CA를 인증 경로 처리 작업에 참여시키는 새로운 인증서 검증 방안

최 연 희† · 박 미 옥† · 전 문 석††

요 약

PKI를 채용한 대부분의 어플리케이션들은 사용자로 하여금 인증서 검증 작업을 수행하도록 한다. 인증서 검증 작업의 사용자 측 수행은 사용자 부담을 증가시키고 검증 속도를 떨어뜨려 전체적인 사용자 시스템의 효율성을 저하시킨다. 본 논문에서는 인증기관(Certificate Authority : CA)을 검증 작업에 참여시킴으로서 사용자 측 부담을 감소시킬 수 있는 새로운 형태의 인증서 검증 방안을 제안하였다. 따라서 제안한 방안은 새로운 검증 서버의 구현 없이 사용자 측의 검증 작업에 대한 부담을 감소시킬 수 있을 뿐만 아니라 검증 작업의 축소로 인해 검증 수행 시간을 향상시킬 수 있다.

A New Certificate Validation Method Allowing CAs to Participate the Certificate Path Validation Processing

Yeon Hee Choi† · Mi Og Park† · Moon Seog Jun††

ABSTRACT

Most applications using the PKI allows a user to execute the certificate validation processing. The efficiency of user system can be declined by the user-side processing resulting the overhead and low speed of the validation processing. Therefore, in this paper, we propose a new certificate validation processing method can decrease the overhead on user by allowing CAs of the hierarchical PKI to participate in the validation processing. Therefore, our proposed scheme can not only reduce the considerable overhead caused by the user-side whole processing without a new implementation of the delegated server but also improve the time spent for the processing by the reduction of the validation processing job on user.

키워드 : 공개키 기반 구조(Public Key Infrastructure), 인증서 검증(Certificate Validation), 인증 경로 처리(Certificate Path Processing), 인증기관(Certificate Authority)

1. 서 론

고도의 정보화 사회가 도래하면서 우리는 매일 인터넷을 통해 이메일을 교환하거나 물건을 사고 원격지에 있는 서버에 접속해 파일을 다운로드하거나 업로드하는 등 다양한 정보 서비스를 이용하고 있다. 그러나 많은 인터넷 사용자들이 개인 정보와 지불 정보의 누출 등 신뢰성과 안정성을 염려해 전자 상거래를 꺼려하고 있다는 것은 주지의 사실이다. 이에 대한 대비책은 정보 시스템에 정보 보호 기능을 채용하는 것이고 정보 시스템의 안전성을 보장할 수 있는 대표적인 정보 보호 기술은 PKI라 할 수 있다. PKI는 공개키 인증서를 효과적으로 저장 및 분배함으로서 다양한 형태의 암호학적 서비스 구현에 이용할 수 있는 종합적인 인증 체계 기반 기술이다. PKI 구축을 위한 기술 중 인증서의 검증 및 현재 상태를 다루는 인증서 검증 기술은 실제 전자 거래에 있어 그 거래의 유효성에 관한 것이므로 가장 신중하게 처리되어야 하며, 금융 거래에서는 특히 실시간으로 검증이 가능하여야 한다[1-3].

인증서 검증을 위해서는 특정 인증서에 대한 인증 경로를 구성해야 하고, 특정 인증서의 상태를 인증서 취소 목록(Certificate Revocation List : CRL)[4] 또는 Online Certificate Status Protocol (OCSP)[5] 서버와 같은 제3의 신뢰 기관을 통하여 확인해야 하며, 인증 경로 상의 인증서들에 대한 유효성을 검증하기 위한 다양한 검사를 수행해야 한다. 특히 인증서 상의 서명 유효성을 검증하기 위해서는 공개키 서명 알고리즘을 통한 암호학적 연산을 행해야함으로

† 준 회 원 : 숭실대학교 대학원 컴퓨터학과
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
논문접수 : 2003년 7월 14일, 심사완료 : 2003년 12월 10일

서 서명의 유효성 검증 과정은 시간적으로 매우 비효율적이며 전체 검증 작업의 시간적 효율성을 저하시키는 주된 요인이 된다. 이러한 모든 검증 과정들은 서버 기반 인증서 검증 기법이 소개되기 전에는 사용자 어플리케이션에서 처리해야 했다. 이는 인증 경로가 복잡해지거나 인증서의 많은 이용에 의해 CRL 크기가 증가할 경우 사용자 측에 상당한 부담이 될 수 있다[6]. 따라서 사용자의 부담을 줄이기 위해 검증 과정중의 일부나 혹은 전부를 온라인 인증서 검증 서버에 위임하기 위한 OCSPv2[7, 8], Simple Certificate Validation Protocol(SCVP)[9], Certificate Validation Protocol(CVP)[10]와 같은 다양한 온라인 검증 프로토콜들이 제안되어왔다. 온라인 검증 서버의 채택은 사용자 측의 인증서 검증 모듈을 단순화하고 검증으로 인한 연산적인 부담을 크게 축소시키는 장점을 제공하는 반면, 검증 서버를 위한 부수적인 시스템 도입의 필요성, 온라인 상태 유지, 서버와 사용자 사이의 키 분배 및 상호 인증서 검증을 위한 메커니즘의 필요성 등의 까다로운 문제가 수반될 수 있다[11, 12].

본 논문에서는 검증 서버를 따로 구축하지 않고, 계층적 PKI 영역의 CA들을 이용하여 사용자 측 검증 작업의 부담을 감소시킬 수 있는 새로운 인증서 검증 방식을 제안하였다. 제안한 방식은 부담의 주요인이 되는 서명의 유효성 검증 작업을 사용자와 CA가 분담하도록 함으로서 사용자 부담을 줄이고 검증 작업을 수행하는데 소요되는 속도가 향상되도록 한다.

2. RFC 2459를 통한 인증서 검증 방식

인증서 검증을 위한 가장 대표적인 알고리즘은 Housley et al.이 RFC 2459[4]를 통해 제시한 알고리즘으로서 이는 크게 경로 설정, 기본 검증, 경로 검증 등의 3가지 과정으로 수행된다[6]. <표 1>은 각 과정의 작업들을 간단히 설명한 것이다.

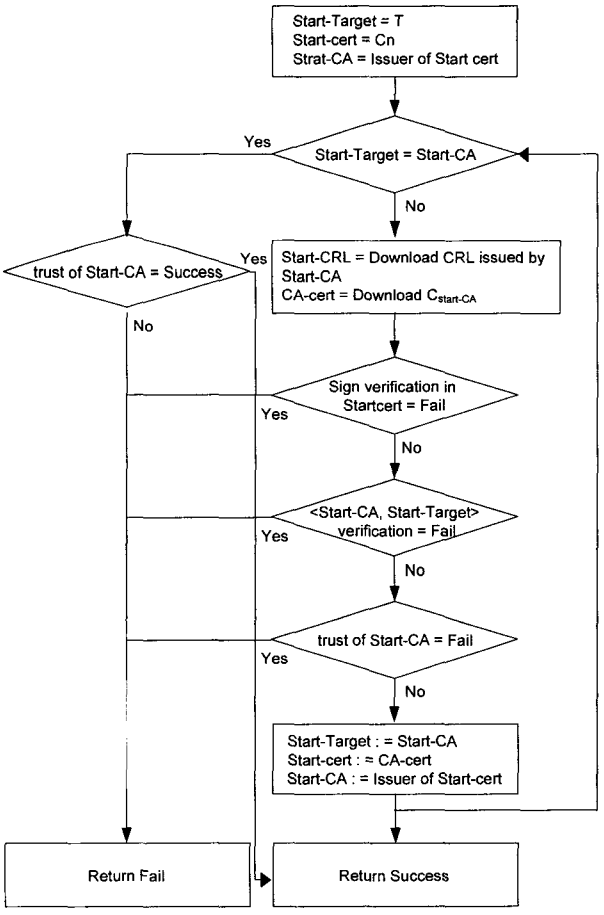
〈표 1〉 인증서 검증 과정

경로 설정	검증에 필요한 인증서 및 CRL 획득
기본 검증	서명의 유효성 검증
	유효기간 검사
	상위 인증서의 주체 이름과 해당 인증서의 발행자 이름이 일치하는가
	인증서 취소 상태 검증 (CRL등)
경로 검증	인증서의 이름 공간 검사
	정책 매핑 검사 및 수행
	인증서 정책 검사

RFC 2459의 인증서 검증 과정을 (그림 1)의 순서도로 표현하기 위해 다음의 표기를 사용할 것이다.

- <x, y> : x가 발행한 y의 인증서
- sign verification : 서명의 유효성 검증
- <x, y> verification : 서명의 유효성 검증을 제외한 <표 1>의 다양한 검사 과정을 통한 검증
- c_x : x가 발행한 인증서
- CAx : CA, x
- T : 타겟

(그림 1)에서 보이는 것처럼, 검증하는 사용자(이하 검증자)는 자체 서명 인증서를 제외한 경로 상의 모든 인증서들에 대해 필요한 인증서와 CRL을 다운로드받아 서명의 유효성 검증 및 <표 1>의 다양한 검증 작업들을 수행함으로서 검증하고자하는 상대방(이하 타겟) 인증서의 유효성을 판단한다. 각 인증서들이 검증을 위한 조건들 중 하나라도 만족하지 못하면 설정된 인증 경로는 유효하지 않은 것으로 판단되어 다른 후보 경로가 입력되어 검증 작업이 다시 수행된다.



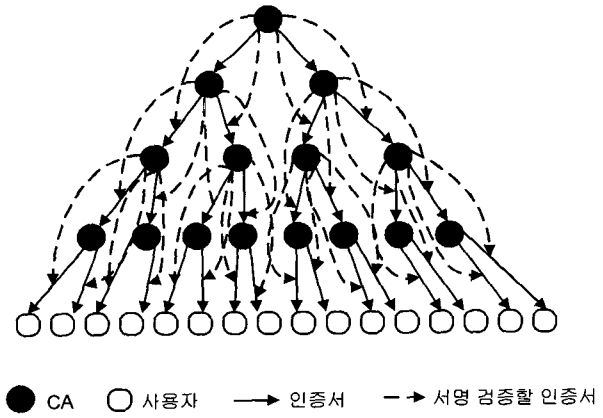
(그림 1) RFC 2459 인증서 검증 과정

검증 작업 중 인증서 서명의 유효성을 검증하는 작업은 인증 경로 길이만큼의 해쉬와 암호화 계산이 수반됨으로서 검증자 측에 많은 부담이 된다. 일반적으로 암호화 시간은 해쉬 계산 시간보다 서명 알고리즘에 따라 대략 1000배 이상의 시간이 더 소요됨으로서 서명 검증 부담의 주된 원인이 된다[13].

3. 제안한 인증서 검증 방식

3.1 CA의 서명 검증

제안한 검증 방식은 계층적 PKI에서 수행하는 것을 원칙으로 하며 같은 키를 사용하여 인증서와 CRL에 서명한다고 가정한다. 계층적 PKI에서 각 CA는 자신이 인증서를 발행해준 사용자나 하위 CA들의 공개키를 알기 때문에 이들이 발행한 인증서들의 서명을 검증할 수 있다. (그림 2)는 각 CA가 서명 검증할 수 있는 인증서들을 나타낸다. (그림 2)에서, 신뢰 CA인 루트 CA(이하 신뢰 CA)가 발행한 인증서들의 서명은 검증자가 직접 검증하고, 신뢰 CA의 자체 서명된 인증서에 대한 서명 검증은 수행하지 않는 것을 기본으로 한다. 각 CA는 서명 검증 작업을 수행한 후 작업의 수행 결과인 디지틀 서명 검증 정보(Digital Signature Validation Data : DSVD)를 발행하여 공개한다.



(그림 2) 각 CA의 서명 검증할 인증서의 위치

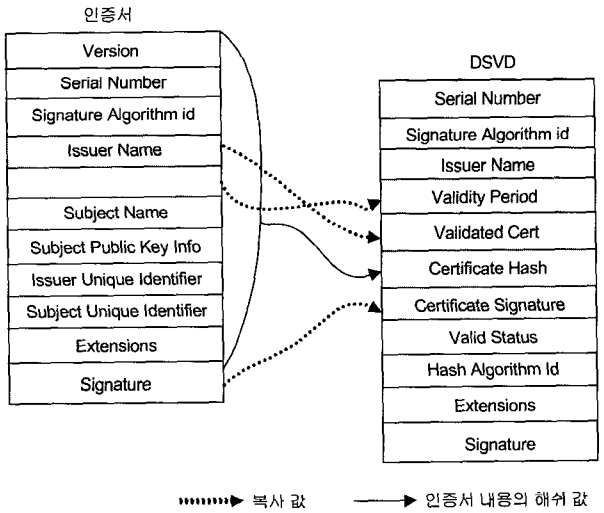
3.2 DSVD(Digital Signature Validation Data)

DSVD는 인증서의 서명 검증 결과를 CA의 비밀키로 서명한 정보로서 <표 2>의 항목들로 구성된다.

<표 2>에서, Validity Period는 검증한 인증서의 Validity Period와 같게 설정됨으로서 검증된 인증서의 변화를 따르게 된다. Certificate Hash와 Certificate Signature는 각각 검증한 인증서 내용의 해쉬값과 서명값으로서 검증자 측의 인증서 서명 검증을 위해 사용되는 항목이다. (그림 3)은 DSVD의 구조를 나타낸다. 각 CA는 하위 CA의 디렉토리 검색 및 CRL 주기를 통하여 인증서들의 상태를 파악하고 이들의 상태에 따라 DSVD 관련 작업을 수행한다. 새로운 DSVD의 생성은 각 CA의 여유로운 시간에 수행될 수 있지만, DSVD의 갱신은 잘못된 검증 결과가 나오지 않도록 인증서 갱신 및 취소 상태를 발견한 후 가능한 빨리 수행되어야 한다.

<표 2> DSVD의 구성

항목 이름	설 명
Serial Number	DSVD의 일련번호
Signature Algorithm id	인증서를 서명하는데 사용하는 알고리즘 식별자
Issuer Name	DSVD의 발행자 이름
Validity Period	DSVD의 유효기간으로서 검증한 인증서의 유효기간과 같게 설정
Validated Cert	서명 검증한 인증서의 id나 인증서 그 자체
Certificate Hash	검증한 인증서 내용의 해쉬값
Certificate Signature	검증한 인증서의 서명값
Valid Status	서명 검증 상태
Hash Algorithm id	서명 검증한 인증서의 내용을 해쉬하는데 사용한 해쉬 알고리즘 식별자
Extensions	부가적인 정보를 입력하기 위한 확장자
Signature	위의 항목에 대한 발행자의 서명값



(그림 3) DSVD 구조

3.3 DSVD를 통한 인증서 서명 검증 알고리즘

검증자는 기존의 암호화 계산을 통한 검증 작업 대신 수집된 DSVD를 통한 인증서 서명 검증 작업을 (그림 4)의 알고리즘을 통해 수행한다. (그림 4)에서 보이는 것처럼, 이 작업은 해쉬 계산만으로 검증이 가능하다. (그림 4)에서, H[c(cont)]와 c(Sig)를 각각 원래 인증서 내용에 해쉬 계산한 값과 원래 인증서의 서명값이라 할 때, 검증자는 다음의 과정을 통해 서명을 검증한다.

① 원래 인증서 내용에 해쉬 계산한 Calculated_Hash값과 DSVD에 포함된 Certificate Hash값을 비교함으로서 인증서 내용의 무결성을 검증한다.

② 원래 인증서의 서명값인 Original_Signature와 DSVD에 포함된 Certificate Signature값을 비교함으로서 인증서 서명의 합법성을 검증한다.

①, ②의 검증이 성공적으로 완료된다 하더라도 검증자는 DSVD나 DSVD를 발행한 CA들이 신뢰할만한 것인지를 확신할 수 없기 때문에 서명의 유효성과 관련한 어떠한 결정도 내릴 수 없다. 따라서 DSVD의 신뢰성을 획득하기 위한 부가적인 DSVD 서명 검증 작업이 수행되어야 할 것이다. 이 작업은 위임 서명 검증 프로토콜 (Delegated Signature Validation Protocol : DSVP)의 수행을 통해 이루어지며 이 작업 또한 해쉬 계산을 통해 수행된다.

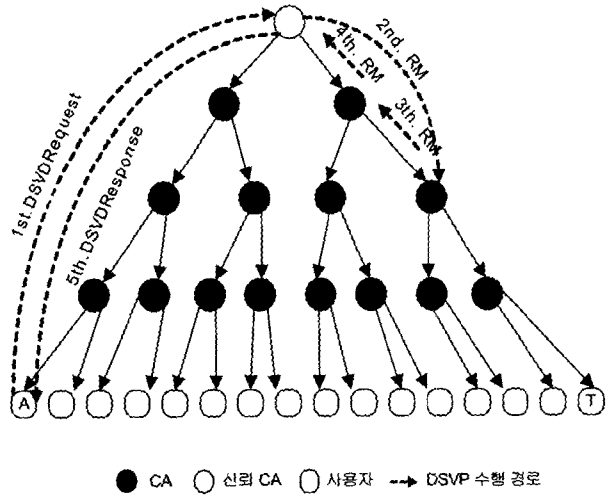
```
Published_ Hash = DSVD(Certificate Hash)
Calculated_ Hash = H[c(cont)]
Published_ Signature = DSVD(Certificate Signature)
Original_ Signature = c(Sig)
IF (Published_Hash = Calculated_Hash AND Published_ Signature
                    = Original_ Signature)THEN
    Return signature_verification_success ;
ELSE
    Return signature_verification_fail ;
```

(그림 4) DSVD를 이용한 서명 검증 알고리즘

3.4 DSVP(Delegated Signature Validation Protocol)

3.4.1 DSVP 수행 과정

제안한 DSVP는 검증자와 경로 상의 CA들 사이에 수행되는 프로토콜로서 경로 상에 존재하는 모든 DSVD들을 요청하고 응답된 DSVD들을 검증하기 위한 것이다.



(그림 5) DSVP 수행 과정

DSVP는 (그림 5)에서 보이는 것처럼 크게 3개의 과정으로 구성된다.

① DSVD 요청 단계 : 검증자는 신뢰 CA에게 경로 상의 DSVD들을 전송해달라는 DSVDRequest를 보낸다. 이 때, DSVDRequest에는 타겟 인증서에 대한 DSVD를 발행한 CA(TDCA : Target-DSVD-issued CA)의 이름이 포함된다.

② DSVD 첨부 단계 : 신뢰 CA는 TDCA에게 타겟에 대한 DSVD의 첨부를 요구하는 릴레이 메시지(RelayMessage : RM)를 생성하여 전송하면 TDCA는 요청된 DSVD를 RM에 첨부하여 계층 상 직속 상위 CA에게 전송한다. 이 과정은 RM이 신뢰 CA에게 되돌아올 때까지 순차적으로 수행된다.

③ DSVD 응답 단계 : 신뢰 CA는 경로 상의 모든 DSVD를 포함하여 DSVDResponse를 생성한다. 생성된 DSVD Response를 검증자에게 보냄으로서 DSVP는 종료된다.

인증 경로 길이가 n이라 했을 때 DSVP 경로 상의 CA들의 수는 n-2로서 최대 n-2개의 DSVD들이 수집된다. DSVP의 수행 경로 상에 존재하는 각 CA는 다음의 2가지 작업을 수행한다.

① DSVD를 RM에 첨부하는 작업 : 각 CA는 요청된 DSVD의 내용에 대한 해쉬값과 DSVD의 서명값을 RM에 첨부하여 계층 상 직속 상위 CA에게 전송한다.

② 하위 CA가 발행한 DSVD의 서명을 검증하는 작업 : CA는 하위 CA가 발행한 DSVD의 서명을 하위 CA의 공개키를 이용하여 암호학 계산을 통해 수행한다. 각 CA는 검증 결과를 디렉토리에 저장하고 RM에 첨부함으로서 DSVD의 합법성 및 DSVD에 대한 부인방지 기능을 제공한다.

3.4.2 메시지

(1) DSVDRequest

DSVDRequest는 검증자가 신뢰 CA에게 경로 상의 모든 DSVD를 전송해달라는 요청문으로서 ASN.1 문법으로 정의하면 다음과 같다. 여기서, certsQueries는 certToValidate와 usefeulDSVDs 항목으로 구성되는데 이들은 각각 검증할 인증서와 경로 상의 DSVD들의 실제값을 나타낸다. useful DSVDs는 DSVP의 수행 생략을 위해 사용되는 항목으로서 신뢰 CA로 하여금 DSVP의 수행을 통해 수집된 RM들을 자신의 디렉토리에 저장하고 저장된 DSVD를 usefulDSVDs 항목에 포함된 현재의 DSVD와 같은지를 확인하는 DSVD 유효성 검증 작업을 수행함으로서 DSVP의 수행을 생략하도록 하기 위한 것이다. 결과적으로 경로 상의 DSVD가 갱신

되거나 취소되지 않은 이상 특정 인증서에 대한 요청 당 한 번의 DSVP만 수행하면 되기 때문에 DSVP 수행에 따른 전송량이 감소됨으로서 검증자는 보다 빠른 응답을 수신할 수 있게 된다.

```
DSVDRequest ::= SEQUENCE{
   nonce            OCTET STRING,
   certsQuries      SEQUENCE OF CertsQuery,
   checks           SEQUENCE of Checks,
   TDCAName         [0] EXPLICIT  GeneralName  OPTIONAL,
   requesterName    [1] EXPLICIT  GeneralName  OPTIONAL,
   signatory        [2] ESSCertID              OPTIONAL,
   requestExtensions [3] EXPLICITExtensions    OPTIONAL,
   signature        [4] EXPLICIT Signature     OPTIONAL}
CertsQuery ::= SEQUENCE{
   certToValidate   CertOrCertRef,
   usefulDSVDs      DSVDvalues}
Checks ::= SEQUENCE{
   responseTime        GeneralizedTime,
   count            [0] OCTET STRING           OPTIONAL,
   RMpath           [1] BOOLEAN DEFAULT FALSE}
Signature ::= SEQUENCE{
   signatureAlgorithm   AlgorithmIdentifier,
   signature            BIT STRING}
```

(2) RelayMessage

RelayMessage는 각 CA에게 그가 발행한 DSVD를 첨부해 달라는 요청문으로서 신뢰 CA에 의해 생성된다. RM은 DSVP 경로 상의 모든 CA들을 거쳐 3.4.1절의 작업 결과가 첨부되어 신뢰 CA에게 되돌아온다. 신뢰 CA에게 전송된 RM을 아래 표기를 이용하여 ASN.1으로 나타내면 다음과 같다.

- 인증 경로 : n
- DSVP 경로 상의 CA의 수 : m(m = 1, 2, …, n-2)
- RMm : DSVP 경로상의 각 CA에 의해 생성된 RM
- DSVDmHash : 각 CA에 의해 첨부된 DSVD 내용에 대한 해쉬 계산값
- DSVDmSign : 각 CA에 의해 첨부된 DSVD의 Signature값
- DSVDmVResult : 경로상의 각 CA에 의해 검증된 직속 하위 CA의 DSVD 검증 결과

```
RelayMessage ::= SEQUENCE{
   nonceRM          OCTET STRING,
   RelayMessage1    SEQUENCE of RM1,
   RelayMessage2    SEQUENCE of RM2,
   Count            [0] OCTET   STRING      OPTIONAL,
   signature        [1] EXPLICIT Signature  OPTIONAL}
RM1 ::= SEQUENCE{
   CAName           [0] EXPLICIT  GeneralName  OPTIONAL,
   DSVD1Hash        BIT STRING,
   DSVD1Sign        BIT STRING,
   signAlgorithm    AlgorithmIdentifier}
RM2 ::= SEQUENCE{
   CAName           [0] EXPLICIT  GeneralName  OPTIONAL,
   DSVD2Hash        BIT STRING,
   DSVD2Sign        BIT STRING,
   DSVD2VResult VResult ;
   signAlgorithm    AlgorithmIdentifier}
                    ..........
VResult ::= CHOICE{
   valid            [0] IMPLICIT NULL,
   invalid          [1] IMPLICIT NULL}
```

(3) DSVDResponse

최종적으로 RM을 수신한 신뢰 CA는 DSVDRequest에 해쉬값을 계산하고, DSVDRequest의 항목들을 복사함으로서, 다음과 같이 DSVDResponse를 생성한다.

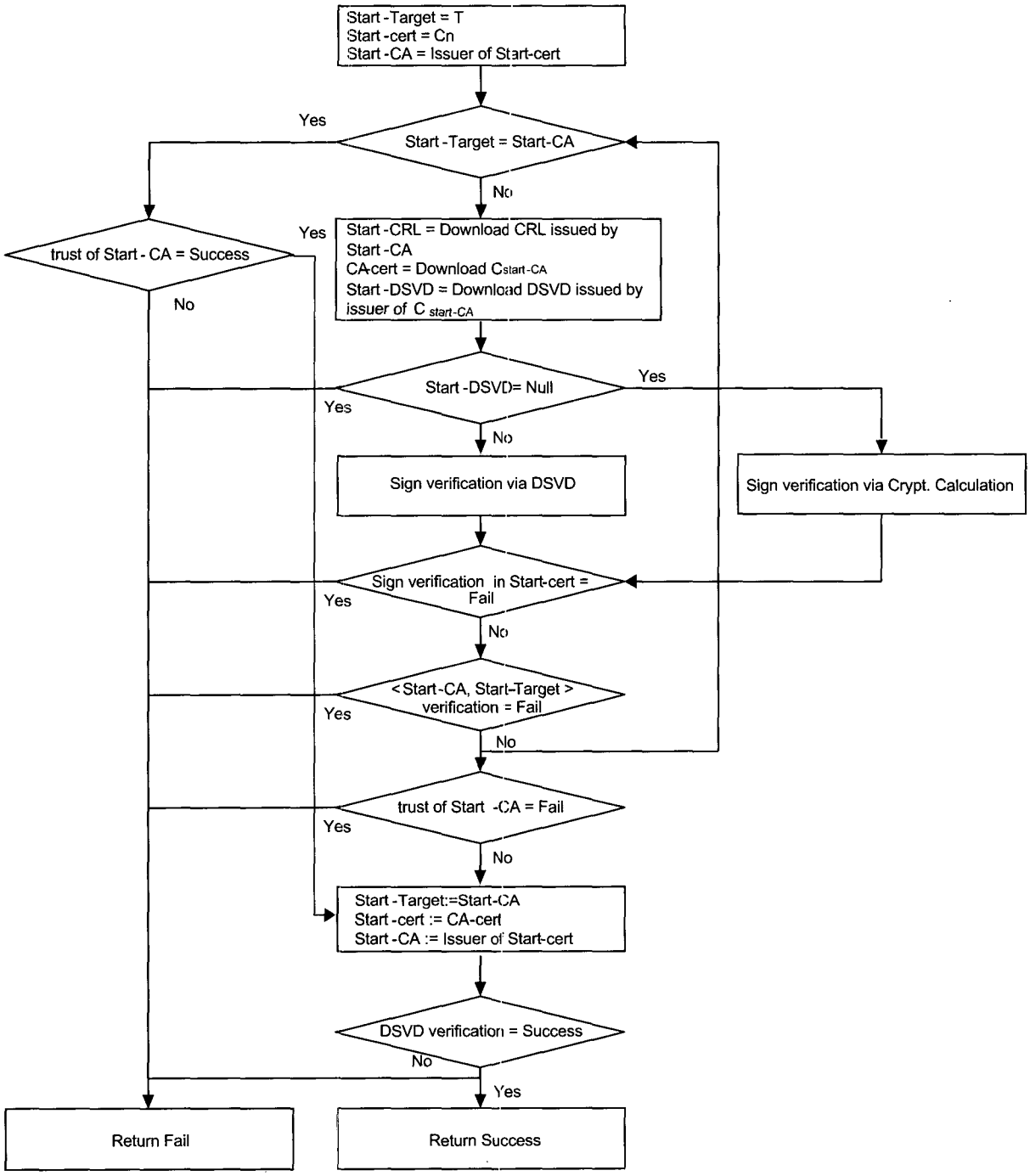
```
DSVDResponse ::= SEQUENCE{
   nonce            OCTET STRING,
   requesterName    [0] EXPLICIT GeneralName     OPTIONAL,
   RequestHash      BITSTRING,
   certToValidate   CertOrCertRef,
   RelayMessages    SEQUENCE of RMs,
   responseTime     GeneralizedTime,
   Count            [1] OCTET STRING             OPTIONAL,
   responsetExtensions [2] EXPLICIT Extensions OPTIONAL,
   signature        [3] EXPLICIT Signature OPTIONAL}
RMs ::= SEQUENCE{
   RelayMessage 1   RM1,
   RelayMessage 2   RM2,
     ..........
   RelayMessagen-2  RMn-2}
RM1 ::= SEQUENCE{
   CAName           [0] EXPLICIT GeneralName     OPTIONAL,
   DSVD1Hash        BIT STRING,
   DSVD1Sign        BIT STRING,
   DSVD1VResult     VResult,
   signAlgorithm    AlgorithmIdentifier}
RM2 ::= SEQUENCE{
   CAName           [0] EXPLICIT GeneralName     OPTIONAL,
   DSVD2Hash        BIT STRING,
   DSVD2Sign        BIT STRING,
   DSVD2VResult     VResult,
   signAlgorithm    AlgorithmIdentifier}
          ..........
```

3.4.3 DSVD 서명 검증

DSVDResponse를 수신한 검증자는 (그림 6)의 과정을 통해 원래의 DSVD에 해쉬를 계산한 값과 서명값을 전송된 해쉬값 및 서명값과 비교함으로서 다음의 사항들을 확신한다.

```
RM1(DSVD1hash) = Computed(DSVD1hash)
RM1(DSVD1sign) = Original(DSVD1sign)
..........
..........
RMn-2(DSVDn-2hash) = Computed(DSVDn-2hash)
RMn-2(DSVDn-2sign) = Original(DSVDn-2sign)
```

(그림 6) 검증자 측의 DSVD 검증 과정



(그림 7) 제안한 인증서 검증 방식

① DSVD 내용의 무결성 : 해쉬값의 비교를 통해 DSVD 내용의 무결성을 검증한다.

② DSVD 서명의 합법성 : 서명값의 비교를 통해 DSVD 서명의 합법성을 확신한다.

③ CA를 통해 검증된 DSVD 서명 결과 : 각 DSVD는 CA에 의해 암호학적으로 서명 검증되어 검증된 결과가 DSVDResponse를 통해 전송됨으로서 검증자는 DSVD에 대한 신뢰성을 더욱 확신한다.

3.5 DSVD의 발행을 통한 인증서 검증 방식

(그림 7)은 제안한 인증서 검증 방식의 순서도를 나타낸 것이다.

제안한 방식에서는 DSVD를 다운로드 받아 인증서에 대한 DSVD가 존재할 경우 해쉬 계산을 통해 서명을 검증하고 DSVD가 존재하지 않을 경우에는 암호화 계산을 통해 서명을 검증한다. 기존 방식에 DSVD 검증 작업이 추가되고 모든 검증이 성공하면 검증자는 DSVD를 신뢰하여 DSVD

의 Valid Status 상태에 따라 타겟 인증서에 대한 서명의 유효성을 판단하게 된다. DSVDResponse의 응답 제한 시간을 주어 이 시간 내에 응답이 들어오지 않는다면 사용자가 직접 암호화 계산을 통해 서명을 검증해야 한다.

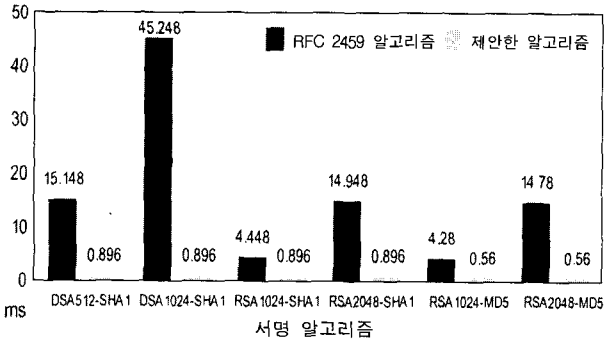
4. 분 석

4.1 서명 검증 소요 시간

이 절에서는 인증기관의 서명 검증 소요 시간은 고려하지 않았고, 검증자 측에서의 서명 검증 작업의 시간 및 속도를 측정함으로서 검증자 측 서명 검증 작업에 대한 부담을 분석하였다. 서명 검증 소요 시간을 분석하기 위해 각 알고리즘의 수행속도를 측정하였고, 이는 2개의 UltraSPAC-II 400MHz CPU를 가진 Sun Enterprise 3500 서버에서 OPENSSL 라이브러리를 사용하여 이루어졌다. 각 알고리즘의 수행 속도는 <표 3>의 시간으로 측정되었다. 여기서 $t_{Algorithm}$은 각 알고리즘의 계산 시간을 *millisecond* (*ms*)로 나타낸 시간이고 $\lambda_{Algorithm}$은 각 해쉬 알고리즘의 ms당 처리되는 비트수인 처리율을 나타낸다. Size(Cont)는 해쉬의 입력 바이트를 나타낸다. <표 3>의 측정 결과에 따라 RFC 2459 와 제안한 방식의 서명 검증 소요 시간은 (그림 8)과 같이 계산될 수 있다. (그림 8)에서는 보다 정확한 성능 향상을 보이기 위해 해쉬 입력 크기가 8192바이트일 경우의 시간을 비교하였다.

〈표 3〉 공개키 알고리즘과 해쉬 함수의 실행 속도

Algorithm	$t_{Algorithm}(ms)$	*Size(Cont)(byte)*	$\lambda_{Algorithm}(bytes/ms)$
DSA512	14.7	-	-
DSA1024	48.8	-	-
RSA1024	4.0	-	-
RSA2048	14.5	-	-
SHA-1	0.448	8192	19887
MD5	0.280	8192	31779



(그림 8) 8192 입력 바이트의 서명 검증 소요 시간

(그림 8)을 통해 해쉬 객체 크기가 상당히 큰 경우라도 제안한 방식이 RFC 2459 방식보다 서명 검증하는데 훨씬 더 적은 시간이 소요되는 것을 알 수 있다.

〈표 4〉 검증 소요 시간 분석을 위한 표기

표 기	의 미
t_{esign}	RFC 2459 방식의 인증서 서명 검증 시간
t_{epath}	RFC 2459 방식의 인증 경로 서명 검증 시간
t_{nsign}	제안한 방식의 인증서 서명 검증 시간
t_{npath}	제안한 방식의 인증 경로 서명 검증 시간
t_{crypt}	서명 검증 시 암호화 계산 시간
t_{hash}	서명 검증 시 해쉬 계산 시간
t_{cert}	암호화 계산을 통한 인증서 당 서명 검증 시간
t_{DSVD}	DSVD를 통한 인증서 당 서명 검증 시간
t_{DSVDR}	DSVPResponse를 통한 DSVD 검증 시간
n_{total}	경로상의 모든 인증서 수
n_{DSVD}	경로상의 DSVD를 통해 서명 검증할 인증서 수
n_{cert}	경로상의 암호화 계산을 통해 서명 검증할 인증서 수

서명 검증 속도 배율을 통해 제안한 방식의 서명 검증 속도를 분석할 수 있다. 속도 배율은 기존 방식에 대한 제안한 방식의 계산 속도의 배수로서 그 배수만큼 속도가 빠르다는 것을 나타낸다. 속도 배율을 계산하기 위해 <표 4>의 표기를 사용한다.

t_{hash}는 초기화를 위한 고정된 설정 시간 t_h와 해쉬 계산 시간으로 구성됨으로서 식 (1)과 같이 계산되며 각 방식의 인증서 당 서명 검증 시간은 각각 식 (2)와 식 (3)으로 계산될 수 있다.

$$t_{hash} = t_h + \frac{Size(cont)}{\lambda_h} \tag{1}$$

$$t_{esign} = t_{hash} + t_{crypt} = t_h + \frac{Sixe(Cont)}{\lambda_h} + t_{crypt} \tag{2}$$

$$t_{nsign} = t_{DSVDR} + 2 \cdot \left(t_h + \frac{Size(Cont)}{\lambda_h} \right) \tag{3}$$

식 (2)와 식 (3)에 따라 서명 검증 속도의 배율은 식 (4)와 같이 계산될 수 있다.

$$SF_{sign} = \frac{t_{sign}}{t_{nsign}} = t_h + \frac{\frac{Size(Cont)}{\lambda_h}}{2 \cdot \left(t_h + \frac{Size(Cont)}{\lambda_h} \right)} = \frac{1}{2} + \frac{1}{2} \left(\frac{1}{\frac{t_h}{t_{crypt}} + \frac{Size(Cont)}{\lambda_h \cdot t_{crypt}}} \right) > 1 \tag{4}$$

여기서 t_h, t_{crypt}, $Size(Cont)$는 모두 양수이고 t_h와 $Size(cont)$가 λ_h와 t_{crypt}보다 대부분 작기 때문에 서명 검증 속도 배율은 항상 1보다 크다. 이것은 제안한 방식의 서명 검증 속도가 기존 방식의 서명 검증 속도보다 항상 1배 이상 빠르다는 것을 의미한다. 인증 경로에 대한 서명 검증 속도 배율은 또한 계산될 수 있다. t_{epath}와 t_{npath}는 각각 식 (5)와 식 (6)으로 계산될 수 있다.

$$t_{epath} = n_{total} \cdot \left(t_h + \frac{Size(Cont)}{\lambda_h} + t_{crypt}\right) = n_{total} \cdot t_h + n_{total} \cdot \frac{Size(Cont)}{\lambda_h} + N_{total} \cdot t_{crypt} \quad (5)$$

$$t_{npath} = (n_{total} - n_{DSVD}) \cdot t_{crypt} + n_{total} \cdot t_h + n_{total} \cdot \frac{Size(Cont)}{\lambda_h} + n_{DSVD} \cdot t_h + n_{DSVD} \cdot \frac{Size(Cont)}{\lambda_h} \quad (6)$$

식 (5)와 식 (6)에 따라 식 (7)이 유도될 수 있다.

여기서 t_h, t_{crypt}, $Size(cont)$는 모두 양수이고 t_{crypt}는 대부분 $t_h + \frac{Size(Cont)}{\lambda_h}$보다 크기 때문에 식 (8)의 범위를 가진다.

$$0 < \frac{t_{crypt} - \left(t_h + \frac{Size(Cont)}{\lambda_h}\right)}{t_{crypt} + \left(t_h + \frac{Size(Cont)}{\lambda_h}\right)} < 1 \quad (8)$$

n_{total}은 n_{DSVD}보다 항상 큼으로서 인증 경로 당 서명 검증 속도 배율 SF_{path}는 식 (9)와 같이 1보다 큰 결과가 유추된다.

$$\therefore SF_{path} = \frac{1}{1 - \frac{n_{DSVD}}{n_{total}} \cdot \left(\frac{t_{crypt} - \left(t_h + \frac{Size(Cont)}{\lambda_h}\right)}{t_{crypt} - \left(t_h + \frac{Size(Cont)}{\lambda_h}\right)}\right)} > 1 \quad (9)$$

결과적으로 n_{DSVD}가 0이 아닌 이상은 제안한 방식의 인증 경로에 대한 서명 검증 소요 시간이 RFC 2459 방식보다 항상 빠르다는 결론을 얻을 수 있었다.

$$\therefore SF_{path} = \frac{t_{epath}}{t_{npahth}} = \frac{n_{total} \cdot t_h + n_{total} \cdot \frac{Size(Cont)}{\lambda_h} + n_{total} \cdot t_{crypt}}{(n_{total} - n_{DSVD}) \cdot t_{crypt} + n_{total} \cdot t_h + n_{total} \cdot \frac{Size(Cont)}{\lambda_h} + N_{DSVD} \cdot t_h + n_{DSVD} \cdot \frac{Size(Cont)}{\lambda_h}}$$

$$= \frac{1}{1 - \frac{\left(n_{DSVD} \cdot t_{crypt} - n_{DSVD} \cdot t_h - n_{DSVD} \cdot \frac{Size(Cont)}{\lambda_h}\right)}{n_{total} \cdot t_{crypt} + n_{total} \cdot t_h - n_{DSVD} \cdot \frac{Size(Cont)}{\lambda_h}}} = \frac{1}{1 - \frac{n_{DSVD}}{n_{total}} \cdot \left(\frac{t_{crypt} - \left(t_h + \frac{Size(Cont)}{\lambda_h}\right)}{t_{crypt} - \left(t_h + \frac{Size(Cont)}{\lambda_h}\right)}\right)} \quad (7)$$

4.2 DSVD 발행과 관련한 CA측 연산

〈표 5〉 DSVD 발행 소요 시간

작업시간 / 알고리즘	서명 검증 시간(ms)	DSVD 서명 생성시간(ms)	전체 소요 시간(ms)
DSA512-SHA1	15.148	12.248	27.396
DSA1024-SHA1	45.248	40.048	85.296
RSA1024-SHA1	4.448	73.648	78.096
RSA2048-SHA1	14.948	483.348	498.296
RSA1024-MD5	4.280	73.480	77.76
RSA2048-MD5	14.780	483.180	497.96

CA는 DSVD의 발행을 위해 인증서의 서명을 검증하고 DSVD의 서명을 생성해야 한다. CA측의 인증서에 대한 DSVD를 발행하는데 소요되는 시간을 측정하면 <표 5>와 같다. <표 5>에서 서명 생성 시간은 4.1절의 분석 환경에서 측정된 시간이고 4.1절과 마찬가지로 해쉬 객체는 8192 바이트로 가정하였다. DSVD 발행 작업은 주체 인증서들이 취소되거나 갱신되는 경우를 제외하고는 주체 인증서 발행 시에 단 한번 수행되고 CA의 여유로운 시간에 발행하는 것을 기본으로 하기 때문에 <표 5>의 시간은 하나의 인증서에 대해 오직 한 번 소요되는 시간으로서 CA측에 부담이 되지 않는다.

〈표 6〉 4-level 20-ary에서의 DSVD의 수

Level	Level당 CA의 수	Level당 Nested Certificate의 수	Level당 DSVD의 수
0	1	160,000	400
1	20	8,000	8,000
2	400	400	160,000
3	8,000	0	0
4	16,000	0	0

i-level m-ary의 균일한 PKI에서 각 CA는 m^2의 DSVD를 균등하게 발행한다. <표 6>은 4-level 20-ary의 PKI에서 NPKI의 nested certificate 발행 수와 제안한 방식의 DSVD의 발행 수를 비교한 것이다. <표 6>에서 보이는 것처럼, 제안한 방식에서는 모든 CA에게 DSVD의 발행에 대한 부담이 일괄적으로 분배됨으로서 특정 CA의 부담이 커지는 문제는 발생하지 않는다.

5. 결 론

본 논문에서는 인증서의 서명 검증 작업을 CA들에게 위임함으로서 사용자 측 검증 작업에 대한 부담을 줄이고 검증 소요 시간을 향상시킬 수 있는 새로운 인증서 검증 방식을 제안하였다. 제안한 방식은 서명 검증 작업에 정적인 상태의 CA들을 참여시킴으로서 성능 좋은 CA들을 적극적으로 이용하도록 하였으며, 이것은 사용자 측의 부담을 덜어주고 검증 속도를 향상시키는 결과를 가져왔다. 속도 배율로 측정한 결과, 제안한 방식의 서명 검증 속도는 RFC 2459 알고리즘을 이용한 기존의 검증 방식들보다 항상 더 빠르다는 결론을 얻을 수 있었다.

CA들은 DSVD와 DSVP와 관련한 모든 작업들을 동시에 수행하지 않고 CA 자체가 고속의 높은 처리 능력을 가지기 때문에 이러한 작업들을 부담 없이 수행할 수 있을 것이다. 또한 인증서 서명 검증과 DSVD 발행 작업은 인증서의 갱신 및 취소가 되지 않는 한은 한번만 이루어지는 작업이고 여유로운 시간에 수행하기 때문에 CA측 부담은 더욱 감소될 것이다. 오히려 인증서 검증 작업은 여러번 수행되기 때문에 한번의 연산적인 부담으로 몇 배의 이득을 얻을 수 있을 것이다.

계층적 PKI에서는 사용자가 신뢰 CA에게 검증 초기에 DSVD에 대한 요청을 하고 신뢰 CA가 저장된 RM을 참조함으로서 빠른 응답이 전송될 수 있다. 또한 DSVD는 CA와 검증자에 의해 두 번 검증되고 검증자가 언제든지 오프라인으로 검증의 신뢰성 여부를 확인할 수 있기 때문에 CA들은 함부로 DSVD의 내용을 위조할 수 없을 것이다.

사용자가 증가하거나 인증서의 변화가 자주 발생할 경우 DSVP가 자주 수행됨으로서 CA간의 메시지 송.수신이 빈번하게 이루어진다. 이에 따라 CA측 보안에 문제가 발생할 수 있음으로서 이를 해결하기 위한 방안이 향후 연구되어야 할 것이다.

참 고 문 헌

[1] 심희원, "DNS를 이용한 상호 연동 및 인증서 검증 방안", http://www.kisa.or.kr/K_trend/KisaNews/200201/focus.html.

[2] N. A. Nazario, "Security Policies for the Federal Public Key Infrastructure," 19th NISSC, October, 1996.

[3] 염홍렬, "<테마특강> 공개키 기반 구조 (PKI) 기술 동향", 전자신문 ET news, http://www.etimesi.co.kr/news/detail.html?id = 200102190074.

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

[5] M. Myers, "X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol-OCSP," IETF RFC 2560, June, 1999.

[6] 황보성, "서버 기반 인증서 검증", http://www.rootca.or.kr/down/down1/Server%20Based%20Certificate%20Validation.pdf.

[7] 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.

[8] 염홍렬, "DPD/DPV 기능을 갖는 OCSPv2표준", 표준화 동향 특집, http://www.kisa.or.kr/K_trend/KisaNews/200108/standardixation_07.html

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

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

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

[12] ETRI ZONE/R&D News, "세계 최초의 통합형 인증서 검증 시스템(CVS·Certificate Validation System) 개발," http://www.etri.re.kr/news/02-03/etri05.htm.

[13] Albert Levi, M.Ufuk Caglayan, "Analytical performance evaluation of nested certificates," Performance Evaluation, Vols.36-37, pp.213-232, August, 1999.

최 연 희

e-mail : lovejung22@hanmail.net

1991년 목포대학교 전산통계학과(학사)

1993년 숭실대학교 전자계산학과(석사)

2004년 숭실대학교 컴퓨터학과 박사과정 졸업 예정

관심분야 : 컴퓨터 통신, 정보 보안, 암호화 알고리즘, PKI

박 미 옥

e-mail : mopark@kingdom.ssu.ac.kr

1991년 조선대학교 전자계산학과(학사)

1993년 숭실대학교 전자계산학과(석사)

2004년 현재 숭실대학교 컴퓨터학과 박사 과정

관심분야 : 이동 통신 보안, 차세대 이동 통신, 정보 보안

전 문 석

e-mail : mjun@computing.ssu.ac.kr

1980년 숭실대학교 전자 계산 학과(학사)

1986년 University of Maryland 전산과 (석사)

1989년 University of Maryland 전산과 (박사)

1989년 Morgan State University 전산수학과 조교수

1989년~1991년 New Mexico State University 부설 Physical Science Lab. 책임 연구원

1991년~현재 숭실대학교 정보과학대학 컴퓨터학과 정교수

관심분야 : 네트워크 보안, 컴퓨터 알고리즘, 병렬처리, VLSI 설계, 암호학