情報保護學會論文誌
第15卷 第4號, 2005. 8

D-OCSP-KIS에서 OCSP Responder의 세션 개인키의 노출을 검출하는 방법*

이 영 교,†‡ 남 정 현, 김 지 연, 김 승 주, 원 동 호
성균관대학교 컴퓨터공학과

A Method for Detecting the Exposure of an OCSP Responder's Session Private Key in D-OCSP-KIS*

Younggyo Lee,†‡ Junghyun Nam, Jeeyeon Kim, Seungjoo Kim, Dongho Won
Computer Engineering, Sungkyunkwan University

요 약

Koga와 Sakurai에 의해 제안된 D-OCSP-KIS는 OCSP Responder의 인증서 수를 줄여줄 뿐만 아니라 클라이언트에게 OCSP Responder의 인증서 상태 검증도 제공하여 주며 통신량, 계산량 그리고 클라이언트의 메모리량을 줄일 수 있는 효율적인 방법이지만 몇 가지 문제점도 가지고 있다. 공격자가 한 시간 주기(예, 1일)에서 OCSP Responder의 세션 개인키를 획득하였다면 OCSP Responder가 인식하지 못하는 경우, 한 시간주기 동안에 OCSP Responder를 사칭할 수 있다. 그리고 그는 가로챈 해쉬값을 이용하여 클라이언트에게 잘못된 응답을 보낼 수 있어 E-commerce상의 서버와 사용자는 심각한 혼란과 손해를 입을 수 있다. 아울러 해쉬 체인의 계산과 배포는 CA에게 부하가 될 수 있다. 따라서 본 논문에서는 D-OCSP-KIS에서 OCSP Responder의 세션 개인키의 노출과 해쉬값의 악용을 검출할 수 있는 방법을 제안하고자 한다. 이 방법에서 각 해쉬값은 OCSP Responder의 인증서 검증을 위해 한번씩만 사용이 되며 CA에서의 해쉬 체인을 위한 부하가 각 OCSP Responder로 분산되어진다.

ABSTRACT

D-OCSP-KIS proposed by Koga and Sakurai not only reduces the number of OCSP Responder's certificate but also offers the certificate status validation about OCSP Responder to the client. Therefore, D-OCSP-KIS is an effective method that can reduce the communication cost, computational time and storage consumption in client, but it has some problems. In case an attacker accidentally acquires an OCSP Responder's session private key in a time period (e.g., one day), she can disguise as the OCSP Responder in the time period unless the OCSP Responder recognizes. She can offer the wrong response to the client using the hash value intercepted. And the server and user on E-commerce can have a serious confusion and damage. And the computation and releasing of hash chain can be a load to CA. Thus, we propose a method detecting immediately the exposure of an OCSP Responder's session private key and the abuse of hash value in D-OCSP-KIS.

Keywords : *D-OCSP-KIS, OCSP Responder, D-OCSP, session private key*

접수일 : 2005년 4월 14일 ; 채택일 : 2005년 6월 15일
* 본 연구는 정보통신부 및 정보통신진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음.
† 주저자, ‡ 교신저자, yglee@dosan.skku.ac.kr

Ⅰ. Introduction

PKI(Public Key Infrastructure)는 공개키 기술을 이용하여 보안(무결성, 인증, 부인방지)을 제공해주는 광범위하고도 강력한 기술이다. PKI의 주 개념은 엔티티(사용자나 기관)의 개인정보와 그의 공개키를 묶어 전자서명을 한 인증서이다. 엔티티의 개인키가 노출되거나 엔티티의 개인정보가 변경되어지면, 엔티티는 CA(Certificate Authority)에게 자신의 인증서에 대한 폐지 요청을 하게 된다. 인증서가 폐지되었는지에 관한 정보를 CSI(Cetificate Status Information)라고 하며 CRLs(Certificate Revocation Lists)은 CSI를 위한 가장 잘 알려진 방법 중에 하나이다.[2,11]

CRL은 간단하다는 장점을 가지고 있지만 사용자와 CRL을 저장하고 있는 CA의 Directory(혹은 Repository)사이에 높은 통신비용을 필요로 한다는 단점을 가지고 있다. 그러므로 CSI의 크기와 통신비용을 줄이기 위하여 현재까지 여러 방법들이 제안되어 오고 있다. 여기에는 Delta-CRL, CRL DP(Distributed Point), Over-issued CRL, Indirect CRL, Dynamic CRL DP, Freshest CRL, CRT(Certificate Revocation Tree), NOVOMODO, Authenticated Directory 등이 있다.[1,2,4,7,9-12,14]

클라이언트나 사용자가 매우 시기적절한 CSI를 원한다면 OCSP(Online Certificate Status Protocol)같은 온라인 인증서 상태 검증 방법이 CRL과 같은 오프라인 방법보다 편리하다.[7] OCSP에서 클라이언트는 CA의 Directory로부터 CRL을 다운로드할 필요가 없으므로 높은 통신비용이 필요하지 않으며 클라이언트에서 CRL 저장을 위한 메모리가 필요하지 않다. 그러나 CSI 요청이 한 OCSP Responder에게 집중되면 OCSP Responder는 DoS(Denial of Service) 공격을 당하게 될 수 있다. DoS 공격 가능성을 줄이기 위하여 OCSP Responder는 짧은 시간 내에서 응답을 위한 서명값을 계산해 놓을 수 있다. 그러나 이것은 다시 재전송 공격의 가능성을 줄 수 있다.[13]

T-OCSP(Traditional-OCSP)에서 하나의 OCSP Responder에 대한 부하를 줄이기 위하여 D-OCSP(Distributed-OCSP)가 도입되었다.[3,13] D-OCSP에서 분산된 각 OCSP Responder들이 동일한 개인키를 갖는다면 개인키의 노출 가능성이 커지게 된다.[14] 그래서 D-OCSP에서 각 OCSP Responder는 일반적으로 다른 개인키를 가지며 클라이언트는 OCSP Responder의 CSI를 검증하기 위하여 모든 OCSP Responder의 인증서를 획득하여야 한다. 이를 위해서는 통신비용과 메모리 사용량이 증가하게 된다. 이러한 문제를 해결하기 위하여 Koga와 Sakurai에 의해 D-OCSP-KIS에서 단일 공개키가 도입되었다.[13] 그러나 D-OCSP-KIS에서는 단일 공개키의 길이가 OCSP Responder의 개수에 비례하여 커지게 되는 데, 염과 이가 제안한 D-OCSP-IBS(Distributed OCSP based on Identity-Based Signature)에서는 단일 공개키의 길이가 일정하거나 짧아지게 된다.[3]

D-OCSP-KIS는 OCSP Responder가 각각 개인키를 보유하지만 하나의 인증서를 갖도록 함으로써 인증서의 수를 줄여줄 뿐만 아니라 클라이언트에게 OCSP Responder의 인증서 상태 검증을 제공해준다. 그러므로, D-OCSP-KIS는 통신비용, 계산시간 그리고 메모리 소모량을 줄여주는 효율적인 방법이다. 그러나, D-OCSP-KIS는 몇 가지의 문제점을 가지고 있다. 예를 들어, 공격자가 한 시간 주기(예, 1 일)에서 OCSP Responder의 세션 개인키를 획득하였다면 마스터 개인키를 획득하지 않는 한 다른 OCSP Responder의 개인키를 계산해낼 수가 없다. 그리고 해쉬값은 역계산이 불가능하므로 이전 시간 주기의 해쉬값을 계산해낼 수 없다. 그러나 공격자는 OCSP Responder가 인식하지 못하는 동안, 한 시간주기 동안에 OCSP Responder를 사칭할 수 있다. 그리고 그는 가로챈 해쉬값을 이용하여 클라이언트에게 잘못된 응답을 보낼 수 있다. 그리고 E-commerce상의 서버와 사용자는 심각한 혼란과 손해를 입을 수 있다. 아울러 해쉬 체인의 계산과 배포는 CA에게 부하가 될 수 있다. 따라서 본 논문에서는 D-OCSP-KIS에서 OCSP Responder의 세션 개인키의 노출과 해쉬값의 악용을 검출할 수 있는 방법을 제안하고자 한다. 이 방법에서 각 해쉬값은 OCSP Responder의 인증서 검증을 위해 한번씩만 사용이 되며 CA에서의 해쉬 체인을 위한 부하가 각 OCSP Responder로 분산되어진다. 논문의 나머지 부분은 다음과 같이 구성되어진다. 2장에서 D-OCSP-KIS를 분석하고 문제점을 제시한다. 3장에서는 D-OCSP-KIS의 문제점에 대한 해결책을 제시한다. 4장에서는 제안한 방법과 기존의 다른 방법들을 비교한다.

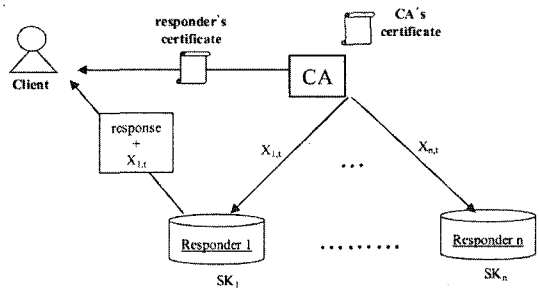


그림 1. D-OCSP-KIS의 개념

마지막으로 5장에서 결론을 맺는다.

II. D-OCSP-KIS와 그에 대한 분석

본 장에서는 D-OCSP-KIS를 분석하고 그에 대한 몇몇 문제점들을 제시한다.

2.1 D-OCSP-KIS

그림 1은 D-OCSP-KIS의 개념을 보여주고 있다. 그림은 CA와 n개의 OCSP Responder 그리고 클라이언트로 구성된다. D-OCSP-KIS는 다음의 특성을 갖는 일방향 해쉬함수 H를 사용한다.

[일방향 해쉬함수의 특성]

1. H는 적어도 전자서명보다 계산에 있어 10,000배 빠르다.
2. H는 입력 길이에 상관없이 20 바이트의 출력을 갖는다.
3. $H(X)=Y$에서 Y가 주어졌을 때, X를 구하는 역계산이 실제로 거의 불가능해야 한다.

[OCSP Responder의 인증서 발행]

1. T를 시간 주기의 전체 수라고 하자. 예를 들어, 각 OCSP Responder의 인증서가 발행 후 365일 후에 폐지된다고 하면 T는 365이다. CA는 다음과 같이 H를 이용하여 T개의 해쉬값들을 생성한다.

$$X_T \xrightarrow{h} X_{T-1} \xrightarrow{h} X_{T-2} \xrightarrow{h} \cdots \xrightarrow{h} X_1$$

2. OCSP Responder의 전체 개수를 n이라고 하자. CA는 다른 입력값 $X_{T,i}$를 이용, 반복적으로 n개의 해쉬체인을 생성한다. $X_{t,i}$는 OCSP Respoder i의 검증을 위한 시간주기 t의 해쉬값을 의미한다. 이 해쉬값들은 CA에 저장된다.

$$X_{T,1} \xrightarrow{h} X_{T-1,1} \xrightarrow{h} X_{T-2,1} \xrightarrow{h} \cdots \xrightarrow{h} X_{1,1}$$
$$\vdots$$
$$X_{T,n} \xrightarrow{h} X_{T-1,n} \xrightarrow{h} X_{T-2,n} \xrightarrow{h} \cdots \xrightarrow{h} X_{1,n}$$

3. CA는 자신의 개인키를 이용하여 다음과 같이 OCSP Responder의 인증서 C_{res}를 발행한다. SN은 인증서의 일련번호이고 V는 유효기간을 I와 S는 각각 인증서의 발행자와 소유자를 나타낸다. 그리고 $X_{1,1},\ldots,X_{1,n}$들은 n개의 OCSP Responder들을 위한 해쉬값들이다.

$$C_{res} = Sig_{K_{CA}}(PK_{res}, SN, I, S, V, X_{1,1}, \cdots, X_{1,n})$$

[OCSP Responder의 개인키 검증]

1. OCSP Responder i의 개인키 SK_i가 시간주기 t에 유효하다면 CA는 OCSP Responder i에게 해쉬값 $X_{t,i}$를 전달한다.
2. OCSP Responder i는 주기 t동안에 클라이언트에게 응답을 보낼때, $X_{t,i}$를 같이 전달한다.

$$< i, t, X_{t,i}, R, Sig_{SK_i}(R) >$$

3. OCSP Responder i로부터 응답을 받으면 클라이언트는 OCSP Responder의 공개키 PK_{res}를 이용하여 전자서명을 검증한다. 그때 클라이언트는 OCSP Responder의 인증서에 포함된 $X_{1,i}$와 수신된 해쉬값 $X_{t,i}$를 이용하여 OCSP Responder의 개인키에 대한 상태검증을 수행할 수 있다. 보다 상세히 클라이언트는 다음의 식을 확인한다. 이 수식이 만족되면, 클라이언트는 SK_i가 유효함을 확인할 수 있다.

$$X_{1,i} = H^{t-1}(X_{t,i})$$

이러한 방식으로 클라이언트는 OCSP Responder의 개인키 상태를 확인할 수 있다. 모든 OCSP Responder들의 개인키가 폐지되지 않는 한 유효기

간 내에 OCSP Responder의 인증서는 폐지되지 않는다.

2.2 D-OCSP-KIS의 특성

1. 통신비용 : OCSP Responder는 각기 다른 개인키를 보유하지만, 인증서는 단 하나이기 때문에 클라이언트는 여러 개의 OCSP Responder의 인증서를 획득할 필요가 없다. 또한 클라이언트는 OCSP Responder들에 대한 CRL을 획득할 필요가 없다. 그러므로 D-OCSP-KIS는 통신비용을 줄여줄 수 있다.
2. 메모리 소모량 : 클라이언트는 여러 개의 OCSP Responder 인증서들과 OCSP Responder에 대한 CRL을 획득할 필요가 없다. 그러므로, D-OCSP-KIS는 또한 클라이언트에서의 메모리 소모량을 줄일 수 있다.
3. 계산 비용 : OCSP Responder의 개인키 상태검증은 서명이 아닌 해싱 연산에 의해 수행된다. 해싱 연산은 전자서명보다 매우 빠르므로 클라이언트는 OCSP Responder의 개인키의 상태 검증을 위한 계산비용을 줄일 수 있다.

2.3 D-OCSP-KIS의 분석

[잘못된 해쉬값의 배포]

일반적으로 개인키의 노출이나 손실, 소유자의 접근 권한 변경, 발행자의 관계변경, 해독 경계 등의 원인으로 유효기간 내에서도 인증서는 폐지될 수 있다.[5] 대부분의 경우, 사용자는 CA에게 자신의 인증서 폐지를 요청하게 되며 CA는 폐지된 인증서들의 목록을 포함하는 CRL을 발행하게 된다. 인증서의 폐지 사유 중에, 개인키는 공격자에 의해 우연히 그리고 비밀리에 노출될 수 있다. 이러한 경우에, 사용자는 CA에게 자신의 인증서 폐지를 요청할 수가 없다. 한 OCSP Responder의 세션 개인키가 공격자에 의해 한 주기 동안(예, 1일)에 노출되었다고 가정하자. OCSP Responder는 CA에게 인증서 폐지를 요청할 수 없다. 그리고 CA는 세션 개인키가 노출되었음에도 불구하고 인증서의 상태를 확인하는, 잘못된 해쉬값을 해당 OCSP Responder에게 배포하게 된다. 결국, E-commerce에서 서버와 사용자는 심각한 혼란과 손해를 입을 수 있다.

[CA에 대한 추가적인 부하]

CA는 표 1과 같이 X-체인을 계산하고 저장하며 각 주기의 시작시마다 각 OCSP Responder에게 해쉬값들을 배포하여야 한다. OCSP Responder의 수가 1,000이고 시간 주기가 1일이라면, CA는 OCSP Responder에게 전체적으로 365,000번의 해쉬값 배포를 해야만 한다. CA는 기본 임무(인증서 발행, 폐지, CRL 발행 등)가 있기 때문에 해쉬값의 계산, 저장 그리고 배포는 CA에게 추가적인 부하가 된다.

[OCSP Responder의 세션 개인키 노출을 검출할 수 없음]

여기에서 세션키는 전자서명용 마스터 개인키의 노출에 대한 피해를 최소화하기 위해서 마스터 키를 서명에 직접 사용하는 대신에 서명키의 전체 사용기간을 나누어 각 짧은 기간에 각기 다른 세션키를 사용하도록 한 개인키이다. 물론 현재의 세션 개인키와 마스터키를 이용하여 다음 세션 개인키를 만들 수 있도록 되어 있다.[15]

예를 들어 한 공격자가 특정 세션의 개인키를 몰래 훔쳤다고 하자. 이 경우, 그는 해쉬값 $X_{t,i}$를 쉽

표 1. 시간 간격과 OCSP Responder의 수에 따른 CA의 부하

시간 간격		1 일	1 시간	1 분	15 초	1 초
1 OCSP Responder인 경우	계산비용	365 hashings	8,760 hashings	525,600 hashings	2,102,400 hashings	31,536,000 hashings
	저장공간	7.3 K bytes	175.2 K bytes	10.3 M bytes	41 M bytes	616 M bytes
	배포횟수	365 번	8,760 번	525,600 번	2,102,400 번	31,536,000 번
1,000 OCSP Responder인 경우	계산비용	365,000 hashings	8,760,000 hashings	525,600,000 hashings	2,102,400,000 hashings	31,536,000,000 hashings
	저장공간	7.13 M bytes	171.10 M bytes	10.06 G bytes	40.04 G bytes	601.56 G bytes
	배포횟수	365,000 번	8,760,000 번	525,600,000 번	2,102,400,000 번	31,536,000,000 번

게 얻을 수 있으나 SK^*를 획득할 수 없기 때문에 다른 OCSP Responder의 개인키는 계산해낼 수 없다. 그리고 그는 H가 일방향 함수이기 때문에 $X_{t+1,i}(H(X_{t+1,i})=X_{t,1})$를 구할 수 없다. 그러므로 공격자는 시간주기 t이후에 클라이언트를 속일 수 없다.[13] 그러나, OCSP Responder R_i가 자신의 세션 개인키가 주기 t동안에 노출되었다는 사실을 모른다면, 공격자는 t가 종료될 때까지 OCSP Responder로 가장할 수 있다. 그는 클라이언트에게 잘못된 OCSP 응답을 줄 수 있고 그리고 E-commerce의 서버와 사용자는 심각한 혼란과 손해를 입을 수 있다.[14]

Ⅲ. OCSP Responder의 세션 개인키의 노출을 검출하는 방법

앞에서 언급한 것처럼, D-OCSP-KIS는 OCSP Responder에게 잘못된 해쉬값을 배포할 수 있고, CA에게 추가적인 부하를 줄 수 있으며, 그리고 OCSP Responder의 세션 개인키의 노출을 검출할 수 없다. 그러므로, 이 장에서 우리는 D-OCSP-KIS에서 OCSP Responder의 세션 개인키의 노출을 검출하는 방법을 제안한다.

3.1 제안된 방법

[제약 사항]

1. n은 OCSP Responder들의 전체 수이고 m은 클라이언트들의 전체 수라고 하자. 일반적으로 n은 m보다 무척 적다($n \ll m$).
2. 최종 사용자는 클라이언트를 통해 CSI 서비스를 얻는다고 하자.
3. 클라이언트는 CA에 등록을 한 후에 OCSP Responder로부터 CSI 서비스를 받는다고 가정하자.

[각 OCSP Responder를 위한 해쉬값의 계산]

1. K를 한 OCSP Responder에서의 서명 사용 총 횟수라고 하자. 예를 들어, 응답에 대해 10,000번의 서명 연산 후에 OCSP Responder의 인증서가 폐지된다고 한다면 K는 10,000이 된다. 그러므로 OCSP Responder의 인증서는 10,000번 서명 연산 후에 폐지되어진다. OCSP Responder는 다음과 같이 H를 이용하여 해쉬값 X_K를 계산할 수 있다.

$$X_0 \xrightarrow{h} X_1 \xrightarrow{h} X_2 \xrightarrow{h} \cdots X_k \cdots \xrightarrow{h} X_K$$

2. OCSP Responder는 m개의 클라이언트들을 위하여 다른 입력값 $X_{j,0}$으로 m 해쉬체인을 반복적으로 계산한다. $X_{j,k}$는 클라이언트 j에서 검증을 위한, 시간 k의 해쉬값을 의미한다.

$$X_{1,0} \xrightarrow{h} X_{1,1} \xrightarrow{h} X_{1,2} \xrightarrow{h} \cdots X_{1,k} \cdots \xrightarrow{h} X_{1,K}$$
$$\vdots$$
$$X_{m,0} \xrightarrow{h} X_{m,1} \xrightarrow{h} X_{m,2} \xrightarrow{h} \cdots X_{m,k} \cdots \xrightarrow{h} X_{m,K}$$

3. 각 OCSP Responder는 다른 입력값 $X_{i,j,0}$으로 n×m 해쉬체인을 반복적으로 계산한다. $X_{i,j,k}$는 클라이언트 j를 위해 배포될, OCSP Responder i에서 계산된 시간 k의 해쉬값을 의미한다.

$$X_{1,1,0} \xrightarrow{h} X_{1,1,1} \xrightarrow{h} X_{1,1,2} \xrightarrow{h} \cdots X_{1,1,k} \cdots \xrightarrow{h} X_{1,1,K}$$
$$\vdots$$
$$X_{1,m,0} \xrightarrow{h} X_{1,m,1} \xrightarrow{h} X_{1,m,2} \xrightarrow{h} \cdots X_{1,m,k} \cdots \xrightarrow{h} X_{1,m,K}$$
$$\vdots \quad \textit{In OCSP Responder 1}$$
$$X_{i,j,k}$$
$$\vdots \quad \textit{In OCSP Responder i}$$
$$X_{n,1,0} \xrightarrow{h} X_{n,1,1} \xrightarrow{h} X_{n,1,2} \xrightarrow{h} \cdots X_{n,1,k} \cdots \xrightarrow{h} X_{n,1,K}$$
$$\vdots$$
$$X_{n,m,0} \xrightarrow{h} X_{n,m,1} \xrightarrow{h} X_{n,m,2} \xrightarrow{h} \cdots X_{n,m,k} \cdots \xrightarrow{h} X_{n,m,K}$$
$$\textit{In OCSP Responder n}$$

4. 각 OCSP Responder는 입력값 $X_{i,1,0},\ldots,X_{i,m,0}$과 모든 중간 해쉬값들을 저장하고 최종값 $X_{i,1,K},\ldots,X_{i,m,K}$를 CA에게 안전하게 전달한다.

[CA에서 OCSP Responder의 인증서 발행]

CA는 각 OCSP Responder로부터 $X_{i,1,K},\ldots,X_{i,m,K}$

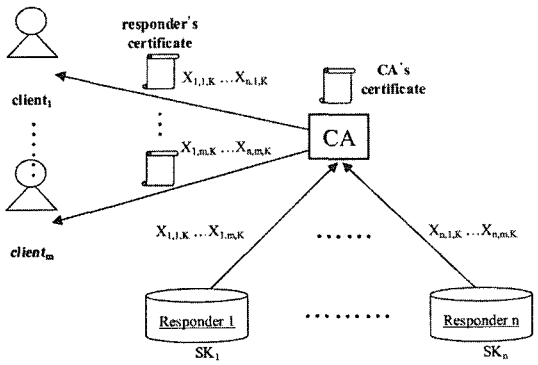


그림 2. 해쉬값의 계산과 OCSP Responder의 인증서 발행

를 모으고 자신의 개인키를 이용하여 클라이언트들에게 배포할 m개의 OCSP Responder의 인증서 C_{client_j}를 발행한다. SN은 인증서의 일련번호이고 V는 유효기간을 I와 S는 각각 인증서의 발행자와 소유자를 나타낸다. 이때, 각 인증서에 포함되는 해쉬값은 서로 상이하다. m개의 클라이언트들에게 배포되는 각 인증서의 내용은 다음과 같다.

$$C_{client_1} = Sig_{K_{CA}}(PK_{res}, SN, I, S, V, X_{1,1,K}, \cdots, X_{n,1,K})$$

Certificate for client 1

$$\vdots$$

$$C_{client_m} = Sig_{K_{CA}}(PK_{res}, SN, I, S, V, X_{1,m,K}, \cdots, X_{n,m,K})$$

Certificate for client m

[클라이언트에서 OCSP Responder의 세션 개인키의 상태 검증]

1. OCSP Responder i가 클라이언트 j에게 응답을 할때, 해쉬값 $X_{i,j,K}$도 함께 전달된다. OCSP Responder는 클라이언트에게 첫 번째 응답에 $X_{i,j,K-1}$을 전달하고 두 번째 응답에 $X_{i,j,K-2}$를 전달한다. 그러므로 마지막 응답에는 $X_{i,j,0}$를 전달하게 된다.

$$< X_{i,j,K-1}, R, Sig_{SK_i}(R) >$$

client j에서의 첫 번째 응답

$$\vdots$$

$$< X_{i,j,k}, R, Sig_{SK_i}(R) >$$

client j에서의 k 번째 응답

$$\vdots$$

$$< X_{i,j,0}, R, Sig_{SK_i}(R) >$$

client j에서의 마지막 응답

2. 클라이언트 j가 OCSP Responder i로부터 응답을 받게 되면, 클라이언트는 OCSP Responder의 공개키 PK_{res}를 이용하여 전자서명을 검증한다. 이때, 클라이언트는 응답에 수신된 해쉬값 $X_{i,j,k}$와 OCSP Responder의 인증서에 포함된 $X_{i,j,K}$를 이용하여 OCSP Responder의 인증서 상태 검증을 할 수 있다. 보다 상세히는, 클라이언트는 첫 번째 응답에서 1회의 해쉬 함수를 수행함으로써 SK_i의 유효함을 확인할 수 있고 두 번째 응답에서는 2회의 해쉬 함수를 수행함으로써 이를 확인할 수 있다. 그리고 마지막 응답에서는 K번의 해쉬 함수를 수행함으로써 확인할 수 있다. 그러므로 클라이언트는 평균 $K/2$(이 경우에는 5,000번)번의 해쉬 함수를 수행함으로써 OCSP Responder의 세션 개인키의 상태검증을 수행할 수 있다.

$$X_{i,j,K} = H(X_{i,j,K-1})$$

첫번째 응답

$$\vdots$$

$$X_{i,j,K} = H^k(X_{i,j,K-k})$$

k 번째 응답

$$\vdots$$

$$X_{i,j,K} = H^K(X_{i,j,0})$$

마지막 응답

3. 이때, 클라이언트는 각 응답에서 카운터 k를 저장하고 이전 응답에서의 카운터와 이를 비교한다. 현재 카운터가 이전 카운터보다 1이 크다면($C_{now} = C_{before} + 1$), 클라이언트는 응답이 유효함을 인식할 수 있으며, 그렇치 않으면 OCSP Responder의 세션 개인키의 노출과 해쉬값의 오용을 인식할 수 있게 된다.

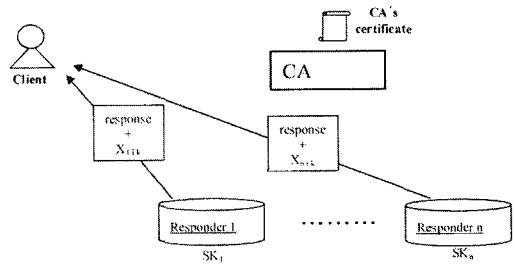


그림 3. OCSP Responder의 개인키 검증

[클라이언트에서 세션키의 노출을 검출하는 절차]

1. 클라이언트는 응답에 포함된 해쉬값 $X_{i,j,K}$를 이용하여 1회의 해쉬 함수를 계산하고 나온 해쉬값을 X_{temp}로 놓는다. 그리고 카운터 C_{now}를 1 증가시킨다.

$$C_{now} \leftarrow C_{now} + 1$$
$$X_{temp} \leftarrow H(X_{i,j,k})$$

2. 해쉬값 X_{temp}를 OCSP Responder i의 인증서에 포함된 $X_{i,j,K}$과 비교를 한다. 비교 결과가 같으면 단계 3으로 가고 그렇치 않으면 $X_{i,j,k}$를 X_{temp}로 놓고 단계 1로 돌아간다.

$$X_{i,j,k} \leftarrow X_{temp}$$

3. 클라이언트는 $C_{now} = C_{before} + 1$를 확인해본다. 성립하면 클라이언트는 응답을 억셉트하고 단계 4로 간다. 성립하지 않으면 클라이언트는 응답을 거절한다.

$$C_{now} \stackrel{?}{=} C_{before} + 1$$

4. 클라이언트는 C_{before}를 C_{now}로 셋하고 C_{now}는 0으로 셋한 뒤 다음 응답을 처리하기 위하여 단계 1로 간다.

$$C_{before} \leftarrow C_{now}$$
$$C_{now} \leftarrow 0$$

Ⅳ. 특성과 비교

이 장에서 우리는 제안된 방법의 특성과 전통적인 D-OCSP, 그리고 D-OCSP-KIS를 제안된 방법과 비교하고자 한다. 자세한 특성은 다음과 같다.

[세션 개인키의 노출과 해쉬값의 오용을 검출]

D-OCSP-KIS에서, 누구나 OCSP Responder에서 클라이언트로 전달되는 그리고 CA에서 OCSP Responder로 전달되는 해쉬값을 획득할 수 있다. 그러므로, OCSP Responder의 세션키를 우연히 획득한 공격자는 OCSP Responder를 사칭할 수 있으며 클라이언트를 속일 수 있다. 그리고 공격자는 OCSP Responder가 이를 인식하지 못하는 한 주기 동안에 클라이언트에게 잘못된 응답을 줄 수 있다. 그러나 제안된 방법에서는 클라이언트가 OCSP Responder의 세션 개인키 노출과 해쉬값의 오용을 즉시 검출할 수 있다.

[OCSP Responder의 개인키 사용 횟수]

제안된 방법에서, OCSP Responder의 개인키는 이의 유효 검증이 일회용 해쉬 함수에 의해 이루어지기 때문에 제한된 횟수까지만 사용된다. 위에서 사용횟수 K는 10,000으로 가정되었다. 이 경우, 클라이언트는 첫 번째 응답에서 1회의 해쉬함수 연산을 수행하며 10,000번째 응답에서는 10,000번의 해쉬 함수 연산을 수행하게 된다. 그러므로, 클라이언트는 표 2에서처럼, 평균 5,000번의 해쉬 연산을 수행하게 된다. 물론 K값은 평균값보다 더 크게(에, 20000, 50000, 100000) 혹은 더 작게(에, 8000, 5000, 2000) 설정될 수 있다. 그러나 검증을 위한 계산시간이 전자서명 시간보다 커지기 때문에, K를 10,000보다 더 크게 설정하는 것은 비효율적이다. 물론 각 OCSP Responder의 인증서에 서로 다른 3개의 해쉬값을 넣음으로써 클라이언트에서의 해쉬 연산 횟수는 줄어들 수 있다. 이 해쉬값들이 서로 다른 초기값으로부터 5,000번의 해쉬 연산을 통해 계산된다고 하자. 전체 사용 횟수 K는 15,000번이 되지만, 각 클라이언트는 검증을 위하여 단지 평균 2,500번의 해쉬 연산을 수행하면 된다. OCSP Responder가 전체 사용 횟수를 다 사용해버린다면, 자신의 개인키가 노출되지 않거나 공개키나 다른 정보가 변경되지 않는 한, 단지 해쉬값을 CA를 통하여 클라이언트에게 전달하면 된다.

[CA의 부하 분산]

D-OCSP-KIS에서는, 표 1에서처럼 CA는 모든 해쉬 체인을 계산해서 저장하여야 한다. 또한, CA는 주기적으로 각 시간 주기의 시작 시에 OCSP Responder들에게 해쉬값들을 배포하여야 한다. 이 작업은 CA와 각 OCSP Responder들 사이에 추가적인 패스를 필요로 하며 CA에게 부가적인 부하가 될 수 있다. 그러나, 제안된 방법에서는 각 OCSP Responder가 해쉬 체인을 계산한다. 그리고 CA는 이를 모아 OCSP Responder들의 인증서를 발행한다. 표 2에서와 같이 CA는 초기에 m개

표 2. 제안된 방법과 다른 방법들의 비교

			일반적인 D-OCSP	D-OCSP-KIS	제안한 방법
Responder의 인증서 구조			유지	변경(+20n byte)	변경(+20n byte)
클라이언트에서 필요한 Responder의 인증서 수			n	1	1
응답의 구조			유지	변경(+60 byte)	변경(+40 byte)
인증서 사용기간			365 일	365일	10,000번(+-가능)
CA의 부하	초기	계산량	n 서명	1 서명	m 서명
		패스	nxm(인증서전달)	m(인증서 전달)	m(해쉬값전달) m(인증서 전달)
	각 주기	계산량	응답x서명연산	mx배포할 해쉬값 계산 혹은 저장	-
		패스	-	nxT 패스	-
클라이언트에서의 계산량			서명 검증(최소)	t-해싱 연산	k-해싱 연산
잘못된 해쉬값 배포가능성			-	o	x
Responder의 개인키 노출 검출			x	x	o
해쉬값의 남용 여부 검출			-	x	o

주의. n : OCSP Responder의 갯수, m : 클라이언트의 개수, T : 시간 주기의 전체 수, x : 지원되지 않음, o : 지원됨

의 클라이언트에게 배포할 인증서를 서명해야 하지만 실제 응답서비스 기간 중에는 주기적으로 각 시간 주기의 시작 시에 OCSP Responder들에게 해쉬값들을 배포(nXT 패스)할 필요가 없게 된다. 또한 이를 위한 해쉬체인의 연산 및 저장 공간이 필요하지 않게 된다. 그러므로 우리가 제안한 방법은 CA에서의 부하가 각 OCSP Responder들에게 분산되어진다.

V. 결 론

본 논문에서 우리는 D-OCSP-KIS에서 OCSP Responder의 세션 개인키의 노출과 해쉬값의 오용을 즉시 검출할 수 있는 방법을 제안하였다. 제안된 방법에서, 해쉬값은 일회용 값이며, 이 방법은 CA에서의 부하가 각 OCSP Responder들에게 분산되어진다. 앞으로의 연구는 OCSP Responder의 개인키 사용 횟수를 늘이고 상태 검증을 위한 해쉬 연산 수를 줄이는 것이다.

참 고 문 헌

[1] A. Malpani, R. Housley, T. Freeman, "Simple Certificate Validation Protocol (SCVP)", *IETF Internet Draft*, June, 2002.

[2] C. Adams, P. Sylvestor, M. Zolotarev and R. Zuccherato, "Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols", *IETF RFC 3029*, February, 2001.

[3] Dae Hyun Yum, Pil Joong Lee, "A Distributed Online Certificate Status Protocol Based on GQ Signature Scheme", ICCSA 2004, *LNCS 3043*, pp.471-480, 2004.

[4] ITU/ISO Recommendation, "X.509 Information Technology Open Systems Interconnection-The Directory:Authentication", Frameworks, 2000.

[5] Jose L. Munoz, Jordi Forne, Oscar Esparza, and Miguel Soriano, "A Certificate Status Checking Protocol for the Authenticated Dictionary", MMM-ACNS 2003, *LNCS 2776*, pp. 255-266, 2003.

[6] Leo Reyzin, "General Time/Storage Tradeoffs for Hash-Chain Re-computation", unpublished manuscript.

[7] M. Myers, R. Ankney, A. Mappani, S. Galperin, C. Adams," X.509 Inter-

net Public Key Infrastructure Online Certificate Status Protocol - OCSP", IETF RFC 2560, June, 1999.

[8] NIST FIPS (Federal Information Processing Standards Publication) 186-1," Digital Signature Standard", December, 1998.

[9] P.C.Kocher, "On Certificate Revocation and Validation", Financial Cryptography (FC'98), *LNCS 1465*, pp.172-177, Springer-Verlag, 1998.

[10] Paul. Kocher, "Quick Introduction to Certificate Revocation Tree(CRTs)", Technical Report, Valicert, 1999.

[11] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF RFC 2458, January, 1999.

[12] R. Housley, W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF RFC 3280, April, 2002.

[13] Satoshi Koga, Kouichi Sakurai, "A Distributed Online Certificate Status Protocol with a Single Public Key, Public Key Cryptography", 2004, LNCS 2947, pp.389-401, 2004.

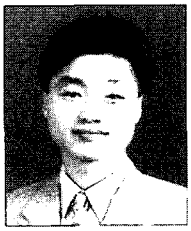
[14] Silvio Micali, "NOVOMODO : Scable Certificate Validation And Simplified PKI Management", 1st Annual PKI Research Workshop Preproceedings, pp.15-25, 2002.

[15] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung, "Key-Insulated Public Key Crytosystems", EUROCRYPT 2002, LNCS 2332, pp. 65-82, 2002.

[16] http://www.eskimo.com/~weidai/benchmarks.html.

〈著者紹介〉

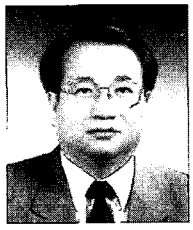
이 영 교(Younggyo Lee) 학생회원
1986년 2월: 한양대학교 전자공학과 (공학사)
1991년 6월: 한양대학교 대학원 전자공학과 (공학석사)
2002년 3월~현재: 성균관대학교 대학원 전기전자 및 컴퓨터공학부 박사수료
1993년 3월~1998년 9월: 대우통신 종합연구소 선임연구원
1999년 2월~2001년 6월: LG 전자/정보통신 중앙연구소 선임연구원
2002년 3월~현재: 인하공업전문대학 정보통신과 강사
2004년 9월~현재: 아주대학교 정보통신대학원 강사
〈관심분야〉 암호이론, 정보통신 보안, 네트워크 이론



남 정 현 (Junghyun Nam) 학생회원
1997년 2월: 성균관대학교 정보공학과 (공학사)
2002년 5월: Computer Science, University of Louisiana, Lafayette(M.S.)
2003년 3월~현재: 성균관대학교 대학원 정보통신공학부 박사수료
〈관심분야〉 암호 프로토콜, 암호이론, 네트워크 보안

김 지 연 (Jeeyeon Kim) 종신회원
1995년 2월: 성균관대학교 정보공학과 (공학사)
1997년 2월: 성균관대학교 대학원 정보공학과 (공학석사)
1996년 12월~현재: 한국정보보호진흥원(KISA) 선임연구원
〈관심분야〉 암호프로토콜, 키관리 등

김 승 주 (Seungjoo Kim) 종신회원
1994년 2월~1999년 2월: 성균관대학교 정보공학과 (학사, 석사, 박사)
1998년 12월~2004년 2월: 한국정보보호진흥원(KISA) 팀장
2004년 3월~현재: 성균관대학교 정보통신공학부 교수
2001년 1월~현재: 한국정보보호학회, 한국인터넷정보학회, 한국정보과학회, 한국정보처리학회 논문지 및 학회지 편집위원
2002년 4월~현재: 한국정보통신기술협회(TTA) IT 국제표준화 전문가
2005년 6월~현재: 교육인적자원부 유해정보차단 자문위원
〈관심분야〉 암호이론, 정보보호표준, 정보보호제품 및 스마트카드 보안성 평가, PET



원 동 호 (Dongho Won) 종신회원
1976년~1988년: 성균관대학교 전자공학과(학사, 석사, 박사)
1978년~1980년: 한국전자통신연구원 전임연구원
1985년~1986년: 일본 동경공업대 객원연구원
1988년~2003년: 성균관대학교 교학처장, 전기전자 및 컴퓨터공학부장, 정보통신대학원장, 정보통신기술연구소장, 연구처장.
1996년~1998년: 국무총리실 정보화추진위원회 자문위원
2002년~2003년: 한국정보보호학회장
현재: 성균관대학교 정보통신공학부 교수, 한국정보보호학회 명예회장, (정통부지정 ITRC) 정보보호인증기술연구센터 센터장
〈관심분야〉 암호이론, 정보이론, 정보보호