Canonical XML 알고리즘의 구현

박종현[0] 김병규 강지훈 한우용*
충남대학교 컴퓨터과학과, 한국전자통신연구원*
{ jhpark, yourovin, jhkang }@cs.cnu.ac.kr, wyhan@etri.re.kr

An Implementation of The Canonical XML Algorithm

Jong-Hyun Park[0], Byung-Kyu Kim, Ji-Hoon Kang, & Woo-Yong Han*
Dept. of Computer Science, Chungnam National University
Electronics and Telecommunications Research Institute*

요 약

인터넷상의 B2B(Business to Business)와 EDI(Electronic data Interchange)가 급속도로 발달 하면서 XML이 메시지 교환 형식으로 활발 하게 이용되고 있다. XML 문서는 논리적으로 동일하나 물리적으로는 여러 가지 표현이 존재할 수 있다는 특성을 가지고 있다. 이러한 점은 XML 디지털 서명과 같이 문서의 물리적 형태로써 유효성 여부를 판단하는 응용 등에서는 치명적인 문제를 야기시킬 수 있다. 이와 같은 단점을 해결하기 위해 W3C에서는 상호 교환되는 XML 문서간의 논리적인 동일성을 물리적 수준에서 확인하기 위한 Canonical XML 알고리즘을 제안하여 사용하도록 권고하고 있다.

본 논문에서는 XML 디지털 서명 환경에서 Canonical XML 알고리즘을 수행하는 XML Canonicalizer를 설계하고 구현하였으며, 우리의 XML Canonicalizer는 XML 디지털 서명에서 뿐만 아니라 XML문서의 교환 시 물리적인 동일성이 요구되는 어떠한 응용에서도 사용 가능 하다.

1. 서론

XML(eXtensible Markup Language) 기술이 인터넷 e-비즈니스 시스템 등에서 메시지 교환 형식으로 이용되면서 이들 XML 문서의 보안은 필수적 요구조건이 되고, 안전한 비즈니스를 수행하기 위해서 XML 디지털 서명은 반드시 지원 되야 한다. 이를 위하여 W3C에서는 XML-Signature 표준 규격[6]을 제안하고 있고, 이 규격에는 XML문서가 물리적으로 동일하다는 것을 보장하기 위하여 Canonical XML 알고리즘을 사용하도록 권고하고 있다. W3C에서는 XML문서를 처리 할 때 물리적인 동일성이 요구되는 어떠한 응용에서도 사용할 수 있도록 XML 문서의 Canonicalization을 위해서 Canonical XML 표준 규격을 정의하고 있다. [7]

XML 디지털 서명을 할 때 교환된 XML 문서가 변형되지 않았음을 보장하기 위하여 문서의 Digest 값을 얻어내야 한다. 이때 물리적으로 동일한 XML 문서를 보장받기 위하여 Canonical XML 알고리즘을 적용할 필요가 있다.

본 논문에서는 Canonical XML 표준 규격에서 정의하고 있는 두 가지 알고리즘을 수행하는 XML Canonicalizer를 설계하고 구현하였다. 우리의 XML Canonicalizer는 XML 문서의 보안을 위한 XML 디지털 서명을 기반으로 개발하였으나, XML 디지털 서명에서 뿐만 아니라 XML을 이용하여 메시지 교환을 하는 모든 응용 프로그램에서 파싱과 같은 처리 과정에서 나타날 수 있는 XML 문서의 물리적 변형에 대하여 변형이 없음을 보장할 수 있다.

* 이 연구는 BK21 충남대학교 정보통신인력양성사업단의 지원을 받았음.

본 논문의 구성은 다음과 같다. 2절에서는 Canonical XML에 대한 개념과 XML-Signature에서 Canonical XML 알고리즘의 적용에 대해 다루고, 3절에서는 본 논문에서 개발한 XML Canonicalizer의 처리흐름과 설계 및 구현에 관해서 설명 하고 Canonical XML 알고리즘을 적용한 XML 문서를 살펴본다. 4절에서는 관련연구에 대해서 기술 하고, 5절에서 향후 연구 방향과 결론을 맺는다.

2. Canonical XML

2.1 Canonical XML의 개념

W3C에서 제정된 Canonical XML version 1.0 규격[7]은 현재 모든 표준화 작업이 끝난 상태로 XML 문서를 정규화 시키기 위한 알고리즘을 기술하고 있다.

Canonical XML 알고리즘은 XML 문서의 특성상 논리적으로는 동일한 문서지만 사용자에 따라서 수없이 많은 형태로 기술될 수 있는 XML 문서를 물리적으로 동일한 형태의 문서로 만들기 위해서 사용된다. 또한 각 응용 프로그램에서 XML 문서를 처리하기 위해서는 반드시 파싱 등의 과정을 거쳐야 하는데 이 처리 과정에서 논리적으로는 문서의 변형이 없더라도 물리적으로는 문서의 변형이 존재하게 된다. 이때에 XML 문서를 물리적으로 동일한 형태로 정규화 하기 위한 규칙이 Canonical XML 알고리즘이다.

2.2 XML 디지털 서명에서 Canonical XML 알고리즘

XML 디지털 서명을 할 때 XML 비즈니스 문서의 변형이 없음을 보장 하기 위해서는 Hash 함수를 이용하여 비즈니스 문서의 유일한 값인 Digest 값을 얻어내어 서명 문서에 첨부하여 송신한다. 수신자 측에서도 송신자 측에서 사용한 방법과 동일한 방법으로 XML 비즈니스 문서의 Digest 값을 생성하여 수신자 측에서 생성하여 서명문서에 첨부한 Digest 값과 비교하여 동일하다면 문서가 변형이 없음을 보장 한다. 이때 서명의 대상이 되는 XML 비즈니스 문서의 처리를 위해서는 파싱 등의 처리 과정을 거친 결과가 Hash 함수의 입력으로 제공된다. 이러한 처리 과정에서 입력으로 제공되는 XML 비즈니스 문서는 논리적으로는 동일하지만 물리적으로는 동일하지 않을 수 있다. 이렇게 되면 유일한 Digest 값은 다르게 나오게 되며, XML 디지털 서명 처리기는 서명의 대상이 된 XML 비즈니스 문서는 동일하지 않다고 잘못된 판단을 하는 오류를 범하게 된다.

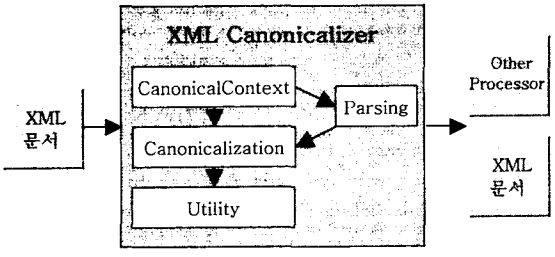
XML 비즈니스 문서의 유효성 검사가 확인되면 XML 디지털 서명의 다음 처리 과정은 서명 문서의 유효성 확인을 위하여 서명 문서를 입력으로 하여 서명알고리즘을 적용한다. 이때 XML 비즈니스 문서의 위치정보와 적용할 알고리즘과 같은 정보를 추출하기 위해 DOM Tree형태로 파싱한 XML 문서를 Canonicalizarion 과정을 거쳐 물리적인 형태의 Canonical XML 문서로 변형한 후 서명 알고리즘의 입력으로 제공한다. 만약 파싱된 DOM Tree구조의 XML문서를 Canonicalization 과정을 거치지 않고 물리적인 XML문서로 변형한다면, XML 비즈니스 문서의 경우와 마찬가지로 수신자 측과 송신자 측에서 입력으로 제공한 XML 문서의 물리적인 형태는 응용에 따라 달라질 수 있으므로 유효한 서명 문서라 할 지라도 유효하지 않다고 판단하는 오류를 범하게 된다. 이러한 이유에서 XML 디지털 서명은 반드시 Canonical XML 알고리즘의 수행이 필수 적이다.

위와 같은 처리과정을 거치는 XML 디지털 서명은 XML 비즈니스 문서의 일부만을 서명하는 경우, 또는 일부만을 변형하여 서명하는 경우가 많다. 이때에 XML문서의 일부분만을 처리 하기 위해서는 문서의 구조적 정보를 얻기 위하여 파싱된 DOM Tree구조의 XML문서를 이용하여 처리한다. 처리된 DOM Tree는 다음 변형 알고리즘의 입력으로 제공 되는데 다음 알고리즘의 입력으로 OctetStream형식의 XML문서를 요구할 경우 DOM Tree구조의 XML문서를 OctetStream형식의 XML 문서로 변형하여야 한다. 이때 물리적인 동일성을 유지하기 위하여 Canonical XML 알고리즘을 사용 한다.

3. XML Canonicalizer

3.1 XML Canonicalizer의 구조

XML Canonicalizer의 구조는 [그림 1]과 같다. 입력으로 제공되는 XML 문서는 OctetStream, Node 또는 NodeSet의 데이터 타입으로 되어 있으며, 주석을 포함하여 정규화 처리를 하는지의 여부를 포함 하고 있다.



[그림 1] XML Canonicalizer의 처리 흐름

CancnicalContext에서는 Canonical XML의 두 가지 알고리즘 중 어떤 알고리즘을 적용 할 것인가 여부와 XML문서의 데이터 타입을 확인한 후 문서의 데이터 타입이 OctetStream일 경우 문서의 처리를 위하여 파싱한 DOM Tree[5]를 생성 한다. 생성된 DOM Tree 형태의 XML 문서 또는 초기 입력 형식이 NodeSet이나 Node 타입의 XML문서는 Canoicalization을 통하여 DOM Tree를 순회 하면서 정규화 한다. 이때 Utility에서 각각 해당 노드의 타입별로 정규화 한다. NodeSet구조는 완전한 XML문서 이외에도 XML문서의 일부분 만을 저장하고 있을 수 있으므로 문서의 일부분만을 Canonicalization할 수 있다. DOM Tree 또는 NodeSet 내부의 모든 노드들을 정규화 한 Canonicalizer의 출력 은 OctetStream 타입의 XML문서이다.

XML 디지털 서명의 경우 다음 처리(Digest 값의 추출)를 위한 입력으로 제공된다.

3.2 XML Canonicalizer의 구현

XML 문서의 정규화는 파싱된 DOM Tree의 노드가 가질 수 있는 12가지 데이터 타입 중에서 7가지의 데이터 타입에 대한 Canonical 처리만으로 정규화를 이룰 수 있다. 다음 [표 1]은 정규화 처리의 대상이 되는 7가지 타입의 노드를 각 노드 별로 처리하기 위한 메소드들 이다.

[표 1] Canonical 메쏘드

Canonical_Element	노드 타입이 Element인 것을 대상으로 Canonical form을 만든다.
Canonical_Attribute	노드 타입이 Attribute인 것을 대상으로 Canonical form을 만든다.
Canonical_Text	노드 타입이 Text인 것을 대상으로 Canonical form을 만든다.
Canonical_CDATASection	노드 타입이 CDATASection인 것을 대상으로 Canonical form을 만든다.
Canonical_EntityReference	노드 타입이 EntityReference인 것을 대상으로 Canonical form을 만든다.
Canonical_ProcessingInstruction	노드 타입이 ProcessingInstruction인 것을 대상으로 Canonical form을 만든다.
Canonical_Comment	노드 타입이 Comment인 것을 대상으로 Canonical form을 만든다.

본 논문에서 구현한 XML Canonicalizer는 OctetStream 또는 NodeSet 타입의 XML 문서를 입력으로 받는다. 만약 입력 XML 문서의 타입이 OctetStream이라면 Parsing 하여 DOM Tree 구조를 얻은 후 Canonicalization처리를 한다. 이때 DOM Tree의 상위 엘리먼트에서 선언 되어 있는 네임 스페이스[8]는 하위의 자식 노드에 영향을 주게 되므로 상위 노드에서부터 하위노드 순으로 순회 하며, 엘리먼트 노드의 end-tag를 닫아 주어야 하므로 하위 노드에서 상위노드로 순회하여야 한다. 따라서 정규화 순서는 최상위 Root 노드에서부터 좌측 자식 노드, 우측 자식 노드의 순서로 전위 순회하면서 각 타입에 맞는 메소드를 호출하여 정규화 한다.

NodeSet 타입의 XML 문서가 Canonicalizer의 입력으로 제공될 경우는 주로 정규화 하려는 문서가 전체문서의 일 부분일 경우 사용된다. NodeSet의 구조 안에 저장되어 있는 노드들은 전위 순회하면서 선택된 순서대로 NodeSet 구조에 저장되므로 저장 되어 있는 첫번째 노드부터 순차적으로 정규화 한다.

3.3 XML 문서의 Canonicalization

우리의 XML Canonicalizer는 W3C의 권고 안을 모두 만족하도록 설계하고 구현 하였다. [그림 3]은 [그림 2]의 XML 비즈니스 문서를 Canonicalization처리한 결과이다.

정규화의 처리 결과를 보면 먼저 내부에 선언된 XML 선언

과 DTD는 제거 되고, PI(Processing Instructions)는 유지된다. 이 때 PI 내부의 내용은 공백을 포함하여 모두 유지된다. 엘리먼트 e1은 내용이 없는 공백 엘리먼트 이므로 end-tag를 닫아주고, tag안의 내용 중 공백은 의미없는 공백이므로 모두 제거한다. 엘리먼트 e2의 속성과 네임 스페이스들은 네임 스페이스와 속성의 순서로 정렬되며, 알파벳 순으로 정렬된다. 엘리먼트 e3은 상위 엘리먼트인 e2에서 선언된 네임스페이스가 중복되므로 중복되는 네임 스페이스 선언이 제거되었고, 엘리먼트 e4와 e5에 선언된 네임스페이스도 상위 엘리먼트에 이미 선언되어 중복되는 선언은 제거된다. 내용을 가진 엘리먼트의 경우는 start-tag와 end-tag사이의 모든 공백들을 의미있는 공백으로 보고 모두 유지시켜 준다. 즉, 엘리먼트 e5의 start-tag앞의 공백은 e3 엘리먼트의 내용이므로 그대로 유지시켜 준다. e5 엘리먼트의 내용 중 Entity 레퍼런스는 DTD에 선언된 값으로 대체된다.

```
<?xml version="1.0"?>
<?xml-stylesheet   href="doc.xsl"
   type="text/xsl"   ?>
<!DOCTYPE doc [ . . .
                <!ATTLIST e9 attr CDATA "default">
                    <!ENTITY ent1 "Hello World">] . . . >
<doc>
   <e1   />
   <e2 a:attr="out" b:attr="sorted" attr2="all" attr="I'm"
      xmlns:b=http://www.ietf.org xmlns:a="http://www.w3.org"
      xmlns="http://example.org">
      <e3 xmlns="" xmlns:a="http://www.w3.org">
         e3's Content <!-- Comment 1 -->
         <e4 xmlns="http://www.ietf.org">
                e4's
                c o n tent
          </e4>          <e5 xmlns="" xmlns:a="http://www.w3.org">
            e5's content &ent1;
          </e5>
      </e3>
   </e2>
</doc>
<?pi-without-data?>
<!-- Comment 2 -->
```

[그림 2] Canonicalization 처리 전의 XML문서

```
<?xml-stylesheet   href="doc.xsl"
   type="text/xsl"   ?>
<doc>
   <e1></e1>
   <e2 xmlns="http://example.org" xmlns:a="http://www.w3.org" xmlns:b=
      "http://www.ietf.org" attr="I'm" attr2="all" b:attr="sorted" a:attr="out">
      <e3 xmlns="" >
         e3's Content <!-- Comment 1 -->
         <e4 xmlns="http://www.ietf.org">
                e4's
                c o n tent
          </e4>          <e5>
            e5's content Hello World
          </e5>
      </e3>
   </e2>
</doc>
<?pi-without-data?>
<!-- Comment 2 -->
```

[그림 3] Canonicalization 처리 후의 XML문서

위 예문의 경우 Canonical XML의 두 가지 알고리즘 중 주석을 포함하여 정규화 하는 알고리즘을 적용했으므로 문서에 포함된 모든 주석은 유지된다.

4. 관련 연구

본 논문에서 연구하고 개발한 XML Canonicalizer는 XML 디지털 서명 환경에서 개발하였다. XML 디지털 서명을 위하여 W3C에서는 XML-Signature Syntax and Processing 규격[6] 제안하고 있다. 그 내용으로는 전자 상거래 등의 사용자 인증이 반드시 필요한 응용에서 XML문서의 디지털 서명을 위하여 필요한 XML 구문과 처리 규칙들을 기술하고 있다.

XML 디지털 서명은 현재 국외의 몇몇 업체에서 연구, 개발하고 있는 실정이며, 이들 업체에서는 XML 디지털 서명을 위해서 반드시 필요한 Canonical XML 알고리즘을 함께 개발하고 있다. WebSig Consortium에서는 실제 여러 여행사들 사이의 e-Commerce 응용을 목적으로 개발 중에 있으며[1], IBM에서는 IBM XML Security Suite for JAVA (XSS4J)[3]를 개발 중에 있고, Baltimore사에서는 X/Secure[4]를, IAIK는 XML Signature Library (IXSIL)[2] 라는 이름으로 Canonical XML 알고리즘을 포함한 XML디지털 서명을 개발 중에 있다. XML-Signature 규격[6]은 이제 막 표준화 작업이 끝났으므로, 관련 구현 시스템들은 실제 여러 응용에서 테스트가 되어야 할 것이다.

5. 결론

본 논문에서 연구하고 개발한 XML Canonicalizer는 XML문서를 송신하고 수신하는 모든 응용프로그램에서 논리적으로는 동일하나 물리적으로는 여러 가지 표현이 존재할 수 있는 XML 문서를 물리적으로 일관된 형태로 정규화 하는 Canonical XML 알고리즘을 효과적으로 수행한다. 특히 문서의 물리적인 변형이 없음을 반드시 보장해야 하는 XML 디지털 서명과 같은 응용에서는 Canonical XML 알고리즘을 필수적으로 적용해야 한다. 또한 우리의 Canonicalizer는 독립적으로 수행 되므로 응용에 따라 특성에 맞게 사용할 수 있다.

향후, Canonical XML과 유사한 기능을 하는 DOMHash를 구현하여 비교 분석하고, XML 디지털 서명 이외의 시스템과 연계하여 통합한 후 성능을 테스트해야 할 것이다.

6. 참고 문헌

[1] C. Geuer-Pollmann, C. Puland, P. Sklavos, & M. Moula "Digital Signatures for Web Content", e-2000 eBusiness and eWork Conference, 2000.

[2] IAIK, XML Signature Library (IXSIL) (http://jcewww.iaik.at/products/ixsil/index.php).

[3] IBM, The XML Security Suite, 2001. (http://www.trl.ibm.com/projects/xml/xss4j/docs/dsig.html)

[4] Baltimore, KeyTools XML Introductions, 2001 (http://www.baltimore.com/keytools/xml/index.html).

[5] W3C, Document Object Model (DOM) Level 1, Recommendation, 1-Oct-1998. (http://www.w3.org/TR/REC-DOM-Level-1).

[6] W3C, XML-Signature Syntax and Processing, Candidate Recommendation, 19-April-2001. (http://www.w3.org/TR/xmldsig-core/).

[7] W3C, Canonical XML Version 1.0, Recommendation 15-March-2001. (http://www.w3.org/TR/xml-c14n)

[8] W3C, Namespace in XML, Recommendation 14- January-1999. (http://www.w3.org/TR/REC-xml-names/).