

KEAMANAN DATA PADA *LEARNING MANAGEMENT SYSTEM*
(LMS)

TUGAS AKHIR

**Karya tulis sebagai salah satu syarat
lulus mata kuliah Keamanan Sistem Lanjut
(EC7010)**

Oleh
FERY JANUAR
NIM : 23205329
Program Studi Teknik Elektro



INSTITUT TEKNOLOGI BANDUNG

2007

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Perkembangan internet saat ini sedemikian pesatnya. Seiring dengan semakin baiknya sarana dan infrastruktur, maka hal itu dengan sendirinya juga mengubah cara hidup manusia termasuk pula dalam hal belajar.

E-learning, seperti juga namanya “*Electronic Learning*” disampaikan dengan menggunakan media elektronik yang terhubung dengan Internet (*world wide web* yang menghubungkan semua unit komputer di seluruh dunia yang terkoneksi dengan Internet) dan Intranet (jaringan yang dapat menghubungkan semua unit komputer dalam sebuah perusahaan). Jika komputer sudah terkoneksi dengan Internet, maka sudah dapat berpartisipasi dalam *e-learning*. Dengan cara ini, jumlah siswa yang dapat ikut berpartisipasi dapat jauh lebih besar dari pada cara belajar secara konvensional di ruang kelas (jumlah siswa tidak terbatas pada besarnya ruang kelas). Teknologi ini juga memungkinkan penyampaian pelajaran dengan kualitas yang relatif lebih standar dari pada pembelajaran di kelas yang tergantung pada “*mood*” dan kondisi fisik dari instruktur. Dalam *e-learning*, modul-modul yang sama (informasi, penampilan, dan kualitas pembelajaran) dapat diakses dalam bentuk yang sama oleh semua siswa yang mengaksesnya, sedangkan dalam pembelajaran konvensional di kelas, karena alasan kesehatan atau masalah pribadi, satu instruktur pun dapat memberikan pelajaran di beberapa kelas dengan kualitas yang berbeda.

Walaupun sepertinya *e-learning* diberikan melalui komputer (yang adalah benda mati), *e-learning* ternyata disiapkan, ditunjang, dikelola dan “dihidupkan” oleh tim yang terdiri dari para ahli di bidang masing-masing, yaitu: *Subject Matter Expert* (SME), *Instructional Designer* (ID), *Graphic Designer* (GD) dan para ahli di bidang *Learning Management System* (LMS). SME merupakan nara sumber dari pelatihan yang disampaikan. ID bertugas untuk secara sistematis mendesain materi dari SME menjadi materi *e-learning* dengan memasukkan unsur metode pengajaran agar materi menjadi lebih interaktif, lebih mudah dan lebih menarik untuk dipelajari. GD

mengubah materi teks menjadi bentuk grafis dengan gambar, warna, dan *layout* yang enak dipandang, efektif dan menarik untuk dipelajari. Para ahli di bidang LMS mengelola sistem di website yang mengatur lalu lintas interaksi antara instruktur dengan siswa, antarsiswa dengan siswa lainnya. Di sini, siswa dapat melihat modul-modul yang ditawarkan, mengambil tugas-tugas dan test-test yang harus dikerjakan, serta melihat jadwal diskusi secara maya dengan instruktur, nara sumber lain, dan siswa lain. Melalui LMS ini, siswa juga dapat melihat nilai tugas dan test serta peringkatnya berdasarkan nilai (tugas ataupun test) yang diperoleh. Jadi, *e-learning* tidak diberikan semata-mata oleh mesin, tetapi seperti juga pembelajaran secara konvensional di kelas, *e-learning* ditunjang oleh para ahli di berbagai bidang terkait.

Sharable Content Object Reference Model (SCORM) adalah spesifikasi standar untuk isi (*content*), yang dikembangkan oleh *Advanced Distributed Learning* (ADL) di Amerika (USA). Dengan dukungan SCORM memungkinkan satu materi bahan ajar dijalankan dalam LMS yang berbeda.

1.2 Perumusan Masalah

Munculnya banyak penyedia jasa pelayanan e-learning, berimbas kepada jumlah *content* yang diperlukan. Para penyedia jasa tersebut juga menggunakan beragam jenis LMS untuk pelayanannya. Dengan adanya suatu standardisasi, dalam hal ini SCORM, maka *content-content* yang dibutuhkan sekarang sudah dapat digunakan oleh berbagai LMS yang sudah SCORM *Comformant*.

1.3 Tujuan

Penelitian ini bertujuan untuk mengetahui tentang keamanan data pada LMS yang sudah SCORM *Conformant*.

1.4 Pembatasan Masalah

Penelitian ini dibatasi hanya untuk mengetahui apa dan bagaimana sebuah LMS dapat mengamankan data-nya.

1.5 Sistematika Penulisan

Sistematika dalam penyusunan laporan ini dapat dijelaskan sebagai berikut.

BAB I. PENDAHULUAN, berisi latar belakang masalah, tujuan, perumusan masalah, pembatasan masalah, dan sistematika penulisan laporan.

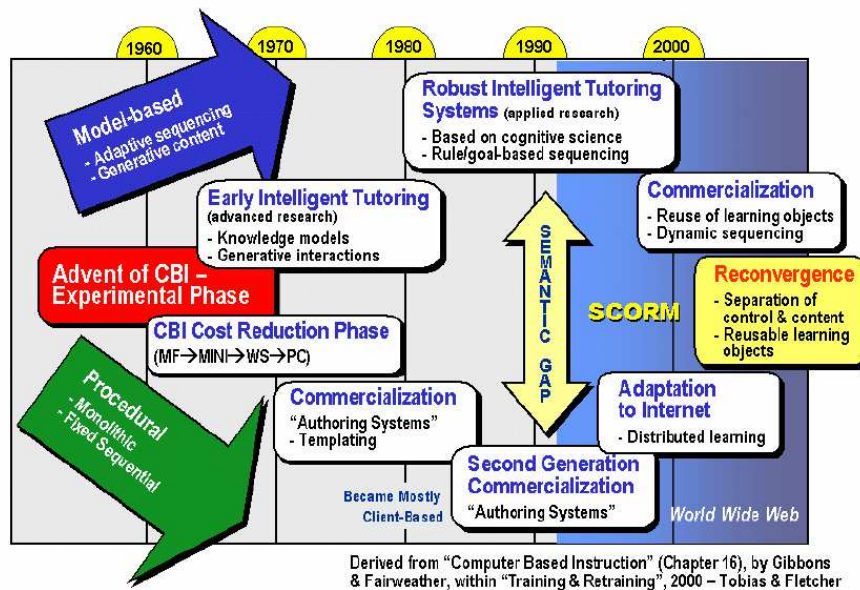
BAB II. DASAR TEORI, membahas teori yang mendukung penyusunan penelitian ini, meliputi pembahasan tentang *Shareable Content Object Reference Model* (SCORM), LMS serta pengenalan XML.

BAB III. XML *Security*, berisi tentang hal-hal yang berhubungan dengan keamanan data pada XML yang digunakan pada LMS.

BAB IV. KESIMPULAN DAN SARAN, berisi kesimpulan hasil penelitian dan memberi saran untuk penelitian berikutnya.

BAB II DASAR TEORI

2.1 Perkembangan Teknologi *E-Learning*



Gambar 2.1 Perkembangan teknologi *e-learning*

Pada tahap awal, pengembangan materi ajar menggunakan pendekatan *Computer-Based Instruction (CBI)*. Tahapan belajar dipandu secara instruksional melalui program yang dirancang khusus pada media mainframe dengan menggunakan bahasa mesin. Dengan perkembangan teknologi komputer, CBI mulai dimanfaatkan pada mesin-mesin *minicomputer*, *workstation* sampai ke PC. Pergeseran ini mengurangi biaya pengembangan materi ajar.

Pada tahap berikutnya sekitar tahun 1960-an, dikembangkan pendekatan baru yang lebih menitikberatkan proses pembelajaran berbasis komputer yang berorientasi pada struktur informasi untuk merepresentasi cara belajar manusia. Pendekatan ini disebut *Intelligent Tutoring System (ITS)*. Pendekatan ini di tahap awal tidak berkembang dengan baik karena beberapa sebab. Pertama, ilmu pengetahuan tentang kognisi

manusia masih relatif belum matang sejalan dengan tahap awal ilmu komputer. Kedua, pemodelan yang kompleks dan sistem berbasis aturan ternyata membutuhkan *computing power* yang tinggi yang belum tersedia saat itu.

Kedua pendekatan tersebut di atas berkembang sejalan dengan semakin matangnya teknologi komputasi. Di tahun 1980-an, teknologi CBI lebih menitikberatkan pada penyempurnaan instruksional komputer menjadi bentuk *template* yang menghindarkan perancang materi ajar dari kerumitan pemrograman komputer. Pendekatan ini menggabungkan isi dan kendali ke dalam satu bundel untuk memperoleh materi ajar yang diharapkan. Sedangkan kelompok kedua terus mengembangkan pendekatan *Intelligent Tutoring System (ITS)* yang memisahkan materi ajar dengan kendalinya. Konsep ini memungkinkan materi ajar dikomposisi secara fleksibel untuk mencapai sasaran belajar yang diharapkan.

Perkembangan teknologi internet di tahun 1990-an telah mengubah banyak kedua pendekatan di atas. Internet memungkinkan diaksesnya beragam informasi dengan mudah dengan memanfaatkan struktur komunikasi yang dibangun pada *common standard*. Materi ajar berbasis web adalah antitesis dari pendekatan CBI karena materi ajar tersebut bebas platform dan dapat disimpan pada *remote-server*. Generasi berikutnya dari materi ajar berbasis web ini mulai memisahkan secara jelas isi (*content*) dengan kendali (*control*). Pada konteks ini, konsep *reusable*, *sharable learning object* dan *adaptive learning strategy* menjadi acuan bagi pengembangan materi ajar elektronik.

2.2 Sharable Content Object Reference Model

Department of Defense (DOD) Amerika Serikat dan *White House Office of Science and Technology Policy (OSTP)* mendirikan *Advance Distributed Learning (ADL) Initiative* pada Nopember 2007. Visi dari *ADL Initiative* adalah untuk memberikan akses kepada kualitas terbaik pembelajaran dan peningkatan hasil pembelajaran, yang dapat disesuaikan dengan kebutuhan individu, serta memberi *cost-effective* kapan dan dimanapun. *ADL Initiatives* bertujuan untuk mempercepat perkembangan berskala besar dari sistem dan perangkat lunak pembelajaran yang *cost-effective* serta dinamis untuk membangkitkan pasar dari produk-produk ini.

2.2.1 Pengenalan SCORM

SCORM merupakan akronim dari *Sharable Content Object Reference Model*. *Reference Model* adalah sesuatu yang menunjukkan jenis-jenis pelayanan apa saja yang dibutuhkan untuk menyelesaikan masalah-masalah tertentu, bagaimana masalah tersebut dapat ditempatkan secara bersama-sama, standar-standar relevan yang terkait, dan bagaimana penggunaannya.

Ada tiga buah kriteria utama untuk *reference model* seperti SCORM. Pertama, harus mampu menghubungkan petunjuk-petunjuk yang dapat dimengerti dan diimplementasikan oleh pengembang *learning content*. Kedua, harus diterapkan, dimengerti dan digunakan oleh banyak *stakeholder* sebisa mungkin. Ketiga, SCORM harus mengizinkan pemetaan dari setiap model spesifik para *stakeholder* untuk pengembangan dan desain instruksional ke dalam dirinya sendiri. *Stakeholder* harus mampu untuk melihat model desain instruksional mereka direfleksikan dengan *reference model* yang mereka pakai pada umumnya.

SCORM merupakan suatu kumpulan spesifikasi-spesifikasi serta standar-standar yang berasal dari organisasi-organisasi berbeda, diantaranya.

- IEEE *Data Model for Content Object Communication*
- IEEE ECMA *Script Application Programming Interface for Content to Runtime Services Communication*.
- IEEE *Learning Object Metadata (LMO)*.
- EIII, *Extensible Markup Language (XML) Schema Binding for Learning Object Metadata Data Model*.
- *IMS content packaging*
- *IMS simple sequencing*.

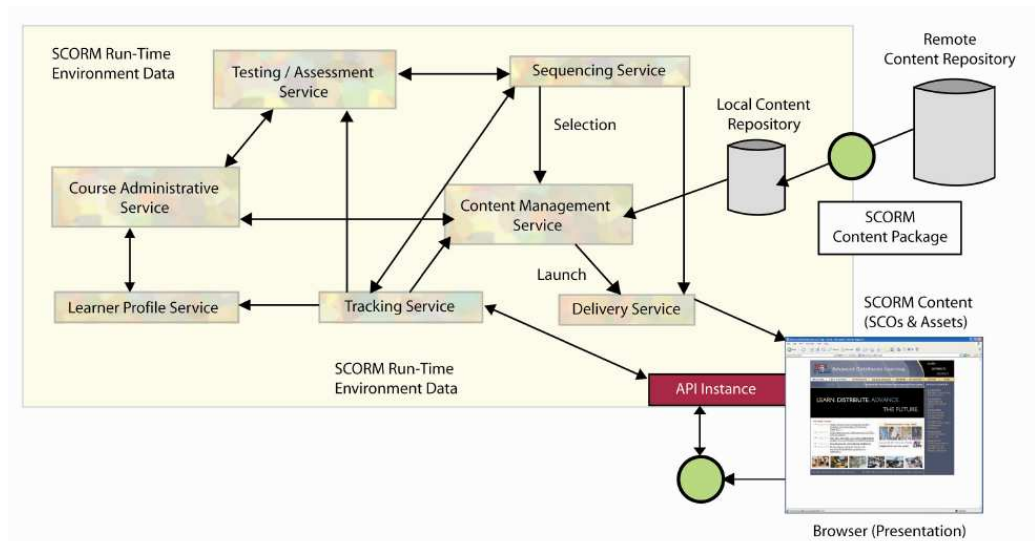
2.2.2 SCORM "ilities"

- *Accessibility*, kemampuan untuk mencari dan mengakses komponen instruksional dari suatu lokasi remote dan mengirimkannya ke banyak lokasi lain.
- *Adaptability*, kemampuan untuk menyesuaikan instruksi kepada kebutuhan pribadi dan organisasi.
- *Affordability*, kemampuan untuk meningkatkan efisiensi dan produktifitas dengan mengurangi biaya dan waktu yang dibutuhkan dalam pengiriman instruksi.
- *Durability*, kemampuan bertahan dari perkembangan dan perubahan teknologi tanpa banyak mengeluarkan biaya untuk mendesain, mengkonfigurasi serta penyimpanan ulang.
- *Interoperability*, kemampuan untuk mengambil komponen-komponen instruksional yang dikembangkan pada suatu lokasi dengan kelengkapan tool atau platform-nya dan menggunakannya di tempat lain dengan tool atau platform yang berbeda.
- *Reusability*, kemudahan menggabungkan komponen-komponen instruksional dalam aplikasi-aplikasi dan konteks-konteks yang bertingkat.

2.2.3 Elemen-Elemen SCORM

- ***Learning Management System (LMS)***

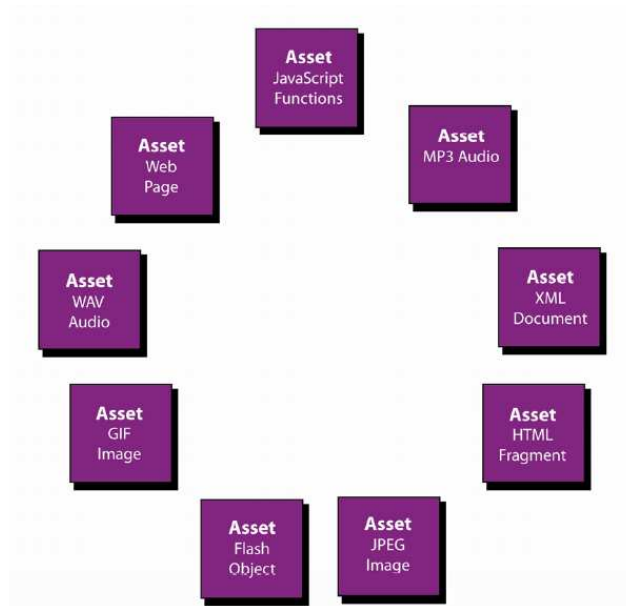
LMS merupakan *catchall term* dalam SCORM, dimana diistilahkan sebagai sebuah *suite* yang terdiri atas desain fungsi-fungsi untuk mengirimkan, melacak, melaporkan dan mengelola isi pembelajaran, mengetahui kemajuan siswa serta interaksi siswa dengan *course*. Sebuah model umum yang menunjukkan komponen atau *service* potensial dari sebuah LMS ditunjukkan pada gambar berikut.



Gambar 2.2 Learning Management System

- **Asset**

Asset adalah blok bangunan yang utama dari sebuah *learning resource*. Asset merupakan representasi elektronik dari media seperti, teks, gambar, suara, obyek penilaian atau bagian data lain yang dapat diolah oleh *web client* dan ditampilkan ke siswa.

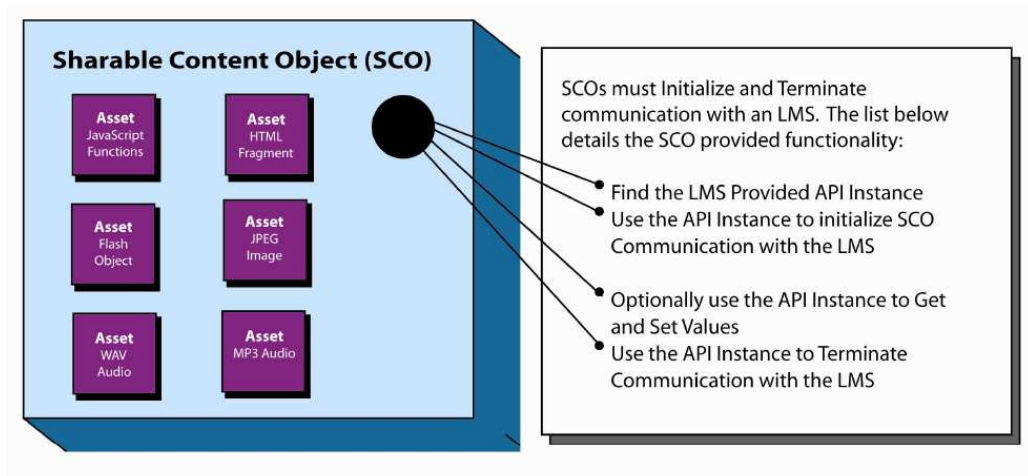


Gambar 2.3 Asset

- **Sharable Content Object (SCO)**

SCO merupakan kumpulan dari satu atau lebih *asset* yang menggunakan SCORM RTE untuk berkomunikasi dengan LMS. Perbedaan utama dengan asset yaitu, SCO berkomunikasi dengan LMS menggunakan aplikasi pemrograman antarmuka *Institute for Electrical and Electronics Engineers (IEEE) ECMAScript*.

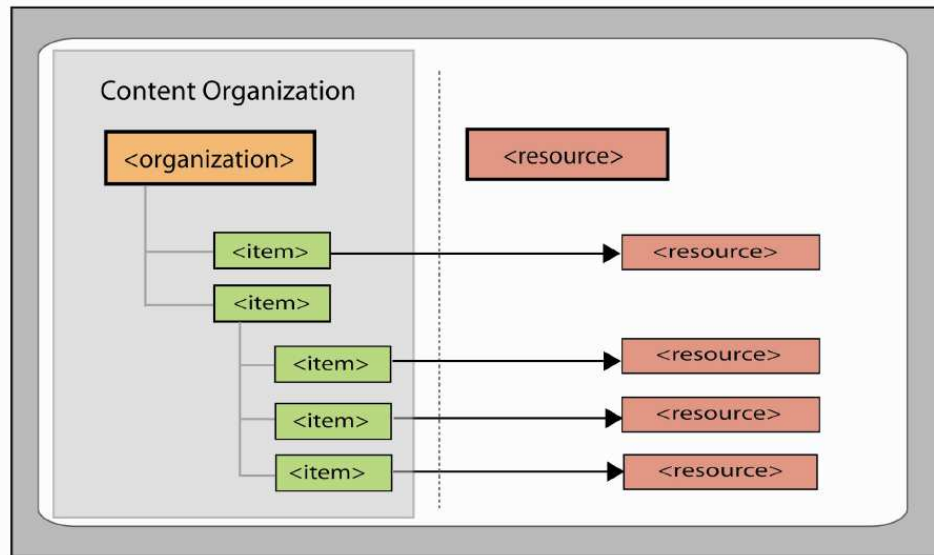
SCO ini adalah unit informasi minimum yang dapat di *load* ke content LMS. Berikut adalah skema dari SCO.



Gambar 2.4 *Sharable Content Object (SCO)*

- **Content Organization**

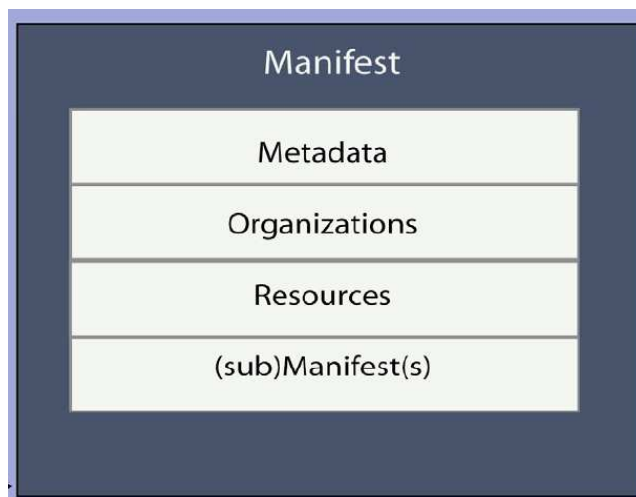
Content organization adalah sebuah representasi atau peta yang mendefinisikan penggunaan yang diharapkan dari isi sampai unit intruksi terstruktur. Peta tersebut memperlihatkan hubungan antara satu aktifitas dengan aktifitas yang lainnya. Berikut merupakan contoh gambar dari *Content organization*.



Gambar 2.5 Content Organization

- **Manifest file**

Manifest merupakan sebuah dokumen XML yang memiliki isi *inventory* yang terstruktur dari sebuah paket. Jika sebuah paket dimaksudkan untuk dikirim ke user, maka manifest akan berisi tentang bagaimana isi dari paket tersebut di organisasikan. Nama standar untuk manifest adalah "imsmanifest.xml" dan harus ditempatkan pada *root directory* dari isi paket. Berikut merupakan struktur dari manifest.

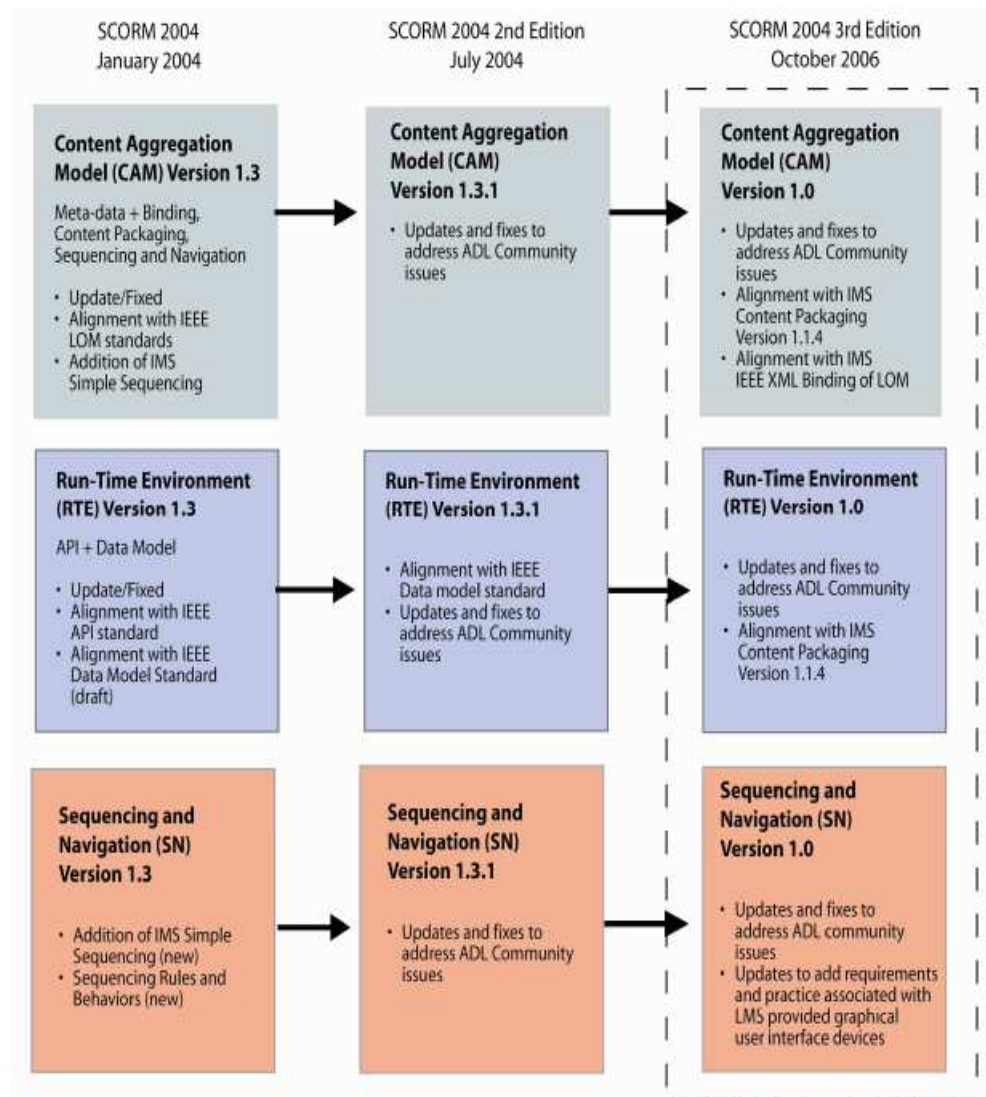


Gambar 2.6 Manifest

- o <Metadata> : data yang menggambarkan isi paket secara keseluruhan.
- o <Organizations> : berisi struktur atau organisasi *content* dari *learning resource*.
- o <Resources>: menjelaskan berkas *learning resource* pada *content package*.
- o <(sub)Manifest(s)>: menguraikan setiap unit intruksi logika yang bersarang (dapat diperlakukan sebagai unit tersendiri).

2.2.4 Evolusi SCORM

Saat ini SCORM sudah memasuki edisi yang ketiga. Berikut merupakan gambar dari evolusi SCORM.



Gambar 2.7 Evolusi SCORM

2.3 XML

XML, singkatan dari *Extensible Markup Language*, adalah bahasa *markup* yang dirancang untuk penyampaian informasi melalui *website* dan juga dapat digunakan untuk pertukaran informasi antar sistem *database*. Bahasa *markup* (*markup language*) merupakan bahasa pemrograman untuk menandai suatu data. Contoh bahasa *markup* yang sudah terkenal adalah *Hypertext Markup Language* (HTML) yang digunakan untuk membuat halaman-halaman *web*.

XML menyediakan format untuk mendeskripsikan data terstruktur atau terurut. Fasilitas yang disediakan XML membuat isi (*content*) suatu data menjadi lebih mudah dipahami. Format data XML, dapat diolah dengan berbagai *tool* pengolah data yang berbeda-beda dan untuk tujuan yang berbeda-beda. Misal, suatu program yang satu digunakan untuk menampilkan data, sedang program yang lain untuk mengedit data.

Beberapa keuntungan menggunakan XML dapat disebutkan sebagai berikut.

- Penyederhanaan aplikasi, dimana *database* yang ditulis dalam XML dapat diakses di mana saja dan memudahkan aplikasi dalam mengolah data karena dapat menghemat memori.
- Pemisahan data dengan presentasi, dimana *tag-tag* dalam dokumen XML dapat menerangkan isi data dan mendefinisikan tentang isi data tersebut. Misal, pada data XML `<nama>budi</nama>`, mengandung penjelasan bahwa budi adalah sebuah nama.
- Berbeda dengan HTML yang digunakan untuk menampilkan data, XML tidak didesain untuk menampilkan data. XML didesain untuk menyimpan dan pertukaran data. Dengan XML data dapat dipertukarkan antar format dari sistem yang tidak kompatibel. Konversi data ke XML dapat mereduksi kompleksitas dan membuat data dapat dibaca oleh aplikasi yang berbeda-beda.
- Pada XML, *file* dengan format teks (*plaintext*) dapat digunakan untuk berbagi data, sehingga lebih mudah membuat data yang berbeda aplikasi yang bekerja dengannya, juga memudahkan untuk *upgrade* sistem ke operasi sistem, *server* dan aplikasi yang baru.

- XML dapat digunakan untuk menyimpan data dalam suatu *file* atau *database*. Aplikasi dapat dibuat untuk menyimpan dan memanggil informasi dari *file* penyimpan untuk menampilkan data tersebut.
- XML bersifat independen terhadap *hardware*, *software* dan aplikasi, menyebabkan aplikasi dapat mengakses *file* XML sebagai sumber data, seperti mengakses *database*. Data dapat digunakan untuk hampir seluruh jenis mesin pembaca data.

2.3.1 Dokumen XML

Dokumen XML berupa *file plaintext* yang menggunakan standar ISO 8859-1 atau *unicode* yang menggunakan UTF-8 dan UTF-16. Simbol *markup* dijelaskan dengan simbol kurung sudut < dan >. <...> dan </...> disebut *tag*. Data, atau *content*, mengisi ruangan di antara *tag-tag* tadi. *Tag* digunakan sebagai tanda untuk menavigasi dokumen. Anatomi dokumen XML terdiri atas dua bagian utama, yaitu *prolog* dan elemen dokumen atau elemen *root*.

2.3.1.1 Bagian Prolog

Terdiri atas deklarasi XML dan komentar.

- Deklarasi

Dimulai dengan <?XML dan diakhiri dengan ?>. Menunjukkan versi XML, tipe encoding dan *file* pendukung dokumen XML. Contoh deklarasi :

```
<?XML version="1.0" encoding="UTF-8" standalone="yes"?>
```

Version "1.0" menunjukkan bahwa *file* XML ditulis sesuai aturan XML versi 1.0. Encoding="UTF-8" menunjukkan bahwa *file* XML ditulis dengan kode UTF-8. Standalone="yes" berarti bahwa *file* XML berdiri sendiri atau tidak ada *file* pendukung lainnya.
- Komentar

Dimulai dengan <!-- dan diakhiri dengan -->, diletakkan setelah deklarasi. Contoh komentar :

```
<!--Data penerbit -->
```

2.3.1.2 Bagian Elemen Dokumen atau Elemen *Root*

Elemen terdiri atas *tag* pembuka dan *tag* penutup. *Tag* pembuka dimulai dengan simbol < dan diakhiri dengan simbol >. *Tag* penutup dimulai simbol </ dan diakhiri dengan simbol >. Nama elemen dapat ditulis bebas, namun tidak boleh menggunakan kata XML dan karakter @, #, \$, %, ^, (,), +, ?, +, ; dan *. *File* XML harus memiliki minimal satu elemen.

2.3.2 *Well Formed*

Dokumen XML harus memenuhi syarat-syarat yang disebut *well formed* untuk menghindari kesalahan, yaitu :

- setiap *tag* pembuka harus ditutup dengan *tag* penutup,
- tidak boleh ada elemen yang *overlapping*,
- setidaknya ada satu elemen utama,
- penulisan atribut harus di antara tanda petik ganda (").

2.3.3 *Document Type Definition (DTD)*

Dokumen XML harus memenuhi spesifikasi dalam *Document Type Definition (DTD)* untuk memastikan validitasnya. DTD adalah deklarasi tipe dokumen, berisi deklarasi yang mendefinisikan elemen, atribut dan fitur-fitur lain dokumen. DTD diletakkan di bagian *prolog* dokumen, dimulai dengan tulisan <!DOCTYPE Nama DTD>, dimana Nama menyatakan nama elemen dokumen. DTD berisi simbol [diikuti serangkaian deklarasi *markup*, diikuti dengan simbol]. Deklarasi *markup* menjelaskan struktur logika dokumen, yaitu mendefinisikan elemen, atribut dan fitur lain dokumen. DTD boleh tidak ditulis, namun menyebabkan dokumen XML tersebut tidak dapat diperiksa validitasnya.

2.3.4 *Schema*

Schema adalah deskripsi komponen dan aturan dari *vocabulary* XML. *Schema* digunakan untuk menggambarkan struktur legal, isi dokumen dan batasan dalam dokumen XML, dengan menyediakan struktur yang valid, batasan, dan tipe data

untuk bermacam-macam elemen dan atribut dokumen XML. Dengan demikian *schema* lebih lengkap dan lebih detail daripada DTD.

2.3.5 Pengurai XML (XML Parser)

Untuk membaca, membuat dan memanipulasi dokumen XML, diperlukan pengurai XML atau *XML Parser*. Pengurai XML berbentuk pustaka atau *software library* yang memberikan layanan-layanan bagi aplikasi yang akan membaca dan mengambil data di dalam dokumen XML. Pengurai XML ini menetapkan *Application Programming Interface* (API) tertentu untuk berinteraksi dengan program aplikasi yang menggunakannya. API mendefinisikan data model dari sebuah dokumen XML kepada aplikasi yang menggunakan pengurai tersebut. Terdapat dua standar API, yaitu *Simple API for XML* (SAX) dan *Document Object Model* (DOM).



Gambar 2.8 Arsitektur pengurai XML.

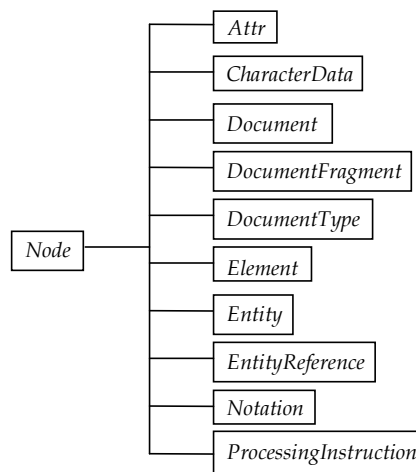
2.3.5.1 *Simple API for XML* (SAX)

SAX menggunakan pustaka pengurai XML dengan fungsi-fungsi dan antarmuka yang memanggil aplikasi pada saat dokumen XML diuraikan. SAX lebih cocok digunakan pada aplikasi-aplikasi yang menggunakan XML sebagai representasi data eksternal dan menggunakan struktur datanya sendiri yang berbeda jauh dari struktur dokumen XML yang digunakannya. SAX memerlukan lebih sedikit memori untuk memproses dokumen XML, karena SAX tidak perlu menyimpan data pada saat dokumen diuraikan.

2.3.5.2 Document Object Model (DOM)

DOM menggunakan struktur data yang disebut *DOM Document Tree*, suatu struktur pohon di memori yang serupa dengan dokumen XML yang sedang diurai. Terdapat satu *node* untuk setiap elemen XML dengan tipenya masing-masing.

Dalam DOM, dokumen XML memiliki tipe *document*. Elemen-elemen di dalam dokumen tersebut umumnya bertipe *Element*. Berbagai atribut yang dimiliki oleh elemen diwakili oleh obyek-obyek bertipe *Attr*. Komentar dan elemen yang berisi teks diwakili oleh *CharacterData*. Tipe-tipe turunan *node* lainnya berupa : *Entity*, *EntityReference*, *Notation*, dan *ProcessingInstruction*.



Gambar 2.9 Hirarki tipe dalam DOM.

Elemen terluar dari suatu dokumen, yang disebut *root node*, bukan merupakan bagian dari dokumen itu sendiri. Sebuah dokumen hanya memiliki satu *node* yang menjadi *root node*. Dokumen yang tidak mempunyai *root node* dikatakan sebagai dokumen kosong (*blank document*). DOM menggunakan *Interface Definition Language* (IDL) untuk mendefinisikan antarmuka berorientasi obyek pada komponen-komponen perangkat lunak dan tidak bergantung pada suatu bahasa pemrograman tertentu, sehingga XML *Parser* dengan standar DOM dapat menggunakan berbagai bahasa pemrograman.

BAB III

XML SECURITY

3.1 XML dan LMS

Seperti dijelaskan pada bab 2, keamanan data pada LMS juga sangat dipengaruhi oleh penulisan kode-kode pemrograman dalam *manifest*-nya. Sedangkan *manifest* tersebut dituliskan menggunakan *syntax* XML. Maka alangkah lebih baik jika kita dapat memahami seluk beluk XML lebih dalam lagi guna memperbaiki keamanan data yang akan diterapkan nantinya dalam LMS.

3.2 Data Privacy dan Authentication

XML merupakan sebuah teknologi yang memungkinkan sebuah data lebih *portable*. XML *security* itu sendiri merupakan sebuah aplikasi *applied security* sampai dengan struktur XML itu sendiri. Dan ini merupakan sebuah cara untuk memisahkan subyek dari *applied security* untuk membedakan antara *data privacy* dengan *authentication*. Jadi, XML *security* merupakan aplikasi dari *data privacy* dan *authentication* dalam struktur XML.

3.3 Encryption

Enkripsi adalah sebuah proses *data encoding* atau pemecahan kode data yang hanya dapat dibaca bila kita memiliki sebuah kunci khusus. Dulu enkripsi hanya berkembang di dunia spionase/perang, tetapi saat ini teknologi tersebut telah merambah ke sektor bisnis dan kalangan pengguna rumahan: Enkripsi adalah tool terbaik untuk melindungi data, privasi, dan rahasia Anda. Yang perlu Anda ketahui tentang enkripsi:

- Mencegah akses yang tidak diinginkan pada dokumen dan pesan e-mail.
- Level enkripsi yang tinggi sukar untuk dibongkar.
- Perubahan dalam peraturan ekspor teknologi kriptografi akan meningkatkan penjualan software enkripsi.

Sebuah program enkripsi, baik itu yang berdiri sendiri (*stand-alone*) atau sudah terdapat pada aplikasi e-mail *client* Anda, memiliki proses yang sama: Data melewati sebuah formula matematis yang disebut algoritma, yang kemudian mengubahnya menjadi data terenkripsi yang disebut sebagai *ciphertext*. Formula ini memerlukan sebuah variabel dari Anda--yang disebut kunci--untuk mengembalikan data tersebut kembali ke bentuk asal, sehingga sangat sulit, bahkan hampir tidak mungkin, seseorang dapat memecahkan kode enkripsi tersebut. Tetapi tentu saja hal ini tidak berlaku jika orang tersebut berhasil mencuri kode enkripsi dari Anda. Jadi, berhati-hatilah dengan kode enkripsi yang Anda miliki.

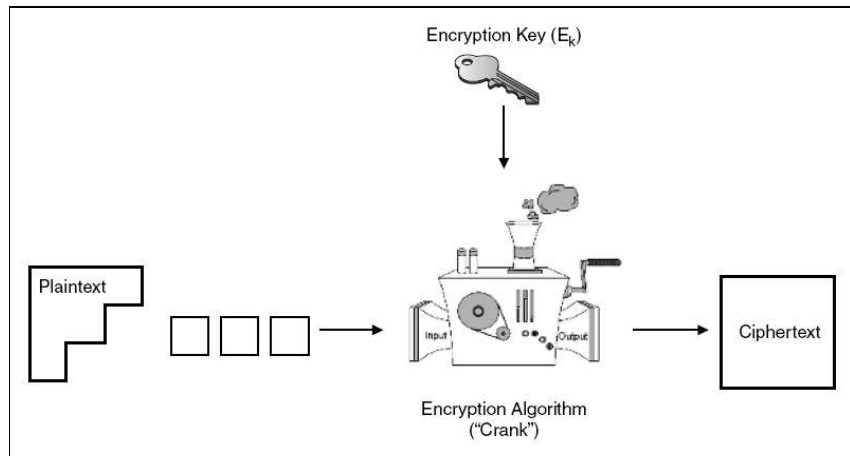
Ada dua jenis enkripsi: simetris dan asimetris (juga disebut sebagai *public key*). Dengan enkripsi simetris, Anda menjalankan sebuah file melalui program dan membuat sebuah kunci yang mengacak file. Kemudian Anda mengirim file terenkripsi melalui e-mail ke si penerima dan secara terpisah mentransmit kunci dekodingnya (mungkin berupa sebuah password atau file data lainnya). Si penerima, dengan menjalankan aplikasi enkripsi yang sama, menggunakan kunci yang Anda berikan untuk menyatukan kembali file yang telah diacak. Enkripsi simetris sangat mudah dan sangat cepat dalam penggunaannya, tetapi tidak seaman enkripsi asimetris, karena seseorang dapat saja mencegat kunci dan mendekoding pesan tersebut. Tetapi karena kecepatannya itu, saat ini enkripsi simetris banyak digunakan pada transaksi e-commerce.

Enkripsi asimetris sangat kompleks--tetapi jauh lebih aman. Diperlukan dua buah kunci yang saling berhubungan: sebuah kunci publik dan sebuah kunci pribadi. Anda membuat kunci publik Anda tersedia bagi siapa saja yang ingin Anda kirim informasi terenkripsi. Kunci tersebut hanya dapat mengenkoding data; ia tidak dapat mendekodingnya. Kunci pribadi Anda terjaga dengan aman bersama Anda. Saat orang-orang hendak mengirim informasi terenkripsi pada Anda, mereka mengenkripsinya menggunakan kunci publik Anda. Saat Anda menerima *ciphertext* tersebut, Anda akan mendekripsikannya dengan menggunakan kunci pribadi Anda. Enkripsi asimetris menambahkan tingkat keamanan pada data Anda, tetapi akibatnya

lebih banyak lagi waktu komputasi yang dibutuhkan, sehingga prosesnya menjadi sangat panjang dan lebih lama.

Enkripsi simetris dan asimetris menggunakan dua buah algoritma yang berbeda untuk menghasilkan *chipertext*. Pada enkripsi simetris, algoritmanya akan memecah-mecah data menjadi potongan-potongan kecil yang disebut blok. Kemudian ia akan mengganti letak huruf, mengubah informasi pada setiap blok menjadi angka, mengkompresinya dan memperbesar ukuran data, dan kemudian menjalankannya melalui formula matematis termasuk kunci di dalamnya. Kemudian algoritma mengulangi proses tersebut, kadang-kadang sampai selusin pengulangan. Pada algoritma enkripsi asimetris, memperlakukan teks sebagai sebuah angka yang sangat besar, terus mengkalikannya menjadi angka yang lebih besar, dan kemudian mengkalkulasi sisanya setelah dibagi dengan angka terbesar ketiga lainnya. Akhirnya, angka sisa ini dikonversi menjadi teks kembali. Program enkripsi dapat menggunakan algoritma yang sama secara berbeda, itu sebabnya mengapa para penerima informasi yang terenkripsi harus memiliki program yang sama dengan si pengirim untuk mengenkoding data yang mereka terima.

Kunci menjadi potongan akhir yang menyusun teka-teki enkripsi, Kunci ini bermacam-macam jenisnya dalam hal panjang dan kekuatannya. Alasan: semakin panjang kuncinya, semakin besar jumlah kombinasi angka yang timbul. Sebagai contoh, bila program enkripsi Anda menggunakan kunci 128-bit, maka kunci Anda tersebut dapat berupa salah satu kombinasi dari 3,4 trilyun milyar milyar milyar kombinasi--atau 2 pangkat 128 kombinasi--dari angka satu dan nol. Seorang cracker mungkin akan lebih beruntung mendapat lotere dibanding ia harus memecahkan enkripsi tersebut menggunakan metode brutal (*brute-force method*, yaitu mencoba menebak kombinasi kunci satu per satu sampai mendapatkan kunci yang benar). Sebagai perbandingan, seorang ahli enkripsi menggunakan metode brutal ini dapat memecahkan kode enkripsi simetris 40-bit dalam waktu 6 jam dengan menggunakan PC biasa di rumah. Walau begitu, enkripsi 128-bit masih memiliki beberapa kelemahan; para profesional memiliki teknik yang canggih yang dapat menolong mereka memecahkan kode yang paling sulit sekali pun.



Gambar 3.1 Enkripsi

3.4 *Meta-Language*

XML bukan merupakan sebuah bahasa pemrograman, namun merupakan sebuah *meta-language*. Yakni, sebuah bahasa yang digunakan untuk mendeskripsikan bahasa lain.

```

<Food>
  <FrenchFries> Curly Fries </FrenchFries>
  <Beers>
    <Good_Beer> Samuel Adams </Good_Beer>
    <Good_Beer> Guinness </Good_Beer>
    <Bad_Beer> Budweiser </Bad_Beer>
    <Bad_Beer> Fosters </Bad_Beer>
  </Beers>
</Food>

```

Gambar 3.2 Meta-language

3.5 *Paradigm-Shift*

XML *Security* menghadirkan suatu *paradigm-shift* yang jelas dari ASN.1-based. Hampir semua *entity* dalam dunia *security* yang berhubungan dengan kriptografi dan *public key infrastructure* (PKI) menggunakan ASN.1 untuk meng-*encode entity*-nya.

```

<KeyValue>
  <RSAKeyValue>
    <Modulus>
      s3mkTQbzXuNFPPDtWd/9jvs8tF5ynBLilbG/sT24OglEol
      1PBvRe+VUJU0eI2SRhN/KtZv4iD2jwT0Sko0eeJw==
    </Modulus>
    <Exponent>EQ==</Exponent>
  </RSAKeyValue>
</KeyValue>

```

Gambar 3.3 Elemen <KeyValue>

```

SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
    NULL
  }
  BIT STRING 0 unused bits
  30 46 02 41 00 B3 79 A4 4D 06 F3 C6 E3 45 3C 50
  ED 59 DF FD 8E FB 3C B4 5E 72 9C 12 E2 95 B1 BF
  B1 3D B8 38 69 44 A2 5D 4F 06 F4 5E F9 55 09 53
  47 88 D9 24 61 37 F2 AD 66 FE 22 0F 68 F0 4F 44
  A4 A3 47 9E 27 02 01 11
}

```

Gambar 3.4 SubjectPublic- KeyInfo biner diterjemahkan menggunakan ASN.1 parser

3.6 XML Signature

XML Signature merupakan *syntax* XML spesifik yang digunakan untuk merepresentasikan *digital signature* diseluruh isi *digital arbitrary*.

```

AlgorithmIdentifier ::= SEQUENCE {
  algorithm OBJECT IDENTIFIER,
  parameters ANY DEFINED BY algorithm OPTIONAL }

```

Gambar 3.5 ASN.1 definition of AlgorithmIdentifier

```

30 0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00

```

Gambar 3.6 AlgorithmIdentifier for RSA with SHA-1

XML signature juga didefinisikan sebagai :

- An XML Signature is generated from a hash over the canonical form of a signature manifest.

- An XML Signature associates the contents of a signature manifest with a key via a strong one-way transformation.

3.7 XML Signature Types

- Different applications require signature delivery in certain ways, with preferred signature types.
- Certain applications require that an XML Signature be modeled as closely as possible to a real, handwritten contract that includes embedded signatures in certain parts within the original document.
- Other applications may process the original data separate from the signature and may require that the original data be removed from the signature itself.

```

<!-- Enveloped Signature -->
<original_document>
  <Signature> ... </Signature>
</original_document>
<!-- Enveloping Signature -->
<Signature>
  <original_document>
  </original_document>
</Signature>

<!-- Detached Signature -->
<Signature> ... </Signature>

```

Gambar 3.7 Enveloped, enveloping, and detached XML signatures

```

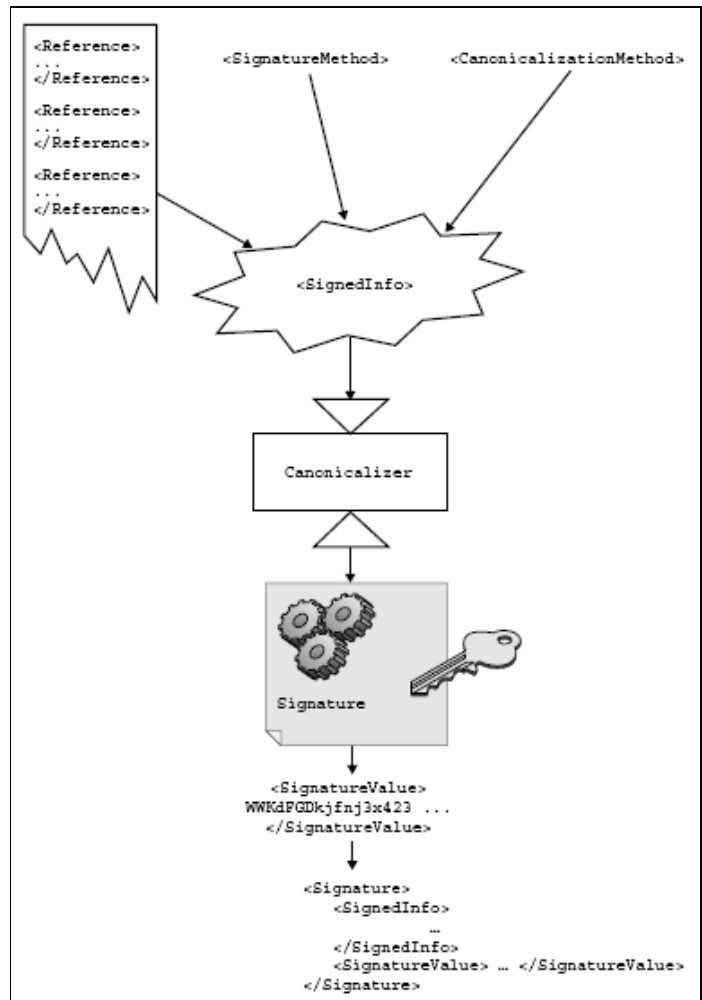
<Signature>
  <SignedInfo>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference URI="http://www.myserver.com/importantFile.xml">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>aZh8Eo2alIke1D5NNW+q3iHrRPQ=</DigestValue>
    </Reference>
    <Reference URI="#ImportantMessage">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>qGh8Eo2alJke1D7NNW+z3iHhRPF=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>
    MI6rfG4dwPFASDFfgAFsdAdASfasdFEVWdMrO/
    Ta7nFCSDAhnTRhy45vJSDcvadrtrEQW2HP0/7J
    tPQTBFEWGVwIqewrfVfewgrtgVfbwdj77ujYdT
    Q+zMtKRQGRElgrwrfht32rwhnbtygrwtrHyKl
    3EFfdasreEDafsf8I=
  </SignatureValue>
  <Object>
    <original_document>
      <very_important_element id="ImportantMessage">
        Milk Chocolate is better than Dark Chocolate!
      </very_important_element>
    </original_document>
  </Object>
</Signature>

```

Gambar 3.8 Enveloping and detached XML Signature

3.8 XML Signature Syntax

Contoh dari syntax XML :



Gambar 3.12 *Signature Generation*

BAB IV

Kesimpulan dan Saran

4.1 Kesimpulan

LMS menjadi salah satu software e-learning yang banyak digunakan oleh penyedia jasa pembelajaran jarak jauh saat ini. Keamanan data dalam LMS ini menjadi sesuatu yang amat vital bagi kelangsungan hidup LMS itu sendiri. Maka, mengetahui bagaimana cara mengamankan datanya menjadi hal yang sangatlah penting. Kebanyakan LMS sekarang ini sudah mengikuti standarisasi dari SCORM. SCORM memberikan suatu tuntunan teknis bagi para developer untuk mengembangkan software-nya agar dapat digunakan oleh banyak pihak yang berkepentingan. SCORM juga mengatur bagaimana caranya berkomunikasi antara data didalam content yang ada dengan keseluruhan data pada LMS tersebut. Data dalam SCORM diatur menggunakan manifest yang dibuat menggunakan XML. Maka, penulisan kode XML haruslah benar-benar aman bagi integritas data di dalam LMS.

4.2 Saran

Bagi software developer yang ingin mengembangkan learning content ataupun LMS, hendaknya memperhatikan benar aspek securitynya. Namun aspek tersebut perlu juga disesuaikan dengan standar yang sudah ada, seperti standar SCORM.

DAFTAR PUSTAKA

- [1] _____ (2006), *SCORM 2004 3rd Edition Overview*. <http://www.adlnet.org> - Official web ADL page.
- [2] _____ (2006), *SCORM 2004 3rd Edition Content Aggregation Model Version 1.0*. <http://www.adlnet.org> - Official web ADL page
- [3] _____ (2006), *SCORM 2004 3rd Edition Run-Time Environment Version 1.0*. <http://www.adlnet.org> - Official web ADL page
- [4] _____ (2006), *SCORM 2004 3rd Edition Sequencing and Navigation Version 1.0*. <http://www.adlnet.org> - Official web ADL page
- [5] _____ (2006), *SCORM 2004 3rd Edition Impacts Summary*. <http://www.adlnet.org> - Official web ADL page
- [6] Dournaee, Blake (2002), *XML Security*. McGraw-Hill.
- [7] Shoikova, Elena & Ivanova, Malinka. (2005), *Learning Design Implementation in SCORM E-Learning Environment*. Technical University of Sofia, Department of Electronics.