2012.
7. Verma, Manish. XML Security, Implements security layers, Part 2, https://www.ibm.com/developerworks/xml/library/x-seclay2. Diakses tanggal 12 Oktober 2012.
8. Hanson, Jeff, Managing XML Encryption with Java, http://www.devx.com/xml/Article/28701/0/page/. Diakses tanggal 12 Oktober 2012.