연산적 측면의 효율성을 향상시키는 신뢰 CA를 이용한 새로운 인증 경로 처리 기법

준회원 최 연 희*, 전 문 석**

A New Certificate Path Processing Scheme employed the Trusted CA for improving an efficiency on the Computational Aspect

Yeon-hee Choi*, Moon-seog Jun** *Regular Members*

요 약

최근에 PKI(Public Key Infrastructure)는 인증 경로 처리 작업을 사용자로부터 Delegated Path Discovery (DPD)와 Delegated Path Validation (DPV) 서버에게로 위임하는 추세이다. DPD/DPV 서버 구현의 최적의 조건은 높은 신뢰성을 제공하면서 저렴한 비용과 단순한 구현을 통해 구축하는 것이다. 본 논문에서는 사용자들이 가장 신뢰하는 신뢰 CA(Trusted CA)에 VA(Validation Authority)기능을 추가하여, 이를 DPD/DPV 서버로 사용하는 새로운 인증 경로 처리 기법을 제안하였다. 제안한 기법은 기존의 신뢰 CA를 위임 서버로 사용하기 때문에 단순한 구현을 통해 높은 신뢰성을 획득할 수 있다. 더불어, 경로 처리 작업으로 인한 신뢰 CA의 부담을 감소하기 위해서, 서명 검증 작업을 경로 상의 다른 CA들에게 위임하고, 반복되는 경로에 대한 처리 작업을 생략하도록 함으로서, 신뢰 CA를 이용한 검증 작업이 보다 효율적이고 빠르게 수행될 수 있도록 하였다.

key Words : PKI, trusted CA, certification path validation processing, DPD, DPV

ABSTRACT

The Public Key Infrastructure (PKI) trends to delegate the certificate path processing to the Delegated Path Discovery (DPD) Server and Delegated Path Validation (DPV) server recently. The most critical factor for the selection of the delegated server is to allow the server to be equipped with a high reliability through a low cost and simple implementation. In this paper, we propose a new certificate path processing scheme employed the trusted CA as the DPD/DPV server by adding the capability of the Validation Authority (VA) to the trusted CA. Since our proposed scheme uses the existing trusted CA as validation server, we can achieve a high trust through a simple implementation for the processing. Besides, we propose an additional scheme for reducing an overhead on the trusted CA. it is obtained by delegating digital signature verification to CAs on the path and by skipping the repeated path processing. As the result, our proposed validation scheme can be performed efficiently with high speed and low computational overhead.

* 숭실대학교 컴퓨터학과 컴퓨터통신 연구실(lovejung22@naver.com), ** 숭실대학교 정보과학대학 정교수
논문번호 : 030212-0519, 접수일자 : 2003년 9월 19일

Ⅰ. 서 론

고도의 정보화사회가 도래하면서 인터넷을 비롯한 정보 통신 기술이 급속하게 발전되었다. 이로 인해 인터넷을 이용한 전자 상거래와 같은 상업적 서비스가 널리 사용되면서 메시지의 무결성과 네트워크를 통하여 상대방의 신분 확인을 수행할 수 있는 인증 기술과 같은 정보 보호의 중요성이 점차 증대되고 있다. 현재 전자상거래 등 보안을 요구하는 정보 보호 시스템은 대부분 PKI를 사용하고 있다.

PKI는 공개키 암호화 방식을 사용하는 암호 시스템에서 사용자의 공개키를 안전하고 신뢰성 있게 공표하는 수단을 제공함으로서, 기밀성, 무결성, 인증, 부인 방지 등 다양한 형태의 암호학적 서비스 구현에 이용할 수 있는 종합적인 인증 체계 기반 기술이다[1][2].

통신하기에 앞서 사용자들은 통신하기를 원하는 상대방 즉, 타겟의 공개키 인증서나 인증서 안에 포함되어 있는 공개키가 올바른 것인지를 확인해야 하는데, 이를 위해서는 적합한 인증 경로를 설정해서 이 설정된 경로를 검증하는 인증 경로 처리 작업을 수행해야 한다. 인증 경로 처리 작업은 경로상에 있는 모든 인증서들 각각에 대해 서명에 대한 유효성 검증, 인증서 취소 상태 검증, 인증서 유효기간 검증, 주체 이름 공간에 대한 허용 이름 공간과 금지 이름 공간의 검증, 인증서 정책 처리 및 검증, 기타 확장 필드의 유효성 검증 등의 다양한 검증 과정을 수행해야 하기 때문에 많은 시간이 소요된다. 검증 작업 중 대부분의 시간은 서명을 검증하는데 소요되는데, 그 이유는 서명 검증을 위해서는 경로상의 각 인증서에 대해 한번 이상의 암호화 및 복호화(이하 암호화) 작업을 수행하기 때문이다. 암호학적 연산은 시간적으로 매우 비효율적인 작업으로서, 이 암호학적 연산을 수반한 서명의 유효성 검증 과정은 검증 작업의 연산적 효율성을 저하시키는 주된 요인이 된다[3],[4]. 따라서, 인증 경로를 처리하는데 필요한 모든 작업들을 사용자가 직접 수행하는 것은 상당한 부담이 될 수 있다.

이러한 문제를 해결하기 위한 다양한 방안들이 제안되어 왔고, 이들 중 가장 대표적인 방안이 인증 경로에 관한 모든 작업들을 온라인 서버에게 위임하는 방법이다. 인증 경로 설정 및 검증 작업만을 전담해서 수행하는 서버를 각각 DPD 및 DPV 서버라고 하는데, DPD/DPV 서버는 구현하는 PKI의 특성이나 환경 또는 사용자의 요구에 따라 새로운 서버를 구축하거나 기존의 서버에 VA의 기능을 추가하여 사용할 수도 있다. DPD/DPV 서버를 따로 두게 되면 복잡한 인증서 검증 모듈을 보유할 필요가 없어 사용자 측 운영 모듈이 단순해지고 경로 처리로 인한 연산적인 부담도 크게 축소될 수 있다. 또한 인증서의 실시간 상태 확인과 상호 연동과 관련된 복잡한 관리 기능을 대행함으로써 사용되는 상호 연동 방법에 관계없이 즉시 상호 연동을 가능하게 할 수 있을 뿐만 아니라 검증과 관련된 정책 설정 및 신뢰 관리를 중앙 집중적으로 제어할 수 있다는 많은 장점을 가진다. 반면에, 위임 서버를 위한 부수적인 시스템 도입이 필요하며, 항상 온라인 상태가 유지되어야 하고, 사용자에게 전적으로 신뢰되어야 하는 DPV 서버의 경우에는 서버와 사용자 사이의 안전한 키 분배 및 상호 인증서 검증 등의 다양한 메커니즘이 제공되어야 하는 등의 까다로운 문제가 수반될 수 있다[5],[6].

따라서 본 논문에서는 DPD 및 DPV 서버로 PKI 영역 내에서 사용자들이 가장 신뢰하는 그들의 신뢰 CA를 사용하는 새로운 인증 경로 처리 방식을 제안하였다. 제안한 방식은 사용자들이 본질적으로 자신들의 신뢰 CA를 신뢰한다는 PKI의 기본 특성 하에서 운영되기 때문에, 사용자들은 신뢰 CA 그 자체나 신뢰 CA가 수행한 모든 경로 처리 작업을 전적으로 신뢰할 수 있다. 뿐만 아니라, 기존의 서버를 사용한다는 측면에서 구현을 위한 비용이 추가로 들지 않고 신뢰성 유지를 위한 사용자와 위임 서버간의 키 분배 및 상호 인증과정 등의 부수적인 검증 과정이 필요하지 않음으로서 구현이 단순화될 수 있다는 장점을 가진다. 제안한 방식에서는 신뢰 CA가 하위 CA나 사용자들에 대해 인증서를 발행하고 그들을 관리하는 작업을 주된 작업으로 행하는 기본적인 운영 체계에는 변함이 없다. 따라서, 신뢰 CA가 이러한 기본적인 작업과 더불어 인증 경로의 설정 및 검증과 관련한 모든 작업을 수행하기에는 다소 부담이 될 수도 있기 때문에, 본 논문에서는 이러한 신뢰 CA의 부담을 감소할 수 있는 부가적인 방안 또한 제안하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 DPD/DPV 및 기존의 검증 알고리즘과 인증서 검증 방식등의 관련 연구들을 간단히 기술한다. Ⅲ장에서는 제안한 방식과 제안한 기법의 효율성을 향상시킬 수 있는 부가적 방안들을 자세히 소개한다. Ⅳ장에서는 제안한 기법의 성능을 분석하고 여러 가지

측면으로 기존의 검증 방식들과 비교한다. 마지막 V장에서는 결론을 내린다.

II. 관련 연구들

1. DPD & DPV

DPD 서버는 인증 경로를 설정하고 검증하는데 필요한 모든 정보들을 관련 디렉토리들로부터 얻을 수 있도록 해주고, 수집된 정보들에 사용자가 요청한 경로 설정 정책이나 디폴트 정책을 적용하여 하나 이상의 유효한 경로를 설정해주는 작업을 수행한다[7]. 수집된 모든 정보들은 이미 서명된 상태이고 검증 작업은 사용자가 직접 수행하기 때문에, DPD 서버는 사용자가 디렉토리를 신뢰하는 정도의 신뢰성만으로 구현이 가능하다. 사용자로부터 DPD 요청을 전송 받은 DPD 서버는 해당 정보들을 수집하고, 수집된 정보들을 이용하여 하나 이상의 적합한 인증 경로를 설정하여 이를 사용자에게 되돌려 준다.

DPD 서버에 의해 설정된 인증 경로는 DPV 서버에 의해 그 유효성이 검증된다. 사용자로부터 DPV 서비스를 요청 받은 서버는 사용자가 요청한 특정 검증 정책이나 디폴트 검증 정책을 이용하여 설정된 경로가 유효한지를 검증한다[7]. 최소의 지연을 중요시하는 응용 업무들에서는 서버에게 검증을 위임하는 것이 큰 장점으로 작용하는데, 이는 타겟 인증서에 대한 검증 요청을 DPV 서버에게 전송하고, 응답을 수신하고, 응답을 인증하는데 걸리는 시간들이 사용자가 직접 인증 경로를 설정하고 검증하는데 걸리는 시간보다 적게 걸리고, 또한 사용자가 이미 설정된 경로를 가지고 있다고 하더라도, 각 인증서에 대한 서명 검증과 관련한 처리 시간이 DPV 서버의 사용과 관련된 지연 시간보다 더 많이 걸리기 때문이다. 또한 일관된 형태로 정의되는 검증 정책을 이용함으로서 검증 정책들의 중앙 집중화된 관리가 필요한 사용자들에게도 큰 혜택이 될 수도 있다.

경로의 설정 및 검증은 사용자가 요청한 정책에 따라 수행된다. 정책은 수행되어야 하는 인증서 설정 및 검증에 대한 일련의 규칙 집합으로서, 크게 3개의 요소, 즉, 인증서 처리의 시작을 위해 사용될 일련의 신뢰 CA들과 인증 경로 처리를 위한 초기 조건들을 명시하는 인증 경로 요구사항들, 인증서 취소 정보의 소스를 나타내는 인증서 취소 요구사항들, 타겟 인증서가 특정 형태들이나 값들을 가지는 특정 확장자들을 포함하도록 요청하기 위한 타겟 인증서의 요구사항들로 구성된다. 이러한 정책 정의들은 꽤나 복잡하고, 어떤 정책들은 루트 자체 서명된 인증서들과 같은 몇 개의 파라미터들의 설정을 하도록 한다. CVP (Certificate Validation Protocol)[8], OCSPv2 (Online Certificate Validation Protocol v2)[9], SCVP (simple Certificate Validation Protocol)[10]와 같은 인증서 처리 온라인 프로토콜들은 사용자로 하여금 요청에 이러한 정책 의존 파라미터들을 포함하도록 해야 한다. 그러나 대부분의 사용자들은 주어진 응용에 대한 정책을 간단히 참조하거나 DPV 서버의 디폴트 검증 정책을 허용한다[7],[8].

2. 인증서 검증

1) rfc 2459 검증 알고리즘

통신하고자 하는 타겟의 공개키가 올바른지를 검증하기 위해서는 검증자 자신의 신뢰 CA와 타겟 사이의 인증 경로가 존재하고 이 인증 경로가 올바른지를 확인하기 위한 경로 설정 및 검증 작업을 수행해야 한다. Housley et al.이 제안한 rfc 2459 인증 경로 검증 알고리즘은 검증을 위한 기본 알고리즘으로 주로 사용된다.

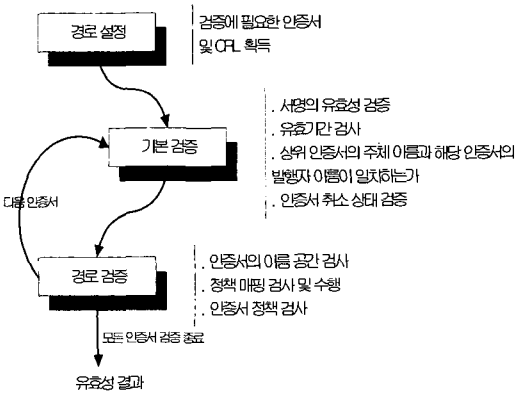


그림 1. rfc 2459 인증 경로 검증 과정

Rfc 2459[4]의 인증서 검증 과정은 크게 경로 설정, 기본 검증, 경로 검증 등의 3가지 과정으로 수행된다. 그림 1 은 이들의 과정을 간단히 도식화한 것이다. 그림 1에서 보이는 것과 같이, 검증자는 경로 설정에 필요한 모든 인증서 및 CRL들을 수집

하여 적합한 인증 경로를 설정하고 설정된 경로에 대한 기본 검증 및 경로 검증을 수행한다. 이 때 검증 작업은 경로상의 모든 인증서들에 대해 수행되며, 각 인증서들은 검증을 위한 조건들 중의 하나라도 만족하지 못하면 설정된 인증 경로는 유효하지 않은 것으로 판단되어 다른 후보 경로를 새롭게 설정하고 이를 입력으로 받아들여 검증을 다시 시작한다. 인증 경로 처리 방식은 검증 알고리즘의 수행 주체에 따라 크게 사용자 검증 방식과 SCVP 검증 방식으로 분류될 수 있다.

2) 인증서 검증 방식

(1) 사용자 검증 방식

이 방식은 사용자가 인증 경로를 설정하고 검증하는데 필요한 모든 작업들을 직접 수행하는 검증 방법으로서, 사용자는 해당되는 인증서,CRL,OCSP 응답들을 다운받아 검증 알고리즘에 따라 직접 경로를 설정하고 검증한다. 따라서 검증 과정 및 결과에 대한 높은 신뢰성을 가지는 반면, 사용자 측의 복잡한 작업의 수행으로 인한 부담이 커지고, 중앙 집중화된 인증서 검증 작업의 수행을 제한하여 궁극적으로 PKI의 확산을 저해하는 원인이 된다.

(2) SCVP 검증 방식

SCVP는 인증서 유효성과 상태를 문의하거나, 인증 경로를 찾거나, 인증 경로에 대한 유효성을 문의하기 위한 대표적인 온라인 프로토콜이다. 이는 SCVP 서버와 사용자간에 수행되며, 사용자는 SCVP 서버에게 인증 경로 설정, 설정된 인증 경로의 검증, 또는 인증서 취소 상태 등을 요청하면, 서버는 사용자의 요청에 따라 인증 경로 및 인증 경로 검증 결과 및 인증서 취소 상태 등을 포함하는 응답을 전송한다. 앞에서도 언급했던 것처럼, SCVP 방식은 응용 업무들이 PKI를 채용함에 있어서 인증서 검증과 상태를 확인하기 위하여 요구되는 절차와 부담을 간단히 하여 사용자 측 프로그램의 부피와 크기를 간소화할 수 있으며, 일관성 있는 중앙 집중화된 인증서 검증 정책을 유지할 수 있도록 한다. 반면에, SCVP 서버를 위한 부수적인 시스템 도입이 필요하며 사용자와 SCVP 서버사이의 상호 인증을 위한 검증 과정이 수반되어야 하는 등의 단점을 가진다[10].

Ⅲ. 제안한 방식

1. 신뢰 CA를 이용한 인증 경로 처리

제안한 방식은 위임 서버를 새로이 구현하는데 부수적으로 따르는 비용과 안전도를 위한 절차 등의 복잡한 문제를 해결하고, 보다 단순한 구현을 통하여 높은 신뢰성을 제공하기 위해 기존의 신뢰 CA를 DPD/DPV 서버로 사용하도록 하였다. 신뢰 CA는 사용자들에 의해 전적으로 신뢰된다는 PKI의 기본적인 특성에 더하여, CA 자체의 신뢰성 및 안전성, 부여된 법적 책임 등의 VA로서의 수행을 위해 기본적으로 필요한 다양한 특성들을 가진다. 또한 이미 신뢰 CA와 사용자간의 공개키를 알고 있음으로 해서 상호 간의 키의 분배 및 상대방의 인증서 검증 작업을 따로 수행하지 않아도 되기 때문에 새로운 서버를 구축할 때보다 구현이 보다 간소화될 수 있다는 장점을 가진다. 그림 2. a는 SCVP 서버를 따로 구축한 온라인 검증 방식을 나타내고, 2.b는 제안한 인증서 검증 방식을 보인다.

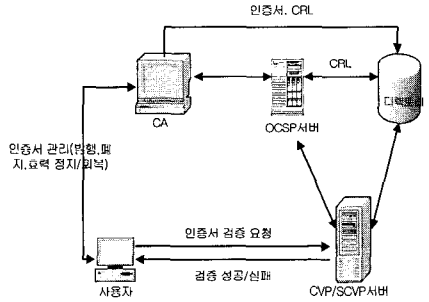


그림 2-a. SCVP 검증 방식

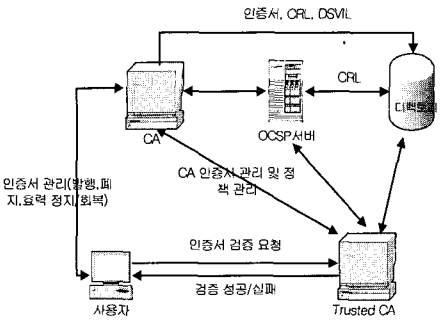


그림 2-b. 신뢰 CA를 이용한 제안한 방식

SCVP 방식에서는 사용자들이 SCVP 서버에게 인증서 검증을 요청하면, SCVP 서버가 필요한 정보들을 수집하여 유효한 경로를 설정 및 검증한 후,

검증 결과를 사용자에게 되돌려주는 방식으로 수행된다. 반면에, 제안한 방식에서는 사용자들이 자신의 신뢰 CA에게 경로 설정 및 검증과 관련한 모든 요청을 전송하게 되고, 신뢰 CA는 필요한 인증서와 CRL외에, DSVIL (Digital Signature Validation Information Lists)을 추가로 수집하여 요청된 인증서의 유효성을 결정한다. DSVIL은 신뢰 CA의 검증에 대한 부담을 덜기 위해 제안한 방안의 하나로서, 각 CA가 인증서의 서명을 검증하여 검증 결과로서 발행한 디지털 서명 검증 정보이다. DSVIL에 대한 자세한 설명은 다음 절에서 기술할 것이다.

SCVP 방식에서는 사용자와 SCVP 서버 사이의 상호 인증을 위해 요청과 응답 시에 각각 자신들의 인증서를 포함해야 한다. 반면에, 제안한 방식에서는, 사용자와 신뢰 CA 사이에 이미 공개키가 알려지고 서로를 신뢰하기 때문에 상호 인증 과정을 따로 수행할 필요가 없다. 따라서, 사용자는 요청 시에 신뢰 CA의 인증서를 정책에 포함하지 않아도 되고, 신뢰 CA도 응답 시에 자신의 인증서를 전송하지 않아도 된다. 또한 기존 신뢰 CA의 운영 모듈에 검증 모듈만을 추가하면 되기 때문에 서버의 구축비용 또한 절감할 수 있다. 결과적으로, 신뢰 CA를 이용함으로서 보다 단순한 구현 절차와 저렴한 비용으로 안전한 DPD/DPV 서버를 구축할 수 있다. 그러나, 검증 작업의 추가로 인해 신뢰 CA의 부담이 커질 수 있기 때문에, 본 논문에서는 이러한 부담을 감소하기 위한 2개의 부가적 방안을 제안하였다.

2. 서명 검증의 위임

1) 배경

제안한 방식에서, 신뢰 CA는 기본적으로 인증서와 CRL 발행 및 관리, 공개키 전달, 루트 CA인 경우의 정책 수립 및 관리, 하위 CA들의 관리 등의 기본적인 작업 외에도 인증서 검증과 관련한 VA로서의 모든 작업을 수행해야 한다. 따라서, 검증과 관련한 부가적 수행은 신뢰 CA에게 부담을 줄 수 있기 때문에, 본 논문에서는 이러한 부담을 감소시키기 위해 인증서들의 서명 검증을 PKI 영역의 다른 CA들에게 위임시키는 방안을 제안하였다. 서명 검증 작업은 반복적인 암호화 계산으로 인해 많은 시간이 소요됨으로서, 검증자에게 연산과 속도 면에서 많은 부담을 주는 가장 큰 원인이 된다. 일반적으로, 인증서의 디지틀 서명은 RSA나 DSA 공개키 암호 알고리즘을 이용하여 서명되거나 검증된다. 우선, 인증서 발행자는 SHA1이나 MD5와 같은 단방향 해쉬 함수를 이용하여 인증서 내용에 대한 해쉬를 계산하고, 이 해쉬 계산한 값에 암호화 알고리즘을 이용하여 서명한다[11]. 검증자는 서명을 검증하기 위해서, 인증서에 서명한 인증서 발행자의 정확한 공개키를 알아야 하고, 한번의 암호화와 해쉬 계산을 함으로서 서명을 검증한다.

서명 검증 작업은 경로 상에 있는 모든 인증서들에 대해 수행되기 때문에, 인증 경로의 서명을 검증하기 위해서는 인증 경로의 길이 만큼의 암호화와 해쉬 계산을 수행해야 한다. 암호화 계산 시간은 해쉬 계산 시간보다 대략 100 - 10000배 정도가 더 소요됨으로서, 연산과 속도 부담의 주된 원인이 된다[12]. 따라서, 새로운 PKI와의 연결 등으로 인해 인증 경로가 기하 급수적으로 커질 경우에, 제안한 방식의 신뢰 CA는 사용자만큼은 아닐지라도 서명 검증으로 인한 연산적인 부담을 가질 것이다.

따라서, 신뢰 CA가 서명 검증을 PKI 영역의 다른 CA들에게 위임하여 서명 검증에 따른 부담을 그들에게 일괄적으로 분배하게 되면, 신뢰 CA의 검증에 대한 부담이 감소될 수 있을 뿐만 아니라, 인증서의 발행 및 하위 CA의 감시등의 작업 외에는 거의 정적인 상태에 있는 CA들을 활용할 수 있다는 CA의 활용도 측면에서 좋은 효과를 가져올 수 있을 것이다.

2) 각 CA의 서명 검증

제안한 기법은 신뢰 CA가 사용자나 하위 CA들의 공개키 및 비밀키를 생성하여 분배하는 것을 기본으로 하고, 각 CA는 같은 키를 사용하여 인증서와 CRL에 서명한다고 가정한다. 따라서, PKI 영역내의 모든 CA들은 자신이 발행한 인증서 주체인 사용자나 하위 CA들의 공개키를 알기 때문에 이들이 발행한 인증서들의 서명을 검증할 수 있다. 그림 3은 계층적 PKI에서 각 CA들이 서명 검증 할 수 있는 인증서들을 나타낸다.

그림 3에서, 신뢰 CA와 사용자들의 인증서를 발행한 CA들을 제외한 모든 CA들은 자신의 하위 CA들이 발행한 인증서에 대한 서명 검증 작업을 수행한 후 서명 검증 정보인 DSVIL을 발행하여 공개해야 한다. 즉, 신뢰 CA는 자신이 검증자이기 때문에 자신이 발행한 자체 서명된 인증서나 직속 하위 CA에게 발행한 인증서에 대해서는 서명 검증할 필요가 없다. 또한 자신의 직속 하위 CA들이 발행

한 인증서의 서명에 대해서는 암호학적 계산을 통해 검증하기 때문에 이에 대한 DSVIL을 발행할 필요가 없다.

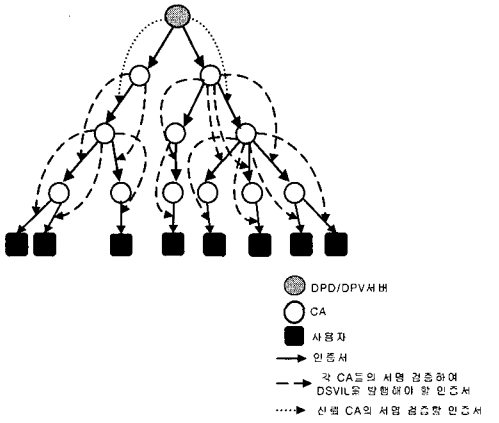


그림 3. 각 CA의 서명 검증할 인증서

신뢰 CA로부터 타겟 A로의 인증 경로가 자체 서명된 인증서 c0를 포함한 {c0, c1, c2, c3, c4, c5}이라고 했을 때, 그림 4는 각 CA들이 수행해야 하는 인증서의 서명 검증 작업을 보인다.

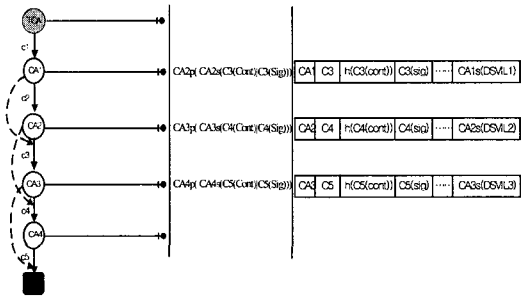


그림 4. 각 CA들의 서명 검증 및 DSVIL 발행

여기서, 신뢰 CA를 TCA라고 했을 때, TCA와 사용자 A의 인증서를 발행한 CA4를 제외한 모든 CA들은 해당되는 인증서들의 서명을 검증해야 한다. 즉, CA1은 자신이 인증서를 발행해 준 CA2의 공개키를 알기 때문에, CA2가 발행한 인증서인 c3의 서명을 검증하고, CA2는 CA3의 공개키를 이용하여 c4를, CA3는 CA4의 공개키를 이용하여 c5를 각각 검증한다. 이들은 검증 결과인 DSVIIL을 발행하여 사용자들에게 공개한다. 여기서, 자체 서명 인증서 c0는 검증되지 않고, c1과 c2에 대해서는 암호학적 계산을 통해서 검증된다. 단, c1에 대해서는 제 3자로부터 위조되지 않았다는 보장이 되는 한, 서명 작업이 생략될 수도 있다.

3) DSVILs

인증서들에 대한 서명을 검증한 CA는 검증한 결과로서 DSVIL을 생성하여, 이를 사용자들에게 공개한다. DSVIL은 다음의 항목들로 구성된다.

DSVIL = (Issuer-Id|SN|Cert-Id|VT|VS|H(Cert(cont))
|Cert(Sig)|Alg|E|Sign)

- ●Issuer-Id : 발행자 이름
- ●SN : 발행자에게 유일한 일련 번호
- ●Cert-Id : 서명 검증한 인증서 Id
- ●VT : 서명 검증한 시간
- ●VS : 검증 상태
- ●H(Cert(cont)) : 검증한 인증서 내용의 해쉬값,
- ●Cert(Sig) : 검증한 인증서의 서명값
- ●Alg : 사용한 서명 및 해쉬 알고리즘
- ●E : 확장자
- ●Sign : 발행자의 서명

여기서, SN은 DSVIL을 식별하기 위한 발행자에게 유일한 일련 번호이고, VS는 인증서의 서명 검증 상태를 나타내고 이는 valid, invalid, *unknown* 의 3가지 형태로 표시된다. H(Cert(cont))와 Cert(Sig)는 각각 서명 검증한 인증서 내용의 해쉬 값과 서명 값으로서, 이들은 인증서의 무결성과 서명의 합법성을 검증하는데 사용된다.

CA들은 항상 하위 CA들의 행위를 감시하여 이들이 새로운 인증서를 발행한다거나 기존의 인증서를 취소한다거나 인증서 내용을 갱신할 경우에는 그에 따라 새로운 DSVIL을 추가, 삭제하거나 기존의 DSVIL의 내용을 갱신해야 한다.

신뢰 CA는 요청된 인증서에 대한 경로 설정을 위해 필요한 인증서, CRL, DSVIL들을 수집하여 적합한 경로를 설정하고, 설정된 경로에 대한 유효성을 판단한다. 이 때, 신뢰 CA는 서명의 검증을 위한 기존의 암호화 작업 대신 수집된 DSVIL을 통한 서명 검증 작업을 다음과 같이 수행함으로서, 검증한 인증서 내용의 무결성과 서명의 합법성을 검사한다.

DSVIL(H(Cert(cont)))=Computed(H(Cert(cont)))
DSVIL(Cert(Sig))=Original(Cert(Sig))

- ●DSVIL( ) : 수집된 DSVIL의 해쉬값과 서명값
- ●Computed(H(Cert(cont))) : 원래 인증서 내용에 해쉬 계산한 값
- ●Original(Cert(Sig)) : 원래 인증서의 서명 값

즉, 원래의 인증서의 내용에 해쉬 계산을 하여

이 값을 DSVIL의 해쉬 값과 비교함으로서 인증서 내용을 무결성을 검증하고, 원래 인증서의 서명 값을 DSVIL의 검증한 인증서에 대한 서명 값과 비교함으로서 인증서 서명의 합법성을 검증한다. 제안한 검증 과정은 암호화 계산 없이 오직 해쉬 계산만으로 검증 기능을 수행함으로서 기존의 서명 검증 과정으로 인한 부담을 상당히 줄일 수 있다.

이 서명 검증 작업을 통해 신뢰 CA는 DSVIL이 원래의 인증서에 대한 서명 검증의 결과라는 것을 확인할 수 있지만, 이 검증 과정만으로는 DSVIL의 서명 검증 결과나 DSVIL을 발행한 CA들이 신뢰할 만한 것인지는 확신할 수 없다. 비록, 대부분의 검증 작업을 신뢰 CA가 직접 수행함으로서 경로상의 CA들이 이미 어느 정도는 신뢰된다 할지라도, 서명 검증을 수행한 CA들이나 그들이 발행한 DSVIL에 대한 신뢰성을 획득하기 위해 DSVIL 검증을 포함한 몇 가지 부가적인 검증을 수행해야 할 것이다. DSVIL의 검증은 신뢰 CA에 의한 경로상의 다른 CA들로의 DSVIL의 요청 및 응답의 과정을 통해 이루어진다.

4) DSVIL의 검증

(1)DSVILRequest/Response & RelayMessage의 전송

요청된 인증서에 대한 유효성을 판단하기 전에 신뢰 CA는 DSVIL의 무결성 및 서명의 합법성을 검사하기 위한 부가적인 검증 작업을 수행한다. 이 작업은 신뢰 CA가 경로상의 DSVIL을 발행한 CA들에게 그들이 발행한 DSVIL의 전송을 요청하는 DSVILRequest, DSVILRequest를 수신한 CA로부터 경로상의 상위 CA들로 순차적으로 전송되는 RelayMessage(RM), RM의 종료와 함께 DSVILRequest에 대한 응답으로 생성되어 신뢰 CA에게 전송되는 DSVILResponse를 통해 이루어진다. 신뢰 CA는 DSVILRequest를 경로상의 DSVIL을 발행한 CA들 중에서 맨 하위의 CA인 DLCA(DSVIL-issued Lowest CA)에게 전송하고, DLCA는 RM을 생성하여 이를 바로 상위의 CA에게 전송한다. 이 RM의 전송은 DSVIL을 발행한 CA들 중에서 맨 상위의 CA, 즉 DHCA(DSVIL-issued Highest CA)에게 까지 전송된다. DHCA는 자신이 RM의 마지막 수신자라는 것을 인지하고, 최종적으로 DSVILResponse를 생성하여 이를 자신의 직속 상위 CA, 즉, 신뢰 CA에게 보냄으로서 모든 전송과정이 완료된다. 이 전송 과정은 그림 5와 같다.

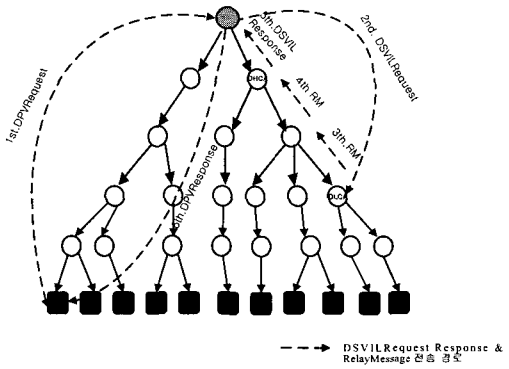


그림 5. DSVIL 검증을 위한 DSVILRequest/Response & RelayMessage 전송경로

그림 5에서, 사용자 A가 사용자 B의 인증서의 유효성을 알고자 할 때, A는 신뢰 CA에게 B의 인증서를 검증해 달라는 DPVRequest를 전송한다. 이를 전송 받은 신뢰 CA는 필요한 인증서, CRL, DSVIL등을 수집하여 유효한 인증 경로를 발견하고, DSVIL을 요청하는 DSVILRequest를 DLCA에게 전송하면, DLCA는 자신이 발행한 DSVIL을 포함한 RM을 생성하여 상위 CA에게 보낸다. RM을 수신한 모든 CA들은 자신이 발행한 DSVIL을 RM에 덧붙이고, Count를 1씩 감소한다. Count가 1인 RM을 수신한 CA, 즉, DHCA는 최종적으로 모든 경로상의 DSVIL들이 포함된 DSVILResponse를 생성하여 이를 신뢰 CA에게 전송함으로서 DSVIL의 요청을 위한 전송 과정은 완료된다. 신뢰 CA는 DSVILResponse의 검증을 수행한 후, 최종적으로 B의 인증서의 유효성을 판단하여, 이 결과인 DPVResponse를 A에게 전송한다.

(2) DSVILRequest

DSVILRequest는 신뢰 CA가 DLCA에게 경로상의 모든 DSVIL의 정보를 전송해달라는 요청문으로서, 이를 수신한 DLCA는 RM을 생성하여 이를 자신의 상위 CA에게 전송한다. DSVILRequest의 구성은 다음과 같다.

여기서, Nonce는 리플레이 공격을 방지하기 위해 포함된 임의의 수로서, 신뢰 CA는 DSVILRequest마다 다른 수를 포함해야 한다. Count는 경로상의 DSVIL의 수로 초기화되고, DSVILRequest를 수신한 DLCA는 이 값에 1을 감

소하여 RM에 덧붙인다.

DSVILRequest=(Req-Id|Rep-Id|Service|Nonce|DCAs|Path|Indicator|Count|Alg|E|Sign)

- ●Req-Id : 요청자 Id
- ●Rep-Id : 응답자 Id(DLCA의 Id)
- ●Service : 요청을 나타내는 서비스 형태
- ●Nonce : replay를 방지하기 위한 난수
- ●DCAs : DSVIL을 발행한 CA들의 목록
- ●Path : 인증 경로상의 Cert-Id목록
- ●Indicator : 응답으로 되돌려줄 정보를 나타내는 지시자(DSVIL내용의 해쉬값, 서명값등)
- ●*Count* : 경로상의 DSVIL의 수
- ●Alg : 서명 및 해쉬 알고리즘
- ●E : 확장자
- ●Sign : 신뢰 CA의 비밀키로 암호화된 서명

신뢰 CA는 DSVILRequest를 DLCA에게 전송한 후 DSVILResponse가 돌아올 때까지 사용자로부터 요청된 타겟 인증서의 유효성을 판단하는 것을 보류해야 한다. 따라서, DSVILRequest나 RM의 메시지 전송 시 발생할 수 있는 딜레이로 인해 DSVILResponse의 응답 시간이 길어지면 유효성 판단 시간도 그만큼 늦어지게 된다. 이는 검증의 시급함을 요하는 사용자들에게는 비효율적인 결과를 초래할 수 있을 것이다. 이러한 문제는 신뢰 CA가 경로 설정을 위해 필요한 DSVIL을 수집함과 동시에, 타겟 인증서에 대한 DSVIL을 발행한 DLCA를 찾아 DSVILRequest를 전송함으로서 해결될 수 있다. 따라서, 신뢰 CA가 설정 및 검증 작업을 수행하는 동안에 각 CA들은 관련된 RM을 생성 및 검증하게 되고, 신뢰 CA의 검증 작업 수행 중, 또는 작업 직후에 DPVResponse가 도착하게 될 것이다.

결과적으로, 신뢰 CA는 중간의 idle한 시간 없이 곧바로 DSVIL을 검증함으로서 사용자에게 보다 빠르게 요청한 타겟에 대한 유효성을 제공할 수 있게 된다. 그러나, 이 방법은 각 CA당 오직 하나의 상위 CA만이 존재하는 계층적 PKI 모델에서만 제한적으로 사용될 수 있다는 한계성을 가진다.

(3) RelayMessage

DSVILRequest를 수신한 DLCA는 DSVILRequest가 잘 형성되었는지를 확인하고, 이를 신뢰 CA의 공개키로 복호화 하여 DSVILRequest에 대한 서명을 검증하고, DCAs와 *Path* 항목에 자신의 이름과 자신이 서명 검증한 인증서가 있는지를 확인한다. 확인한 후 자신이 발행한 DSVIL을 DSVIL1이라 했을 때, DSVIL1의 내용에 해쉬 계산을 하여 RelayMessage1을 생성하고 RM1을 포함한 RelayMessage를 생성하여 RM을 자신의 상위 CA에게 전송한다. DLCA의 RM의 구성은 다음과 같다.

RelayMessage1=(CA-Id|H(DSVIL1(Cont))|DSVIL1(Sig))
RelayMessage=(DSVILRequest|Service|RelayMessage1|*Count*|Sign)

- ●CA-Id : 자신의 Id
- ●H(DSVIL1(Cont)) : DSVIL1 내용에 해쉬한 값
- ●DSVIL1(Sig) : DSVIL1의 서명값
- ●Service : RM을 나타내는 서비스 형태
- ●Count : DSVILRequest에서 1 감소
- ●Sign : DLCA의 비밀키로 암호화된 서명

이 때, RM에서 DSVILRequest는 원래의 DSVILRequest의 복사값이어야 하고 *Count*는 1을 감소해야 한다. DLCA는 RM을 생성한 후 이를 경로상의 상위 CA에게 전송한다. 상위 CA는 하위 CA의 공개키를 이용하여 RM의 서명을 검증하고, RelayMessage1에서 하위 CA가 생성한 DSVIL1의 서명을 다음의 과정으로 검증함으로서 DSVIL1의 정당성을 확인한다.

DLCAp(DSVIL1(Sig)) = h(DSVIL1(Cont))
RM(H(DSVIL1(Cont))) = H(DSVIL1(Cont))
비교 : h(DSVIL1(Cont)) = H(DSVIL1(Cont))

- ●DLCAp : DLCA의 공개키
- ●RM(H(DSVIL1(Cont))) : RM을 통해 전송된 H(DSVIL1(Cont))

즉, RM1의 DSVIL(Sig)에 DLCA의 공개키를 적용하여 DSVIL1의 내용에 해쉬한 값을 추출하고, 이를 RM1의 해쉬값과 비교함으로서 DSVIL1의 서명을 검증하게 된다.

상위 CA는 위의 검증 결과를 포함하고 Count를 하나 줄인 RM2를 생성하여 이를 RM에 덧붙이고, DSVIL1의 서명 검증 결과를 자신의 디렉토리에 저장하여 후에 DLCA가 자신이 발행한 DSVIL1의 내용에 대한 부인을 할 경우에 이에 대한 증거로서 제시함으로서 부인 봉쇄 기능을 제공하게 된다. RM은 최종 DHCA에게까지 순차적으로 전송되고, RM을 전송받은 모든 CA들은 위의 RM 생성 및 하위 CA가 보내온 DSVIL의 검증 과정을 수행하게 된다. 따라서 n이 경로상의 DSVIL의 수라 할 때, RM1을 제외한 RMn은 다음의 형식을 가진다.

RelayMessagen = (CA-Id|H(DSVILn(Cont))|DSVILn(Sig)|DSVILn-1(Status))
RelayMessage = (DSVILRequest|service|RelayMessage1|RelayMessage2|...|RelayMessagen|Count|Sign)

여기서 DSVILn-1(Status)는 하위 CA로부터 전송된 DSVILn-1의 서명 검증의 결과이다. 최종적으로 RM을 수신한 DHCA는 관련된 수행을 한 후 다음과 같이DSVILResponse를 생성하여 자신의 신뢰 CA에게 전송한다.

DSVILResponse=(h(DSVILRequest)|Req-Id|Nonce|Service|RelayMessage1|..|RelayMessagen|Count|Sign)

DHCA는 DSVILRequest에 해쉬값을 계산하고, DSVILRequest로부터Req-Id, Nonce를 복사한다. 또한, DSVILResponse의 서비스 형태를 표시하고 Count는 1을 감소함으로서, DSVILResponse를 생성한다. 따라서 모든 전송 과정이 정확히 이루어졌다면DSVILResponse의 Count값은 0이 되어야 한다. DSVILResponse를 수신한 신뢰 CA는 DSVILResponse의 서명을 검증하고 다음의 과정을 통해 DSVILRequest와 경로상의 DSVIL들의 무결성 및 합법성을 검사한다. 즉, DSVILResponse를 통해 전송된 DSVILRequest의 해쉬값을 원래의 DSVILRequest에 해쉬 계산한 값과 비교하고, 서비스 형태를 검사한다. 또한 DSVILResponse의 Req-Id와 Nonce가 원래의 DSVILRequest의 값과 같은지를 확인하여 DSVILRequest의 무결성을 검사한다. 또한 RM1-RMn까지의 각 DSVIL들의 해쉬값과 서명 값들을 공개된 값들과 비교함으로서 DSVIL내용의 무결성과 서명의 합법성을 검증한다.

H(OriginalDSVILRequest)=h(DSVILRequest)
DSVILRequest(Req-Id)=DSVILResponse(Req-Id)
DSVILRequest(Nonce)=DSVILResponse(Nonce)
RM1(h(DSVIL1(Cont)))=PubDSVIL1(h(DSVIL1(Cont)))
RM1(DSVIL1(Sig))=PubDSVIL1(Sig)
.....
.....
RMn(h(DSVILn(Cont)))=PubDSVILn(h(DSVIL1(Cont)))
RMn(DSVILn(Sig))=PubDSVILn(Sig)

위의 검증이 모두 성공하면 신뢰 CA는 경로상의 모든 DSVIL의 내용 및 검증 결과를 신뢰하여, 결과적으로 DSVIL을 발행한 모든 CA들을 신뢰하게 된다.

(4) 서명 위임의 신뢰성

제안한 기법은 다음 조건의 만족을 통해, DSVIL 및 DSVIL을 발행한 CA의 신뢰성을 보장한다.

① DSVIL의 검증된 인증서에 대한 해쉬 값들은 원래 인증서에 해쉬 계산한 값들과 같아야 한다.

② DSVIL의 검증된 인증서에 대한 서명 값들은 원래 인증서의 서명 값들과 같아야 한다.

③DSVResponse의 DSVIL의 해쉬 값과 서명 값들은 공개된 DSVIL의 해당 값들과 같아야 한다.

④하위 CA로부터 RM을 전송 받은 상위 CA는 하위 CA가 전송한 DSVILn-1에 대한 서명을 검증하고 이를 저장한다.

신뢰 CA는 ①과 ②를 통해 서명 검증한 인증서 내용의 무결성 및 서명의 합법성을 확신하게 되고, ③을 통해 DSVIL의 무결성 및 합법성을 확인하게 된다. ④를 통해 DSVIL을 수행한 사용자의 악의적인 행위를 방지하고, 그러한 수행 결과에 대한 부인을 하는 경우 증거 자료를 제출함으로서 부인 봉쇄 기능을 제공할 수 있다. 위의 4가지 조건을 모두 만족하게 되면 신뢰 CA는 인증서를 발행한 발행자와 인증서의 서명을 검증한 경로상의 CA들을 신뢰하고 그들이 수행한 작업 결과를 신뢰할 수 있게 된다

(5) 제안한 검증 알고리즘

제안한 방법은 기존의 검증 알고리즘의 서명 검증을 위한 암호화 계산 모듈을 제안한 해쉬 계산 모듈로 대체함으로서 구현될 수 있고, 검증의 마지막 단계에서 경로상의 DSVIL들의 해쉬를 계산하여 이를 전송된 DSVILResponse의 해당되는 내용들과 비교하는 DSVILResponse의 검증 모듈을 추가해야 한다.

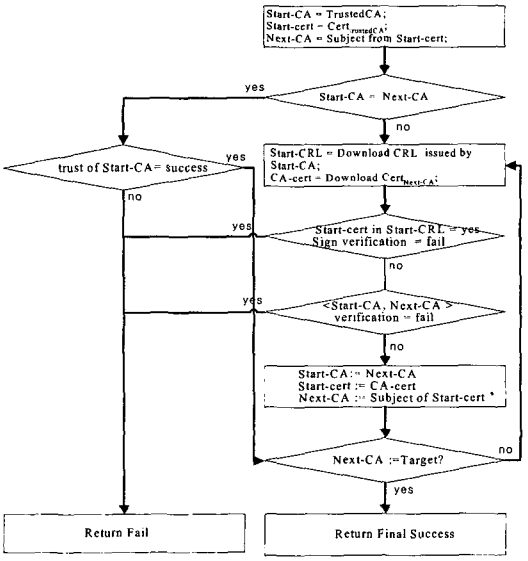


그림 7. rfc 2459의 인증 경로 검증 과정

그림 7과 그림 8은 각각 rfc2459에서 기술한 인증 경로 검증 알고리즘과 제안한 알고리즘의 수행 과정을 순서도로 나타낸 것이다.

rfc 2459검증 알고리즘이 각 CA로부터 필요한 인증서와 CRL을 다운 로드 받은 후 인증서의 서명을 직접 검증하는 대신에 제안한 알고리즘은 필요한 인증서와 CRL외에도, 그 CA가 발행한 DSVIL도 다운 로드 받게 된다.

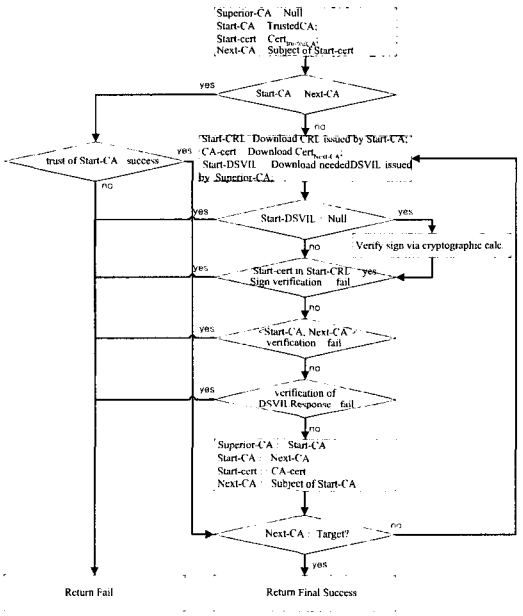


그림 8. 제안한 인증 경로 검증 과정

기존의 알고리즘이 각 인증서들에 대한 암호화 작업을 통해 서명을 검증하는 반면에, 제안한 알고리즘에서는 다운 로드한 DSVIL의 해쉬 계산을 통해 서명 검증을 수행한다. 단, 필요로 되는 DSVIL이 존재하지 않을 경우에는 암호화 계산을 통해 직접 서명 검증 작업을 한다.

제안한 알고리즘은 기존 알고리즘에 하나의 모듈이 더 추가되는데, 이것은 인증서에 대한 기존의 검증을 수행한 후, DSVILResponse에 대한 검증을 부가적으로 수행하는 것이다. 수집된 DSVIL의 내용과 DSVILResponse로부터 전송된 DSVIL의 내용을 비교함으로서 DSVIL에 대한 검증을 하고, 경로상의 모든 인증서에 대해서 서명 검증을 포함한 모든 검증 과정이 성공적으로 종료되면 최종적으로 검증 결과는 success 가 된다. 만약 응답 수신의 제한 시간 내에 응답이 입력되지 않는다면 검증의 최종 결과는 unknown 이 되고, 검증자는 암호학적 계산을 통해 서명 검증을 수행해야 한다.

기존 알고리즘의 검증 시간은 아래의 시간들로 표시될 수 있다. 여기서 인증 경로의 길이는 n이고 신뢰 CA 자신이 발행한 인증서에 대해서는 서명 검증을 수행하지 않는다고 가정한다.

. CRL들과 인증서들을 다운로드하는 시간 : T_{down}
. CRL들을 검색하는 시간 : T_{search}
. 인증서 당 서명 검증 시간 : $C_{sign} = C_{hash} + C_{crypt}$
. 경로 상의 모든 인증서 서명 검증 시간 :
$T_{sign} = C1_{sign} + C2_{sign}+..+Cn_{sign}$
. 서명을 제외한 인증서 검증 시간 : T_{verify}

반면 제안한 알고리즘의 검증 시간은 다음과 같다.

. CRL,인증서,DSVIL의 다운로드 시간 : $T_{down+\alpha}$
. CRL, DSVIL의 검색 시간 : $T_{search+\beta}$
. 서명을 제외한 인증서 검증 시간 : T_{verify}
. 경로상의 모든 인증서 서명 검증 시간 :
$T_{sign} = C1_{hash}+C2_{hash}+\cdot+C_{n\text{-}1sign} = (n\text{-}1)C_{hash} + C_{crypt}$
. DSVILResponse를 검증하는 시간 :
$T_{dsvil} = DSVIL1_{hash}+DSVIL2_{hash}+..DSVILn\text{-}2_{hash}$
$+DSVILRequest_{hash} = (n\text{-}2)DSVIL_{hash}+ DSVILRequest_{hash}$

여기서 기존 알고리즘에 비해 제안한 알고리즘은 다운로드와 검색 시에 DSVIL의 다운로드와 검색과 관련한 $\alpha+\beta$ 의 시간이 더 걸리지만, 이 시간은 성능 평가 시에 거의 취급하지 않는 아주 적은 시간이기 때문에 유념하지 않아도 될 것이다. 두 가지 기법의 가장 두드러진 차이는 서명 검증 시간과 DSVILResponse의 검증시간의 유무라 할 수 있다. 2가지 기법의 전체 서명 검증 시간을 비교하면 다음과 같다. 여기서는, 인증서, DSVIL, DSVILRequest의 크기를 같은 것으로 가정하고, 기존 알고리즘과 제안한 알고리즘의 서명 검증 시간을 각각 EVT_{sign}과 NVT_{sign}이라고 했을 때 이들은 다음과 같이 계산될 수 있다.

. $EVT_{sign} = C1_{sign}+..+Cn_{sign} = n(C_{hash} + C_{crypt})$
. $NVT_{sign} = (C1_{hash}+C2_{hash}+\cdot Cn\text{-}2_{hash}+Cn\text{-}1_{sign})+T_{dsvil} =$
$(C1_{hash}+C2_{hash}+\cdot Cn\text{-}2_{hash}+Cn\text{-}1_{sign})+(C1_{hash}+C2_{hash}+...$
$+Cn\text{-}2_{hash}+DSVILRequest_{hash})=(n\text{-}1)C_{hash}+C_{crypt}+$
$(n\text{-}2)DSVIL_{hash}+DSVILRequest_{hash}=C_{crypt}+(2n\text{-}2)C_{hash}$

제안한 알고리즘에서는 기존 알고리즘에 비해 해쉬 계산 량이 증가하지만 이는 검증 시간에 별다른 영향을 끼치기 않는다. 성능 평가의 중요한 요소

는 많은 시간이 소모되는 암호화 계산 작업 횟수라 할 수 있다. 인증 경로의 길이가 n이라고 했을 때, 제안한 방식은 자체 서명 인증서와 신뢰 CA가 발행한 인증서를 제외한 경로 상의 모든 인증서에 대한 DSVIL이 존재하는 최상의 경우에, 오직 1번의 암호화 계산만을 수행하면 된다. 따라서 n번의 암호화 작업을 해야하는 기존 방식에 비해 훨씬 빠른 검증 시간과 적은 부담을 제공한다.

3. CVIL(Certificate Validation Information Lists)을 이용한 인증 경로 처리 과정의 생략

(1) 배경

사용자들이 특정 인증서에 대한 경로 설정 및 검증 작업을 요청할 때, 요청된 인증서가 같은 정책 하에서 같은 인증 경로로 이전에 검증된 적이 있는 인증서일 경우에는 신뢰 CA는 똑같은 경로 처리 작업을 반복해서 수행해야 한다. 이러한 반복적인 처리 작업의 수행은 작업의 비효율성을 야기 할 수 있을 뿐만 아니라 연산적으로도 부담이 될 수 있다.

그림 9는 사용자 A와 C가 타겟 B에 대한 검증을 신뢰 CA에게 요청했을 때의 DSVIL전송 경로를 나타낸 것이다. 그림 9에서 보이는 것처럼, 특정 타겟에 대한 DSVIL의 요청이 있을 때마다, 신뢰 CA를 포함한 경로 상의 CA들은 같은 검증 작업을 반복해서 수행해야 한다. 잦은 DSVIL에 대한 요청은 네트워크 트래픽을 더욱 복잡하게 함으로서 그에 따른 딜레이가 생길 수 있게 된다.

이러한 문제를 해결하기 위해 신뢰 CA로 하여금 자신의 디렉토리에 자신이 경로 처리한 인증서에 대한 인증서 검증 정보인 CVIL을 저장함으로서, 전에 처리된 적이 있는 인증서에 대한 요청이 있을 경우에, 요청된 인증서에 대한 CVIL을 참조함으로서 경로 처리 작업을 생략할 수 있도록 하였다.

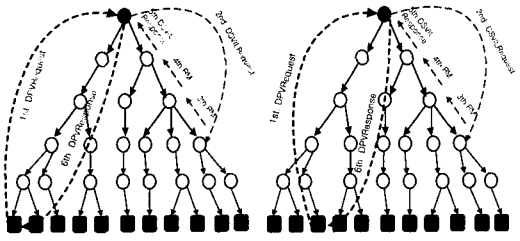


그림 9. 사용자 A와 C가 요청한 B에 대한 DSVIL 전송 경로

(2) CVIL

신뢰 CA는 사용자들로부터 요청된 인증서에 대한 경로 처리 작업을 수행한 후 자신의 디렉토리에 CVIL을 저장한다. CVIL은 처리된 인증서의 id, 검증 시간, 경로 설정 상태, 경로 검증 상태와 함께 검증 데이터를 자신의 비밀키로 서명한 정보로서, 설정된 인증 경로나 인증 경로의 유효성을 확인하기 위해 사용된다. 여기서, 검증 데이터는 인증 경로, 인증서 취소 상태 정보, DSVIL, 타임 스템프 토큰들로 구성된다. CVIL의 항목들을 ASN.1으로 표기하면 그림 10과 같다.

```
CVILInfo ::= SEQUENCE {
    storedInfo                  STOREDInfo,
    signature                   [0] EXPLICIT Signature }
STOREDInfo ::= SEQUENCE {
    certprocessed               CertId,
    validationTime              GeneralizedTime,
    discovery status            DiscoveryStatus,
    validation status           ValidationStatus,
    validationdata              ValidationData}
DiscoveryStatus ::= CHOICE{
    RequiredInfoPresent         [0] IMPLICIT NULL,
    PartialRequiredInfoPresent  [1] IMPLICIT NULL,
    NonrequiredInfoPresent      [2] IMPLICIT NULL}
ValidationStatus ::= CHOICE{
    valid                       [0] IMPLICIT NULL,
    invalid                     [1] IMPLICIT NULL,
    unknown                     [2] IMPLICIT NULL}
ValidationData :: = SEQUENCE {
    certificateValues           [0] CertificateValues,
    revocationValues            [1] RevocationValues,
    DSVILValues                 [2] DSValidationValues
    certPathRefs                [3] CertPathRefs}
Signature ::= SEQUENCE {
    signatureAlgorithm          AlgorithmIdentifier,
    signatureValue              BIT STRING}
```

그림 10. CVILs의 ASN.1 표기

각 사용자들은 자신이 원하는 설정 정책과 검증 정책을 신뢰 CA에게 지시할 수 있다. 어떠한 정책을 사용하느냐에 따라 출력되는 결과는 다를 수 있을 것이다. 이러한 모든 정책들을 통한 모든 결과를 신뢰 CA의 디렉토리에 저장하는 것은 비효율적일 뿐만 아니라 거의 불가능할 것이다. 따라서, 제안한 기법에서는 오직 신뢰 CA의 디폴트 정책을 통해 수행된 CVIL만을 저장하고 갱신한다. 이것은 제안한 기법이 효율적으로 수행되기 위해서는 사용자들로 하여금 신뢰 CA의 디폴트 정책을 사용하도록 권장되어야 한다는 것을 의미한다. 그러나, 이것은 강제적인 사항이 아니기 때문에, 사용자들은 필요하다면 언제든지 자신이 원하는 정책을 신뢰 CA에게 지시하여 원하는 형태의 결과를 얻을 수 있다.

따라서, 하나의 신뢰 CA 영역에서 전체 m개의

인증서에 대한 요청이 발생했을 경우, 신뢰 CA는 자신의 디렉토리에 m개의 CVIL들만 저장하게 된다. m이 크지 않다면, 신뢰 CA가 이들을 저장하는 데 별다른 무리가 없지만, m이 커지게 되면 CVIL들을 신뢰 CA의 디렉토리에만 저장하는 것이 부담이 될 것이다. 따라서, 이러한 정보를 CRL 분배 방식과 같이 분산해서 저장한다거나, OCSP 서버를 따로 둘 경우에는 이 정보만 OCSP 서버에 저장하여, 해당되는 정보를 필요한 경우에 요청/응답 형식으로 얻도록 하는 방안을 이용하는 것도 효율적일 것이다.

(3) 인증 경로 처리 메카니즘

제안한 기법의 요청 및 응답의 전반적인 형태와 과정은 CVP를 기본으로 한다. 따라서, 사용자는 신뢰 CA에게 자신이 직접 사용할 설정 및 검증 정책을 지시할 수 있고, 신뢰 CA는 이 정책에 따라 인증서 처리를 수행하면 된다. 만약, 요청에 어떠한 정책도 포함되어 있지 않을 경우에는 신뢰 CA 자신의 디폴트 정책을 사용하여 처리하고, 이 정책을 응답에 포함하여 전송해야 한다. 그림 11은 CVIL을 이용한 처리 과정을 순서도로 나타낸 것이다.

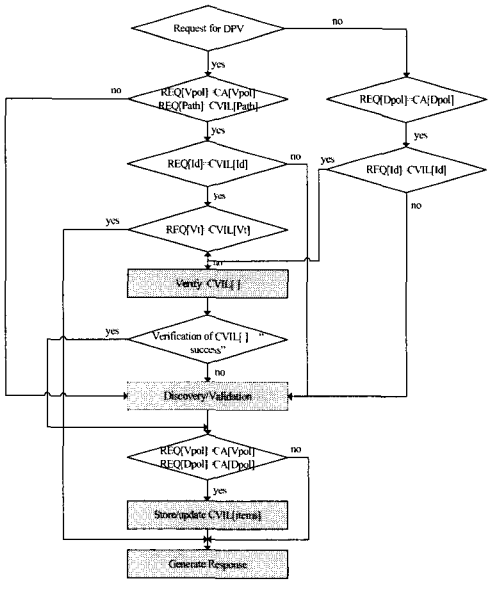


그림 11. CVIL을 이용한 인증 경로 처리 과정

신뢰 CA는 전송된 요청이 DPD에 대한 것인지, DPV에 대한 것인지를 판단한다. DPD에 대한 요청일 경우에, 설정 정책이 신뢰 CA의 디폴트 정책인지를 확인하고, 디폴트 정책인 경우 CVIL의 검색을 통해 이미 설정된 경로가 존재하는지를 검사한다. 있다면 CVIL의 유효성 상태를 검사하여 검사가 실패하거나, 기존의 설정된 경로가 없다면 경로 설정 작업을 수행해야 한다. 요청된 정책이 디폴트 정책이 아닌 다른 정책일 경우에도 마찬가지로 경로 설정 작업을 수행해야 한다. 이 때 CVIL의 유효성 검사는 다음과 같이 수행된다.

```
CVIL[CRLs] = CollectedCRLs
CVIL[Certs(cont)] = CollectedCerts(cont)
CVIL[DSVILs(cont)] = CollectedDSVILs(cont)
```

즉, CVIL의 인증 경로 항목에 있는 CRL과 DSVIL들이 원래의 상태 그대로 존재하는지, 인증서들의 내용이 변하지 않았는지를 확인함으로서 CVIL의 유효성 상태를 검사한다. 모든 내용들이 원래의 상태 그대로 존재한다면 이는 저장된 CVIL이 유효하다는 것을 의미하기 때문에 설정 작업을 수행할 필요 없이 요청에 따라 CVIL에 포함된 정보들을 응답에 포함하여 사용자에게 전송한다. 모든 설정 작업이 완료된 후에는 디폴트 정책을 통해 나온 결과만을 CVIL로 저장하거나 갱신해야 한다.

DPV에 대한 요청일 경우에도 검증 시간을 추가로 더 확인하는 것을 제외하고는 같은 과정으로 검증 작업이 수행된다. 즉, 검증이 요청된 시간이 저장된 CVIL에 있는 과거의 시간과 같거나, 시간이 같지 않더라도, CVIL이 유효한 것으로 판명된다면 어떠한 검증 작업도 수행할 필요 없이 전적으로 CVIL을 참조하면 된다. 그러나 CVIL이 유효한 것으로 판명되지 않는다면 신뢰 CA는 유효한 경로가 얻어질 때까지 계속해서 다른 후보 경로들을 설정하고 검증 해야한다. 경로 설정 과정과 마찬가지로, 디폴트 정책에 의한 검증이 완료된 후, CVIL의 값은 새로운 값으로 갱신되어야 한다.

결과적으로, CVIL의 참조는 신뢰 CA 상의 페취 및 부담을 줄이는 결과를 가져오기 때문에, 제안한 기법이 효율적으로 적용되기 위해서 사용자들은 가능한 한 신뢰 CA의 디폴트 정책들을 사용하여야 할 것이다.

Ⅳ. 비교 및 분석

1. 신뢰 CA의 연산

3장에서 기술한 각 알고리즘의 서명 검증 시간을 토대로 기존 알고리즘의 서명 검증 속도에 대한

제안한 알고리즘의 서명 검증 속도의 배율을 계산할 수 있다. 속도 배율은 기존 방법에 대한 제안한 방법의 계산 속도의 배수를 나타낸 것으로, 그 배수만큼 속도가 빠르다는 것을 나타낸다. 여기서, t_{crypt}를 RSA나 DSA와 같은 공개키 암호화 시스템에서의 암호화 계산을 millisecond(ms)로 나타낸 시간이고, t_{hash}를 해쉬 계산하는데 필요한 총 시간이라고 가정한다. 이 때, t_{hash}는 초기화를 위한 고정된 설정 시간 t_h와 해쉬 계산 시간으로 구성되는데, 해쉬 계산 시간은 해쉬 계산하는 객체의 크기에 따라 달라질 수 있고, 이는 식 1과 같이 계산될 수 있다. 식 1에서, Size(Cont)는 해쉬하고자 하는 객체의 크기를 나타내고, h는 ms당 처리되는 비트수를 나타내는 처리율을 나타낸다. 검증 속도를 구하기 위해, n을 인증 경로의 길이라고 하고, 해쉬 계산하는 객체들이 같은 크기를 갖는다고 가정했을 때, 기존 알고리즘과 제안한 알고리즘의 검증 속도인 EVT_{speed}와 NVT_{speed}는 각각 다음과 같이 계산될 수 있다.

$$t_{hash} = t_h + Size(cont)/\lambda_h \quad (1)$$

$$DSVIRe\,p/Req = 2_{tcrypt} + 2_{thash} = 2_{tcrypt} + 2(t_h + Size(Cont)/\lambda_h)$$

$$EVT_{speed} = n(t_{crypt} + t_{hash}) = nt_{crypt} + nt_{hash}$$

$$= nt_{crypt} + n(t_h + Size(Cont)/\lambda_h) \quad (2)$$

$$NVT_{speed} = 1\,t_{crypt} + 2n - 2t_{hash} + DSVIRe\,p/Req = 1\,t_{crypt} +$$

$$2n - 2(t_h + Size(Cont)/\lambda_h) + 2\,t_{crypt} + 2(t_h + Size(Cont)/\lambda_h)$$

$$= 3\,t_{crypt} + 2n(t_h + Size(Cont)/\lambda_h) \quad (3)$$

따라서 기존 알고리즘에 대한 제안한 알고리즘의 서명 검증 속도의 배율, Sf는 다음과 같다.

$$\therefore Sf = EVT_{speed} / NVT_{speed} =$$

$$n_{tcrypt} + n(t_h + Size(Cont)/\lambda_h \Big/ 3_{tcrypt} + 2n(t_h + Size(Cont)/\lambda_h) \quad (4)$$

표 1은 [12]에서 제공한 공개키 알고리즘과 해쉬 함수의 수행 속도를 나타낸 것이다.

표 1. 공개키 알고리즘과 해쉬 함수의 실행 속도

Algorithm	$t_{Algorithm}$(ms)	$\lambda_{Algorithm}$(bits/ms)
DSA512	33.909	-
DSA1024	113.968	-
RSA1024	4.457	-
RSA2048	17.152	-
SHA-1	0.0093	36057.0
MD5	0.009	68267.0

표 1의 수행 속도에 따라 기존 알고리즘에 대한 제안한 알고리즘의 서명 검증과 관련한 속도 배율을 계산해보면 표 2와 같다. 여기서 D와 S는 각각 DSA와 RSA알고리즘을 나타낸다.

표 2. 서명 검증 속도 배율

n	서명 검증 속도 배율					
	D512 -SHA1	D1024 -SHA1	R1024 -SHA1	R2048 -SHA1	R1024 -MD5	R2048 -MD5
2	0.666	0.666	0.662	0.665	0.664	0.666
3	0.998	0.999	0.987	0.996	0.992	0.998
4	1.330	1.332	1.309	1.326	1.318	1.329
5	1.661	1.665	1.627	1.656	1.642	1.660
6	1.992	1.997	1.941	1.984	1.963	1.990
7	2.325	2.330	2.252	2.311	2.282	2.319
8	2.652	2.662	2.569	2.638	2.599	2.648
9	2.981	2.994	2.865	2.963	2.914	2.977
10	3.310	3.326	3.166	3.288	3.227	3.304
100	31.079	32.629	21.496	29.155	24.673	30.543
500	122.218	150.390	44.304	96.967	60.320	114.247
1000	192.942	274.002	51.079	136.717	73.614	173.777

표 2에서 보이는 것처럼, n이 3이하일 경우를 제외하고는 어떠한 알고리즘을 사용하든지 간에 제안한 방법이 기존의 서명 검증 방법보다 빠른 속도를 보였다. 제안한 방법은 암호화의 횟수를 현저하게 줄였기 때문에 암호화 속도가 비교적 느린 DSA와 RSA2048에서 더 빠른 속도를 보였으며, 상대적으로 암호화 속도가 빠른 RSA1024에서는 속도 배율이 약간 둔화되는 것을 알 수 있었다. 또한 n의 크기가 커질수록 제안한 방법의 속도 배율이 훨씬 향상되는 것을 알 수 있었다.

2. DSVIL을 발행한 CA들의 연산

DSVIL의 발행의 부담은 각 CA마다 고르게 분배된다. 이는 NPKI[3][12]에서 Nested certificate 발행 시에 상위 CA로 갈수록 발행 횟수가 증가하는 것과는 비교되는 측면이라 할 수 있다. 신뢰 CA와 사용자들의 인증서를 발행한 CA들은 DSVIL을 발행하지 않기 때문에 DSVIL발행과 관련한 어떠한 부담도 없다. 그러나 그 외의 모든 CA들은 해당되는 인증서들의 서명을 검증하여 DSVIL을 발행해야 하기 때문에 그에 따른 부담을 가질 수 있다.

i-level m-ary의 균일한 PKI에서 각 CA는 m^2의 DSVIL을 발행해야 한다. 표 3은 4-level 20-ary에서의 각 CA가 발행하는 DSVIL의 수를 나타낸다. 표 3에서 보이는 것처럼 신뢰 CA및 사용자와 그의 인증서를 발행한 CA들을 제외한 모든 CA는 20^2 즉, 400개의 인증서에 대한 DSVIL을 발행을 해야 한다.

표 3. 4-level 20-ary에서의 DSVIL의 수

Level	Level당 CA의 수	Level당 DSVIL의 수
0	1	0
1	20	8,000
2	400	16,000
3	8,000	0
4	16,000	0

DSVIL의 발행 작업은 주체 인증서들이 취소되거나 갱신되는 경우를 제외하고는 주체 인증서 발행 시에 단 한번 발행되고, 각 CA의 idle time에 발행되기 때문에 그다지 큰 부담을 야기하지 않는다. 만약 검증 시에 원하는 인증서의 DSVIL이 발행되지 않은 상태라면 신뢰 CA는 암호화 계산을 통해 직접 서명 검증하기 때문에 DSVIL의 발행 시기가 인증서 검증에 어떠한 악영향도 미치지 않는다. 오히려, 작업의 제한성 때문에 idle한 시간이 많은 성능 좋은 CA들을 활용한다는 측면에서 전체적인 PKI의 효율성이 향상될 수 있다.

DSVIL의 요청 시에 수행되는 검증의 계산 량은 많지 않기 때문에 그다지 큰 부담이 되지 않지만, 잦은 요청은 CA들 사이의 전송량을 증가시킴으로서 부담을 가중시킬 수 있다. 이 문제는 신뢰 CA에 저장된 CVIL을 참조함으로서, 검증에 참여하는 각 CA들은 연산적인 부담 없이 자신에게 할당된 작업들을 효율적으로 수행할 수 있을 것이다.

3. 검증 방식의 비교

표 4는 제안한 방식을 사용자 검증 방식 및 SCVP 검증 방식과 여러 가지 측면에서 비교한 것이다.

표 4. 다른 경로 처리 방식들과의 비교

	사용자 검증 방식	SCVP 검증 방식	제안한 검증 방식
통신 모델	오프라인	온라인	온라인
신뢰점	신뢰 CA	SCVP 서버	신뢰 CA
검증자	사용자	SCVP 서버	신뢰 CA
서명 검증자	사용자	SCVP 서버	CAs/사용자
사용자/서버간 상호검증	필요	필요	불필요
새로운 서버 구축	불필요	필요	불필
수행 서비스	X	DPD/DPV	DPD/DPV
사용자 부담	많음	적음	적음
CA 활용도	낮은	낮음	높음

표 4에서 보이는 것과 같이 사용자 검증 방식이 오프라인으로 운영되는 반면 제안한 방식과 SCVP 방식은 온라인으로서, 각각 SCVP 서버와 사용자의 신뢰 CA에게 의존하게 된다. 따라서, SCVP 방식이 사용자와 서버간의 신뢰성을 유지하기 위한 키 분배 및 상대방 인증서의 검증 작업이 부수적으로 필요한 반면, 제안한 방식은 사용자와 신뢰 CA 상호간의 부수적인 검증 작업이 필요하지 않다.

SCVP 방식과 제안한 방식에서는 인증 경로 처리와 관련한 모든 작업을 SCVP 서버와 신뢰 CA에게 모두 위임하기 때문에 사용자측의 검증에 대한 부담이 전혀 없다. SCVP 서버와는 달리 신뢰 CA는 기본적인 운영 작업 외에 검증 작업을 추가로 수행해야하는 부담이 있기 때문에, 서명 검증을 다른 CA들에게 위임함으로서 부담을 감소할 수 있다.

따라서, CA를 검증 작업에 참여시킴으로서 다른 방식들보다 CA의 활용도가 매우 높게 된다. 또한 CVIL을 이용하여 인증 경로 처리 작업의 생략이 가능함으로서 신뢰 CA의 연산적인 부담 및, 서명의 위임으로 인해 야기될 수 있는 네트워크 트래픽의 복잡성 또한 감소시킬 수 있다.

V. 결 론

본 논문에서는 사용자의 신뢰 CA를 DPD/DPV 서버로 사용함으로서 간단한 구현을 통해 높은 신뢰성을 유지시키는 동시에, 서명 검증을 다른 CA들에게 위임하고, 반복되는 인증 경로처리 작업을 생략할 수 있도록 함으로서 신뢰 CA상의 연산과 속도의 효율성을 증가시키는 새로운 인증 경로 처리 기법을 제안하였다. 제안한 방법은 기존의 서버를 사용한다는 측면에서, 새로운 서버의 구축 및 유지 비용이 들지 않고, 사용자와 서버사이의 서로의 신뢰를 확인하기 위한 부수적인 키 분배 및 상대방의 인증서 검증 과정이 필요하지 않음으로서 구현이 단순화될 수 있다. 또한 신뢰 CA 자체의 높은 신뢰성과 요청 및 응답의 다양한 인증 및 무결성 검사를 통해 통신의 안전성이 보장된다.

또한, 제안한 기법은 서명 검증에 CA들을 참여시킴으로서 성능 좋은 CA들을 적극적으로 활용하도록 하였으며, 이것은 신뢰 CA의 연산적인 부담을 덜어주는 결과를 가져왔다. 속도 배율로 측정한 결과, 제안한 기법의 서명 검증 속도는 인증 경로의 길이가 3 이상일 때, 항상 다른 기법들보다 더 빠르다는 것을 알 수 있었고, 이는 인증서 길이가 커지고, 암호화 속도가 느린 알고리즘을 사용할수록 더 빨라진다는 결론을 얻을 수 있었다. 비록 대부분의 CA들이 인증서의 서명 검증 및 DSVIL의 발행,

DSVILRequest/Response와 RelayMessge의 생성 및 검증과 관련한 다수의 암호화 작업을 수행하여야 할지라도, CA 자체가 고속의 높은 처리 능력을 가지며 인증서 서명 검증과 DSVIL발행 작업은 새로운 인증서의 발행 및 기존 인증서의 갱신이나 취소가 되지 않는 한은 한번만 이루어지는 작업이기 때문에, 이들은 서명 검증과 관련한 모든 작업들을 무리 없이 수행할 수 있다.

서명의 위임이 자주 수행되면 CA들 사이의 전송량이 많아져 네트워크 트래픽이 복잡해지고 그에 따른 딜레이가 발생할 수도 있다. 이러한 문제는 처리한 인증서에 대한 CVIL을 신뢰 CA의 디렉토리에 저장하여, 같은 인증서에 대한 인증 경로 처리 요청이 있을 경우에 이를 참조함으로서 해결될 수 있었다.

사용자 수가 증가할 경우, 신뢰 CA에 저장된 CVILs의 크기 또한 비례적으로 증가하게 됨으로서, CVIL들을 CRL 분배 방식과 같이 분산해서 저장하거나, 다른 서버에 저장시키는 대응 방안이 앞으로 연구되어져야 할 것이다.

참 고 문 헌

[1] 이만영, 김지홍, 류재철, 송유진, 염흥열, 이임영, 전자 상거래 보안 기술, 생능 출판사, p155-206, 1999

[2] 심희원, DNS를 이용한 상호 연동 및 인증서 검증 방안

[3] Albert Levi, M.Ufuk Caglayan, An Efficient Dynamic and Trust Preserving Public Key Infrastructure , Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000), 2000.

[4] R. Housley, W. Polk, D. Solo, Internet X.509 Public Key Infrastructure Certificate and CRL Profile , IETF RFC 2459, January 1999.

[5] M.Branchaud, J.Linn, "Extended Validation Models in PKI : Alternatives and Implications", 1st Annual PKI Research Workshop--Proceedings, 2001.

[6] ETRI ZONE/R&D News, "세계 최초의 '통합형 인증서 검증시스템 (CVS·Certificate Validation System) 개발",

[7] D. Pinkas, R. Housley, "Delegated Path Validation and Delegated Path Discovery Protocol Requirements" IETF RFC 3379, September 2002

[8]D. Pinkas, Certificate Validation Protocol , IETF draft-ietf-pkix-cvp-01.txt, October 2002.

[9] M. Myers, A. Malpani, D.Pinkas, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol, version 2" IETF draft-ietf-pkix-ocspv2-text-01.txt, December 2002.

[10]Ambarish Malpani, Paul Hoffman, Russ Housley, and Trevor Freeman, Simple Certificate Validation Protocol(SCVP) , IETF draft-ietf-pkix-scvp-06.txt, July 2001.

[11] R.Housley, T.Pork, Planning for PKI, John Wiley&Sons,Inc, p138-150, 2002.

[12] Albert Levi, M.Ufuk Caglayan, Analytical performance evaluation of nested certificates , Performance Evaluation , vols. 36-37, p213-232, August 1999.

최 연 희(Yeon-Hee Choi) 준회원

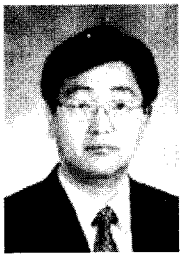
1991년 2월 : 목포대학교 전산통계학과 졸업

1993년 2월 : 숭실대학교 전자계산학과 석사

1996년9월~현재 : 숭실대학교 컴퓨터학과 박사과정

<주관심분야> 암호학, 정보 보안

전 문 석(Moon-Seog Jun) 정회원



1980년 2월 : 숭실대학교 전자계산학과 졸업

1986년 2월 : University of Maryland 전산과 석사

1989년 2월 : University of Maryland 전산과 박사

1991년3월~현재 : 숭실대학교 정보과학대학 정교수

<주관심분야> 네트워크 보안, 컴퓨터 알고리즘, 병렬처리, VLSI 설계, 암호학