타원곡선 암호 시스템을 이용한 보안 메일 시스템의 설계 및 구현

(Design and Implementation of a Secure E-Mail System using Elliptic Curve Cryptosystem)

이 원 구 † 김 성 준 †† 이 희 규 †† 문 기 영 ††† 이 재 광 ††††
(Won-Goo Lee) (Sung-Jun Kim) (Hee-Gyu Lee) (Ki-Young Mun) (Jae-kwang Lee)

요 약 컴퓨터와 네트워크의 보급이 일반화되면서 인터넷을 통한 정보 전달이 일상 생활처럼 되고 있다. 또한 인터넷, 무선통신, 그리고 자료교환에 대한 증가로 인해 다른 사용자와 접속하기 위한 방식은 빠르게 변화하고 있다. 그러나 이러한 전자메일에도 많은 문제가 존재한다. 기존의 전자메일은 간단한 방법으로 내용을 열람하거나 변조할 수 있어 중요한 정보나 사생활 노출의 위험에서 벗어날 수 없다. 이러한 데이타에 대한 보안이 기대에 미치지 못하고 있기 때문에 암호학적으로 강력한 전자메일 시스템의 개발이 시급하다. 본 논문에서는 기본적인 정보보호 서비스 외에 기존의 전자메일 시스템에서는 제공되지 않는 배달 증명 및 내용 증명 기능을 제공하고 자바 암호 API를 사용하여 안전한 키 교환이 가능하도록 하였다.

키워드 : 타원곡선, 타원곡선 암호시스템, 보안 메일, 메일 시스템, 전자우편

Abstract As computers and networks become popular, distributing information on the Internet is common in our daily life. Also, the explosion of the Internet, of wireless digital communication and data exchange on Internet has rapidly changed the way we connect with other people. But secure mail is gaining popularity abroad and domestically because of their nature of providing security. That is. it has been used a variety of fields such as general mail and e-mail for advertisement. But, As the data transmitted on network can be easily opened or forged with simple operations. Most of existing e-mail system don't have any security on the transmitted information. Thus, security mail system need to provide security including message encryption, content integrity, message origin authentication, and non-repudiation. In this paper, we design and implement secure mail system with secure key agreement algorithm, non-repudiation service, and encryption capability to provide services for certification of delivery and certification of content as well as the basic security services.

Key words : Elliptic Curve, Elliptic Curve Cryptosystem, Secure Mail, Mail System. E-Mail

1. 서 론

기존의 전자메일 시스템은 기본 표현방식으로 MIME (Multipurpose Internet Mail Extensions)을 사용하고 있어 악의적인 메일서버 관리자라면 해당 메일서버를 사용하는 모든 사람의 프라이버시를 획득할 수 있으며, 또한 시스템 침투를 통하지 않더라도 일반 해커라면 스니퍼(sniffer) 프로그램 등을 통해 해당 내용을 알 수 있게 된다. 기존 전자메일 시스템의 또 다른 취약점으로는 대표적인 전송 프로토콜인 SMTP와 POP3 프로토콜의 취약점으로 이들 프로토콜은 데이타에 어떠한 처리도 하지 않으므로 전자메일 데이타를 그대로 노출시켜 전자메일 데이타를 가로채기만 하면 내용이 편집되는 위협에서 벗어날 수 없다. 따라서 본 논문에서는 기존의 전자메일 시스템이 가지고 있던 근본적인 보안 취약성을 보완하여 안전하고 신뢰할 수 있는 전자메일 시스템

† 학생회원 : 한남대학교 컴퓨터공학과
wglee@netwk.hannam.ac.kr
†† 비 회 원 : 한남대학교 컴퓨터공학과
sjkim@netwk.hannam.ac.kr
june@netwk.hannam.ac.kr
††† 비 회 원 : 한국전자통신연구원 능동보안기술연구팀 연구원
kymoon@etri.re.kr
†††† 비 회 원 : 한남대학교 컴퓨터공학과 교수
jklee@netwk.hannam.ac.kr
논문접수 : 2001년 9월 15일
심사완료 : 2002년 5월 8일

을 개발하고자 한다. 이에 본 논문에서는 기존의 암호화 알고리즘보다 암호학적 강도가 우수한 타원곡선 암호 알고리즘 이용하여 암호화 및 복호화, 그리고 인증 및 서명을 구현하였고, 이러한 암호 알고리즘을 이용하여 키 교환부터 발신자 인증 및 서명을 구현하게 되었다.

2. 관련 연구

인터넷에서는 전자 메일 시스템에서의 정보보호에 대한 취약성을 보완하기 위하여 1993년 2월에 IETF (Internet Engineering Task Force)의 PEM(Privacy Enhanced Mail) WG와 IRTF(Internet Research Task Force)의 공동연구로 PEM에 대한 RFC 문서를 발표하였다. 여기에는 암호화 기법을 기반으로 하는 키 관리 및 분배에 대해서 기술되어 있으며 메시지 기밀성, 메시지 무결성, 송신자 신분 인증, 송신 부인 방지 등의 정보보호 서비스를 제공하고 있다.

또한 PGP(Pretty Good Privacy)는 Philip Zimmermann을 중심으로 한 인터넷상의 지원자들이 만든 제품으로 대칭형 암호화 알고리즘으로 DES를 사용하지 않고 IDEA/RSA를 사용한다는 것과, RSA Public Key 인증구조가 PEM 등 국제표준과는 달리 사이버 세상의 문화와 밀접하게 비슷한 Trust of Web이라는 구조를 가지고 있는 특징이었다.

PEM은 CA(Certificate Authority) 등을 요구하며 그 사양이 방대하고 복잡해 현실적으로 사용되지 못하고 몇 가지 형태로 단순화 되어갔다. 그 가운데 대표적인 것이

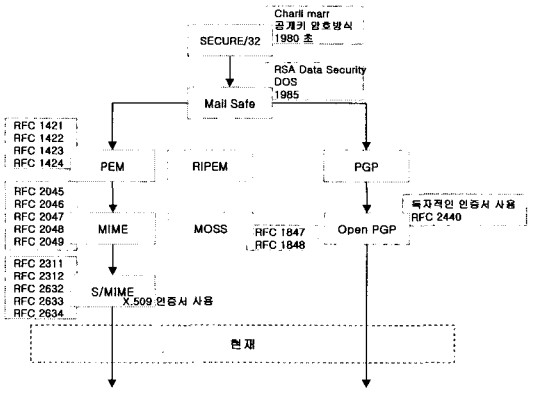


그림 1 전자 메일 프로토콜 현황

바로 RIPEM(Riordan's Internet Privacy Enhanced Mail)이다. RIPEM은 PEM의 복잡한 사양을 단순화했으며 구현사례도 많이 나와 상당히 보급되기도 했다. 그러다가 PEM, RIPEM의 복잡성을 단순하게 만들기 위해 메일사양보다는 문서의 양식 (MIME)만을 정의한 MOSS (MIME Object Security Service)가 제안되었으나 MOSS 사양은 너무 유동적이어서 이 사양만으로는 호환성을 유지하기가 쉽지 않다. 최근 미국 RSA에서 제안한 S/MIME은 MOSS와 비슷하지만 상호 호환성에 초점을 맞춘 사양을 제안한 전자메일 보안 프로토콜로서 마이크로소프트의 Exchange, Eudora, Netscape 등의 전자메일 시스템에서 지원된다.

표 1 메시지 보안 프로토콜 비교

	PEM	S/MIME	PGP	MOSS
제안자	IEFT, IRTF	RSA Data Security	Phil Zimmerman	IETF
발표 년도	1993	1995	1991, 1994	1995
표준 문서	RFC 1421~4	MIME (RFC 1521)에 PKCS #7	적용 없음	RFC1848
제공 서비스	기본서비스	기본서비스	기본서비스	기본서비스
상호 연동성	없음	없음	없음	없음
인코딩 방식	Character	ASN.1/Base64	Character	Character
X.509 사용 여부	사용함	사용함	사용 안함	사용 안함
특 징	-최초의 메시지 보안 프로토콜(80년대 후반) -공개키 방식만 구현 -메시지 처리 방식은 표준화되지 않음 -높은 보안성을 제공하나 구현이 복잡함	-기본 시스템은 PKCS#7 채택 -MIME과 호환되지 않음(PKCS# 10채택으로 해결) -기존 메시지 처리프로토콜에 보안서비스 구현 -해커가 암호화된 메시지의 서명과 암호화된 메시지 구분 가능 -다양한 상호개발 툴킷 제공 -표준을 따르며, 널리 사용되는 메일 에이전트에 탑재되어 있으나 미국의 수출제한법으로 미국 외에서는 56 비트까지 만을 사용가능	-별도의 라이센스가 없음 -PEM,MOSS,S/MIME과 기능적으로 유사 -다른 시스템과 키관리 체계가 호환성이 전혀 없음 -PEM 에 비해 보안성이 취약하나 현이 용이 -사용자 서로간에 증 가능 -표준방식을 지원하지 않으므로 공개키 기반구조(PKI)에 적합하지 않음	-공개키 관련 정보 처리를 위한 메시지 내용의 형태를 정의함 -공개키 쌍 관리를 위한 표준은 채택하지 않음

ASN.1(Abstract Syntax Notation One) IRTF (Internet Research Task Force)
IETF (Internet Engineering Task Force) RSA(Rivest-Shamir-Adleman)

S/MIME은 그 우수한 확장기능으로 인해 PGP와 더불어 전세계의 인터넷 메일 보안을 이끌어 갈 대표적인 보안 프로토콜로 자리잡아가고 있으며, 지금까지의 전자메일 프로토콜 현황은 그림 1과 같다.

다음 아래의 표 1은 지금까지 언급했던 전자메일 보안과 관련되어 현재까지 연구되어온 메시지 보안 프로토콜을 비교·정리한 것이다[1].

2.1 전자 메일 시스템

2.1.1 SMTP

SMTP는 Simple Mail Transfer Protocol의 약자로서 모든 RFC 표준 규약의 메일 서버들이 서로 다른 메일서버 간에 메일을 주고받는데 사용되는 규약(Protocol)이다[1]. 단점으로는 7비트의 ASCII 텍스트만 처리할 수 있다는 것으로 인해 일반화되고 있는 멀티미디어 자료를 전송할 수 없으며 영문 텍스트 외에 다른 글자들은 전송할 수 없다. 또한 어느 한 사이트에서 MTA(Message Transfer Agent)를 향상한다고 해서 그 기능이 혁신적으로 개선되지 않는다는 단점을 가지고 있다[1].

2.1.2 POP3

POP3(Post Office Protocol - Version 3)는 SMTP를 제공하지 못하는 호스트들이 SMTP를 제공하는 워크스테이션을 서버로 하여 메일을 처리할 수 있도록 고안된 서비스로, 메일 서버에 저장되어 있는 메일을 원하는 시간과 장소에서 가져오는 프로토콜이다[2]. POP3의 문제점은 사용자가 자신의 사서함에 배달된 메일을 선택적으로 가져올 수 없다는 것이다.

2.2 메시지 보안 프로토콜

인터넷 전자메일은 전자문서가 목적지에 도착할 때까지 여러 호스트들을 거치는 과정에서 제 3 자에 의해 도청이나 가로채기 또는 변조되기 쉬운데, 이러한 데이타 무결성 및 기밀성은 메시지 보안 프로토콜로서 해결할 수 있다[3].

2.2.1 PEM

PEM은 대칭키 또는 비대칭키 암호화 알고리즘을 사용하여 암호화(enveloping) 한 후 PEM에서 사용하는 인코딩 방식에 따라서 문자로 변환하여 전자메일을 사용하여 전송하며 인증(Authentication), 무결성(Integrity), 기밀성(Confidentiality), 부인방지(Non-repudiation), 키 관리(Key Management)와 같은 서비스를 제공하지만 다자간 (Multipart) 전자메일이 지원되지 않고, 키 발급을 위하여 엄격한 인증기관의 계층구조를 요구하기 때문에 사용이 점차 줄고 있다[4].

2.2.2 PGP

PGP(Pretty Good Privacy)는 전자메일용 암호화 도구로 기밀성, 인증, 무결성, 부인방지 등의 기능을 제공한다. 초기의 PGP에서는 데이타를 암호화하기 위한 대칭형 알고리즘으로 IDEA를 사용하였으며, 대칭형 암호화키를 암호화하는데는 RSA를 사용하였다. 단점으로는 키 관리에 있어서 시간이 많이 소모되며 상당량의 수작업을 필요로 하고 또한 다수 사용자 그룹에 적용하기에는 그 관리가 너무 어렵다는 점을 들 수 있다.

2.2.3 MIME

MIME(Multipurpose Internet Mail Extensions)은 RFC 1521과 1522에 규정돼 있으며, 영어 외 문자나 멀티미디어 데이타 등을 전자메일으로 보내기 위한 기술에 사용되는 표준이다[5, 13]. 인터넷 표준 포맷의 헤더를 확장해서 메시지 종류를 인식할 수 있도록 했으며 첨부파일이나 멀티미디어 데이타를 포함하고 있는 전자메일의 포맷에 관한 표준을 가리킨다.

2.2.4 S/MIME

S/MIME(Secure Multipurpose Internet Mail Extensions)은 인터넷 MIME 메시지에 전자서명과 암호화기능을 첨부한 프로토콜로 안전한 MIME 데이타를 주고받을 수 있는 방법을 제공한다[6]. 널리 대중적으로 쓰이는 인터넷 MIME 표준에 근거하여 S/MIME은 안전한 보안 서비스를 제공한다.

2.3 타원곡선 암호시스템(Elliptic Curve Cryptosystem)

소수 p로 나눈 나머지를 구하는 모듈러 연산을 통해 이산대수 문제를 해결하는 것이 이산대수 문제에 대한 유일한 수학적 구조는 아니다. 1985년에 Neil Koblitz와 Victor Miller는 타원곡선 상에 정의된 이산대수 문제를 통해 타원곡선 암호 시스템을 제안하였다.

2.3.1 타원곡선의 정의

q를 어떤 소수의 멱승이라고 할 때, $K = F_p$는 q개의 원소를 갖는 유한체이고, K'는 K의 대수적 닫힘(algebraic closure)이라고 하자($K' = U_{n\geq 1} F_{q^n}$). $x_1 = ux_2,\ y_1 = uy_2,\ z_1 = uz_2$을 만족시키는 $u \in K^*$가 존재할 때, $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$라고 정의되는 $K^3 \setminus \{(0,0,0)\}$에서의 관계 ~에 대한 동치류의 집합을 K에 대한 사영 평면(projective plane) $P^2(K)$이라고 한다. 이 중에서 (x, y, z)를 포함하는 동치류를 $[x, y, z]$라고 표기한다.

다항식(Weierstrass equation)이란, $a_1, a_2, a_3, a_4, a_5, a_6 \in K'$에 대하여

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

와 같은 형태의 3차 동차식(homogeneous equation)을 말한다. 위의 식을 다음과 같이 표현했을 때,

$$F(X,Y,Z) = Y^2Z + z_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0 \quad (1)$$

이를 만족시키는 $P=[X,Y,Z]\in P^2(K^*)$에서 $\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}$가 모두 0이면 위의 등식을 "$P$에서의 특이(singular)하다"라고 하고, 그렇지 않으면 "P에서 비특이(non-singular)하다"라고 한다.

앞에서의 정의를 기초로 타원곡선을 정의하면, 식 (1)을 만족하는 모든 점 P에서 비특이 하다면 이를 타원곡선(elliptic curve) E라고 하며, 수학식으로는 다음과 같이 정의한다.

$$E = \{[X,Y,Z]\in K^* \mid Y^2Z + A_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, (\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}) (X,Y,Z)\neq 0, a_1, a_3, a_2, a_4, a_6 \in K^*\}$$

$Z=0$되는 점은 $\{0,1,0\}$으로 유일하게 E위에 있고, 이 점을 무한원점(point at infinity) O라 부른다.

2.3.2 타원곡선 상에서의 암호기법

유한체 F_q위에서 정의된 타원 곡선 $E(F_q)$에서의 이산대수 문제는 다음과 같이 "주어진 점 $P, Q\in E(F_q)$에 대해 $Q=mP$를 만족하는 자연수 m이 존재한다면 m을 찾는다."라는 것으로 정의한다.

이 m을 밑(base) P로 하는 Q의 이산대수라 부르며, m은 모듈라 $Ord(P)$에 의해 유일하게 결정된다. (P의 위수 $Ord(P)$는 $mP=o$되는 최소의 자연수이다.)

타원 곡선을 이용한 암호 시스템은 임의의 정수 s를 선택하여 감추고 $Q=sP$를 공개하므로 이 암호 시스템을 공격하는 공격자는 타원 곡선에서의 이산 대수 문제를 풀면 정보를 얻게 된다. 따라서, 이산 대수 문제가 어려우면 안전한 암호 시스템을 구성할 수 있다.

유한체에서 이산 대수 문제를 풀기 위하여 여러 가지 알고리즘이 개발되었는데, 그 중 타원 곡선에 적용할 수 있는 것은 Shank의 baby-step giant-step 알고리즘과 Pohling-Hellman 알고리즘이 있다. 이 두 알고리즘은 군의 위수(order)를 나누는 가장 큰 소인수의 이중근에 비례하는 실행시간을 갖고 있어 이중근 알고리즘이라 하고 타원 곡선의 위수가 40 자리 이상의 소수로 나누어지면 이산 대수 문제가 충분히 어렵다고 알려져 있다. 따라서, 유한체의 크기가 2^{130} 이상이면 충분한 암호학적 강도를 가지게 된다.

2.3.3 타원곡선 연산

그림 2에서와 같이 타원곡선상의 점에 대한 연산은 다음에 보여지는 알고리즘을 수행하는 것으로 계산된다.

① $P+O=O+P=P(\forall P\in E(Z_p))$

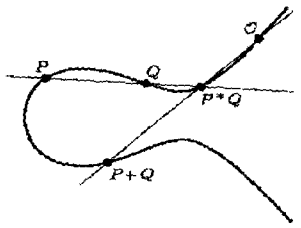


그림 2 타원곡선 연산

② $P=(x,y)\in E(Z_p) \Rightarrow (x,y)+(x,-y)=O$
$(-P=(x,-y))$

③ $P=(x_1,y_1)\in E(Z_p), Q=(x_2,y_2)\in E(Z_p)$
$\langle P\neq Q\rangle \Rightarrow P+Q=(x_3,y_3)$

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$ 이고,

$$\lambda = \frac{y_2-y_1}{x_2-x_1} \text{ if } P\neq Q \text{ (Addition)}$$

$$\frac{3x_1^2+a}{2y_1} \text{ if } P=Q \text{ (Doubling)}$$

선택한 임의의 두 점을 더할 때, 두 가지의 경우로 계산되는 것을 볼 수 있다. 첫 번째, 경우는 선택된 두 점이 다를 경우에 사용하는 연산으로 Addition이라고 불리고, 두 번째, 경우는 선택된 두 점이 같은 경우에 사용하는 연산으로 Doubling이라고 불린다.

2.3.4 타원곡선 암호의 안정성

공개키 암호시스템의 이론적 안전도를 조사하기 위해서는 먼저 시스템을 공격하는 데에 있어 그 시스템의 기반이 되는 수학적 문제를 푸는 것이 요구되는가를 분석하는 것이다. 실제 소인수 문제에 기반을 둔 공개키 암호시스템(RSA), 이산대수문제에 기반을 둔 공개키 암호시스템(DSA) 및 타원곡선 이산대수문제에 기반을 둔 타원곡선 암호시스템 모두 수년간 정밀한 분석이 있어 왔고, 그러한 시스템 공격방법은 그 기반이 되는 수학적 문제를 푸는 것이라고 알려져 있다. 그러므로 이제 "기반되는 어려운 문제"가 무엇인지를 분석하여야 한다. 타원곡선 이산대수문제는 타원곡선의 작은 클래스인 초특이 타원곡선(Supersingular elliptic curve)일 때 상대적으로 쉽다는 것이 알려져 있다. 그러나 이러한 취약점은 쉽게 확인될 수 있으며, 그러므로 쉽게 피할 수도 있다. 모듈러 p에서의 소인수 분해 문제와 이산대수 문제는 일반적으로 Sub-exponential time 알고리즘이 알려져 있다. Sub-exponential time 알고리즘이란 여전히 어려운 문제로 알려져 있으나, 완전지수복잡도(fully exponential time)알고리즘만을 허용하는 문제보다는 쉽다는 것을 의미한다. 이 알고리즘의 수행시간은 상수 c

에 대하여

$$O(\exp((c+o(1)))(\ln n)^{1/3}(\ln\ln n)^{2/3})$$

이다. 타원곡선 이산대수문제를 위한 최상의 알고리즘은 완전지수복잡도 알고리즘이며, 수행시간은 $O(\sqrt{p})$이다. 이것은 타원곡선 이산대수문제가 소인수문제나 이산대수문제보다 어렵다는 것을 의미한다.

타원곡선 암호시스템과 RSA 및 DSA를 깨는데 알려진 최상의 알고리즘으로 그 수행시간을 비교하여 보면, 표 2와 같다.(현재 허용되는 보안수준은 10^{12}MIPS year이다.)

표 2 타원곡선 암호와 RSA의 키 크기 비교

타원곡선 키 크기(bit)	RSA 키 크기(bit)	MIPS Year
160	1024	10^{12}
320	5120	10^{36}
600	21000	10^{78}
1200	120000	10^{168}

3. 전자 메일 시스템의 설계

전자메일 시스템은 그림 3과 같이 여러 개의 UA (User Agent)와 MTA(Mail Transfer Agent) 등으로 구성되며, 사용자 A가 사용자 B에게 메일을 전송하려 한다면 다음과 같은 과정을 거치게 된다[2].

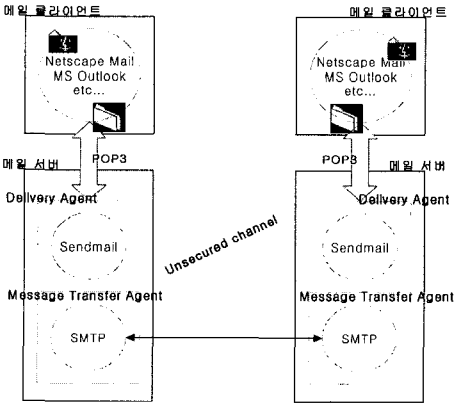


그림 3 전자 메일 시스템의 구조

위와 같은 인터넷 전자메일 시스템은 메시지 전송에 있어서는 효율적이며 신뢰성은 있지만, 메시지의 불법 누출, 불법 변조, 메시지 송·수신자의 신원 조작, 송·수신 사실의 부인 등이 가능하며, 인터넷의 특성상 송·수신자의 신원을 정확히 확인하기가 어렵다[1].

3.1 키 교환 모델

공개된 통신로에서 암호 통신을 하기 위하여 둘 이상의 개체는 암·복호화에 사용될 키를 사전에 공유하는 키 교환 과정을 수행하게 된다. 즉, 발신자와 수신자가 안전하게 메일을 주고받기 전에, 반드시 상호 키 교환이 필요하게 된다. 본 논문에서는 이러한 안전하게 키 교환을 하기 위해 ECDH(Elliptic Curve Diffie-Hellman) 알고리즘을 이용한 키 교환 모델을 구현하였다. 이 모델은 타원곡선 상에서 이산 대수를 계산하는 난이도를 이용하여 안전한 키 교환을 지원하게 된다. 이러한 키 교환 과정을 구현한 모델은 다음의 그림 4와 같다[1].

두 사람(Alice와 Bob)이 보호되지 않는 통신로 상에서 비밀정보를 나누어 가지려고 한다고 가정하자.

타원곡선 E와 점 $P(\in E)$를 선택하여 공개한다.

① Alice와 Bob은 각각 정수 a와 b를 선택하여 비밀로 간직한다.

② Alice와 Bob은 각각 $aP(\in E)$와 $bP(\in E)$를 계산하여 서로 나누어 가진다.

③ Alice와 Bob은 각각 $Q = a(bP)$와 $Q = b(aP)$를 계산하여, Q를 공유한다.

여기에서 aP 및 bP를 가지고, a와 b를 구하는 것은 타원곡선 이산대수 문제이다.

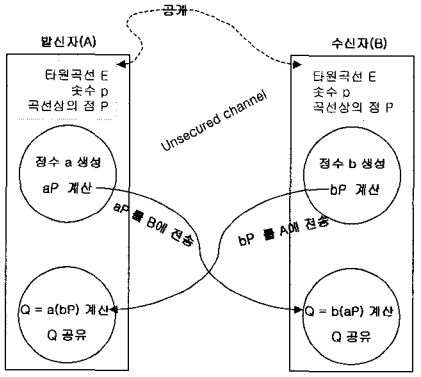


그림 4 ECDH 알고리즘을 이용한 키 교환 모델

3.2 암호화 모델

본 전자 메일 시스템에서 보안 서비스를 제공하기 위한 암호 모델은 아래에 제시된 알고리즘들을 기반으로 한다. 특히 EC-ElGamal 알고리즘은 자바 암호 패키지에서 기본으로 제공되지 않는 알고리즘으로 EC-ElGamal 알고리즘을 위한 키 생성, 암호 및 서명 클래스들을 새롭게 작성하고, JDK 디렉토리의 lib/security 디렉토리에 있는

java.security 파일에 다음과 같이 EC-ElGamal 알고리즘을 위한 새로운 프로바이더를 삽입하였다.

security.provider.3 = hannam.crypto.Provider

그리고 새로이 등록된 Provider 클래스에는 다음과 같이 EC-ElGamal 알고리즘과 이의 구현 클래스에 대한 매핑들을 put 명령어를 사용하여 추가하였다.

put("KeyPairGenerator.EC-ElGamal",
"hannam.crypto.EC-ElGamalKeyPairGenerator");
put("Cipher.EC-ElGamal",
"hannam.crypto.EC-ElGamalCipher");
put("Signature.EC-ElGamal",
"hannam.crypto.EC-ElGamalSignature");

1) 암호 알고리즘

(1) 메시지 암호화 알고리즘 : Rijndael 알고리즘

Rijndael 알고리즘은 AES에 의해서 채택된 관용 암호 알고리즘이며, 데이타를 128, 192, 256비트의 키를 이용한다. 관용 암호 알고리즘이기 때문에 암호화와 복호화에 사용되는 키가 하나이고 속도가 빠르기 때문에 메시지를 암호화하는데 사용된다. Rijndael은 키 길이가 변하는 것과 블록 길이가 변하는 것과 함께 블록 암호를 되풀이하게 되며, 아래와 같은 크게 네 개의 라운드 변환(round transformation) 과정에 의해 암호화 및 복호화 과정을 수행하게 된다.

- The ByteSub transformation
- The ShiftRow transformation
- The MixColumn transtormation
- The Round Key addition

(2) 전자 서명 알고리즘 : EDCSA(Elliptic Curve Digital Signature Algorithm)

ECDSA는 DSA를 타원곡선으로 변형시킨 것이다. ECDSA와 DSA의 중요 차이점은 r의 생성에 있다. DSA는 r을 임의의 $g^k \bmod p$를 선택/계산한 후, $\bmod q$를 계산하여 얻는다. 그러나 ECDSA에서 r은 임의의 점 kP의 x좌표를 $\bmod n$하여 얻는다. ECDSA가 160비트 q와 1024비트 p를 가진 DSA와 비슷한 안전도를 갖기 위해서는 매개변수 n이 약 160비트이면 된다. 이 경우 DSA와 ECDSA는 같은 서명길이(320비트)를 갖는다.

표 3 DSA와 ECDSA 기호 비교

DSA	ECDSA
q	n
g	P
x	d
y	Q

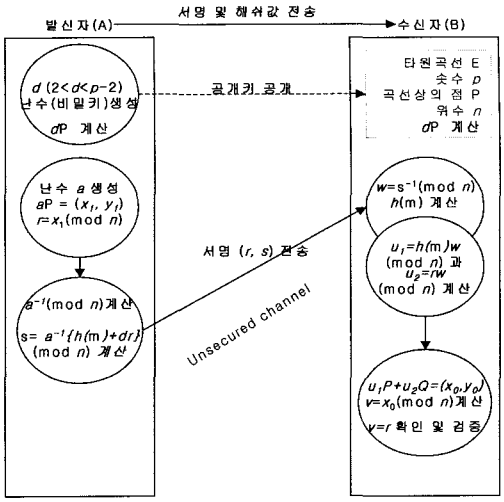


그림 5 ECDSA를 이용한 전자서명 모델

IEEE P1363과 ANSI X9F1 표준에서 현재 표준화 추진 중에 있다. 다음 표 3은 DSA와 ECDSA의 기호를 비교한 것이며, 그림 5는 ECDSA 알고리즘을 이용하여 전자서명을 하는 과정을 보여준다.

[ECDSA 키 생성] A가 키를 생성한다고 가정하자.

① Z_p에서 정의된 타원곡선 E를 선택한다. $\#E(Z_p)$는 큰 소수 n에 의해 나누어져야 한다.

② 위수 n인 점 $P \in E(Z_p)$를 선택한다.

③ 구간 $[2, n-2]$에서 통계적으로 유일하고 예측 불가능한 정수 d를 선택한다.

④ $Q = dP$를 계산한다.

⑤ Alice의 공개키는 (E, P, n, Q)이고, 비밀키는 d이다.

[ECDSA 서명 생성] 메시지 m에 A가 서명한다고 가정하자.

① 구간 $[2, n-2]$에서 통계적으로 유일하고 예측불가능한 정수 k를 선택한다.

② $kP = (x_1, y_1)$과 $r = x_1 \bmod n$을 계산한다(x_1은 정수로 간주).

③ $r = 0$이면, ①단계로 되돌아간다.

④ $k^{-1} \bmod n$을 계산한다.

⑤ $s = k^{-1}\{h(m) + dr\} \bmod n$을 계산한다.(h:SHA-1)

⑥ $s = 0$이면, ①단계로 되돌아간다.

⑦ 메시지 m에 대한 서명은 (r, s)이다.

[ECDSA 서명 검증] B가 A의 서명 (r, s)을 검증한다고 가정하자.

① Alice의 인증된 공개키 (E, P, n, Q)를 얻는다.

② r과 s가 구간 $[1, n-1]$에 있는 지 확인한다.

③ $w = s^{-1} \bmod n$과 $h(m)$을 계산한다.

④ $u_1 = h(m)w \bmod n$과 $u_2 rw \bmod n$을 계산한다.

⑤ $u_1P + u_2Q = (x_0, y_0)$와 $v = x_0 \bmod n$을 계산한다.

⑥ $v = r$을 확인한다.

(3) 세션키 암호화 알고리즘 : EC-ElGamal 알고리즘

타원곡선 E와 $P(\in E)$를 공개하고, 메시지 $M=(M_1, M_2)$ $(\in F_{2^m} \times F_{2^m})$이라 가정하자.

<B가 A에게 메시지를 보내고자 할 때>

[B]

① 임의로 정수 a를 선택

② aP를 계산하여 공개

[A]

① 임의로 정수 k를 선택

② 점 kP와 점

$k(aP) = a(kP) = (x, y)$를 계산

③ $x, y \neq 0$이면,

(kP, M_1x, M_2y)를 Alice에게 보낸다.

<메시지를 읽기 위하여(Alice)>

① $a(kP) = (x, y)$를 계산

② $\frac{M_1x}{x} = M_1$, $\frac{M_2y}{y} = M_2$를 계산

2) 메시지 암호화

발신자가 메시지를 암호화하여 생성하는 암호 메시지는 암호화된 세션키 부분, 암호화된 메시지 부분, 그리고 서명 부분으로 구성된다. 그림 6은 다음에 기술한 메시지 암호화 과정을 거쳐 암호 메시지를 생성하는 과정에 대해 보여주며, 설계에 대한 구현 코드를 기술한다.

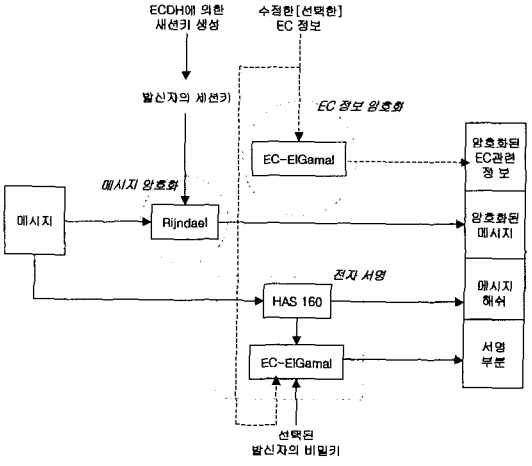


그림 6 Rijndael 알고리즘을 이용한 메시지 암호화 모델

① 메시지를 Rijndael 알고리즘을 사용해서 발신자의 세션키로 메시지를 암호화한다. 그리고 이탤릭체와 함께 진하게 강조된 글자는 사용된 알고리즘을 보여주고 있다.

```
byte[] bodyPlaintext = body.getBytes();
Cipher cipher = Cipher.getInstance("Rijndael");
cipher.init(Cipher.ENCRYPT_MODE, sessionKey);
byte[] bodyCiphertext = cipher.doFinal
(bodyPlaintext);
```

② 메시지 암호화에 사용된 Rijndael 세션키(선택된 EC 정보)를 EC-ElGamal 알고리즘을 사용해서 암호화한다.

```
cipher = Cipher.getInstance("EC-ElGamal");
cipher.init(Cipher.ENCRYPT_MODE,
theirPublicKey);
byte[] sessionKeyCiphertext =
cipher.doFinal(sessionKey.getEncoded());
```

③ HAS160 알고리즘을 사용해서 메시지 다이제스트를 생성하고, 이 메시지 다이제스트를 ElGamal 알고리즘으로 암호화하여 전자 서명을 생성한다.

```
MessageDigest md =
MessageDigest.getInstance("HAS160");
md.update(bodyPlaintext);
byte[] msg_digest = md.digest();
Signature s = Signature.getInstance("ECDSA[또는
EC-ElGamal]");
s.initSign(ourPrivateKey);
s.update(msg_digest);
byte[] bodySignature = s.sign();
```

3) 메시지 복호화

수신된 암호 메시지는 메시지의 각 부분별로 복호화 및 서명을 증명하는데, 그림 7은 이러한 과정을 보여주고 있다.

① EC-ElGamal 알고리즘을 사용해서 Rijndael 세션키(선택된 EC 정보)를 복호화한다.

```
PrivateKey ourPrivateKey =
mKeyManager.getPrivateKey();
Cipher cipher = Cipher.getInstance("ElGamal");
cipher.init(Cipher.DECRYPT_MODE,
ourPrivateKey);
byte[] sessionKeyPlaintext =
cipher.doFinal(sessionKeyCiphertext) ;
SecretKeyFactory skf =
SecretKeyFactory.getInstance("DES");
```

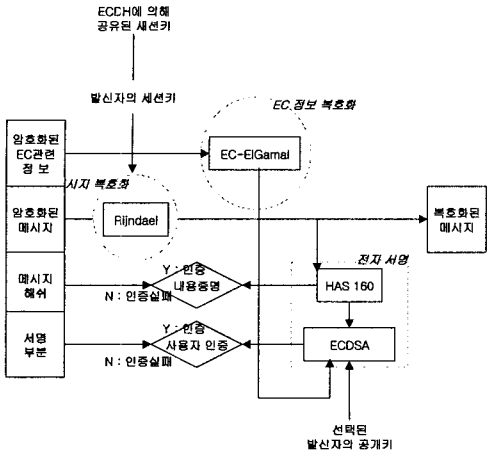


그림 7 Rijndael 알고리즘을 이용한 복호화 모델

```
DESKeySpec desSpec = new
DESKeySpec(sessionKeyPlaintext, 0);
SecretKey sessionKey =
skf.generateSecret(desSpec)
```

② Rijndael 알고리즘을 사용해서 복원된 Rijndael 세션키(선택된 EC 정보를 가지고 암호문을 복호화 한다.

```
cipher = Cipher.getInstance("Rijndael");
cipher.init(Cipher.DECRYPT_MODE, sessionKey);
byte[] plaintext = cipher.doFinal(bodyCiphertext);
```

③ 복원된 평문 메시지를 HAS160 알고리즘을 사용해서 메시지 다이제스트로 변환하고, ECDSA[EC ElGamal] 알고리즘을 사용해서 전자 서명을 증명하게 된다.

```
MessageDigest md =
MessageDigest.getInstance("HAS160");
md.update(plaintext);
byte[] msg_digest = md.digest();
Signature s = Signature.getInstance("ECDSA");
s.initVerify(theirPublicKey);
s.update(msg_digest);
if (s.verify(bodySignature)) {
new MessageBox(this, "서명 증명 성공", "서명이
올바르게 증명되었습니다");
}
else {
new MessageBox(this, "서명 증명 실패", "서명이
올바르지 않습니다");
}
```

3.3 배달증명 모델

그림 8은 제안된 배달증명 방식을 기반으로 한 배달증명 모델을 통해서 의도된 수신자가 올바르게 메일 메시지를 수신하였음을 송신자에게 증명하는 과정을 보여주고 있다.

즉, 발신자가 배달증명 서비스를 요청했을 때, 메시지 암호화와 동일한 과정으로 암호화된 세션키, 암호화된 메시지, 및 서명된 메시지로 구성된 암호 메시지를 생성하고, 이 메시지에 다음과 같이 배달증명을 요청하는 플래그를 함께 붙여서 수신자에게 송신한다. 그리고 수신자 쪽의 UA는 수신된 메시지에서 플래그를 확인하여 배달증명 요청 플래그가 있다면, 전자 서명을 증명하여 증명이 성공하면, 암호화된 메시지를 수신자의 비밀키를 사용하여 서명한 후에 발신자의 암호화된 메시지와 함께 처음 메시지를 송신한 발신자에게 회신한다. 발신자는 배달 증명 요구에 대한 회신 메시지임을 메시지에 삽입된 플래그를 통해서 확인하고, 메시지의 전자서명을 증명하여 의도된 수신자에게 메일이 올바르게 배달되었음을 확인한다. 이러한 절차를 통해서 향후에 발생할 수 있는 전자 메일 메시지의 수신에 대한 수신자의 부인을 방지할 수 있다. 추가적으로, 본 논문에서 가장 중요한 모델이며, 다른 보안 메일 시스템과 비교할 수 있는 모델이다. 따라서, 이러한 일련의 과정에 대한 서술을 다음과 같은 단계별로 보다 자세히 설명하면 다음과 같다.

<단계 1> 발신자가 배달증명 서비스를 요청했을 때, 메시지 암호화와 동일한 과정으로 암호화된 세션키, 암호화된 메시지, 및 서명된 메시지로 구성된 암호 메시지를 생성하고, 이 메시지에 다음과 같이 배달증명을 요청하는 플래그를 함께 붙여서 수신자에게 송신한다.

```
String unbroken = "Certification of delivery:" +
                  base64.encode(plaintext);
```

<단계 2> 수신자 쪽의 UA는 수신된 메시지에서 플래그를 확인하여 배달증명 요청 플래그가 있다면, 전자서명을 증명하여 증명이 성공하면, 암호화된 메시지를 수신자의 비밀키를 사용하여 서명한 후에 발신자의 암호화된 메시지와 함께 처음 메시지를 송신한 발신자에게 회신한다.

```
if (return_flag  == 1) {
PrivateKey ourPrivateKey =
mKeyManager.getPrivateKey();
MessageDigest md =
MessageDigest.getInstance("HAS160");
md.update(bodyCiphertext);
byte[] msg_digest = md.digest();
```

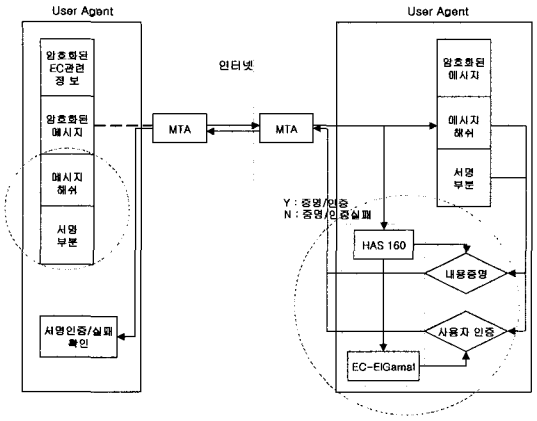


그림 8 암호화 알고리즘을 이용한 배달 증명 모델

```
Signature s = Signature.getInstance("ECDSA");
s.initSign(ourPrivateKey);
s.update(msg_digest);
byte[] bodySignature = s.sign();
.....
```

<단계 3> 발신자는 배달 증명 요구에 대한 회신 메시지임을 메시지에 삽입된 플래그를 통해서 확인하고, 메시지의 전자서명을 증명하여 의도된 수신자에게 메일이 올바르게 배달되었음을 확인한다. 이러한 절차를 통해서 향후에 발생할 수 있는 전자 메일 메시지의 수신에 대한 수신자의 부인을 방지할 수 있다.

```
if (receipt_flag == 1) {
.....
Signature s = Signature.getInstance("ECDSA");
s.initVerify(theirPublicKey);
s.update(msg_digest);
if (s.verify(bodySignature)) {
new MessageBox(this,"***** 서명이 올바르게 증!
*****\n", "*****[배달"+ "증명]\n" + theirName +
"가 올바르게 메일을 수신하였음을 확인합니다.
****\n");
}
else {
new MessageBox(this,"서명 증명 실패!","\n" +
"*****배달증명에 실패하였습니다."+ "*****");
}
.....
}
```

4. 안전한 전자 메일 시스템의 구현

본 자바 전자 메일 시스템의 핵심은 메시지의 암호화와 복호화 및 배달증명을 담당하는 CipherMail 클래스와, 메일 메시지를 저장하고 관리하는 Message 클래스이다. 그림 9는 Message 클래스를 사용하는 Cipher Mail 애플리케이션 모듈을 구성하는 클래스들의 구성도이다.

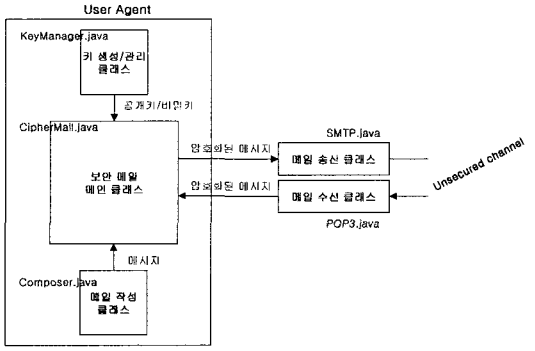


그림 9 전체 모듈 구성도

4.1 메시지 작성 및 송신

메시지를 작성하기 위해서는 시작 윈도우의 메뉴에서 메시지 작성 메뉴 혹은 버튼을 선택한다. 메일 메시지는 크게 세 가지 모드로 전송될 수 있다. 첫 번째는 보안 기능과 배달증명을 사용하지 않고 평문 그대로를 전송하는 모드이다. 어떤 보안 기능도 사용하지 않으므로 보안상의 취약점을 지니게 되지만, 빠르게 메일을 전송할 수 있다. 두 번째는 보안 기능을 사용하는 것으로 메일 메시지에 대한 암호화 및 서명 등을 제공하여 안전하게 수신자에게 메일을 전송할 수 있다. 세 번째는 보안 기능과 배달 증명 서비스를 사용하는 것으로 기본 정보보호 서비스를 제공할 뿐만 아니라, 의도된 수신자가 메일을 올바르게 수신하였음을 확인할 수 있도록, 수신자가 서명한 메시지를 회신하게 된다.

그림 10에서는 메시지 작성 윈도우의 메뉴 바에서 '보안기능 사용' 메뉴와 '배달 증명 사용' 메뉴를 선택하여 기본 보안 기능과 배달증명 서비스를 제공받을 수 있도록 설정하였다. 마지막으로 메시지 전송 버튼을 누르면 그림 11과 같이 메일 메시지의 암호화 및 서명 등의 과정을 거쳐 의도된 수신자에게 메시지를 전송한다.

위의 그림 11에 나타난 메시지 작성에 대한 소스 파일에 아래에 보여진다. 전체적인 메일 박스에 대한 소스는 언급하지 않고, 메시지 작성하기 이한 CipherMail

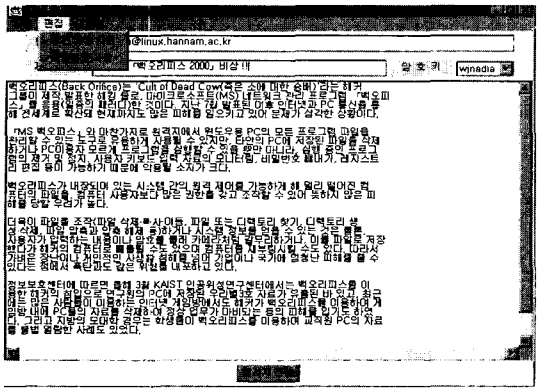


그림 10 메시지 작성

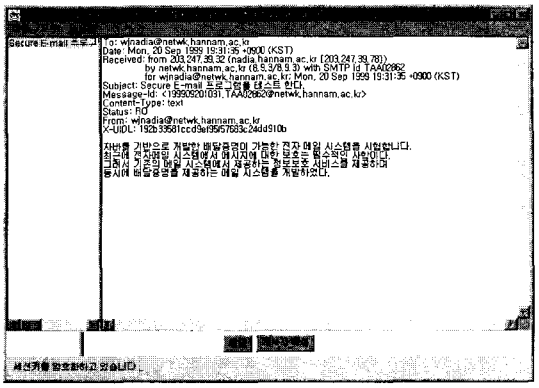


그림 11 메시지 암호화 및 서명

```
public class Composer extends Frame implements ActionListener,
ItemListener {
        protected TextField mToField, mSubjectField, CcField;
        protected Choice mKeyChoice;
        protected TextArea mMessageArea;
        protected Button mSendButton;
        protected CipherMail mCipherMail;
        protected Label ToLabel, SubLabel, KeyLabel, CcLabel;
        protected MenuItem MsgItem, ExitItem, aboutItem;
        protected CheckboxMenuItem flag1_Item,flag2_Item;
    public Composer(CipherMail cm, KeyManager km) {
        super("새로운 메시지");
                ..... 중략
 public void sendMessage(Message m, String remoteName) {
        try {
            String host = mPreferences.getProperty("SMTP");
            String email = mPreferences.getProperty("Email");
            String body = m.getBody();
            try {
               if (secure_flag) m.setBody(encrypt(body,
                  remoteName));
            }
            catch (Exception e) {
               System.out.println("암호화 : " + e.toString());
               setStatus(" " + "죄송하지만, 이 메시지는 전송될 수
               없습니다! : " + e.toString());
               return;
            }
// Send the message .
            setStatus("    " + host + "에 연결중입니다.");
            SMTP smtp = new SMTP (host);
            smtp.login();
            setStatus("    " + "메시지를 전송중입니다...");
            smtp.send(email, m);
            smtp.quit();
            setStatus("    " + "메시지 전송완료");
        } catch (IOException ioe) {
                        setStatus(ioe.toString());
        } }
```

파일의 Composer 클래스 부분만을 보여주고 있다. 이어서 그림 11에 나타난 암호화 및 서명에 대해서도, 실질적인 세션키 암호화, 메시지 암호화 및 전자서명에 대해서만 소스를 공개하였고, 간단한 주석을 통하여 암호화 및 전자서명 과정을 기술하였다.

```
protected String encrypt(String body, String theirName) throws
Exception {
        setStatus("    " + "암호키를 가져오고 있습니다...");
        String ourName = mKeyManager.getName();
        PrivateKey ourPrivateKey =
        mKeyManager.getPrivateKey();
        PublicKey theirPublicKey =
        mKeyManager.getPublicKey(theirName);
// 세션키 생성.
        setStatus("    " + "세션키를 생성하고 있습니다...잠시
        기다리세요 ! " );
        KeyGenerator kg =
        KeyGenerator.getInstance("Rijndael");
        kg.init(new SecureRandom());
        Key sessionKey = kg.generateKey();
        if (devery_flag = 1) Rijndael_key.put(theirname,
        sessionKey);
// 메시지 본문 암호화
        setStatus("    " + "메시지를 암호화하고 있습니다...");
        byte[] bodyPlaintext = body.getBytes();
        Cipher cipher = Cipher.getInstance("Rijndael");
        cipher.init(Cipher.ENCRYPT_MODE, sessionKey);
        byte[] bodyCiphertext =
        cipher.doFinal(bodyPlaintext);
// 세션키 암호화
        setStatus("    " + "세션키를 암호화하고 있습니다...");
        cipher = Cipher.getInstance("ElGamal");
        cipher.init(Cipher.ENCRYPT_MODE, theirPublicKey);
        byte[] sessionKeyCiphertext =
        cipher.doFinal(sessionKey.getEncoded());
// 메시지 서명
        setStatus("    " + "메시지를 서명하고 있습니다..." ) ;
        MessageDigest md =
        MessageDigest.getInstance("MD5");
        md.update(bodyPlaintext);
        byte[] msg_digest = md.digest();
        Signature s = Signature.getInstance("ElGamal");
        s.initSign(ourPrivateKey);
        s.update(msg_digest);
        byte[] bodySignature = s.sign();
// Embed everything in the new body.
        setStatus("    " + "암호 메시지를 생성하고 있습니다...");
        ByteArrayOutputStream byteStream = new
        ByteArrayOutputStream();
        DataOutputStream out = new
        DataOutputStream(byteStream);
                        .... 중략
    }
```

4.2 메시지 수신 및 서명 증명

메시지를 수신한 수신자의 메일 프로그램은 배달 증

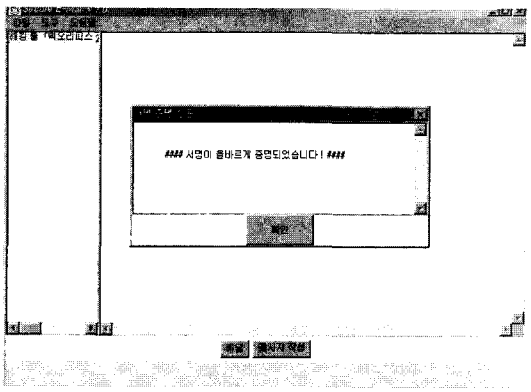


그림 12 수신된 메시지의 서명 증명

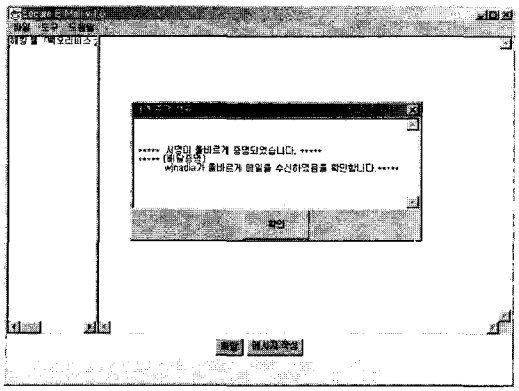


그림 13 회신된 메시지의 배달증명

```
protected String Cipher(String body, int i) throws Exception {
        .....중략
 // Base64로 디코딩
        .....중략
//수령확인 메시지인지 판단하여 수령확인 메시지라면 배달증명을
위한 서명의 증명작업을 수행한다.
        .....중략
 // 세션키 복호화
        .....중략
// 메시지 본문 복호화 및 메시지 다이제스트 생성
        .....중략
 // 서명 증명
            setStatus("    " + "서명을 증명하고 있습니다...");
        PublicKey theirPublicKey;
        try { theirPublicKey =
        mKeyManager.getPublicKey(theirName); }
        catch (NullPointerException npe) {
          setStatus("#### " + theirName + "의 서명을 증명할 수
              없습니다" + " : 키를 가지고 있지 않습니다. ####");
          new MessageBox(this,"서명 증명 실패!","\n\n\n " +
            "#### " + theirName + "의 서명을 증명할 수
            없습니다 !" + ": 키를 가지고 있지 않습니다. ####");
          return new String(plaintext);
        }
        Signature s = Signature.getInstance("ElGamal");
        s.initVerify(theirPublicKey);
        s.update(msg_digest);
        if (s.verify(bodySignature)) {
          setStatus("####" + theirName + "의 서명이 올바르게
              증명되었습니다 ! ####");
          new MessageBox(this,"서명 증명 성공!","\n\n\n " +
            "####" + theirName + "의 서명이 올바르게
            증명되었습니다 ! ####"); }
```

```
protected String Cipher(String body, int i) throws Exception {
        .....중략
        // Base64로 디코딩
        .....중략
// 세션키 복호화
        setStatus("    " + "세션키를 복호화하고 있습니다...");
        String ourName = mKeyManager.getName();
        PrivateKey ourPrivateKey =
        mKeyManager.getPrivateKey();
        Cipher cipher = Cipher.getInstance("ElGamal");
        cipher.init(Cipher.DECRYPT_MODE, ourPrivateKey);
        byte[] sessionKeyPlaintext = cipher.doFinal
        (sessionKeyCiphertext) ;
        SecretKeyFactory skf =
        SecretKeyFactory.getInstance("DES");
        DESKeySpec desSpec = new
        DESKeySpec(sessionKeyPlaintext, 0);
        SecretKey sessionKey = skf.generateSecret(desSpec);
// 메시지 본문 복호화 및 메시지 다이제스트 생성
        setStatus("    " + "메시지 본문을 복호화하고
        있습니다...");
        cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE, sessionKey);
        byte[] plaintext = cipher.doFinal(bodyCiphertext);
        MessageDigest md =
        MessageDigest.getInstance("MD5");
        md.update(plaintext);
        byte[] msg_digest = md.digest();
        ... 중략
        Signature s = Signature.getInstance("ElGamal");
        s.initVerify(theirPublicKey);
        s.update(msg_digest);
        if (s.verify(bodySignature)) {
          setStatus(" ");
          new MessageBox(this,"배달 증명 성공!","\n\n\n
            " + "#### 송신된 메시지에 대한 수령확인 메시지의
          배달증명이 성공하였습니다 ! ####");
        }
        else {
            setStatus(" ");
          new MessageBox(this,"배달증명 실패!","\n\n\n
            " + "#### 수령확인 메시지의 서명이 올바르지 않습니다
          ! ####");
        } } return new String(plaintext); }
```

명을 요구하는 메시지임을 메시지 내의 플래그를 인지하여 확인하고, 서명을 증명하여 그림 12와 같이 서명이 올바르게 증명되었음을 다이얼로그 박스에 출력한다. 그리고 수신된 메시지 중에서 암호화된 메시지만을 서명하고 발신자의 암호화된 메시지와 함께 "Receipt:" 플래그를 메시지의 처음에 첨부하여 발신자에게 회신한다.

4.3 메시지 회신 및 부인방지

회신 메시지를 수신한 발신자는 메시지에 붙어있는 플래그가 배달 증명에 대한 회신 메시지를 나타냄을 확인한다. 그리고 수신된 메시지의 서명을 증명하여 배달 증명이 확인되었음을 나타내는 다이얼로그 박스를 그림 13과 같이 출력한다.

5. 비교 분석

보안 전자 메일 시스템은 PGP와 PEM 및 인터넷 익스플로러나 넷스케이프와 같은 범용 브라우저와 인증 서버를 기반으로 한 최신의 상용화된 전자 메일 시스템 등이 있다. 본 논문에서 구현한 메일 시스템은 앞에서 열거한 기타 메일 시스템들에서와 같이 메시지 기밀성, 메시지 무결성, 송신자 인증, 송신자 부인 방지 서비스 등을 제공하며, 더불어 배달증명 서비스를 제공한다. 또한 현재 일부 웹을 기반으로 한 웹 전자 메일 시스템에서 전자 메일 메시지를 수신하였음을 통보해 주는 기능을 제공하지만, 암호 기반의 서명을 통해서 수신 부인 방지를 제공하지는 못하고 있는 실정이다. 그리고 현재까지 대부분의 전자 메일 시스템은 내용 증명 서비스를 제공하지 못하고 있다. 표 4는 본 논문에서 구현한 메일 시스템과 PGP, PEM, MIME, 그리고 S/MIME 등의 메시지 보안 프로토콜을 비교하였다. 또한, 표 5에서는 기존의 범용 브라우저 및 최신의 상용 전자 메일 시스템에서 제공하고 있는 정보보호 서비스에 대한 비교를 보여주고 있다[1, 6, 15, 18].

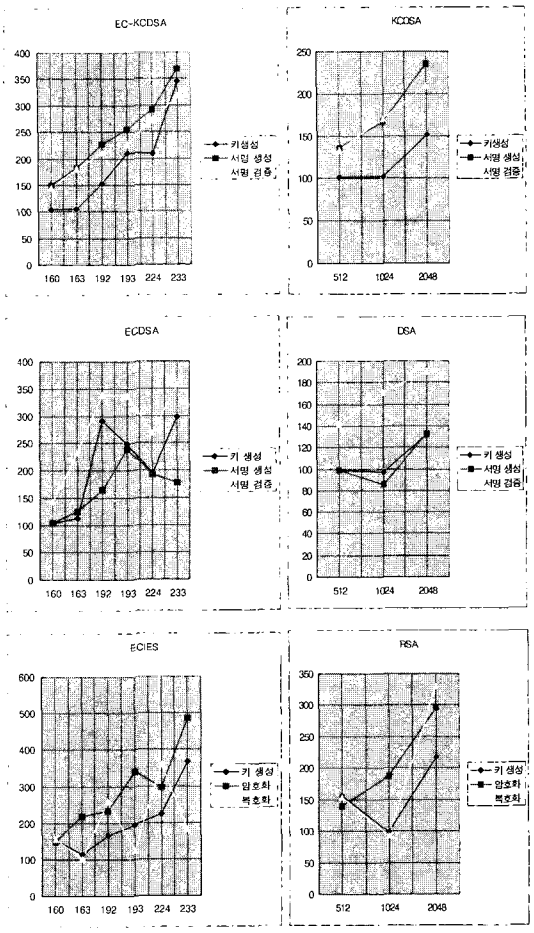
표 4 제안된 메일 시스템과 메시지 보안 프로토콜과의 비교

정보보호 서비스	제안된 시스템	PEM	PGP	MIME	S/MIME
메시지 기밀성	제공함	제공함	제공함	제공함	제공함
메시지 무결성	제공함	제공함	제공함	제공함	제공함
송신자 인증	제공함	제공함	제공함	제공함	제공함
송신부인방지	제공함	제공함	제공함	제공함	제공함
배달 증명	제공함	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음
내용 증명	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음

표 5 제안된 메일 시스템과 상용화 메일 시스템의 비교

정보보호 서비스	제안된 시스템	범용 브라우저	D 사	e 사	J 사
메시지 기밀성	제공함	제공함	제공함	제공함	제공함
메시지 무결성	제공함	제공함	제공함	제공함	제공함
송신자 인증	제공함	제공함	제공함	제공함	제공함
송신부인방지	제공함	제공함	제공함	제공함	제공함
배달 증명	제공함	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음
내용 증명	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음	제공하고 있지 않음

또한, 기존의 전자 메일 시스템 상에서 사용하는 알고리즘과 본 논문에서 구현한 타원 곡선 알고리즘과의 암호학적 강도에 대한 성능 비교를 해보면 다음과 같다.



즉, 기존의 암호화 알고리즘을 포함한 모든 알고리즘의 암호학적 강도를 유지하기 위해서는 위의 그림에서 볼 수 있듯이 훨씬 더 적은 키 길이를 갖는 타원곡선 암호 알고리즘으로 대체할 수 있다. 이러한 작은 키로 높은 암호학적 강도를 유지하는 것은 향후 자바카드나 스마트 카드와 같은 적은 용량의 메모리로 커다란 암호학적 강도를 유지해야 하는 분야에서 쉽게 적용될 수 있다고 본다.

6. 결 론

현재 네트워크 환경에서 널리 사용되고 있는 전자메일 시스템은 많은 보안상의 취약점에 노출되어 있다. 이러한 전자메일의 취약점을 극복하기 위해서 다양한 보안 메일

시스템들이 소개되고 있는 추세이지만 사용자에게 만족스런 서비스를 제공하지 못하고 있다. 본 논문에서는 기존의 메일 시스템에서 제공되는 기본 보안 서비스를 제공하며, 의도된 수신자가 메시지를 올바르게 수신하였음을 증명하는 배달증명 서비스와 내용이 변경되지 않았음을 증명하는 내용증명 서비스, 그리고 메시지 교환 이전에 안전하게 키를 교환하기 위한 키 교환 모델을 설계·구현하였다. 구현은 자바 암호 API를 기반으로 하였으며, 이를 포함한 자바 플랫폼은 네트워크 및 보안 서비스를 제공하는 데 필수적인 모든 요소들을 클래스로 갖추고 있기 때문에 개발자에게 프로그램의 작성을 용이하게 한다.

향후 보안 메일 시스템에서는 부인방지 서비스 및 안전한 키 교환 서비스를 제공하면서도, 기존의 시스템(암호화하지 않은 채, 메일을 전송하는 시스템)과 전송속도 차이가 나지 않는 전자 메일 시스템을 구현함으로써, 상호간에 신뢰하면서도 빠른 메일 서비스를 제공하는 방법을 모색해야 할 것이다.

참 고 문 헌

[1] 최용락, 소우영, 이재광, 이임영, "통신망 정보 보호", 그린출판사, 1995.

[2] 조한진, 김봉한, 이재광, "정보보호 서비스를 위한 Secure E-mail 시스템 설계", 한남대학교 산업기술연구소, 1998.

[3] 손진욱 편저, "Java 2 Programing Bible," 정보문화사, 1999.

[4] 박춘식, "배달 및 내용 증명이 가능한 전자메일", 통신정보보호학회지, 제7권 제2호, 1997. 6.

[5] 강명희, "인터넷 메일 시스템에서의 정보 보호 서비스 구현", 광운대학교 전자계산학과 석사학위 논문, 1995

[6] 이재용, 이기수, 장춘서, "PGP를 이용한 WWW 메일 시스템의 설계 및 구현", 한국정보과학회 가을 학술발표논문집, 제24권 제 2호, 1997.

[7] 홍주영, 윤이중, 김대호, "전자우편 시스템의 보호 방식 분석", 통신정보보호학회지 Vol.4 No.2, 1994. 6

[8] Jonathan Knudesen, "Java Cryptography," O' REILLY, 1998.

[9] Sun Microsystems, "Java 2 SDK, Standard Edition *Documentation*," 1999.

[10] Scott Oaks, "Java Security," O' REILLY, 1998

[11] Elliotte Rusty Harold, "Java Network Programing," O' REILLY, 1997.

[12] Bruce Schneier, "Applied Cryptography," John Wiley & Sons Inc., 1996.

[13] J.Zhou and D. Gollmann, "A Fair Non-repudiation Protocol," Proc. of the 1996 IEEE Symposium on Security and Privacy, 1996.

[14] Stephen T. Kent,"Internet Privacy Enhanced Mail," CACM, Vol. 36, No. 3, 1993.

[15] Stephen T. Kent,"An Overview of Internet Privacy Enhanced Mail," INET '93, June 1993.

[16] N. Borenstein, "MIME(Multipurpose Internet Mail Extensions)" Part One: "Mechanisms for Specifying and Describing the Format of Internet Message Bodies," RFC 1521, 1993.

이 원 구

2000년 한남대학교 컴퓨터공학과(공학사). 2002년 한남대학교 대학원 컴퓨터공학과(공학석사). 2002년 한남대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 컴퓨터네트워크, 정보통신 정보보호

김 성 준

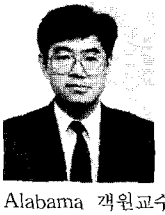
2000년 한남대학교 컴퓨터공학과(공학사). 2000년. 한남대학교 대학원 컴퓨터공학과(공학석사). 2002년 한남대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 컴퓨터네트워크, 정보통신 정보보호

이 희 규

1998년 우송대학교 컴퓨터과학과(공학사). 2000년 한남대학교 대학원 컴퓨터공학과(공학석사). 2002년 현재 한남대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 컴퓨터네트워크, 정보통신 정보보호.

문 기 영

1986년 2월 경북대학교 전자공학과 졸업(학사). 1989년 2월 경북대학교 대학원 전산전공(석사). 1992년 1월 ~ 1994년 3월 대우정보시스템 기술연구소 대리. 1994년 3월 ~ 현재 한국전자통신연구원 능동보안기술연구팀 과제책임자. 관심분야는 XML 정보보호, EC정보보호, 분산처리



이 재 광

1984년 광운대학교 전자계산학과(이학사). 1986년 광운대학교 대학원 전자계산학과(이학석사). 1993년 광운대학교 대학원 전자계산학과(이학박사). 1986년 ~ 1993년 군산전문대학 전자계산학과 부교수. 1997년 ~ 1998년 University of Alabama 객원교수. 1993년 ~ 현재: 한남대학교 컴퓨터공학과 부교수. 관심분야는 컴퓨터 네트워크, 정보통신 정보보호