서명값 은닉을 이용한 XML 전자서명 모델 설계

고훈*, 김대원** 신용태**
*대진대학교 컴퓨터공학과
**숭실대학교 컴퓨터학과
e-mail:skoh21@daejin.ac.kr, shin@comp.ssu.ac.kr

XML Signature Model Design using Signature Value Hide

Hoon Ko*, Dae-won Kim,**Yong-tae Shin**
*Dept of Computer Engineering, Dae-jin University
**Dept of Computer Science, Soong-sil University

요 약

최근 전자상거래 확산에 따라 전자서명, 키관리 및 인증서비스가 새롭게 부상하고 있으며, 여기에 적용하기 위한 정보보호 기반기술로는 암호화 기술, 인증기술, 전자서명, PKI(Pulbic Key Infrastructure) 및 WPKI(Wireless PKI)등이 있다. 암호화 기술은 합법적 참여자들 간에 메시지 변/복조 규칙에 대한 약속을 정하고, 이 규칙에 따라 송신하려는 메시지를 암호화시켜 전달 혹은 보관하며, 메시지 수신시 또는 접근 권한이 있는 사람이 필요에 따라 이를 복호화 하도록 하는 기술을 말한다. 전자서명은 종래의 종이 문서에 표기하던 수기 서명이나 인장 효과를 전자적 매체내에 저장 또는 전송되는 전자 문서상에 효과적으로 부여하는 전자적 서명 방식이다. e-business 활성화를 위한 정부의 적극적인 참여와 지원을 바탕으로 많은 전자상거래 관련 사업들이 등장하고 있다. 이러한 전자상거래에 사용되는 기반 기술로서 XML(eXtensible Markup Language)기술이 사용되고 있다. 본 연구에서는 서명값을 XML 문서 안에 포함해서 전송 하는 방법으로 문서의 무결성과 비밀성을 보장하고자 한다.

1. 서론

최근의 정보 네트워크와 컴퓨팅 기술의 발달은 많은 사람들이 통신하고 사업하는 방식을 변화시키고 있으며 공공분야와 민간분야 모두에게 폭넓은 영향을 주고 있다. 다양한 방법의 상거래가 수년 동안 행해져 왔지만 인터넷과 같은 개방 분산된 네트워크는 새로운 상거래의 가능성을 불러일으키고 있다. 이러한 새로운 종류의 상거래는 사업자와 소비자에게 넓은 시장, 다양한 제품과 서비스, 저 비용의 거래 등을 포함한 광범위한 이익을 제공한다. 전자서명 서비스 및 암호화, 복수서명 기능, 시점확인 기능 등을 추가한 기업관리 시스템은 기존 오프라인 상의 업무 프로세스와 비교해 볼 때 상당히 간략화 되었으며, 비용적인 측면에서도 상당한 이익을 창출하였다. 이러한 장점에 비해 전자상거래는 정보보안 분야에서 많은 취약점을 안고 있다. 보안기술의 사용, 즉 암호화, 인증과 같은 보안기술의 사용과 예측 가능한 규제환경은 전자거래에서 이러한 사업자와 소비자의 신뢰를 확립하기 위한 기반을 형성한다. 본 논문의 구성은 다음과 같다. 2장에서는 제안하는 모델을 제시하고 3장에서는 제시한 모델에 대한 설계 및 구현을 하고 4장에서 결론 및 향후 연구 방향을 제시 한다

2. 제안 모델

기존의 전자서명 모델은 문서를 해쉬 함수를 통해서 생성되는 해쉬 값에 서명자의 비밀키를 이용해서 서명을 한 후에 원래의 문서와 같이 전송하게 된다. 본 연구에서 제안하는 모델은 XMLDSIG에서 제안한 기본 조건을 따르고 추가적으로 문서의 서명값을 XML 문서 안에 포함해서 문서를 암호화 한 후에 상대방에게 전달하게 된다. 본 연구에서 제안하는 모델의 그림 1과 같다. 먼저 XML 문서를 해쉬 함수를 통해서 해쉬값을 얻고 이 값을 서명자의 개인키를 이용해서 서명하게 된다. 여기서 생성된 서명값을 XML 문서 안에 포함해서 개방 통신로를 통해서 전송하게 된다.

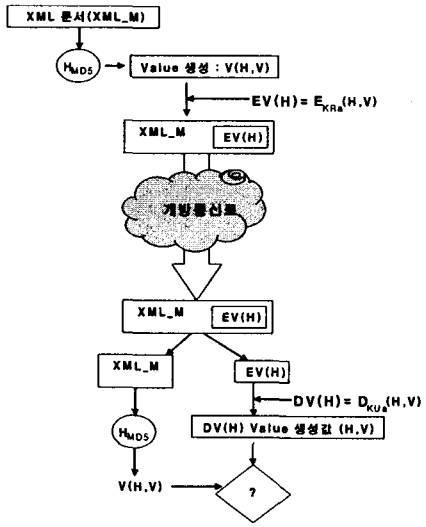

그림 1. XML 서명 생성 모델

■ 기호정의

- XML_M : XML 문서
- H : Hash 함수
- MD5 : MD5 이용
- $V(H,V)$: 해쉬 함수를 통해서 생성된 값
- $EV(H))$: 해쉬 값 암호화 한 값
- KR_a : 서명자의 개인키
- KU_a : 서명자의 공개키
- $DV(H)$: 해쉬 값 복호화 값
- H_{MD5} : MD5를 이용한 해쉬

2-1. 서명 생성 과정

그림 2은 XML 문서의 전자 서명 생성 과정을 보여주고 있다.

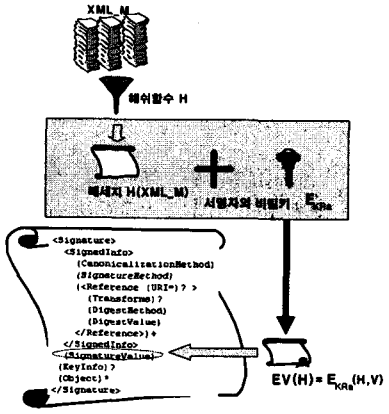

그림 2. 서명 생성 과정

먼저 XML 문서(XML_M)이 해쉬 함수를 통해 처리되어 나온 결과 H(XML_M)이 서명자의 비밀키를 이용해서 서명값을 생성하게 된다.

[서명하기]

(1) $H_{MD5} \leftarrow (XML\ M)$

(2) $V(H,V) \leftarrow H_{MD5}(XML\ M)$

(3) $EV(H) \leftarrow E_{KR_a}(H,V)$

(4) $Make\,XML\ M\,include\,EV(H)$

(5) $Send\,to\,B\,(XML\ M\ include\,EV(H))$

XML 서명은 간접 지정을 통해 임의의 디지털 콘텐츠에 적용된다. 데이터 객체에 다이제스트 값이 계산되어 그 결과 값이 원소 안에 들어가고, 다음에 그 원소가 다이제스트 되어 암호학적으로 서명 된다. :

```
<Signature ID?>
        <SignedInfo>
                <CanonicalizationMethod/>
                <SignatureMethod/>
                (<Reference URI? >
                        (<Transforms>)?
                        <DigestMethod>
                        <DigestValue1>
                </Reference>)+
        </SignedInfo>
                :
        <SignatureValue1>EV(H)
        (<KeyInfo1>)?
        (<Object ID1?>)*
                :
                        // 확장가능 //
</Signature>
                                :
```

생성된 서명값은 (SignatureValue1)안에 추가되게 된다.

2-2. 서명 검증 과정

그림 3은 전자서명 된 문서의 검증 과정을 보여주고 있다. 서명 값이 추가되어 전송되어 온 XML 문서는 수신측에서 서명값과 원래의 문서와 분리하게 된다. 원래의 문서는 $MD5$ 해쉬 함수에 의해서 $V(H,V)$를 생성한다. 분리된 서명값은 수신측에서 서명자의 공개키에 의해서 복호화 하게 된다. 복호화 된 값과 해쉬 함수를 통해서 생성된 값을 비교하게 되어 서명의 결과를 얻게 된다.

[검증하기]

(1) $Rece.\,from\ A\,(XMLM\ include\ EV(H))$

(2) $Separate\ EV(H)\ from\ XMLM$

(3) $V_B(H,V) = H_{B\ MD5} \leftarrow (XML\ M)$

(4) $DV(H) = D_{B\ KU_a}(H,V)$

(5) $if\ (V_B(H,V) == DV(H))$

$printf("signature\ OK")$

$else$

$printf("signature ERR.")$

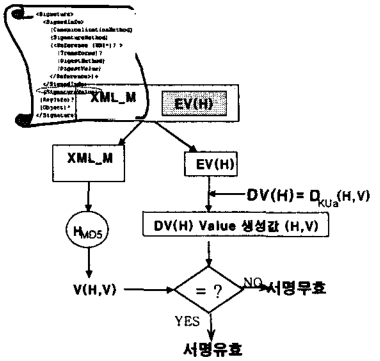

그림 3. 서명 검증 과정

2-3. 키 교환

Diffie-Hellman은 제한된 영역에서 멱의 계산에 비하여 이산대수 로그문제의 계산이 어렵다는 이론에 기초를 둔다. 이산대수 문제란 $y = g^x \ mod \ p$일때 g, x, p를 알고 y를 구하는 것이 쉽지만 y, p, g를 알고 x를 찾는 것이 어렵다는 것에 기반을 둔다. 키 교환 프로토콜은 다음과 같다.

A	공개정보 p, g, y_A, y_B	B
X_A : 비밀키	←	$X_B \in$ 비밀키
$y_A \equiv g^{X_A} mod\, p$	y_B	$y_B \equiv g^{X_B} mod\, p$
$K_{AB} \equiv (y_B)^{X_A}$	→	$K_{BA} \equiv (y_A)^{X_B}$
$\equiv g^{X_A X_B} \ mod\, p$	y_A	$\equiv g^{X_A X_B} \ mod\, p$

DH 알고리즘은 전송 당사자 양쪽이 서로 인증되지 않았으므로, "중간자" 공격에 취약하며, 이를 보완하기 위하여 추가적으로 전자서명 프로토콜이 사용된다.

3. 모델 설계 및 구현

3-1. 서명 생성

3-1-1. 해쉬값 생성

본 연구에서 서명을 위한 다이제스트는 MD5를 이용했다. 먼저 주어진 알고리즘으로 메시지 축약 인스턴스를 생성하는 단계이다.

MessageDigest md = MessageDigest.getInstance("MD5");

그리고 축약을 생성할 데이터를 읽어 들인다.

FileInputStream in = new FileInputStream(filename);

마지막으로 버퍼에 데이터를 입력하고 메시지 축약을 생성한다.

```
while((length = in.read(buffer)) != -1)
    md.update(buffer, 0, length);
byte[] raw = md.digest(); .
```

처리되어 나온 결과는 아래와 같다.

DigestValue : *?~RW?□SP?y}?ZT???w{?wI?R?zQ=*

위의 결과를 XML_M의 문서 〈DigestValue〉 태그 사이에 첨가 한다.

〈DigestValue〉?F??~AAX}W?G?E?{?X+P?A==〈/DigestValue〉

3-1-2. 서명 생성

[서명알고리즘]

(1) *getinstance() : 인스턴스 생성. 서명 생성 타입을 결정.*

Signature signature = Signature.getInstance("DSA");

(2) *initSign() : 전자서명을 사용자의 비밀키로 초기화*

signature.initSign((PrivateKey)keystore.getKey(alias, storepass));

(3) *Update() : 전자서명 메시지 입력*

```
while((length = ins.read(buffer)) != -1)
        signature.update(buffer, 0, length);
        ins.close();
```

(4) *sign() : 전자서명 생성*

FileOutputStream out = new FileOutputStream(signfile);

byte[] raw = signature.sign();

(5) *modPow() : 생성된 전자서명값 암호화*

BigInteger encText = m.modPow(e, n);

위의 서명 알고리즘에 따라서 구현한 결과이다.. 먼저 (4)단계에서 생성된 서명 값은 아래와 같다.

Sig.Value : *}Q?xQ|?UY?v?w???G?+WY?==*

생성된 서명값을 암호화 한 결과이다.

암호화된 Sig.Value : 0,ㄱ/œ~楸翎樓?h?E?ㄱ:??뵤= [illegible]

이렇게 처리해서 나온 결과를 XML 문서에 첨가한 결과는 아래와 같다.

```
<Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<SignatureMethod Algorithm="d:#Signature#DSA-MD5"/>
<Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
<Transforms>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
</Transforms>

<DigestMethod Algorithm="d:#Signature#MD5"/>
<DigestValue> )Q?xO|?UY?v?w???G?+WY?==</DigestValue>
</Reference>
</SignedInfo>

<SignatureMethod Algorithm="d:#Signature#DSA"/>
<SignatureValue>0,ㄱ ?œ ?楸翎樓?↕h?E?ㄱ ?:??뵤=    #??ㅗ? 暮濤</SignatureValue>

<KeyInfo>
<KeyValue>

<DSAKeyValue>
<P>...</P><Q>...</Q><G>...</G><Y>...</Y>
</DSAKeyValue>
</KeyValue>
</KeyInfo>
</Signature>
```

그림 4. 결과

3-2. 서명 검증

[검증알고리즘]

(1) *getinstance() : Signature 객체 생성.*

Signature signature = Signature.getInstance("DSA");

(2) *initVerify() : 전자서명한 사용자의 공개키로 초기화*

Signature.initVerify(keystore.getCertificate(alias).getPublicKey());

(3) *Update() : 전자서명에 사용한 메시지를 추가*

```
while((length = ins.read(buffer)) != -1)
        signature.update(buffer, 0, length);
ins.close();
```

(4) *modPow() : 생성된 전자서명값 복호화*

```
BigInteger dec = encText..modPow(d, n);
```

(5) *verify() : 전자서명 인증*

```
if(signature.verify(raw))
    System.out.println("Signature OK.");
else
    System.out.println("Signature ERROR"");
```

전자

4. 결론 및 향후 과제

전자서명 모델은 최근의 전자상거래 환경에서 많이 사용될 것으로 기대가 되는 모델이다. 이에 본 연구에서는 전자서명 모델에 대해서 간단한 구현을 해 보았고 무결성, 인증 요구사항을 충족시킬 수 있는 전자서명 모델임을 확인하였다. 설계된 조건의 구현은 자바를 이용하여 구현 하였다. 추후 연구될 사항은 본 연구에서는 구현하지 못한 안전한 개인의 지문을 이용한 키 생성 및 안전한 키 교환 방법에 대한 보다 중점적인 연구가 필요하고, 마지막으로 본 연구는 사용자 한명에 대한 서명 및 검증을 위하여 설계하지 않았다. 본 논문은 다자간의 계층적인 서명 모델의 첫 단계로 서명값을 포함한 전자서명 모델을 설계 및 구현 하였다. 추후, 본 연구를 바탕으로 다자간의 계층적인 서명 모델에 대한 연구가 필요하다.

참고문헌

[1] National Bureau of Standards, "Data Encryption Standard," *Federal Information Processing Standards Publication 46*, Jan. 1977.

[2] IST, NIST SP 800-16, "*Information Technology Security Training Requirement*", 1998.

[3] S. Bellovin and M. Merritt. "Augmented Encrypted Key Exchange : A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise," ACM Conference on Computer and Communications Security 1993, pp. 244~250.

[4] Richard E. Smith *"Internet Cryptography"* Addison Wesley, 5th-Edition, 2002.

[5] D.P. Jablon, "Strong Password-only Authenticated Key Exchange," *ACM Computer Communications Review*, vol.26, no.5, pp.5-26, October 1996.

[6] 이원구, 이재광, "자바기반의 타원곡선 알고리즘을 이용한 보안 메일 시스템의 설계 및 구현," 한국정보호진흥원, 2001

[7] "정보보호산업 시장동향 및 전망, 주간기술동향", 한국전자통신연구소, 1055호, 2002.

[8] 성태경, "인터넷 정보보안에 관한 연구", 산업연구 제11집, 1999. pp241~246