인터네트 멀티미디어 전자우편의 보안 처리를 위한 라이브러리 함수 개발

윤 성 순[†] · 윤 기 송[††]

요 약

인터네트의 사용의 급격한 증가와 멀티미디어 정보의 보편화에 따라 이를 이용하는 다양한 전자우편 프로토콜들이 나타나고 있다. 이러한 전자우편의 상용화와 전자우편을 통한 개인정보 및 주요정보 보호를 위해서는 암호화 기법을 이용한 보안기능이 필수적이다. 본 논문에서는 멀티미디어를 지원하는 인터네트 전자우편(MIME; Multi-Purpose Internet Mail Extention)에 보안 기능을 추가한 MOSS(MIME Object Security Service)의 효과적 처리에 필요한 라이브러리 함수개발에 관하여 기술하였다. 이를 위하여 MIME 구조로 나타나는 암호화와 복호화, 전자결재 및 검증등의 보안기능 처리를 위한 MOSS의 특성 및 구조에 관하여 기술하고 이를 실질적으로 개발하기위한 자료구조 및 필요한 함수들을 정의하였으며 개발시 고려하여야 할 사항들에 대하여 기술하였다.

Library Function Development for Internet MOSS(MIME Object Security Service)

Seong Soon Yoon[†] · Ki Song Yoon[††]

ABSTRACT

As Internet and its users grow dramatically and multi-media data are getting common, many types of electronic mail applications are showing up. Internet s/w developers and users would like to use electronic mail system for commercial purposes. They also would like to protect their electronic mail somehow. For these purpose, the security feature using cryptography is one of the essential functions. In this paper, we describe the library function development for handling MOSS(MIME Object Security Service), the security version of MIME (Multi-Purpose Internet Mail Extension). For these purpose, we explained the security features and structures of MOSS and defined the necessary functions and we also discussed design issues for the MOSS implementation.

1. 서 론

최근 인터네트는 세계적으로 그 규모 및 사용자가 급증하고 있으며 다양하고 새로운 응용 프로그램들이 나타나고 있으나, 가장 많이 사용되는 응용 프로그램은 역시 전자우편시스템이라 할 수 있다. 많은 인터네트 사용자들은 전자우편을 통하여 정보를 주고받고 있으며 전자우편을 통하여 업무를 수행하고 있다. 특히 멀티미디어 데이타의 보편화에 따라 전자우편의 활용 범위는 더욱 증대되고 있으며, 이를 지

† 정 회 원:시스템공학연구소 선임기술원
†† 정 회 원:시스템공학연구소 선임연구원
논문접수:1996년 3월 20일, 심사완료:1996년 6월 20일

원하는 멀티미디어 전자우편 역시 보편화되는 추세이다. 이에 따라 개인, 기업체 또는 국가의 주요정보가 전자우편 시스템을 통하여 전달되고 있으며 전자문서도 공식적으로 그 실체를 인정받게 되었다.

이와같이 전자우편을 통한 주요 정보의 전달이 확대되고 전자우편 사용자가 증가하면서 전자우편을 수많은 불특정 인터네트 사용자로부터 보호하고 내용의 변조방지 및 송수신자를 확인하는데 필요한 기밀성, 무결성, 전자결재, 인증, 부인봉쇄와 같은 보안 기능들이 전자우편에도 필요하게되었다. 특히 전자상거래의 활성화에 따라 인터네트를 통한 Ordering System, Payment System, Home Banking System 등 다양한 전자상거래 응용시스템과 이를 지원하는 프로토콜들이 나타나고 있으며, 전자우편은 이러한 전자상거래 응용시스템에 필요한 데이타를 전달하는 직접적인 도구로 사용될 수 있을 뿐 아니라, 전자상거래에 필수적으로 사용되는 사용자, 상인, 금융기관의 공개키의 생성, 송수신, 인증등 Certificate을 이용한 공개키 관리가 필수적이다. 이러한 Certificate 및 공개키 관련 정보 처리는 일반적으로 CA(Certificate Authority)를 중심으로 전자우편을 통하여 이루어진다. 따라서 전자우편은 WWW, Certificate Management와 함께 전자상거래의 하부 구조로서 주요한 역할을 하며 보다 빠르고 효율적이며 강력한 보안기능을 요구하고 있다.

현재 인터네트 환경에서 쓰이고 있는 보안기능이 포함된 전자우편 시스템으로는 인터네트 전자우편의 최초 보안 표준인 PEM(Privacy Enhanced Mail)[11, 12, 13, 14], 멀티미디어를 지원하는 MIME(Multi-Purpose Internet Mail Extension)[6, 7]에 보안기능이 추가된 MOSS(MIME Object Security Service)[2, 3]와 비표준인 PGP(Pretty Good Privacy) 및 S/MIME 등이 있다. PEM은 기본적으로 텍스트만을 지원하고 전자서명 기능을 반드시 포함하나 암호화 기능은 선택적으로 사용할 수 있으며 공개키는 CA를 통한 Certificate을 이용하여 인증한다. PGP는 인터네트 표준은 아니지만 실제로 가장 널리 사용되고 있는 De facto 표준이라 할 수 있다. 멀티미디어를 지원 하지 않으나 최근 PGP와 MIME을 결합한 PGP/MIME이 논의되고 있다[4, 5]. 공개키의 인증은 "Web of trust" 방식에 의해 상호 신뢰할 수 있는 사용자간에 이루어 지므로 공개키 인증의 한계가 있다. S/MIME은 RSA사에서 제안한 프로토콜로서 PEM을 기반으로 MIME을 지원하나 사용하는 암호화기법의 특허문제가 있으며 현재까지 널리 사용되지 않고 있으나 RSA사를 중심으로 진행되고 있다. 이와같이 이들은 키관리, 보안기능, 메세지형식, 특허사용, 효율성에 있어 각각의 특성 및 문제점을 가지고 있으며 IMC(Internet Mail Consortium) 등을 통하여 기능 향상 및 상호 연동문제등에 관하여 활발하게 논의되고 있다. MIME은 인터네트에서 멀티미디어를 지원하는 전자우편 프로토콜로 정착하여 사용되고 있다. 실제로 이러한 인터네트 전자우편의 보안 기능을 이용하여 미국정부의 필요한 물자를 구입에 이용하는 시범 프로젝트들이 진행되고 있고, 이러한 추세에 따라 MIME에 보안기능을 추가한 MOSS사용의 확산이 예상된다. 본 논문에서는 가장 최근의 인터네트 멀티미디어 보안 전자우편의 표준인 MOSS 메세지를 처리하는데 필요한 MOSS 라이브러리 함수 개발에 대하여 기술하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 라이브러리 함수 개발을 위한 MOSS의 특성 및 구조에 대하여 살펴보고, 3 장에서는 MOSS 메세지 처리를 위한 자료구조를 정의하고 각각의 보안 기능과 이를 위해 필요한 데이타 구조를 설정한다. 4 장에서는 MOSS와 보안 모듈에 대한 각각의 라이브러리 함수를 정의하고 5 장에서는 라이브러리 함수를 이용하는 알고리즘과 MOSS 메세지를 보여주며 끝으로 6 장에서는 결론에 관하여 기술한다.

2. MOSS 메세지의 특성 및 구조

2.1 MOSS의 특성

MOSS의 특성은 다음과 같다. 첫째, MOSS는 MIME의 메세지 형식을 가진다. 따라서 MOSS 메세지는 Header와 Body로 구성되어 있고 MIME이 지원하는 모든 멀티미디어 데이타의 암호화 및 전자서명의 보안 기능을 제공한다. 또한 MOSS 메세지 자체가 하나의 Content-Type을 갖는 MIME형식이므로 다른 multipart MIME 메세지의 일부분으로 쓰일 수 있다. 둘째, 암호화와 전자서명이 독립적으로 사용될 수 있다. 따라서 전자서명 MOSS(Signed MOSS)와 암호화 MOSS(Encrypted MOSS)를 비롯하여 전자서명 MOSS의

MOSS 암호화(MOSS Encryption of Signed MOSS Message)와 암호화 MOSS의 전자서명(MOSS Signature of Encrypted MOSS Message)도 사용할 수 있으나 일반적으로 전자서명 MOSS, 암호화 MOSS 및 전자서명 MOSS의 MOSS 암호화가 일반적인 MOSS 사용형태이다. 셋째, MOSS는 공개키 액세스를 위하여 Certificate를 사용하지 않고 전자우편주소(Email Address), 무작위 문자열(Arbitrary String), 사용자 고유이름(Distinguished Name), 공개키(Public key), 공개키 발행자 이름과 공개키의 일련번호(Issuer Name and Serial Number)를 사용한다.

넷째, MOSS 메세지에 보안 기법을 명시하므로 암호화 또는 전자 서명에 필요한 보안 기법이 독립적이다.

2.2 MOSS 메세지의 구조

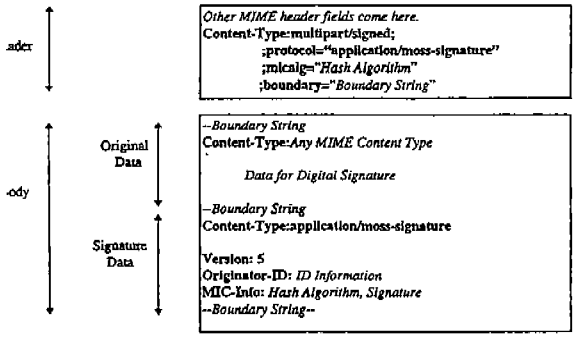
모든 MOSS 메세지는 MIME 형식을 가진다. MOSS의 Content-Type은 multipart이며 전자서명 MOSS와 암호화 MOSS의 두가지의 형식이 있다. 이 두가지의 MOSS 메세지는 항상 두개의 서브 MIME 메세지를 포함한다. 이 두개의 서브MIME 메세지는 Boundary String에 의하여 구분되고 일반 MIME 메세지와 동일한 형식을 갖는다. 즉 Header와 Body를 가진다. 이때 메세지 내부에 Boundary String과 동일한 문자열이 나타나지 않도록 주의 하여야 한다.

2.2.1 전자서명 MOSS 메세지의 구조

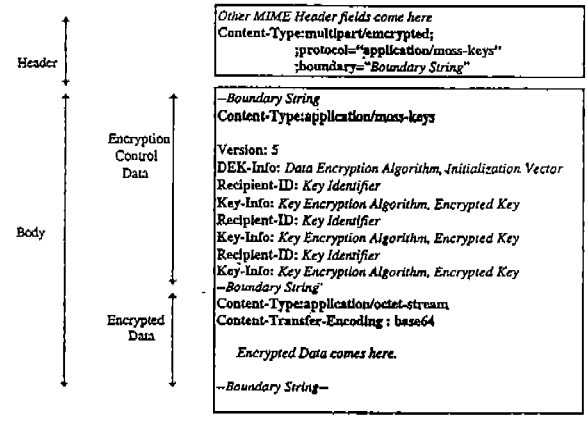
전자서명 MOSS메세지는 항상 Content-Type 값으로 'multipart/signed'를 가지며 parameter로 protocol, micalg, boundary를 반드시 가져야하며 protocol은 항상 'application/moss-signature'이다. 첫째 서브 메세지는 전자서명을 필요로 하는 MIME메세지이며 둘째 서브 메세지의 Header는 반드시 Content Type으로 'application/moss-signature'를 가진다. Body부분은 Header Field의 양식으로나타나며 'Version:' Field는 MOSS Version, 'Originator-ID:' Field는 송신자의 공개키 식별자(Originator's Public Key Identifier), 'MIC-Info' Field는 Hash기법, Hash 암호화 기법과 전자서명의 결과를 각각 나타낸다. 전자서명 MOSS 메세지의 구조는 (그림 1)과 같다.

2.2.2 암호화MOSS 메세지 구조

암호화MOSS 메세지는 항상 Content-Type 값으로 'multipart/encrypted'를 가지며 parameter로 protocol, boundary를 가져야하며 protocol은 항상 'application/moss-keys'이다. 첫째 서브 메세지는 암호화 또는 복호화에 필요한 Control 데이타를 나타내는 MIME메세지이며 Header는 반드시 Content-Type으로 'application/moss-keys'를 가진다. Body부분은 Header Field의 양식으로 나타나며 'Version:' Field는 MOSS Version, 'DEK-Info' Field는 메세지 암호화 기법 및 암호화 초기값(Initialization Vector), 'Recipient-ID:' Field는 수신자의 공개키 식별자, 'Key-Info:' Field는 메세지암호화 키의 암호화 기법 및 메세지 암호화 키의 암호화결과를 각각 나타낸다. 다수의 To:Cc:Bcc:수



(그림 1) 전자서명 MOSS
(Fig. 1) Signed MOSS



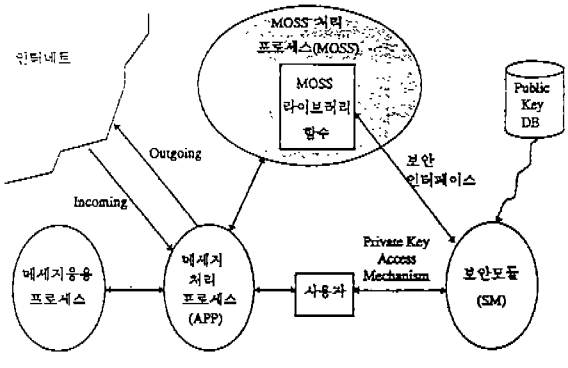
(그림 2) 암호화 MOSS
(Fig. 2) Encrypted MOSS

신자가 있을경우 'Recipient-ID'와 'Key-Info'는 각 수신자들의 공개키정보에 따라 수신자 숫자만큼 나타난다. 둘째 서브메세지의 Header는 반드시 Content Type으로 'application/octet-stream'을 가지며 Body부분은 암호화된 결과의 Encoding된 형식으로 나타난다. 암호화 MOSS의 경우 일반적으로 base64 기법을 사용한다. 암호화 MOSS 메세지의 구조는 (그림 2)와 같다.

3. MOSS 라이브러리 개발 환경

3.1 MOSS 메세지 처리

본 모델에서는 실제 보안관련 기능은 IDUP-GSS-API(Independent Data Unit Protection Generic Security Service API)[1] 또는 GSS-API(Generic Security Service API)[8]등과 같은 인터페이스를 통하여 보안 모듈이 수행하고 보안 모듈은 무작위 문자열 공개키 식별자(Public Key Identifier)에 의해 공개키를 액세스 할 수 있다고 가정한다. 또한 메세지 처리 프로세스를 이용하는 응용프로세스 또는 사용자의 전자서명 및 복호화를 위한 비밀키는 로칼 환경의 Trusted 시스템에 안전하게 저장 및 액세스 할 수 있다고 가정한다. 본 모델에서 메세지처리 프로세스는 MTA (Message Transfer Agent), UA(User Agent), MEA(Message Enabled Applications)등이 될 수 있으며 메세지 응용 프로세스는 WWW, EDI등 다양한 응용 시스템이 주로 CMC(Common Message Call), CGI(Common Gateway Interface)등을 통하여 메세지 처리 시스템을 이용할수 있다((그림 3) 참조).



(그림 3) MOSS 메세지 처리 환경
(Fig. 3) MOSS Message Handling Environment

현재 인터네트의 전자우편 프로토콜은 7bit데이타를 지원하는 SMTP[15, 16]와 8bit데이타를 지원하는 ESMTP[9, 10]가 공존하고 있다. non-7bit 데이타 MOSS 메세지를 인터네트를 통하여 전송하고자 할 때 암호화 MOSS는 모든 non-7bit 데이타가 암호화된후 다시 encoding되어 7bit(ASCII) 데이타 형태로 전달되므로 문제가 없으나. 전자서명 MOSS의 첫째 서브메세지는 8bit, binary를 포함한 MIME의 모든 Content-Type을 가질 수 있으므로 전자우편 전달과정에서 Most Significant Bit이 reset될 경우 수신된 전자서명 MOSS의 전자서명 확인에 문제를 야기시킬 수 있다. 따라서 송신자의 전자서명 MOSS는 전자서명 산출이전에 반드시 7bit으로의 변환이 있어야한다. 이 변환작업은 메세지처리 프로세스 또는 MOSS 처리 프로세스에 의해 실행 될 수 있다. 또한 시스템은 그 종류에 따라 line delimiter는 LF, CRLF, CR, None등과 같이 다양하게 나타난다. 즉 송신자가 사용하는 시스템의 line delimiter와 수신자가 사용하는 시스템의 line delimiter가 다를 수 있다. 따라서 송신자의 전자서명은 표준화된 line delimiter 즉 CRLF로 계산하여야 하고 수신자는 수신된 전자서명 MOSS의 검증을 위하여 모든 로칼 line delimiter는 CRLF로 변환후 검증하여야한다.

3.1.1 Incoming MOSS 메세지 처리

일반적으로 모든 Incoming 메세지의 Input은 MIME 형태 즉 최소한의 Header field 와 Body를 가진다. 암호화 MOSS의 경우 복호화된 데이타의 다음 처리를 위하여 필요한 데이타를 Header field에 갖는다.

- 전자서명 MOSS 처리
  - APP(메세지 처리 프로세스)는 MOSS(MOSS 처리 프로세스)에게 전자서명 MOSS메세지를 전송한다.
  - MOSS는 첫째와 둘째 서브메세지를 분리한다.
  - MOSS는 첫째 서브메세지의 line delimiter를 CRLF로 표준화하여 hash값을 산출하고 둘째 서브메세지에서 암호화되어 있는 hash값을 SM(보안모듈)을 통하여 복호화한 후 두 hash값을 비교하여 그 결과를 APP에게 돌려준다.
- 암호화 MOSS 처리

- APP는 MOSS에게 암호화MOSS메세지를 전송한다.
- MOSS는 첫째와 둘째 서브메세지를 분리한다.
- MOSS는 첫째 서브메세지에서 메세지 암호화기법, 암호화 초기값, 수신자 공개키 식별자, DEK 암호화기법, 암호화된 DEK값을 구하고 둘째 서브메세지의 Body를 Decoding한 후 SM을 통하여 복호화하고 그값을 APP에 돌려준다.

3.1.2 Outgoing MOSS 메세지 처리

일반적으로 Outgoing MOSS 메세지의 Input은 MIME 메세지형식으로 Header와 Body를 가진다.

• 전자서명 MOSS 처리
- APP가 MOSS에게 전자서명을 위한 메세지를 전송한다.
- MOSS는 메세지를 표준화(Canonicalization)시킨다.
  · 전자서명MOSS 메세지가 7bit데이타가 아니면 endoding하고 encoding scheme을 Header field에 추가한다.
  · 메세지의 모든 로칼 line delimiter를 CRLF로 변환한다.
- 7bit으로 변환된 메세지(로칼 line delimiter는 그대로 유지)로 전자서명 MOSS의 첫째 서브메세지를 만든다.
- MOSS는 송신자 공개키 식별자, 표준화된 데이타를 가지고 SM을 부른다.
- SM은 전자서명기법, Hash 기법, 전자서명 값을 돌려준다.
- MOSS는 위의 값으로 전자서명 MOSS의 둘째 서브메세지를 만든다.
- MOSS는 첫째와 둘째 서브메세지로 전자서명 MOSS메세지를 만들어 APP에 돌려준다.

• 암호화 MOSS
- APP가 MOSS에게 전자서명을 위한 메세지를 전송한다.
- MOSS는 수신자ID list와 메세지를 가지고 SM를 부른다.
- SM은 암호화된 메세지값, 암호화기법, 암호화초기값, 메세지암호화키(DEK)의 암호화기법, 암호화된 DEK을 돌려준다.
- MOSS는 암호화된 메세지를 가지고 암호화MOSS의 첫째 서브메세지를 만들고 다른 값으로 둘째 서브메세지를 만든다.
- MOSS는 첫째와 둘째 서브메세지로 암호화MOSS 메세지를 만들어 APP에 돌려준다.

3.2 라이브러리 함수에 필요한 자료구조

MOSS 메세지 처리를 위한 자료구조는 기본적으로 첫째 메세지 처리 프로세스와 MOSS처리 프로세스 간의 데이타 전송을 위한 자료구조와 둘째 MOSS메세지를 처리하기위한 자료구조가 필요하며 셋째 이를 위한 보조 자료구조가 필요하다.

3.2.1 데이타 전달 자료구조

메세지처리 프로세스와 MOSS처리프로세스는 시스템 구조 디자인에 따라 하나의 프로세스에 포함하여 필요한 자료구조의 포인터만을 전달할 수도 있으며 두개의 독립된 프로세스로 하여 로칼 시스템 환경에 맞는 IPC(Interprocess Communication)를 이용할 수도 있다. MOSS처리프로세스가 제공하는 보안기능은 전자서명, 암호화, 전자서명메세지의 암호화, 복호화, 전자서명확인, 복호화 및 전자서명확인등 여섯가지의 기능을 제공한다. 이러한 보안기능을 요구하고 제공하기 위하여는 각 기능별로 공통적인 데이타와 서로다른 데이타가 필요하나 본 연구개발에서는 아래와 같이 하나의 자료구조를 이용하여 데이타를 전달하도록 한다.

```
typedef struct {
  Security_Service_Type   servicetype;
  Sign_Algo               signalgo;
  Hash_Algo               hashalgo;
  Data_Encrypt_Algo       dataencryptalgo;
  Key_Encrypt_Algo        dekencryptalgo;
  Encode_Algo             encodealgo;
  char                    *originatorID;
  char                    **recipientID; /* To, Cc 수신자list */
  char                    **bccrecipientID; /* Bcc수신자 list */
  char                    *input, *output;
  int                     validationresult;
  int                     errorcode;
```

```
} Security_Data;
```

위 자료구조에서 originatorID, recipientID, bccrecipientID는 로칼 시스템 환경에 따라 다섯가지 공개키 식별자(Public Key Identifier; Email Address, Arbitrary String, Distinguished Name, Public Key, Issuer Name and Serial Number)중 하나 또는 여러개를 선택한다. input, output변수는 데이타 전달방식에 따라 버퍼의 포인터 또는 화일명으로 사용할 수 있다. recipientID와 bccrecipient는 To:Cc:와 Bcc:에 해당하는 수신자의 리스트를 각각 나타내며 Outgoing 암호화 MOSS의 경우 별도의 처리가 필요하다. 위 자료구조를 위하여 다음과 같은 보조 자료구조 또한 필요하다.

```
typedef enum {
    SIGN,
    ENCRYPT,
    DECRYPT,
    VALIDATE,
    SINGANDENCRYPT,
    DECRYPTANDVALIDATE
} Security_Service_Type;
typedef enum {
    RSA,
    DSA
} Sign_Algo;
typedef enum {
    MD2,
    MD5
} Hash_Algo;
typedef enum {
    DES_CBC, DES_ECB,
    RC2, RC4,
    FEAL,
    IDEA
} Encrypt_Algo;
typedef enum {
    RSA,
    DSA
} Key_Encrypt_Algo;
typedef enum {
    BASE64,
    QUOTED_PRINTABLE
} Encode_Algo;
```

3.2 MOSS 처리를 위한 자료구조

Incoming MOSS메세지의 분석과 Outgoing메세지를 작성하기 위하여 자료구조가 필요하다. 모든 MOSS 메세지는 다음과 같은 특성을 가지고 있다.

- MIME 형식을 갖는다.
- multipart content-Type을 가진다.
- Boundary String이 있다.
- 항상 두개의 서브 메세지를 가지며 Boundary String에의해 구분되고 Body에는 Boundary String이 나타나지 않아야한다.
- 서브 메세지는 MIME 형식으로 header와 Body를 갖는다. Header가 없을 경우는 text/plain으로 간주한다.
- 전자서명 MOSS와 암호화 MOSS의 Control Data를 갖는 서브메세지의 Body는 아래의 두가지 예와 같이 Header 형식(Field Name ":"Field Value)을 갖는다.

```
Version:5
DEK-Info:DES-CBC,D488AAAE271C8159
Recipient-ID:EN,2,galvin@tis.com
Key-Info:RSA,ISbC3IR01BrYq2rp493X+Dt7WrVq3V3/U/YXbxOTY5cmiy1/=
7NvSqqXSK/WZq051N99RDUQhdNxXI64ePA0FWQ6RGoiCrRs+Dc95oQh7EFEPoT=
9P6jyzcV1NzZVwfp+u

Version:5
Originator-ID:PK,MHkwCgYEVQgBAQICAwADawAwaAJhAMAHQ45ywA357G4f=
qQ61aoC1fO6BekJmG4475mJkwGIUxvDkwuxc/EFdPkXDGBxzdGrWiuh5K8k18=
KRGJ9wh1HU4TrghGdlm0Lw8gG67Dmb5cBhY9DGwq0CDnrpKZV3cQIDAQAB,EN,=
2,galvin@tis.com
MIC-Info:RSA-MD5,RSA,PnEvyFV3sSyTSiGh/HFgWUIFa22jbHoTrFIMVERf=
MZXUKzFsHbmKdowJ1JR56Oolmo+t7WjRfzpMH7MOKgPgzRnTwk0T5dOcP/1fb=
sOVJj1eV7vTe9yoNp2P8mi/hs7
```

이러한 Nest 특성을 이용하여 MOSS 메세지는 다음과 같은 자료구조를 이용하여 나타낸다.

```
typedef struct {
    Header          *header;
    Body            *body;
    MOSS_Struct     *child;
    MOSS_Struct     *next;
} MOSS_Struct;

typedef struct {
    Header_Type     *name;
    char            *value;
    Parameter       *param;
    Header          *next;
} Header;

typedef struct {
    Parameter_Type  paramtype;
    char            *paramvalue;
    Parameter       *next;
} Parameter;

typedef struct {
    Content_Type        contenttype;
    content_Encoding    contentencoding;
    int                 length;
    void                *value;
} Body;

tyepdef enum
    /* SMTP, MIME, MOSS에서 정의 하는 모든 Header Field */
    To, Cc, Bcc, From, Subject, Date, Message_ID, Received, Sender,
    Return_Path, MIME_Version, Content_Type, Content_Type_Encoding,
    Content_ID, Content_Length, Apparently_To, Apparently_From,
    Resent_From, Reply_To, Sesent_To, Resent_Reply_To,
        ....., others
} Heder_Type;

tyepdef enum {
    /* MIME, MOSS에서 지원하는 모든 Content Type & Subtype */
    text_plain, text_richtext,
    multipart_mixed, multipart_alternative, multipart_parallel,
    multipart_digest, multipart_encrypted, multipart_signed,
    application_octet_stream, application_postscript,
    application_moss_keys, application_moss_signature,
    message_rfc822, message_partial, message_external_body,
    image_jpeg, image_gif,
    audio_basic,
    video_mpeg
        ....., others
} Content_Type;

typedef enum
    /* MIME, MOSS에서 지원하는 모든 Parameter Type */
    charset, mode, size, padding,
    protocol, micalg, boundary,
        ....., others
} Parameter_Type;
```

4. 라이브러리 함수

4.1 MOSS 관련 함수

4.1.1 Incoming MOSS

Moss_To_MOSS_Struct (char *input, MOSS_Struct *mossstruct)

위 함수는 MOSS 메세지를 분석하여 MOSS_Struct 구조로 변환한다.

- input은 Input buffer pointer 또는 Input 화일 이름
- mossstruct는 Input MOSS 메세지의 MOSS_Struct 구조

Get_Header (MOSS_Struct *mossstruct, Header_Type headertype, char *string)

위 함수는 MOSS 메세지에서 필요한 Header field 값을 돌려준다.

- mossstruct는 Input MOSS 메세지의 MOSS_Struct 구조
- headertype은 필요한 Header field 이름
- string은 지정한 Header field의 값을 나타낸다.

Get_Parameter (Header *header, Header_Type headertype, char *string)

위 함수는 주어진 Header field에서 필요한 Parameter값을 돌려준다.

- header는 MOSS 메세지내의 특정한 Header
- parametertype은 필요한 Parameter field 이름

- string은 Parameter field의 값을 나타낸다.

4.1.2 Outgoing MOSS 메세지 관련 함수

Make_MOSS_Struct (MOSS_Struct *mossstruct)

위 함수는 MOSS_Struct 구조를 만든다.

Add_Header (Header *header, Header_Type header-type, char *string)

위 함수는 MOSS_Struct 구조에 특정한 값을 가진 Header field를 추가한다.

- header는 MOSS_Struct 구조의 Header list의 시작점을 가르킨다.
- headertype은 추가하고자하는 Header field의 종류
- string은 추가하고자하는 Header field의 값을 가진다.

Add_Parameter (struct Parameter *parameter Parameter_Type parametertype, char *string)

위 함수는 MOSS_Struct 구조의 특정 Header field에 특정한 값을 가진 Parameter field를 추가한다.

- parameter는 MOSS_Struct 구조의 특정 Header field의 특정 Parameter list의 시작점을 나타낸다.
- paramertype은 Parameter field의 종류를 나타낸다.
- string은 parameter field의 값을 나타낸다.

Add_Body (MOSS_Struct *mossstruct, Body *body)

위 함수는 MOSS_Struct 구조에 Body 구조를 연결시킨다.

MOSS_Struct_To_MIME (MOSS_Struct *mossstruct, char *output)

위 함수는 MOSS_Struct 구조를 MOSS 메세지로 변환한다. 모든 Header 리스트에 있는 Header 정보를 이용하여 Header를 구성하고 Body 구조의 정보에 따라 Encoding하고 그 Encoding 기법을 Header에 추가한다.

- mossstruct는 두개의 서브메세지의 MOSS_Struct를 포함하는 MOSS_Struct
- output은 Outgoing MOSS 메세지

Boundary_String_Generation (char *input)

위 함수는 input에 나타나지 않는 임의의 문자열을 돌려준다. 이는 두 MOSS 서브메세지를 분리하는데 사용된다.

4.1.3 기 타

MOSS_Struct_Free (MOSS_Struct *mossstruct)

위 함수는 MOSS_Struct내의 모든 할당된 메모리를 시스템으로 환원 시킨다.

- mossstruct는 특정 MOSS_Struct 구조를 나타낸다.

Base64 (int flag, char *in, *out)

위 함수는 주어진 데이타를 BASE64 기법으로 Encoding 또는 Decoding 한다.

- flag은 Encoding 또는 Decoding을 나타낸다.
- in, out은 Input과 Output을 각각 나타낸다.

Quoted_Printable (in flag, char *in, out)

위 함수는 주어진 데이타를 Quoted Printable 기법으로 Encoding 또는 Decoding 한다.

- flag은 Encoding 또는 Decoding을 나타낸다.
- in, out은 Input과 Output을 각각 나타낸다.

4.2 보안 기능 관련 함수

4.2.1 Incoming

Corresponding_Recipient (mossstruct, user, recipientID)

위 함수는 Incoming 암호화 MOSS 메세지에서 수신자가 다수인 경우 복호화를 위한 해당 수신자의 recipientID를 사용자 데이타를 이용하여 찾는다.

Input

- mossstruct Incoming dkaghghk MOSS메세지의 MOSS_Struct 구조를 갖는다.
- user는 사용자 데이타

Output

- recipientID해당 수신자의 공개키 식별자

Decrypt (mossstruct, recipientID, dataencryptalgo, dekencryptalgo, initialvector, encrypteddek, datafordecryption, decrypeddata)

위 함수는 Incoming 암호화 MOSS 메세지에서 복호화에 필요한 데이타를 MOSS_Struct구조에서 찾아 보안모듈을 이용하여 복호화한 메세지를 돌려준다.

Input

- mossstruct는 Incoming MOSS 메세지의 MOSS_Struct 구조를 갖는다.
- recipientID는 수신자의 비밀키를 액세스할 수 있는 Key Identifier 암호화 MOSS 메세지는 To, Cc, Bcc등 다수의 수신자를 위한 RecipientID와 Key-INFO 데이타를 가지고 있다. 암호화 MOSS 메세지처리는 그중 하나의 수신자만을 처리한다. 따라서 전자우편 처리 프로세스는 지정된 수신자 정보를 MOSS 처리 프로세스에게 알려주어야 한다.
- dataencryptalgo는 메세지암호화 기법
- dekencryptalgo는 메세지암호화키 암호화 기법
- initialvector는 메세지 암호화에 필요한 초기 값
- encrypteddek는 암호화된 메세지 암호화 키 값
- datafordecryption는 암호화된 메세지

output

- decrypteddata는 복호화된 메세지

Incoming_Canonicalize(dataforsignature, canonicalized-data)

위 함수는 Incoming 전자서명 MOSS 메세지에서 전자서명된 부분을 추출하여 로칼 Line Delimiter를 모두 CRLF 형태로 변형시킨다.

Input

- dataforsignature는 전자서명 MOSS 메세지의 전자서명된 메세지를 나타낸다.

Output

- canonicalizeddata는 line delimiter가 모두 CRLF 형태로 변형된 결과.

4.2.2 Outgoing

Outgoing_Canonicalize(dataforsignature, canonicalized-data)

위 함수는 MOSS라이브러리를 이용하는 메세지 처리 프로세스로 부터 받은 Input 데이타(Appdata)에서 전자서명을 위한 메세지를 추출하여 첫째 모든 데이타가 7bit 데이타여부를 확인하고 non-7bit (8bit 또는 binary) 데이타의 경우 메세지 성격에 따라 base64 또는 Quoted Printable 방식으로 Encoding하고 그 방식을 메세지 Header에 추가하여 모든 전자서명을 위한 데이타를 7bit으로 만든다. 둘째 전자서명을 위한 메세지의 로칼 Line Delimiter를 모두 CRLF 형태로 변형시킨다.

Input

- dataforsignature는 전자서명 할 메세지를 나타낸다.

Output

- canonicalizeddata는 Line Delimiter가 모두 CRLF 형태로 변형된 7bit 데이타

Sign (Appdata, canonicalizeddata, originatorID, hashalgo, signalgo, signature)

위 함수는 Outgoing MOSS 메세지를 만들기위하여 전자서명확인에 필요한 데이타를 APPdata에서 찾아 보안모듈을 이용하여 canonicalize된 메세지의 signature(암호화된 Hash)를 돌려준다.

Input

- Appdata 는 메세지 처리 프로세스로 부터 받은 Input 데이타
- canonicalizeddata는 Line Delimiter가 모두 CRLF형태로 변형된 7bit 데이타
- originatorID는 송신자의 비밀키를 액세스할 수 있는 Key Identifier(본 논문에서는 전자서명을 위한 송신자의 비밀키는 Trusted System에 의해 안전하게 관리되고 있다고 가정한다.)
- hashalgo는 canonicalize된 메세지의 Hash기법
- signalgo는 Hash결과를 암호화하는 기법

Output

- signature는 암호화된 Hash

Bcc_Handling (recipientID, bccrecipientID)

위 함수는 bcc 수신자를 위한 암호화 MOSS 메세지를 처리한다. 암호화 MOSS 메세지는 모든 To, Cc 수신자의 Recipient-ID 및 Key-Info 데이타를 포함하고 Bcc 수신자의 Recipient-ID 및 Key-Info데이타는 해당 Bcc수신자 이외의 To, Cc 그리고 타 Bcc수신자의 MOSS 메세지에는 나타나서는 안된다. 따라서 To, Cc 수신자는 모두 동일한 암호화 메세지를 받게되고 모든 Bcc 수신자는 각각 다른 암호화 MOSS메세지를 받는다. 이함수는 내부에서 encrypt ()를 부른다.

Input

- recipientID는 To, Cc 수신자 list
- bccrecipient는 Bcc 수신자 list

Encrypt (APPdata, recipientID, dataencryptalgo, dekencryptalgo, initial_vector, encrypteddek, dataforencryption, encrypteddata)

위 함수는 Outgoing MOSS 메세지를 만들기위하여 전자서명확인에 필요한 데이타를 APPdata에서 찾아 보안모듈을 이용하여 암호화한 메세지를 돌려준다.

Input

- APPdata는 메세지 처리 프로세서로 부터 받은 Input 데이타
- recipientID는 수신자의 공개키를 액세스할 수 있는 Key Identifier
- dataencryptalgo는 메세지암호화 기법
- dekencryptalgo는 메세지암호화키 암호화 기법
- initialvector는 메세지 암호화에 필요한 초기 값
- encrypteddek는 암호화된 메세지 암호화 키값
- dataforencryption는 Input 메세지

Output

- encrypteddata는 암호화된 메세지

5. MOSS 라이브러리 함수 사용 알고리즘

5.1 MOSS 처리 Procedure

```
Procedure Incoming_MOSS (MOSS_Struct)
{
  Security Service Type 확인
  MOSS_To_Moss_Struct()
  Get_Header(content_type)
  if (content_type == multipart/signed) {
    Get_Parameter()          /* Check correct parameters. */
    Get the first subbody for signed data
    Incoming_Canonicalize()
    Corresponding_Recipient()  /* Find sender's public key id. */
    get the second body for signature control data
    Quoted_Printable()       /* Decode the data. */
    Base64()                 /* Decode digital signature */
    Validate()
    return }

if (content_type == multipart/encrypted) {
  Get_Parameter()            /* Check correct parameters. */
  get the first body for encryption control data
  Quoted_Printable()         /* Decode the data. */
  Base64()                   /* Decode DEK */
  get the recipient's private key identifier for decryption
  get the second body for encrypted data
  Base64()                   /* Decode the data. */
  Decrypt()
  return }
}
Procedure Outgoin_MOSS (MOSS_Struct)
{
  Check Security Service Type
  if (SIGN) {
    Make_MOSS_Struct()       /* Create new MOSS_Struct. */
    Add_Header()             /* Add proper headers */
    Add_Parameter()          /* Add Proper parameters */
    Add_Body()               /* Add the data body */
    Outgoing_Canonicalize()
    Sign()                   /* Get the digital signature */
    Base64()                 /* Encode the signature */
    Add_Header()             /* Add proper headers */
    Construct control body
    Quoted_Printable()       /* Encode the whole control body */
    Add_Body()               /* Add thee control body */
    Boundary_String Generation()
    MOSS_Struct_To_MOSS()
    return }

if (ENCRYPT) {
  Make_MOSS_Struct()         /* Create new MOSS_Struct. */
  Add_Header()               /* Add proper headers */
  Add_Parameter()            /* Add Proper parameters */
  Encrypt()                  /* Encrypt data */
  Base64()                   /* Encode the encrypted dek */
  Add_Header()               /* Add proper headers */
  Construct control body
  Quoted_Printable()         /* Encode the whole control body */
  Add_Body()                 /* Add the control body */
  Construct encrypted data body
  Add_Header()               /* Add proper headers */
  Base64()                   /* Encode the encrypted data */
  Add_Body()
```

Boundary_String_Generation()
MOSS_Struct_To_MOSS()
return }

5.2 MOSS 메세지

본 논문에서 제시한 라이브러리를 이용하여 Tanm 컴퓨터 플랫폼에서 MOSS 처리 시스템을 구현였다. 암호화 기법으로는 기존의 데이타 암호화 기인 DES 및 RSA 기법을 이용하여 구현하였으나 OSS는 다양한 암호화, 전자서명 및 Hash기법을 수 할 수있도록 구성되어 있으므로 향후 보다 강력한 호화 기법을 사용할수 있다.

본 논문에서 제시한 라이브러리 함수를 이용하여 현한 전자우편의 보안 처리시스템을 통하여 위

27K JPEG Image를 안전하게 전송하기위한 전자서명MOSS와 암호화MOSS메세지를 아래와 같이 구성하였다.

5.2.1 전자서명MOSS

```
To: Kildong Hong
Subject: An example of Signed Image
MIME-Version: 1.0
Content-Type: multipart/signed;
 protocol="application/moss-signature";
 micalg="rsa-md5"; boundary="Signed Boundary"

--Signed Boundary
Content-Type: image/jpg
Content-Transfer-Encoding: base64

/9j/4AAQSkZJRgABAgEAZABkAAD/7QE0UGhvdG9zaG9wIDMuMAA4QklNA+0AAAAAABAAZAAAAAEA
AgBkAAAAAQACOEJJTQPzAAAAAAAIAAAAAAAAAAA4QklNJxAAAAAAAAoAAQAAAAAAAAACOEJJTQP1
AAAAAABIAC9mZgABAGxmZgAGAAAAAAABAC9mZgABAKGZmgAGAAAAAAABADIAAAABAFoAAAAGAAAA
AAABADUAAAABAC0AAAAGAAAAAAABOEJJTQP4AAAAAABwAAD/////////////////////////////

    . . . . . (중 략) . . . . . .

/wDBKn+tRr6cf2llpJJdc/bI/wAFOk/TnhM/7Zu190e0blkpJIdX9bn/AAe7+1Kg77XOuzz+ks1J
JTofrMa7I/tJ/wBa/kf9JZySSm+ftHfZH9pMPX/kROkygKSSm4fWnXZPblJ3ra/Q84lUOklNn9NO
seXKcevpET81VSSU3WevGmzy5U2/adv5neJ3LPSSU6lP2qdPT5E7t3/fE5+2btPTnWI3LKSSU2P0
vl/O/wDT/wDSaSrpJKf/2Q==

--Signed Boundary
Content-Type: application/moss-signature
Content-Transfer-Encoding: quoted-printable

Version: 5
Originator-ID: STR,2,This is my string for key identifier.
MIC-Info:RSA-D5,RSA,XUKzFIKgPgzRwJlJRMZbmKtIo56OoMVERfoTrFVRo+t7WFaH2OjfzpM=
H7M3sSyTSGhs7vTe2P8mi/h/HFgWUnTwkPnEvImsHIy/1fb92jbs7F0T5dOcPiOVJjleVyoNp

--Signed Boundary--
```

5.2.2 암호화MOSS

```
To: Kildong Hong
Subject: An example of Encrypted Image
MIME-Version: 1.0
Content-Type: multipart/encrypted;
 protocol="application/moss-keys";
 boundary="Encrypted Boundary"

--Encrypted Boundary
Content-Type: application/moss-keys
Content-Transfer-Encoding: quoted-printable

Version: 5
DEK-Info: DES-CBC
Recipient-ID: STR,3, This is a string for your key identifier.
Key-Info: RSA,9t7WrVIS3I3X+DRU/Yrp4q35c01BrbCYq23/miy1/VXbxOTYSK/64ePWZqop+=
cVwz995oEPfu1N0QRs9Rhd7NbNA5SXiqqXvIxDU9DNFWQEFV67oTQh+iCrcRGzZyP6j

--Encrypted Boundary
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

YRzJrPBmkj8pfUh7N0F6rpHyfgZQu0gtg/6hJ5HZ3kVMFnakPJqgMucsUxRZBeU9Uwd2VlLTjMrn
J+6FWK+YF2sl/toKQFGQJ5LiRbo8BNIawiXNsdlwEOcjlmykfOWdm/tq7ISNHlLldjvkecjeMQ5K
c7D46mKPrIDyBVxPwJ7EQ9JSPk4H5H4DN0CGEGDkpVsmvmfukQY4ejhvsDz+uilkVsrctUSNUT8j
WuCZVuAf/p5vL5XItRLlYx5HjbOGvAQywze3GFtNNeGWcBqfxlPTx8ApWjPBddA/cEDjWJl0cvhF

. . . . . (중 략) . . . . . .

fA+nLTfp0wX+Iq8Z7m9D2mHVP0BByejTaCeqhdLehN9Fhkrox3LiTl10XN7cdyieUShRhqvxnhiT
AdZTiECDTeOa94KQOvZbwzmHg6dZzR3E/yr5xn8zjfN96i/Mx2hl71pdrVWeOEbVjCLk2c3Js09X
5SOzS0R1mWOeB0SJrtsdecW4nu/nH6W+igwfSrNviUSVL/L6bB/xMD8pdzws7XanuzLvx1uqNwGJ
R2eRijLD+SYnCDgSZbkDu8f9pXAiWPOX860StDcXK0cf488/2997iOCH811oLGPY

--Encrypted Boundary--
```

6. 결 론

본 연구개발은 인터네트 멀티미디어 보안 전자우편 표준인 MOSS 메세지 처리를 위하여 MOSS의 특성을 분석하여 이를 처리하는데 필요한 기본적인 라이브러리 함수 개발에 대하여 기술하였다. 인터네트의 발전에 따라 전자우편 시스템의 중요성이 증대하고 있으며 전자우편 보안 표준화에 대하여 많은 논의가 진행되고 있다. 본 연구개발을 이용하여 인터네트의 보안 관련 표준을 신속하게 개발함으로서 표준의 문제점을 찾아내고 개발결과의 성능을 지속적으로 향상시켜 전자상거래를 비롯한 다양한 응용 프로그램들과 연계하여 사용할 수 있는 방안을 검토하고자 한다. 기본적으로 암호화를 통한 전산망 보안 기술은 전산망이 발전 함에따라 국가적, 상업적 또는 개인적으로 반드시 필요한 기술이므로 본 연구 개발을 통하여 보안 기능을 요구하는 다양한 전자우편 처리뿐 아니라 보안기술을 요구하는 타 전산망 분야에 보다 효율적으로 적용하여 궁극적으로 안정적인 전산망 사용에 기여하고자 한다.

참 고 문 헌

[1] C. Adams, "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)" Internet Draft draft-ietf-cat-idup-gss-04.txt, Feb. 1996.

[2] J. Galvin, S. Murphy, S. Crocker, N. Freed, "Security Multiparts for MIME:Multipart/Signed and Multipart/Encrypted," RFC1847, Oct. 1995.

[3] S. Crocker, N. Freed, J. Galvin, S. Murphy, "MIME Object Security Services," RFC1848,

Oct. 1995.

[4] Michael Elkins, "MIME Security with Pretty Good Privacy (PGP)," INTERNET DRAFT draft-elkins-pem-pgp-02.txt, November 1995.

[5] Kazuhiko Yamamoto, "PGP MIME Integration" Internet-Draft draft-kazu-pap-mime-00.txt, Oct. 1995.

[6] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," RFC1521, Sep. 1993.

[7] K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text," RFC1522, Sep. 1993.

[8] J. Linn, "Generic Security Service Application Program Interface," RFC1508, Sep. 1993.

[9] J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker, "SMTP Service Extensions," RFC1425, Feb. 1993.

[10] J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker, "SMTP Service Extension for 8bit-MIME-transport," RFC1426, Feb. 1993.

[11] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures," RFC1421, Feb. 1993.

[12] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management," RFC1422, Feb. 1993.

[13] D. Balenson, "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers," RFC1423, Feb. 1993.

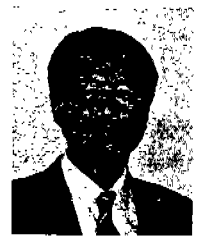
[14] B. Kaliski, "Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services," RFC1424, Feb. 1993.

[15] J. Postel, "Simple Mail Transfer protocol," RFC-821, Aug., 1982.

[16] D. Crocker, "Standard for the format of ARPA Internet text messages," RFC822, Aug., 1982.

윤 성 순

1986년 경북대학교 전자공학과 졸업(학사)
1986년~1994년 대우통신(주) 종합연구소 선임연구원
1995년~현재 시스템공학연구소 선임기술원
관심분야: 네트워크, 시큐리티, 데이타베이스



윤 기 송

1984년 부산대학교 조선공학과 졸업(학사)
1988년 New York City University 대학원 전산학(공학석사)
1993년 New York City University 대학원 전산학(공학박사)
1993년~현재 시스템공학연구소 선임연구원
관심분야: 컴퓨터 네트워크, 컴퓨터 보안, 병렬처리, 분산처리