확장된 DNS 보안 메커니즘의 설계 및 구현

심 희 원† · 김 진 성†† · 심 영 철††† · 임 찬 순†††† · 변 옥 환†††††

요 약

DNS는 인터넷 사용에서 가장 기본이 되는 네이밍 서비스를 제공하므로 인터넷 보안을 위해서는 DNS의 보안이 반드시 제공되어야 한다. 최근 IETF에서는 DNS 데이터베이스 내용과 질의 및 응답 메시지의 무결성을 제공하고 DNS를 사용하여 호스트의 공개키를 분배하는 방안을 제안하였다. 본 논문에서는 IETF 표준을 기본으로 하고 사용 및 관리가 용이하도록 기능이 추가된 안전한 DNS의 설계 및 구현에 대해 설명한다. 확장된 안전한 DNS에서는 DNS 서버가 공개키 기반 구조의 디렉토리 시스템으로 사용되어 사용자의 공개키 인증서를 저장하고 분배할 수 있는 기능을 제공한다. 또 Web 기반의 관리자 인터페이스와 보안 로그 기능이 추가되었으며 새로운 암호화 알고리즘이 쉽게 추가될 수 있도록 하였다.

Design and Implementation of the Extended DNS Security Mechanism

Hee-Won Shim† · Jin-Sung Kim†† · Young-Chul Shim†††
Chan-Soon Lim†††† · Ok-Hwan Byeon†††††

ABSTRACT

The DNS provides naming services which are the basis for the application of the Internet and the security of the DNS should be provided for the security of the Internet. Recently IETF proposed a method which guarantees the integrity of DNS database contents and DNS queries/replies and distributes host public keys. In this paper we describe the design and implementation of the secure DNS which is built based on the IETF proposal and extended to facilitate its use and management. In the extended secure DNS, DNS servers are used as the directory system in a public key infrastructure and stores/distributes user public key certificates. The Web-based management interface and security log functions are added and the extended secure DNS is being built so that new cryptographic algorithms can be easily added.

1. 서 론

최근 정보통신 기술의 발달과 더불어 전 세계적으로 확장되고 있는 네트워크의 첨단기술 덕분에 인터넷을 이용하여 전 세계에 산재해 있는 정보와 자원에 대한 접근이 용이하게 되었다. 따라서 기하급수적으로 네트워크 사용자가 늘어나게 되었으나 이에 따른 부작용도 심각하게 발생하고 있다. 인터넷은 기본적으로 개방성을 갖는 네트워크이므로 많은 사람들이 네트워크에 용이하게 접근할 수 있다는 장점이 있지만 침입자들도 마찬가지로 쉽게 타인의 컴퓨터에 접근하여 귀중한 정보를 유출시키거나 또는 변경시킴으로써 많은 피해를 발생시키고 있는 상황이다[1].

† 준 회 원 : 홍익대학교 대학원 전자계산학과
†† 준 회 원 : 홍익대학교 대학원 정보공학과
††† 종신회원 : 홍익대학교 컴퓨터공학과 교수
†††† 정 회 원 : 한국전자통신연구원 연구원
††††† 정 회 원 : 한국전자통신연구원 책임연구원
논문접수 : 1998년 9월 3일, 심사완료 : 1998년 11월 13일

이러한 인터넷 보안의 문제는 여러 관점에서 해결되어야 하지만 그 중 시급히 해결되어야 할 문제점 중의 하나는 DNS(Domain Name System) 문제이다. DNS는 도메인 이름을 IP 주소로 또는 그 역으로 변환시켜주는 네이밍 서비스로서 WWW, TELNET, FTP, E-MAIL 등과 같은 모든 인터넷 응용 프로그램들이 사용하고 있다. 그러므로 이와 같은 인터넷 응용들에 대한 보안성이 제공될 수 있기 위해서는 DNS의 보안 문제가 먼저 해결되어야 한다. 이와 같이 DNS는 인터넷에서의 가장 중요한 기반구조이지만 보안을 제공하기 위한 연구는 근래에 들어서야 시작된 상황이다.

IETF는 안전한 DNS 서비스를 위하여 기존 DNS의 네이밍 서비스 기능 외에 다음의 세 가지 보안 기능이 추가된 것을 권고하고 있다[2]. 첫째는 인증된 공개키의 분배 서비스이고, 두 번째는 서버에 저장된 내용의 무결성과 데이터 인증 서비스이며, 마지막은 질의 또는 응답 메시지의 무결성을 제공하는 것이다. 이러한 보안 서비스를 제공하기 위한 핵심은 3개의 새로운 자원레코드이다. 인증된 공개키의 저장이나 분배를 위해서 KEY 자원레코드를 정의하였으며, 데이터의 무결성과 인증 서비스를 부여하기 위해서 SIG(SIGnature) 자원레코드를 정의하였다. 또한 존재하지 않는 자원레코드의 부재를 인증하기 위해서 NXT(NeXT)라는 자원레코드를 정의하였다.

이러한 보안 서비스 외에 안전한 DNS에는 다음과 같은 기능들이 추가될 필요가 있다. 첫째는 안전한 DNS가 공개키 기반구조로서 인증서를 제공하고 이를 인증하는 체계를 갖추는 것이다. 사용자가 자신의 공개키(public key)를 등록하거나 타인의 공개키를 획득할 수 있는 시스템이 PKI(Public Key Infrastructure)이다. PKI에서는 사용자가 인증기관에 자신의 공개키를 등록하면 인증기관은 자신의 개인키(Private key)로 사용자의 공개키에 서명하고 서명된 공개키를 인증서로 발행한다. 이러한 인증서는 디렉토리 시스템안에 저장되고 사용자들은 이 디렉토리 시스템으로부터 다른 사용자의 인증서를 획득할 수 있게 된다. PEM Privacy Enhanced Mail)에서와 같이 전세계의 사용자를 포함할 수 있는 PKI의 구축이 제안되었지만[3, 4] 실제로는 매우 작은 규모로만 만들어져 특정한 인터넷 응용 프로그램을 사용하거나 또는 특정 기관에 속하는 적은 수의 사용자 그룹에만 적용되고 있는 실정이다. 따라서 전세계적인 PKI를 형성하려면 사용자나 호스트에 대한 네이밍 체계와 디렉토리 시스템의 표준이 설정되어야 하며 안전한 DNS는 구조상 이러한 요구조건에 상당히 적합하다. 그러므로 안전한 DNS를 PKI의 디렉토리 시스템으로 사용하면 대규모의 PKI 구축이 훨씬 용이해질 것이다.

둘째는 사용자 인터페이스의 구현이다. 안전한 DNS는 기존 DNS에 비해 존(zone) 파일이 상당히 커져 이들의 관리가 쉽지 않게 되었다. 새로운 KEY, SIG, NXT 자원레코드가 추가되고 빈번하게 발생하는 자원레코드의 등록, 삭제, 갱신등으로 이들을 각각 제어하려면 많은 시간과 노력이 필요하게 되었다. 따라서 자동화된 관리자 인터페이스를 통해 GUI 환경에서 이들을 쉽게 관리하는 기능이 요구된다. 그밖에 DNS의 운영상 서비스 중 발생하는 통계적 데이터를 분석하여 이를 그래프 형태로 출력하는 기능과 보안관련 도구의 핵심이라 할 수 있는 로깅 기능도 필수적인 요소이다.

셋째는 다양한 암호화 모듈의 접속이다. 총 256개의 사용 가능한 암호화 알고리즘 중 현재 RFC2065에 정의된 암호화 알고리즘은 RSA/MD5, Diffie-Hellman, DSA, Elliptic curve 등을 포함한 7개이다. 따라서 새로운 암호화 알고리즘을 쉽게 적용할 수 있는 암호화 알고리즘에 대한 인터페이스의 정의가 필요하다.

본 논문에서는 앞에서 기술한 바와 같이 사용자 인증서의 저장과 분배기능을 제공하고, 사용자 인터페이스와 로그 기능을 포함하며, 다양한 암호화 알고리즘이 용이하게 추가될 수 있는 안전한 DNS의 설계 및 구현에 대해서 설명한다.

본 논문의 구성은 다음과 같다. 2장에서는 RFC2065에 정의된 안전한 DNS의 기능과 PKI에 대해 설명한다. 3장은 사용자 인증서 저장소로서의 안전한 DNS 서버가 어떻게 확장되어야 하는가에 대해 설명한다. 4장에서는 안전한 DNS의 구조와 암호화 인터페이스 및 관리자 인터페이스 등의 설계 및 구현에 대해 설명하고 마지막 5장에서는 결론으로 본 논문을 끝맺는다.

2. 안전한 DNS와 PKI

본 장에서는 IETF의 RFC 2065에 정의된 안전한 DNS의 기본 사양과 IETF의 PKIX(Public Key Infrastructure)분과의 드래프트에 의해 정의되고 있는 X.509 기반 PKI에 대해 설명한다.

2.1 안전한 DNS

RFC2065에 정의된 안전한 DNS는 다음 세 가지의 보안 서비스를 제공한다.

- **키분배** : 존(zone), 호스트, 사용자들은 한 쌍의 공개키(public key)와 개인키(private key)를 보유하고 이 중 공개키는 DNS 서버에 저장되며 사용자의 요구에 따라 분배된다.
- **DNS 서버에 저장된 내용에 대한 무결성과 데이터 인증** : DNS 서버 내에 저장되어 있는 자원 레코드들은 DNS 서버에 의해 전자서명되고 이 전자서명은 별도의 자원 레코드로서 저장된다.
- **DNS 질의와 응답에 대한 무결성과 데이터 인증** : DNS 질의와 응답 패킷의 헤더와 내용은 DNS 클라이언트와 서버에 의해 서명된다.

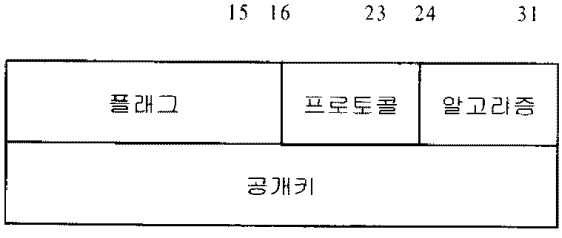
다음에서는 이들 확장된 보안 서비스들에 대해 좀 더 자세히 기술한다.

2.1.1 키 분배

존, 호스트, 사용자들과 같은 개체들은 공개키와 개인키의 쌍을 가지고 이 중 공개키는 DNS 서버 내에 KEY 자원 레코드로 다음과 같이 저장된다.

foo.host.example. IN KEY RDATA

RDATA는 foo.host.exmaple의 공개키를 (그림 1)의 형식으로 저장된다.



(그림 1) KEY RDATA의 형식
(Fig. 1) Format of KEY RDATA

플래그(flag)는 키가 사용자나 호스트 또는 존 중 어떠한 개체에 속하는 것인지와 용도에 대해 기술하고 있다. 프로토콜(protocol)은 어떠한 프로토콜이 이 키를 사용할 수 있는지를 나타낸다. 그리고 알고리즘(algorithm)은 공개키의 암호화 알고리즘을 명시한다.

DNS는 개체들의 인증서와 CRL(Certificate Revocation List)을 다음과 같은 형식으로 저장한다.

foo.host.example. CERT RDATA

여기서 RDATA는 다음의 내용을 포함한다.

- **유형** : 인증서가 X.509, SPKI(Simple Public Key Infrastructure), PGP(Pretty Good Privacy) 등의 유형 중 어떠한 것인지를 명세한다.
- **키 태그** : 인증서 내의 공개키가 KEY 자원 레코드의 공개키 중 어떠한 것에 해당하는지정의한다.
- 인증서인지 또는 CRL인지에 대한 명세

2.1.2 자원 레코드의 무결성과 데이터 인증

DNS 서버에 저장되어 있는 자원 레코드에 대한 무결성과 데이터 인증은 기본적으로 SIG 레코드에 의해 보장된다. SIG 자원 레코드는 다른 자원 레코드에 대한 서명을 가지고 있으며 이 서명은 서명되는 자원 레코드가 속하는 존의 개인키로 만들어진다. SIG 자원 레코드는 다음의 형식으로 저장되며

foo.host.example. IN SIG RDATA

RDATA는 다음의 내용을 포함하고 있다.

- **유형** : 서명되는 자원 레코드의 유형이 NS, A, MX, CNAME인지를 나타낸다.
- **알고리즘** : 서명을 위해 어떠한 해쉬 알고리즘과 암호 알고리즘이 사용되었는지 정의한다.
- **서명 만료** : 서명의 유효기간이 언제 만료되는지 알려준다.
- **서명 시각** : 서명된 시간을 알려준다.
- **서명자의 이름**
- **서명**

2.1.3 질의와 응답의 무결성과 데이터 인증

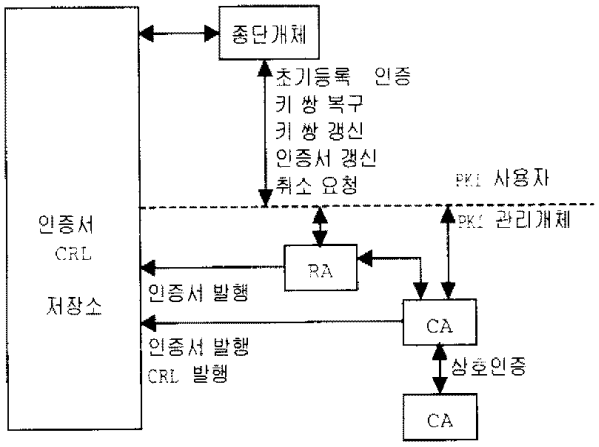
DNS 질의와 응답 패킷의 내용과 헤더는 이 질의 및 응답을 만드는 DNS 클라이언트나 서버의 개인키로 서명되어 무결성과 데이터 인증이 보장된다.

2.2. PKI (Public Key Infrastructure)

(그림 2)는 IETF의 PKIX 분과에 의해 발표된 드래프트 표준에서 제안된 X.509 기반 PKI의 구성 요소와

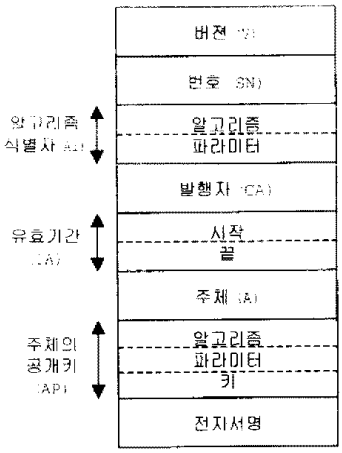
이들 사이의 관계를 보여주고 있다[5,6]. 각 구성 요소들은 다음과 같이 정의된다.

- **종단 개체(End Entity)** : PKI 인증서의 사용자나 인증서의 주체가 되는 종단 시스템
- **CA(Certification Authority)** : 인증기관
- **RA(Registration Authority)** : 등록기관으로서 인증기관이 자신의 관리기능 중 일부를 넘겨줄 수 있는 선택 사항임
- **저장소(Repository)** : 인증서와 CRL을 저장하고 이를 필요로 하는 종단 개체들에게 분배해 주는 시스템이나 분산 시스템의 집합 종단 개체는 CA나 RA를 통하여 다음의 관리 연산을 수행한다.



(그림 2) X.509 PKI의 구성요소
(Fig. 2) Components of X.509 PKI

- **초기 등록과 인증** : 종단 개체가 자신의 신원을 CA나 RA에 알리면 CA나 RA는 이 종단 개체의 공개키 인증서를 생성하여 종단 개체에 전해 주고 인증서 저장소에 저장한다.
- **키 쌍 복구** : 선택 사항으로서 CA가 사용자의 개인키를 저장하고 있어서 만약 사용자가 자신의 개인키를 분실한 경우 이 CA로부터 복구할 수 있다.
- **키 쌍 갱신** : 모든 키 쌍은 새로운 키 쌍으로 교체되어야 하고 이에 따라 새 인증서가 발급된다.
- **인증서 갱신** : 특별한 환경 상의 변화가 없었다면 인증서의 유효기간이 지난 경우 유효기간이 연장된다.
- **취소 요청** : 보안상의 문제가 발생한 경우 권한이 있는 사용자가 CA에게 인증서의 취소를 요청할 수 있다.



(그림 3) X.509 인증서
(Fig. 3) An X.509 Certificate

X.509 표준의 인증서는 (그림 3)과 같이 구성되며 다음과 같은 형식으로 표현된다.

$$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, TA, A, AP\}$$

여기서

$Y\langle\langle X\rangle\rangle$ = 인증기관 Y에 의해 발행된 사용자 X의 인증서

$Y\{I\}$ = I와 I에 대한 Y의 전자서명

X.509 시스템의 문제점은 사용자가 공개키 인증서를 등록하고 찾을 수 있기 위해서는 매우 크고 복잡한 PKI가 구축되어 있어야 하는 것이다. 이 문제를 해결하기 위해서 PGP에서는 매우 간단한 방법을 채택하고 있다[1, 7]. PGP에서 사용자는 다른 사용자의 공개키 인증서를 이 사용자나, 제 3의 다른 사용자, 또는 키 서버로부터 전자우편이나 FTP를 사용하여 받아 온다. 그러나 PGP 패킷 형식에 따르면 공개키 인증서는 사용자 이름이나 서명을 포함하고 있지 않다. PGP 인증서는 단지 공개키와 알고리즘 그리고 유효기간 만을 포함하고 있다. 그러므로 PGP 인증서에는 한 명 이상의 사용자의 ID 패킷과 전자서명 패킷이 따라야 한다. PGP 환경에서 공개키를 안전하게 분배할 수 있는 방법은 신뢰된 키 서버를 사용하는 것이다. 사용자는 자신의 공개키와 사용자 ID 패킷을 키 서버에게 보내면 키 서버는 자신의 개인키로 전자서명을 하고 다른 사용자의 요청이 있으면 이 서명된 공개키 인증서를 보내 준다.

3. 인증서 저장소로서의 DNS 서버

본 장에서는 DNS 서버가 X.509 표준에 의거한 IETF PKIX 구조에서 인증서 저장소로서 어떻게 사용될 수 있는가와 이를 위해서 DNS가 어떻게 변경되어야 하는가에 대해 설명한다.

3.1. DNS 서버 내의 인증서와 CRL의 저장

X.509 PKI의 CA는 DNS 서버와 같이 계층적 구조를 이루고 있다고 가정한다. 그리고 PGP 사용자는 자신의 공개키를 PGP CA를 통하여 등록한다고 가정한다. 하나의 CA는 하나의 DNS 서버에 연결되어 있어서 이 CA가 발행하는 인증서와 CRL은 모두 이 DNS 서버에 저장된다. 그러나 어떠한 CA와도 연결되지 않은 DNS는 존재할 수 있다. 만약 DNS 서버와 CA가 서로 연결되어 있으면 DNS 서버는 자신의 공개키를 이 CA에 등록하며 반대로 이 CA는 자신의 이름이 이 DNS 서버에 등록되고 CA의 공개키 인증서도 이 DNS 서버에 저장된다.

사용자는 전자우편 주소로서 유일하게 식별이 되며, 이 사용자 전자우편 주소가 인증서의 주체 이름(subject name)으로 사용된다. PGP의 경우에는 전자우편 주소가 이미 인증서의 주체 이름으로 사용되고 있으므로 전혀 문제가 없다. 그러나 X.509에서는 {attribute, value}의 리스트가 식별자로서 사용되고 있으며 이 식별자는 전자우편 주소와는 완전히 다른 형태를 가지고 있으므로 문제가 될 수 있다. 그러나 X.509에서는 선택사항인 Subject Alternative Name 필드를 사용하여 다른 형태의 이름을 사용할 수 있도록 허용하고 있다. 그러므로 X.509 인증서에서 이 Subject Alternative Name 필드에 전자우편 주소를 저장하고 Subject 필드는 빈 공간으로 남겨 놓음으로써 전자우편 주소를 X.509 인증서의 주체 이름으로 정의한다.

서비스에 대한 이름은 포트 번호나 서비스 이름을 사용하는 방법이 있다. 전자우편 서비스는 일반적으로 25번의 포트를 사용하므로 daehan.ac.kr이라는 호스트에서의 전자우편 서버의 도메인 이름은 25.daehan.ac.kr이라고 할 수 있다. 그러나 만약 서비스가 고정된 포트 번호를 사용하지 않는 경우라면 포트 번호 대신에 서비스 이름을 호스트 이름에 붙여서 사용한다. 즉 daehan.ac.kr라는 호스트가 db라는 이름을 가지고 데이터베이스 서비스를 제공하는 경우 포트 번호가 고정되어 있지 않다면 이 서비스의 도메인 이름은 db.daehan.ac.kr이라고 정한다.

IETF 드래프트와는 달리 인증서에는 CERT의 자원 레코드 유형을 CRL에는 CRL의 자원 레코드 유형을 사용한다. 다음의 X.509 예에서 daehan.ac.kr이라는 도메인 이름을 가지는 학교에 host라는 호스트가 등록이 되어 있고 또 이 호스트에 kim이라는 사용자와 ca라는 이름의 서비스가 존재할 때 이 대학의 DNS 서버에는 다음과 같은 정보가 저장된다.

```
kim.host.daehan.ac.kr.    CERT    RDATA-1 ;
                                        kim의 인증서
ca.host.daehan.ac.kr.     CERT    RDATA-2 ;
                                        ca의 인증서
ca.host.daehan.ac.kr.     CRL     RDATA-3 ;
                                    ca에 의해 발행된 CRL
```

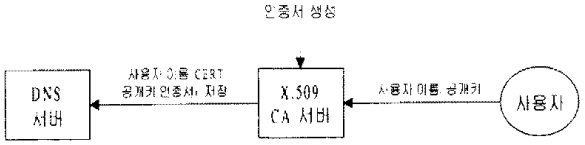
인증서와 CRL에 다른 자원 레코드 유형을 사용하는 이유는 어떤 개체에 대해 주어진 RDATA가 인증서인지 또는 CRL인지 쉽게 구별할 수 있기 위해서이다. 만약 IETF와 같이 인증서와 CRL에 같은 자원 레코드 유형을 사용하게 되면 DNS 클라이언트가 CA의 인증서만을 요구하고 싶더라도 이를 명세할 방법이 없어 결국 인증서와 함께 CRL도 응답으로 받게되는 문제가 발생한다. CERT와 CRL 유형의 자원 레코드에는 유형과 데이터의 두 필드가 존재한다. 유형 필드는 인증서나 CRL이 X.509, PGP, SPKI 등 중 어느 유형인지를 정의하고 데이터 필드는 실제 인증서나 CRL을 저장하고 있다. IETF 드래프트에 있는 키 태그 필드를 제외시킨 이유는 인증서에 있는 공개키가 별도의 KEY 자원 레코드로 중복되어 저장될 필요가 없다고 판단하였기 때문이다.

DNS 서버는 주기적으로 데이터베이스 내의 모든 인증서를 검사하여 인증서의 유효기간이 만료되었는지를 검사한다. 만약 유효기간이 지났으면 인증서를 데이터베이스에서 삭제한다.

3.2. 인증서와 CRL의 발행

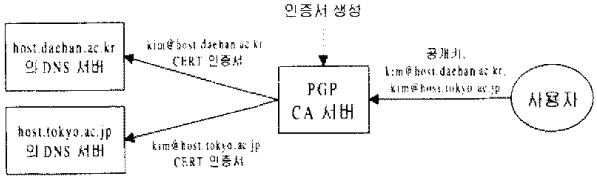
X.509에서 CA는 인증서와 CRL을 자신과 연결되어 있는 DNS 서버에 저장한다. 사용자가 자신의 공개키를 X.509 CA에 보내면 CA는 이 공개키에 서명을 함으로써 인증서를 생성하고 이 인증서를 전자우편 주소 형태의 주체 이름과 같이 DNS 서버에 보낸다. 주체

이름과 인증서를 받은 DNS 서버는 인증서를 이 주체 이름의 CERT 자원 레코드 유형으로 저장한다. 이 과정을 도식적으로 보이면 (그림 4)와 같다. DNS 서버가 CA로부터 CRL을 받으면 CRL의 서명을 점검한 후 CRL 자원 레코드 유형으로 저장한다. 이 때 이 CRL 자원 레코드의 이름으로는 CA의 이름이 사용된다. DNS 서버는 이 CRL에 의해 취소된 공개키의 인증서들을 자신의 데이터베이스에서 찾아 삭제한다.



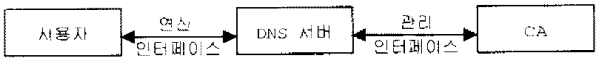
(그림 4) X.509 인증서의 생성 및 저장
(Fig. 4) Generation and Storage of X.509 Certificates

PGP 사용자는 자신의 공개키와 이 공개키에 해당하는 한 개 이상의 사용자 ID를 PGP CA에 보냄으로써 공개키를 등록한다. 하나의 공개키에 대해 CA는 하나의 인증서를 만들지만 이 공개키에 여러 사용자 이름이 주어진 경우 이 인증서는 각각의 사용자 이름별로 저장되어야 한다. 예를 들어 공개키가 kim@host.daehan.ac.kr과 kim@host.tokyo.ac.jp의 두 사용자 이름으로 등록이 되었다면 CA는 이 공개키에 대한 인증서를 생성한 후 이 인증서를 kim@host.daehan.ac.kr이라는 이름으로 먼저 DNS 서버에 저장하고 다음 kim@host.tokyo.ac.jp라는 이름으로 다시 DNS 서버에 저장한다. 첫 번째의 저장 요청은 host.daehan.ac.kr 호스트를 담당하는 DNS 서버에 전달되어 저장되고 두 번째의 저장 요청은 host.tokyo.ac.jp 호스트를 담당하는 DNS 서버에 전달되어 저장된다. (그림 5)는 이 과정을 도식적으로 보여준다.



(그림 5) PGP 인증서의 생성 및 저장
(Fig. 5) Generation and Storage of PGP Certificates

사용자는 PGP CA에 키 취소 인증서(key revocation certificate)를 보냄으로써 공개키의 취소를 요청할 수 있다. 이 취소 인증서는 사용자의 개인키로 서명이 되어 있으므로, CA는 이 서명을 확인한 후 이 취소 인증서에 해당하는 모든 사용자 ID를 인증서에서 검색해 낸다. 만약 앞의 예와 같이 등록된 공개키 인증서가 취소된다면 CA는 이 취소 인증서에서 kim@host.daehan.ac.kr과 kim@host.tokyo.ac.jp의 두 사용자 ID를 추출할 것이다. CA는 키 취소 인증서를 자신이 연결되어 있는 DNS 서버에 처음에는 kim@host.daehan.ac.kr의 이름으로 다음에는 kim@host.tokyo.ac.jp의 이름으로 보내고 이 두 키 취소 인증서는 각각 host.daehan.ac.kr 호스트와 host.tokyo.ac.jp 호스트를 담당하는 DNS 서버에 전달될 것이다.



(그림 6) DNS 서버의 인터페이스
(Fig. 6) Interface of a DNS server

DNS 서버는 (그림 6)과 같이 연산 인터페이스와 관리 인터페이스의 두 인터페이스를 가진다. 연산 인터페이스를 사용하여 DNS 클라이언트는 DNS 질의를 보내고 DNS 응답을 받으며 일반적으로 UDP가 사용된다. 관리 인터페이스는 CA가 인증서와 CRL을 등록하기 위해 사용된다. 관리를 위하여 새로운 인터페이스 표준을 정의하는 대신에 DNS의 동적 갱신을 위한 확장된 인터페이스를 사용한다. 사용자나 관리자가 DNS 서버에 요청을 보내고자 하면 적절한 매개변수 값을 가지고 res_query라는 리졸버 라이브러리 루틴을 호출하여 질의 메시지를 생성한다. res_mkquery는 다음과 같이 사용된다.

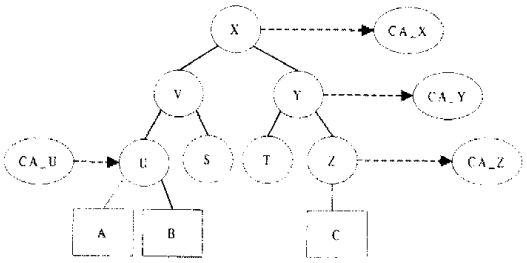
res_mkquery(op, dname, class, type, data, datalen, newrr, buf, buflen)

이들 매개변수들 중 operation_code(op)와 type의 두 매개변수가 확장되어야 한다. operation_code는 어떤 연산이 수행되어야 하는지를 정의한다. 보통 이 매개변수에는 질의나 역질의 값이 주어져 표준 질의(standard query)나 역질의(inverse query)를 요구할 수 있다. 그러나 DNS의 동적 갱신을 위하여 이 매개변수가 UPDATEA나 UPDATED와 같은 값을 가질 수 있도록 하는 확장안이 제안되어 있다[8]. operation_code 매

매개변수가 UPDATEA인 경우 새로운 자원 레코드의 삽입을 요청하는 것이며 UPDATED인 경우는 자원 레코드의 삭제를 요청하는 것이다. 그러므로 CERT나 CRL 유형의 자원 레코드를 저장하기 위하여서는 operation_code의 값을 UPDATEA로 준다. type 매개변수는 어떤 유형의 자원 레코드에 대한 연산을 수행하는지를 정의하므로 CERT나 CRL의 두 값을 가질 수 있도록 확장되어야 한다. 이들의 확장은 UNIX에서 /include/arpa/nameserv.h라는 이름의 파일에 행해진다. 인증서를 전달하는 패킷의 길이는 짧을 것이나 X.509의 경우 CRL은 매우 길 수 있다. 그러므로 관리 인터페이스에서는 TCP를 사용한다.

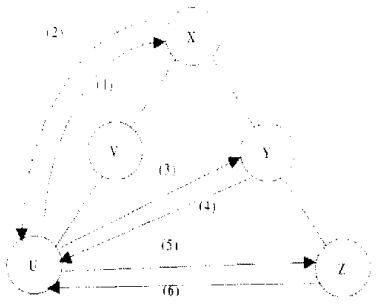
3.3. DNS 서버로부터의 공개키 인증서의 획득

DNS 서버로부터 인증서나 CRL을 가져오기 위해 DNS 클라이언트는 res_mkquery라는 리졸버 라이브러리 루틴을 사용하여 질의 메시지를 만든다. operation_code 매개변수는 질의로 주어지고 type 매개변수는 CERT나 CRL로 주어진다.



(그림 7) DNS 서버와 CA의 계층 구조 예
(Fig. 7) An Example of Hierarchies of DNS Servers and CAs

먼저 X.509 인증서를 DNS 서버에서 가져오는 과정을 설명한다. (그림 7)은 S, T, U, V, X, Y와 Z라는 이름을 가진 일곱 개의 DNS 서버들의 계층구조를 보여주고 있다. 이중 U, X, Y와 Z의 네 DNS 서버는 CA와 연결되어 있다. 두 사용자 A와 B가 U.V.X의 존에 등록되어 있고 사용자 C는 Z.Y.X의 존에 등록되어 있다고 하자. 사용자 A는 CA_U의 공개키와 루트 CA인 CA_X의 공개키를 가지고 있다고 가정한다. 만약 A가 B의 인증서를 원한다면 질의는 DNS 서버 U에 전달되고 U는 CA_U의 개인키로 서명된 B의 공개키 인증서를 받을 것이다. A는 CA_U의 공개키를 가지고 있으므로 B의 공개키 인증서를 확인할 수 있다.



(그림 8) 반복적 방법에 의한 DNS 질의의 처리
(Fig. 8) Processing a DNS Query Using the Interactive Method

만약 A가 다른 존에 등록되어 있는 C의 인증서를 요청하면 이 요청은 반복적(iterative) 방법이나 순환적(recursive) 방법으로 처리된다. 반복적 방법의 경우 이 질의는 (그림 8)과 같이 처리되고 각 메시지는 다음의 정보를 전달한다.

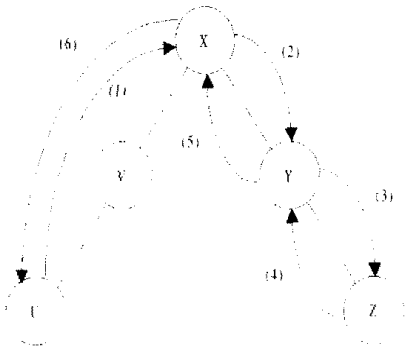
(1) U가 루트 DNS 서버인 X에게 C의 인증서를 요청한다.
(2) 루트는 Y의 주소를 U에게 보낸다.
(3) U는 Y에게 C의 인증서를 요청한다.
(4) Y는 Z의 주소를 U에게 보낸다.
(5) U는 Z에게 C의 인증서를 요청한다.
(6) Z는 C의 인증서를 U에게 보낸다.

메시지 (6)에서 Z는 CA_Z의 개인키로 서명된 C의 인증서를 보낼 것이다. 그러나 사용자 A는 CA_Z의 공개키를 가지고 있지 않으므로 DNS 질의에 대한 응답으로 받은 C의 공개키 인증서를 확인할 수 없다. CA_Z에 의한 서명을 확인할 수 있으려면 CA_Z에 대한 인증서가 필요할 것이다. CA_Z의 인증서는 상위의 CA인 CA_Y에 의해 서명되어 있을 것이므로 또 다시 CA_Y의 공개키가 필요하다. 결론적으로 사용자 A는 C의 인증서를 확인할 수 있기 위하여 다음과 같은 인증서 경로가 필요하게 된다.

CA_X<<CA_Y>>, CA_Y<<CA_Z>>, CA_Z<<C>>

DNS 서버가 다른 서버의 주소나 사용자의 인증서를 답으로 보낼 때 자신에 연결된 CA의 공개키 인증서도 함께 보냄으로써 사용자 A는 위와 같은 완전한 공개키 경로를 획득할 수 있다. 즉 DNS 서버 X, Y, Z는 각각 CA_X, CA_Y, CA_Z의 인증서를 메시지 (2),

(4), (6)에서 U에게 보낸다. DNS 서버 U가 A에게 최종 응답을 보낼 때 C의 인증서는 DNS 응답의 응답 섹션(answer section)에 저장하고 X, Y, Z의 인증서는 DNS 응답의 추가 섹션(addtional section)에 저장하여 보낸다. A는 루트 CA의 공개키를 가지고 있다고 가정하였으므로 X로부터 받은 루트 CA인 CA_X의 공개키를 확인할 수 있으며 상기의 인증서 경로에 따라서 사용자 C의 인증서를 확인할 수 있다.



(그림 9) 순환적 방법에 의한 DNS 질의의 처리
(Fig. 9) Processing a DNS Query Using the Recursive Method

순환적 방법의 경우에는 질의는 (그림 9)와 같이 처리되고 각 메시지는 다음의 정보를 전달한다.

(1) U는 루트 DNS 서버인 X에게 C의 인증서를 요청한다.
(2) X는 Y에게 C의 인증서를 요청한다.
(3) Y는 Z에게 C의 인증서를 요청한다.
(4) Z는 C의 인증서를 Y에게 보낸다.
(5) Y는 C의 인증서를 X에게 보낸다.
(6) X는 C의 인증서를 U에게 보낸다.

이 경우에도 사용자 A는 반복적 방법과 같은 인증서 경로가 필요하다. 이것은 각 DNS 서버가 사용자 인증서를 답으로 돌려 보내거나 또는 이 답을 다른 DNS 서버에게 전달할 때 자신과 연결되어 있는 CA의 인증서도 같이 전달함으로써 가능해진다. 그러므로 앞의 예에서 사용자 A는 DNS 응답의 답 섹션에서 사용자 C의 인증서를 찾을 수 있음은 물론 이 응답의 추가 섹션에서 CA_X, CA_Y, CA_Z의 인증서도 찾을 수 있게 된다.

그러나 사용자가 PGP 인증서를 원하는 경우에는 X.509의 경우와 달리 전체의 인증서 경로가 필요하지 않고 단지 원하는 사용자의 인증서만을 돌려 보내면 된다.

DNS 서버가 요청된 인증서를 찾지 못한 경우에는 에러 메시지를 보내게 된다. 그리고 이 에러 메시지를 포함하고 있는 응답 메시지의 무결성과 데이터 인증은 이 응답을 보내는 DNS 서버의 전자서명에 의해 보장된다. 만약 이 응답 메시지의 무결성이 보장되지 않으면 침입자는 인증서를 포함하고 있는 메시지를 포획(capture)한 후 이 메시지를 버리고 대신 요청된 인증서를 찾을 수 없다는 거짓 응답 메시지를 보낼 수가 있다. 이러한 시나리오는 DNS 서버에 대한 서비스 거부(denial of service) 공격이 된다.

4. 안전한 DNS 설계와 구현

본 장에서는 안전한 DNS 서버의 요구사항과 전체 구조에 대해 설명하고 여러 구성요소 중 암호화 모듈과 관리자 인터페이스의 설계 및 구현에 대해 자세히 설명하였다.

4.1 안전한 DNS 서버의 구조

먼저 안전한 DNS가 제공하는 세 가지 보안서비스의 구현을 위해 필요한 요구사항에 대해 알아본다.

첫째 보안서비스는 공개키의 저장과 분배서비스이다. 이는 DNS 서버를 공개키의 저장장소로 이용하고 요청에 의해 저장된 공개키를 분배하는 서비스를 제공하는 것이다. 따라서 존 파일에 KEY 자원레코드를 저장하고 리졸버의 질의에 대해 해당 KEY 자원레코드를 분배하는 기능이 필요하다.

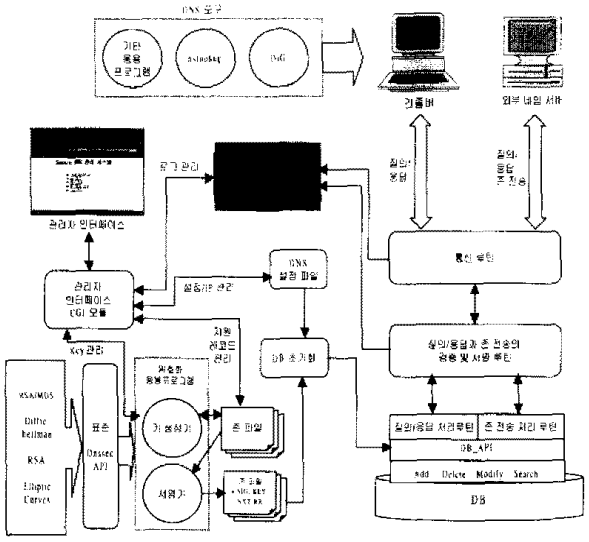
두번째 서비스는 서버에 저장된 내용에 대한 무결성과 데이터 인증서비스이다. 안전한 DNS는 요청된 자원레코드를 데이터베이스에서 찾아 응답할 때 응답되는 모든 자원레코드들마다 각각 전자서명한 SIG 자원레코드를 부가적으로 보낸다. 따라서 응답을 받은 DNS나 리졸버는 각각의 자원레코드에 해당하는 SIG 자원레코드를 검증하여 해당 자원레코드가 틀림없이 해당 존으로부터 전송된 것임을 검증한다. 이를 구현하기 위해서는 기존의 존파일을 인식하여 자동으로 각 자원레코드마다 해당 존의 개인키로 전자서명한 SIG 자원레코드를 생성하여야 한다. 또한 해당 자원레코드가 없는 것에 대한 인증을 제공하기 위해 존 파일을

재도니컬 순서로 정렬하여 NXT 자원레코드를 생성하는 별도의 작업이 필요하다.

세번째 서비스는 메시지의 응답과 질의에 대한 무결성 서비스이다. 즉 DNS와 DNS간 또는 DNS와 리졸버간의 메시지 전송 동안 어떠한 변조나 손실이 없음을 보장하는 것이다. 따라서 DNS 헤더를 포함한 모든 응답이나 질의 메시지를 전자서명한 특별한 SIG 자원레코드를 DNS 메시지의 마지막 부분인 부가적 부분(additional section)에 포함시켜 전송하는 작업이 필요하다.

이러한 모든 보안서비스는 암호화 모듈에 기반하여 키를 생성하고, 전자서명하며 이를 확인하는 작업에 의해 구현된다. 따라서 여러 암호화 알고리즘을 지원하면서도 하나의 공통된 API에 의해 사용할 수 있는 암호화 API가 필요하다. 암호화 키의 생성 및 관리, 암호화 인터페이스의 변경, 보안 로그 데이터베이스의 생성 및 관리등 많은 기능이 추가되어야 하므로 이의 편리한 관리를 위해 관리자 인터페이스가 필요하게 되었다.

구현된 안전한 DNS는 (그림 10)과 같이 DNS 서버 중심 모듈과 암호화 모듈, 그리고 관리자 인터페이스와 리졸버로 나뉘게 된다. DNS 서버의 중심 모듈은 통신 루틴, 질의/응답 처리 루틴, 존 전송 처리루틴과 DB 초기화 루틴으로 구성되고 암호화 모듈은 키 생성기와 서명기 등의 암호화 응용프로그램과 질의/응답의 서명 및 검증 루틴으로 구성된다.



(그림 10) 안전한 DNS의 구조
(Fig. 10) The Structure of a Secure DNS

DNS 관리자가 새로운 호스트와 이의 IP 주소를 존 파일에 입력하면 이 호스트에 대한 공개키와 개인키를 키 생성기를 사용하여 생성한 후 이중 공개키는 KEY 자원레코드로 존 파일에 저장한다. 호스트와 IP 주소의 입력과 키의 생성 및 저장은 관리자 인터페이스를 통해 수행된다.

KEY 자원레코드가 포함된 존 파일은 다시 서명기라는 응용프로그램에 의해 분석되어 각 자원레코드마다 전자서명을 한 SIG 자원레코드가 추가된다. 또한 해당 자원레코드가 없다는 것을 인증하는 NXT 자원레코드도 동시에 형성된다. 이렇게 KEY, SIG, NXT 자원레코드가 추가된 존 파일로부터 DNS 서비스를 위한 데이터베이스가 생성된다.

DNS 클라이언트인 리졸버나 외부 DNS 서버와 교환되는 질의 및 응답은 DNS 서버 중심 모듈과 암호화 모듈의 질의/응답 서명 및 검증 루틴에 의해 처리된다. 통신 루틴을 통해 수신된 질의 및 응답은 먼저 서명을 확인한 후 질의/응답 처리 루틴에 의해 처리된다. 외부로부터의 질의/응답에 대해 발생하는 추가 질의나 응답은 먼저 전자서명이 된 후 통신 루틴을 통해 발신된다. 질의/응답의 서명 검증이나 처리 과정 중 발생하는 보안 이벤트들은 로그 데이터베이스에 저장된다.

4.2 암호화 모듈의 설계 및 구현

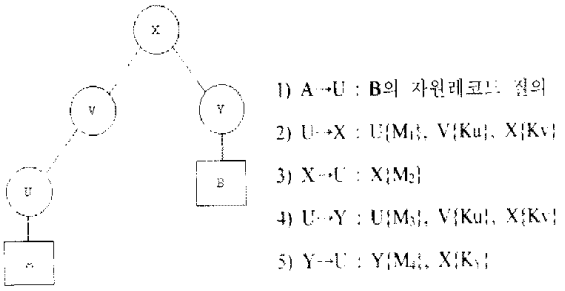
암호화 모듈에서는 키를 생성하고 저장하며 저장된 자원레코드의 무결성을 위하여 각 자원레코드들에 대한 전자서명을 SIG 자원레코드로 저장한다. 또 외부로부터 들어온 질의와 응답의 전자서명을 검증하고 외부로 나가는 질의 및 응답에 전자서명을 한다. 본 연구에서는 RSA/MD5, Diffie-Hellman, DSA 키의 생성 및 저장이 구현되었으며 이때 키의 길이는 각각 512에서 1024bit까지 생성할 수 있다. 또한 RSA/MD5, DSA/SHA-1의 전자서명 방식이 구현되었다.

본 절에서는 질의/응답의 서명 검증시 문제점과 이의 해결 방안에 대해 설명하고 다양한 암호화 알고리즘을 효율적으로 인터페이스하기 위한 구현 방법에 대해 설명한다.

4.2.1 질의/응답의 효율적인 검증을 위한 공개키 체인 전송

이미 설명한 바와 같이 안전한 DNS에서는 질의와 응답 메시지에 대한 무결성을 제공하기 위해 메시지의

마지막에 DNS 헤더를 포함한 모든 메시지에 대한 전자서명이 SIG 자원레코드로 추가된다. SIG 자원레코드를 검증하기 위해서는 전자서명한 존의 공개키가 필요하게 된다. 그러나 이 공개키의 무결성과 데이터 인증을 보장하기 위해서는 루트에서부터 메시지를 생성하여 보낸 존까지의 공개키 체인이 필요하게 된다.



(그림 11) 공개키 체인 전송방식
(Fig. 11) The Method of Public Key Chaining

이 문제점과 해결방안을 (그림 11)을 사용하여 구체적으로 설명한다. 그림에서 Kc는 C의 공개키이며 C{M}은 메시지 M과 이에 대한 C의 전자서명이다. 그림에서 A가 B의 자원레코드를 U에 요청하게 되면 U는 자신의 로컬 데이터베이스에 B의 자원레코드가 없으므로 루트인 X에게 B의 자원레코드를 질의하게 된다. 이때 U는 메시지 질의에 대한 무결성서비스를 제공하기 위해 DNS 메시지 M_1을 자신의 존키인 Ku로 전자서명한 SIG 자원레코드를 메시지와 함께 보낸다. X에서는 U가 보낸 SIG 자원레코드를 검증하기 위해 U의 공개키가 필요하므로 Ku를 메시지에 함께 보낼 수 있다. 하지만 이 메시지 내의 정보만을 사용하여 X가 메시지에서 추출한 U의 공개키가 틀림없이 U의 것이라는 것을 확인하는 것은 불가능하다.

루트 X가 Ku를 확인할 수 있으려면 먼저 Ku에 대한 V의 전자서명이 필요하고 다시 이 전자서명을 확인할 수 있으려면 V의 공개키 Kv가 필요하다. 계속해서 Kv의 공개키를 확인할 수 있으려면 Kv에 대한 루트 X의 전자서명이 필요하다. X의 입장에서 Kv와 Kv에 대한 X의 전자서명은 자신의 존 파일에서 찾을 수 있지만 Ku에 대한 V의 전자서명은 자신의 존 파일에서 찾을 수 없으므로 V에게 DNS 질의를 통하여 찾아와야 하고 이로 인해 DNS 질의/응답은 매우 늦어지게 된다.

X가 U에게 보낸 메시지 M_2는 X의 개인키로 전자서명되고 이는 U에 의해 쉽게 검증된다. 이것은 본 구현에서 모든 DNS 서버는 루트의 공개키인 Kx를 가지고 있도록 하였기 때문이다.

그러나 U가 X로부터 받은 Y의 주소를 사용하여 Y에게 보낸 메시지 M_3의 검증은 더욱 복잡해진다. M_3은 U의 개인키로 전자서명되어 있으므로 이의 확인을 위해 X가 M_1의 검증을 위해 필요로 하던 같은 정보가 필요하다. 이를 위해 Y는 X로부터 Kv와 Kv에 대한 X의 전자서명을 V로부터 Ku와 Ku에 대한 V의 전자서명을 얻어와야 하므로 DNS 질의/응답은 더욱 늦어지게 된다.

이러한 문제를 해결하기 위해 본 구현에서는 DNS 서버가 메시지 M을 전자서명하여 보낼 때 자신의 공개키 뿐만 아니라 자신이 속한 존에서부터 루트까지의 추가 키 및 전자서명을 보낸다. 이 추가 키 및 전자서명은 이 메시지를 받은 DNS 서버가 루트의 공개키만을 가지고 있더라도 다른 DNS 서버와 교신하지 않고 이 메시지에 대한 전자서명을 확인할 수 있도록 하는 최소의 필요한 정보이고 이를 공개키 체인이라 한다. U가 X나 B에게 메시지를 전자서명하여 보낼 때 자신의 공개키인 Ku, Ku에 대한 V의 전자서명, V의 공개키인 Kv, Kv에 대한 루트 X의 전자서명이 Ku의 공개키 체인이 된다. (그림 11)의 오른쪽에는 반복적 방법(iterative method)으로 DNS 질의가 처리될 때 전달되는 메시지와 공개키 체인이 설명되어 있다.

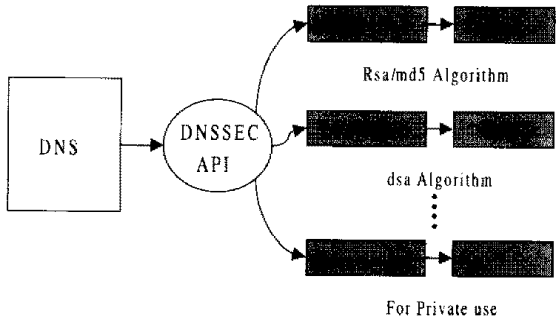
이 공개키 체인은 메시지의 부가적 부분에 포함하여 전달한다. 이러한 작업은 매우 많은 오버헤드를 가져올 것이라 예상되지만 도메인 네임의 루트로부터 대부분의 DNS가 3-5개의 존으로 구성된 것을 고려할 때 6-10개의 부가적인 자원레코드로 완벽한 보안서비스를 제공할 수 있으므로 보안레벨이 높은 서비스에 적용할 때 적합하다. 이러한 공개키 체인방식의 또 하나의 문제점은 기존의 DNS 서버들이 UDP 방식으로 메시지를 전송하는 데 있다. 즉 하나의 패킷 크기가 작아서 평균적으로 6-8개의 자원레코드만을 전송할 수 있도록 정해져서 많은 경우에 메시지가 절단되어 전송되게 된다. 따라서 TCP를 이용한 DNS 서비스를 제공하여 이러한 문제점을 해결하였다.

4.2.2 암호화 알고리즘의 접속

RFC2065에 의해 DNS에서 현재 정의된 알고리즘은

가지이며, 개인적인 용도나 간접적으로 키의 위치를 지정하는 갈림키 등 예약된 것을 제외하더라도 250개 정도의 알고리즘의 추가가 가능하다. 그러나 새로운 암호화 알고리즘을 추가할 때마다 다른 모듈과의 의존성 때문에 알고리즘의 추가가 쉽지 않다. 따라서 본 연구에서는 암호화 알고리즘을 쉽게 추가할 수 있는 암호화 인터페이스를 UNIX 기반의 FreeBSD와 Solaris에서 동시에 구현하였다.

현재 구현된 암호화 인터페이스의 구현방식은 (그림 12)와 같으며 독립된 암호화 알고리즘들과 DNSSEC API, 이들을 사용하는 DNS나 응용프로그램으로 구성된다. DNS는 각각의 암호화 알고리즘을 표준화된 방식으로 제어하기 위해 DNSSEC API를 제공하고 암호화 알고리즘들 역시 이 DNSSEC API를 통해 DNS로 결과를 되돌려 보낸다. 따라서 DNS 본체에서 자원레코드에 전자서명하거나 KEY를 검증하는 등의 작업을 수행할 때 직접 암호화 알고리즘의 세부적인 함수 이름을 모르고 있어도 DNSSEC API에 정의된 표준에 의해 쉽게 세부적인 함수를 이용할 수 있다.



(그림 12) 암호화 알고리즘의 운영방식
(Fig. 12) Operations of the Cipher Interface

암호화 알고리즘과 DNSSEC API 사이에는 알고리즘에 의존적인 링크(link)라는 계층이 있는데 이 계층의 목적은 완전히 독립적인 암호화 알고리즘을 DNS의 목적에 맞는 함수로 재정의 하기 위한 부분이다. 각각의 암호화 알고리즘은 서로 다른 입출력 방식을 제공하고 있으며 특정 작업을 수행하기 위해서 호출되는 함수들도 알고리즘에 따라 모두 다르다. 그러나 DNSSEC API에서 모든 암호화 알고리즘을 포함하는 표준을 제공하기는 매우 어렵다. 이 문제점은 DNSSEC API와 암호화 알고리즘 사이의 링크계층에 의해 해결하였다. 이 링크 계층에서는 DNSSEC의 표준 API를 통한 암호화 알고리즘의 호출은 각 알고리즘의 고유한 호출형태로 변환시키고 각 알고리즘으로부터 고유한 형태로 출력되는 내용을 DNSSEC API의 표준형태로 변환시킨다. 따라서 암호화 알고리즘이 추가되거나 삭제될 때 링크 계층에 의해 DNSSEC API의 수정이 최소화 되었다.

암호화 알고리즘은 링크 계층과 연결하여 하나의 독립된 모듈로 구성되고, 이 모듈은 표준화된 DNSSEC API의 함수를 호출하여 모든 알고리즘을 표준화된 방식으로 쉽게 사용할 수 있다. 즉 알고리즘에 독립적으로 해당 API 함수의 이름과 알고리즘 번호만으로 원하는 기능을 수행할 수 있다. DNS에서 필요로 하는 암호화 알고리즘 표준 API는 (그림 13)과 같다.

```
int dnssec_check() : Function to check if certain alg is
                     supported
ex) switch(alg) { ... case KEY_RSA: return 1 ...
int dnssec_sign() : Incremental signing routine takes flags to
                    select which steps to perform
ex) switch(key1->alg) { ... case KEY_RSA: rsa_cmp( ....
- int dnssec_verify() : Incremental verify routine
ex) switch(key1->alg) { ... case KEY_RSA: rsa_sign( ....
- int dnssec_genkey() : Function to generate new KEY for
                        DNS
                      : rsa_verify등의 rsaref_link에 정의된 함
                        수를 사용한다.
  KEY *dnssec_getkey() : Function to retrieve private KEY
                         from storage private keys are
                         stored internally and are only
                         decoded once.
- void dnssec_putkey() : Function to write out a private key
- KEY *dnssec_pubkey() : Function to convert KEY RR into a
                         KEY structure
- KEY *dnssec_readpubkey() : Function that reads in public
                             key
- int dnssec_writepubkey() : Function to write out public key
- int dnssec_encpubkey() : Function to return a public key in
                           DNS format base64 encoded
- void dnssec_freekey() : Releases all memory pointed to by
                          key structure
```

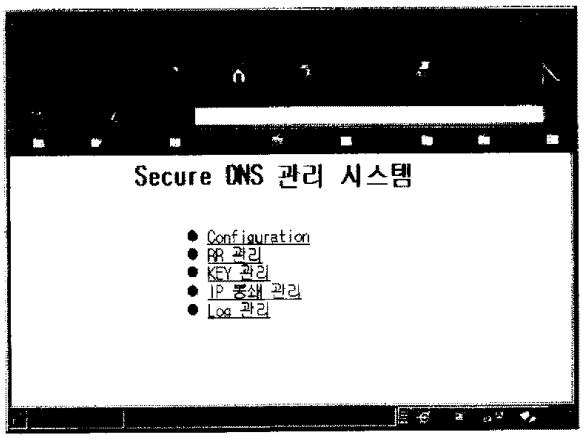
(그림 13) 암호화 알고리즘의 표준 API
(Fig. 13) Standard API of Cipher Algorithms

4.3 관리자 인터페이스

기존의 DNS에서 안전한 DNS로 진화하면서 혁신적인 보안 서비스를 제공하지만 이에 따라 발생하는 부작용도 적지 않다. 그중 가장 눈에 띄는 점은 상당히 커진 존 파일이다. 이는 새로운 KEY 자원 레코드의 추가와 각각의 자원 레코드마다 전자서명으로 인증 서비스를 제공하는 SIG 자원 레코드 그리고 정렬된 두 자원 레코드 사이에 어떠한 자원 레코드도 존재하지

많음을 증명하기 위해 필요한 NXT 자원 레코드로 존 파일의 부피가 상당히 커져 이들의 관리가 쉽지 않게 되었다. 게다가 빈번하게 발생하는 자원 레코드의 갱신뿐만 아니라, 새로운 자원 레코드의 등록과 기간이 만료된 자원레코드의 삭제등을 일일이 제어하기에는 많은 시간과 노력이 필요하게 되었다. 따라서 자동화된 관리자 인터페이스를 통해 GUI 환경에서 이들을 쉽게 관리하는 기능이 요구된다. 그밖에 DNS의 운영상 서비스 중 발생하는 여러 종류의 이벤트들을 수집 및 분석하여 관리자에게 출력하는 로그 기능과 보고 기능이 필요하다.

관리자 인터페이스는 Web 브라우저를 기반으로 관리자와 상호 작동하는 방식으로 여러 작업을 수행한다. 관리자 인터페이스의 세부적인 항목은 (그림 14)와 같이 DNS 설정에 관련된 항목과 자원 레코드의 관리, KEY 관리, IP 봉쇄관리와 마지막으로 로그의 관리 항목을 선택할 수 있도록 하였다.



(그림 14) 안전한 DNS 서버의 관리자 인터페이스
(Fig. 14) The Management Interface for the Secure DNS Server

DNS 설정 항목은 관리자가 직접 DNS 운영에 따른 설정파일을 접근하여 대화식으로 조정할 수 있도록 한다. 즉, DNS의 재시동이나 현재 상태등을 볼 수 있으며 이외에도 존파일의 위치나 로그파일의 위치등 설정파일들의 위치에 대한 설정을 하게 된다.

자원레코드의 관리는 특정 자원레코드에 대한 삭제와 등록, 갱신을 가능하도록 하며, 이에 따라 서명기라는 응용프로그램을 구동시켜 SIG와 NXT 자원레코드의 자동 갱신을 가능하게 한다. 만약 새로운 호스트가 존 파일에 등록이 된다면 그에 해당하는 공개키가 자동으로 생성되어 KEY 자원레코드로 존 파일에 추가되며 이를 입력하여 새로운 존 파일이 생성되게 된다.

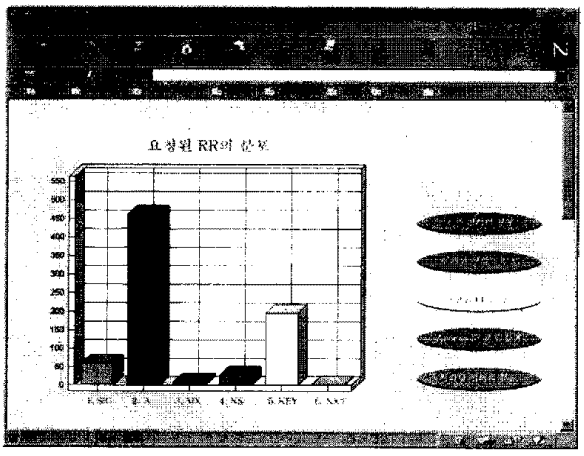
키의 관리는 키 생성기라는 응용프로그램을 이용하여 구현되며 개체의 이름을 입력하여 원하는 알고리즘으로 공개키와 개인키 쌍을 생성하는 기능을 수행한다. 또한 생성된 모든 키는 하나의 파일에 모아서 저장되며 KEY 자원레코드로의 변환 기능도 수행한다. 모아진 키들은 분리될 수 있으며 E-mail이나 파일에 저장이 가능하다.

IP 봉쇄 관리는 특정 IP에 대해 안전한 DNS 서비스를 봉쇄하는 등의 정책을 수행할 수 있도록 지원한다. IP 봉쇄관리 기능은 DNS 서버의 통신모듈을 수정하여 지정된 IP에 대해서 응답과 질의를 봉쇄하는 방법으로 구현된다.

보안 관련 도구의 가장 중요한 요소 중의 하나는 로그의 관리이다. 로그 정보는 안전한 DNS의 운영정책에도 많은 영향을 끼칠 수 있다. 실패한 질의의 분포를 분석하여 접근허가 리스트를 관리할 수도 있으며, 보안사고 이후라도 로그 정보를 분석하여 파급효과를 줄일 수 있다. 또한 이러한 로그는 요청된 자원레코드의 분포등과 같은 통계적 결과를 그래프 형태로 출력할 수 있다.

보통 로그정보는 평균 1초에 5번의 질의가 있을 때 한 개의 로그당 200-500byte가 소요되므로, 엄청난 디스크 용량이 필요하다. 따라서 효율적인 로그관리를 위해서 제한된 로그 정보의 정의가 필요하다. 성공적인 질의에 대한 로그는 질의한 주체의 IP(4byte), 요청한 자원 레코드 종류(4byte), 시간(HH:SS), 성공여부 플래그(1byte)등을 저장한다. 반면 실패한 질의에 대한 로그는 성공적인 질의보다 비교적 많은 정보가 필요하다. 따라서 접근한 주체의 IP(4byte), 목적지의 도메인 네임(255byte), 요청한 자원레코드 종류(3byte), 오류코드(2byte), 시간(HH:SS), 성공여부 플래그(1byte)등이 로그 정보로 필요하다. 보여진 로그정보는 체계적인 관리가 필요하다. 또한 로그 정보를 분석하여 부분적인 DNS 정책에 적용이 가능하다. 일단 보여진 로그파일은 1일 단위로 24시에 파일을 닫는다. 이렇게 닫혀진 파일은 분석기에 의해 다시 열려져 분석된다. 분석기는 전체 질의의 개수, 실패한 질의의 개수, 요청된 자원레코드의 분포, 질의의 시간당 분포, 실패한 질의에 대한 IP 분포와 횟수, 날짜 등을 분석 파일에 저장한다. 분석 작업이 끝난 로그 파일중 성공한 질의에 대한 항목은 디스크의 용량 낭비를 막기 위해 삭제되

시간 실패한 항목은 보안시고 이후에 중요한 자료가 되므로 남겨둔다. 분석기는 1일, 1달, 1년 단위로 분석 파일을 재 분석하여 통계적 정보를 파일에 출력한다.



(그림 15) 요청된 자원레코드의 분포
(Fig. 15) Distribution of Reguested Resource Records

이렇게 관리된 통계정보는 관리자의 요청에 의해 예를 들어 (그림 15)와 같이 그래프 형태로 화면에 출력하거나 파일에 출력할 수 있으며, 부가기능으로 이러한 통계정보에 의해 실패한 질의의 횟수가 많은 호스트의 IP는 DNS에 접근하는 것을 봉쇄할 수도 있다.

로그 정보는 DNS 서버의 내부에서 처리된 결과를 돌려줄 때 가로채어 필요한 정보를 파일에 남기도록 한다. 필요한 정보는 앞서 말한 바와 같이 주체의 IP, 목적지의 도메인 네임, 요청한 자원 레코드 종류, 시간, 성공여부(오류코드)등을 저장하여야 한다.

5. 결 론

본 연구에서는 인터넷에서 가장 중요한 기반구조인 DNS에 보안 서비스를 확장시켜 이를 구현하였으며, 그 외에 부가적으로 필요한 기능 등을 정의하고 이를 설계하였다.

구현된 안전한 DNS는 기본적으로 3가지의 보안 서비스를 제공한다. 첫째로는 인증된 공개키의 저장과 분배이고, 둘째로는 서버에 저장된 데이터의 무결성과 데이터 인증서비스이며, 셋째로는 질의 또는 응답 메시지의 무결성을 제공하는 것이다.

본 논문에서는 이러한 안전한 DNS의 기본 보안 서비스 외에 안전한 DNS가 사용자 인증서를 저장 및 분배하기 위해 PKI내의 인증서 저장소로 사용되고, 편리한 관리자 인터페이스와 로그 기능을 제공하며 또 다양한 암호화 알고리즘이 쉽게 추가될 수 있도록 설계 및 구현한 사항들에 대해 설명하였다.

참 고 문 헌

[1] William Stallings, "Network and Internetwork Security : Principles and Practice," Prentice Hall, 1995.

[2] IETF RFC 2065, "Domain Name System Security Extenstions," 1997.

[3] IETF RFC 1422, "Privacy Enhancement for Internet Electronic Mail : Part II : Certificate-Based Key Management," 1993.

[4] IETF RFC 1424, "Privacy Enhancement for Internet Electronic Mail : Part IV : Key Certification and Related Services," 1993.

[5] IETF Draft, "Internet X.509 Public Key Infrastructure Certificate and CRL profile," 1998.

[6] IETF Draft, "Internet X.509 Public Key Infrastructure Certificate Management Protocols," 1998.

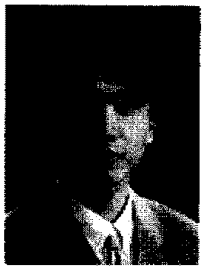
[7] C. Kaufman, R. Perlman, & M. Speciner, "Network Security : Private Communication in a Public World," PTR Prentice Hall, 1995.

[8] IETF RFC 2136, "Dynamic Updates in the Domain Name System," 1997.

[9] IETF Draft, "Domain Name System Security Extensions," 1998, 3.

[10] IETF Draft, "Storing Certificates in the Domain Name System," 1998.

[11] IETF RFC 1995. "Incremental Zone Transfer in DNS," 1996.



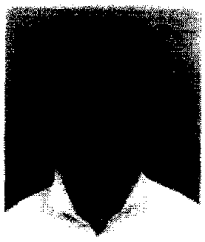
심 희 원

e-mail : hwshim@cs.hongik.ac.kr

1998년 단국대학교 전자계산학과 (학사)

1998년~현재 홍익대학교 전자계산학과 석사과정

관심분야 : 분산 병렬처리, 네트워크 보안, 멀티 캐스팅



김 진 성

e-mail : jskim@cs.hongik.ac.kr

1998년 홍익대학교 컴퓨터공학과(학사)

1998년~현재 홍익대학교 정보공학과 석사과정

관심분야 : 분산 병렬처리, 네트워크 보안, Web

심 영 철

e-mail : shim@cs.hongik.ac.kr

1979년 서울대학교 전자공학과(학사)

1981년 한국과학기술원 전기·전자과(석사)

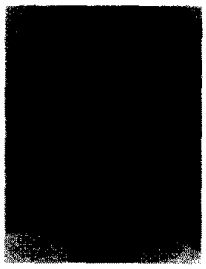
1991년 University of California, Berkeley 전산학(박사)

1981년~1984년 삼성전자 대리

1991년~1993년 University of California, Berkeley 연구원

1993년~현재 홍익대학교 컴퓨터공학과 조교수

관심분야 : 분산 병렬처리, 인터넷 프로토콜, 네트워크 보안, 망관리



임 찬 순

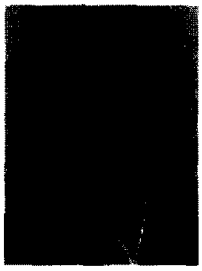
e-mail : csim@garam.kreonet.re.kr

1994년 2월 중앙대학교 전자계산학과 졸업(공학사)

1996년 2월 중앙대학교 컴퓨터공학과 대학원 졸업(공학석사)

1996년 12월~현재 한국전자통신연구원 연구원

관심분야 : 망 관리, 인터넷 보안, 방화벽 시스템



변 옥 환

e-mail : ohbyeon@garam.kreonet.re.kr

1979년 2월 한국항공대학교 통신정보공학과 졸업(공학사)

1985년 2월 인하대학교 전자공학과 대학원 졸업(공학석사)

1993년 2월 경희대학교 전자공학과 대학원 졸업(공학박사)

1978년 9월~1998년 5월 시스템공학연구소

1998년 5월~현재 한국전자통신연구원 책임연구원

관심분야 : 망 관리, SNMP, 인터넷 보안, 방화벽 시스템