通信情報保護學會誌
第5卷 第4號, 1995. 12

전자우편 보안 -PGP-

박 현 동*, 류 재 철*, 임 채 호**, 변 옥 환**

요 약

PGP(Pretty Good Privacy)는 전자우편에 사용되는 보안도구로써 현재 인터넷(Internet)상에서 광범위하게 사용되고 있다. 전자우편에서 필요로 하는 기밀성, 무결성, 송신자 인증, 송신부인방지 등의 기능을 모두 제공해 주고 있고 특히, 사용되는 키에 대한 인증을 하나의 기관에서 전담하지 않고, PGP의 사용자들에 의한 키 인증방식을 채택하고 있는 것이 특징이다. 본고에서는 PGP의 효과적인 보급을 위해 전자우편에 필요한 기본 암호 메카니즘과 PGP의 구조 및 사용방법을 살펴본다.

1. 서 론

인터넷(Internet)이 보편화되어 감에 따라 학술, 연구용의 범위를 벗어나 이제는 기업과 일반 사용자들이 인터넷의 주요한 사용자로써 떠오르고 있다. 인터넷의 사용자들이 그들의 분야에 국한되지 않고 가장 많이 사용하고 있는 인터넷 서비스 중의 하나가 전자우편(Electronic mail)이다. 매년 약 2배씩의 성장률을 보이고 있는 인터넷의 배경에는 사용자들의 전자우편 사용증가가 큰 몫을 차지하고 있다[1]. 이처럼 전자우편은 시간과 공간을 초월하는 자원으로써, 현재 사용범위가 점차 확대되고 있지만, 이제까지의 전자우편은 보안상에 있어 너무나도 취약한 면을 가지고 있다[2]. 어느 사용자가 다른 사용자에게 보낸 전자우편은 목적지에 도달할 때까지 여러 호스트(host)들을 거치게 되며, 전자우편은 봉투에 넣어져서 밀봉되는 일반적 편지와는 달리 보내는 곳의 주소뿐만 아니라 내용까지도 그대로 보이는 엽서와 같은 구조를 가지고 있기 때문에 이런 과정에서 얼마든지 탈취, 변조될 가능성이 있다.

이러한 상황에서 보안을 유지하는 길은 내용을 암호화하여 중간에서 가로챈다 하더라도 다른 사람은 알아 볼 수 없게 만드는 방법이다. 이를 위하여 현재 인터넷에서는 PGP(Pretty Good Privacy)와 PEM(Privacy Enhanced Mail)등이 전자우편의 보안도구로써 사용되고 있다. PGP와 PEM은 보내고자 하는 내용을 암호 알고리즘을 이용하여 암호화함으로서 전자우편을 엽서가 아닌 밀봉된 봉투에 넣어서 보내는 개념이다. 일반우편은 배달도중에 누군가가 몰래 봉투를 뜯어 내용

* 충남대학교
** 시스템공학연구소

을 바꾼 다음에 다시 봉투를 붙여 놓을 수 있지만, PGP나 PEM에서는 특정한 키가 있어야만이 내용을 볼 수 있도록 되어 있기 때문에 기밀성(Confidentiality), 인증(Authentication), 무결성(Integrity), 부인방지(Nonrepudiation) 등의 기능을 지원한다[3]. 여기서 기밀성이란 메일의 내용을 송,수신자이외에는 알아 볼 수 없도록 하는 것을 말하고, 인증이란 메일을 보낸 송신자가 적법한 송신자임을 확인하는 것을 뜻한다. 메일이 송신도중 불법적인 데이타 변조가 발생할 수 있는데 이를 검사할 수 있는 기능으로 무결성 검사가 있으며, 송신자가 M이라는 메일을 보내고 M′을 보냈다고 주장하거나 아예 M 자체를 보낸적이 없다고 주장할 수 있는데 이 경우에는 부인방지 기법을 사용하여 이러한 문제를 해결할 수 있다[4]. 즉, 전자우편 보안을 위해서는 기밀성, 인증, 무결성, 부인방지 기능들이 구현되어야하며, PGP와 PEM에서는 이와 같은 기능을 제공하고 있다.

PEM은 IETF(Internet Engineering Task Force)에서 인터넷 표준으로 준비중인 것으로 높은 보안성을 가지고 있지만, 구현의 복잡성 등의 이유로 현재 널리 사용되지는 않고 있다. 반면에 PGP는 Phil Zimmermann 이라는 사람이 독자적으로 개발하였고, PEM에 비해 보안성에서는 조금 취약한 면을 보이고 있지만 구현이 쉽고, 키인증등의 권한을 한 곳에 집중시키지 않고 사용자 스스로가 가진다는 사실 등에 힘입어 현재 널리 사용되고 있다[1]. 따라서 본 고에서는 PGP를 중심으로 전자우편 보안체제를 살펴보고자 한다.

2장에서는 전자우편 보안체제를 이해하는데 필요한 암호학의 기본 개념을 소개하고, PGP 구조를 3장에서 기술한다. 4장에서는 PGP 도구의 사용방법을 설명하고, 5장에서는 결론을 맺는다.

2. 기본 암호 메카니즘

정보보호를 위한 대책으로서 여러 형태가 있지만, 가장 경제적이면서도 보안수준에 따라 효율적이고 계층적인 보안대책을 제공할 수 있는 것은 기술적인 측면에서의 암호방식을 이용하는 방법이다. 암호방식이란 보호하려는 정보를 작은 길이의 암호키로 관리하는 것을 말하며 정보보호를 위해서는 암호키를 안전하게 관리해야 한다. 암호방식은 암호키의 분배와 관리방법에 따라 관용 암호방식(Conventional cryptosystem)과 공개키 암호방식(Public key cryptosystem)으로 나눌 수 있다.

암호방식의 기본구성을 보면 그림 2-1과 같다. 비밀정보를 전달하려는 송신자는 평문 M을 암호화 알고리즘 E와 암호화 키 Ke를 이용하여 암호문 C를 생성시켜 전달한다. 수신자는 전송된 암호문 C를 수신하여 복호화 알고리즘 D와 복호화 키 Kd를 이용하여 송신자가 보내고자 했던 평문 M을 얻는다. 이때 제삼자가 전송되는 암호문 C를 가로채게 되더라도 복호

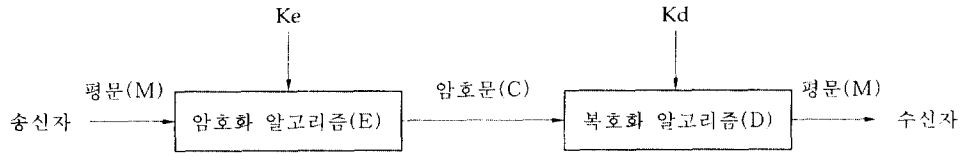


그림 2-1 암호방식의 기본구성

화 알고리즘 D와 복호화 키 Kd를 알지 못하면 C로부터 M을 얻을 수가 없다. 그러나 제삼자가 평문의 성질을 알고 있으면 암호문 C의 통계적 성질과 그 밖의 부대정보를 이용하여 C에서 M을 얻으려고 시도할 수 있다.[5]

2.1 관용 암호방식

관용 암호방식이란 암호화 키 Ke와 복호화 키 Kd가 서로 같은 경우로 대칭(Symmetric) 암호방식이라고도 한다. 비밀정보를 교환하려면 사전에 비밀키(Secret Key) K를 노출되지 않게 나누어 가진 다음, 평문 M을 암호화 알고리즘과 비밀키 K를 이용해 암호문 C를 생성시키고, 수신자는 복호화 알고리즘과 비밀키 K로 평문 M을 얻는다 (그림 2-2 참조)[6].

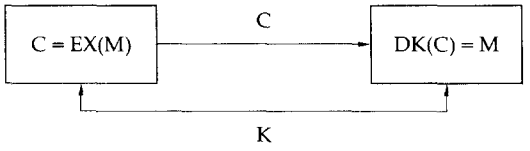


그림 2-2 관용 암호방식

관용 암호방식에서 암호화 알고리즘 E와 복호화 알고리즘 D는 다음의 세가지 조건을 만족해야 한다.[5]

조건 1 : DK(EK(M)) = M 이 임의의 비밀키 K에 대해서 항상 성립해야 한다.

조건 2 : 알고리즘 E, D가 임의의 입력에 대해서도 출력을 계산하는 계산량이 적어야 한다.

조건 3 : 알고리즘 E, D가 공개되어도 비밀키 K를 제삼자가 추측하여 생성하는 것이 매우 어려워야 한다.

조건 1, 2를 만족하는 알고리즘 E, D를 관용 암호방식이라 하고, 부대정보(Side Information)에 대해서 E, D가 조건 3을 만족할 때만이 부대정보에 대하여 안전하다고 한다. 알고리즘 E, D로 구성되는 관용 암호방식이 모든 부대정보에 대하여 안전하다면 알고리즘 E, D를 비밀로 할 필요가 없으므로, 알고리즘을 공개할 수 있어서 대량 생산을 통한 제작비용의 감소나 공동시스템의 이용을 기대할 수 있다. 물론, E, D를 비밀로 하는 것이 안전하지만 다수의 사용자가 보다 편리하고 저렴하게 암호시스템을 구축하기 위해서는 알고리즘의 공개를 불가피하다고 보고 있다. 공개된 E, D의 안전성은 오로지 암호키를 바꾸어 사용함으로써 보장받기 때문에 암호키의 생성, 보관 및 분배에 신중을 기해야 한다. 가입자가 n명일 때 서로 교환해야 할 비밀키의 수가 nC2 = n(n-1)/2로 가입자의 증가에 따라 키의 수가 급증하며 키분배가 문제가 되므로 적절한 암호키 분배방식이 필요하다. 이러한 방식을 사용하는 알고리즘으로는 Lucifer, DES, FEAL, IDEA 등이 있다.

2.2 공개키 암호방식

공개키 암호방식의 대표적인 것으로는 소인수 분해의 어려움을 이용한 RSA방식이 있으며, 관용 암호방식의 단점인 키분배의 문제를 해결한 방법으로 관용방식과 다르게 두개의 키를 사용하고 있다. 기본적인 구조는 그림 2-3과 같다[6].

암호화 키와 복호화 키를 분리하여 암호화 키는 모두에게 공개하고 복호화 키는 각자가 비밀리에 보관하므로 암호화 키는 공개키라 하고 복호화 키는 비밀키라고 한다. 송신자는 수신자의 공개키로 평문 M을 암호화하여 암호문 C를 전송하면 수신자는 자신이 비밀리에 보관하던 비밀키로 평문 M을 복원한다. 즉,

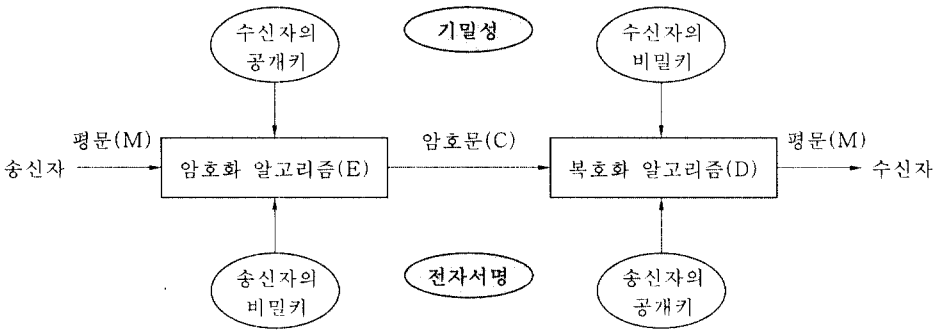


그림 2-3 공개키 암호방식의 기본 구조

메세지의 기밀성(Confidentiality)을 위해서는 수신자의 공개키로 암호화한다. 수신자의 공개키로암호화된 메세지는 수신자의 비밀키를 사용해야 복호화할 수 있다. 그러므로 수신자가 아닌 다른 사람은 메세지의 내용을 알아 볼 수 없게 된다.[5]

메세지의 전자서명(Digital Signature)을 주기 위해선 송신자의 비밀키로 암호화한다. 암호화된 결과는 송신자가 아닌 다른 사용자는 비밀키를 모르기 때문에 만들수 없는 암호문이 된다. 그러나 송신자의 공개키를 가지고 있는 사람들은 복호화를 할 수 있어서 메세지의 기밀성은 기대할 수 없지만 송신자가 보냈다는 증거가 될 수 있어 메세지의 전자서명을 가능하게 한다. 예전의 종이 서류에 의한 문서교환이 컴퓨터 통신망을 통한 교환방식으로 대체되면서 정보의 전달과정에서 상대방의 신분확인, 메세지의 무결성 보장, 사용자의 정당성 및 송수신자간의 분쟁 조정 등이 필요하게 됐다. 전자문서 교환상에서 이러한 기능을 전자적으로 구현하는 것이 전자서명이다. 기밀성, 전자서명등을 모두 이용하려면 먼저 송신자의 비밀키로 암호화한 다음, 다시 수신자의 공개키로 암호화한다. 수신자는 수신자의 비밀키로 복호화하고, 송신자의 공개키로 다시 복호화한다.

가입자는 공개키와 비밀키 두개가 필요하게 되므로 전체 가입자가 n명일 때 필요한 키의 수는 2n이고 실제로 비밀리에 보관해야 하는 비밀키의 수는 n개이므로 관용 암호방식보다 보관해야 할 키의 수가 적고 또한 공개키를 공개하므로 키분배가 필요 없어 키 관리가 용이하다. 그러나 단점으로는 공개키 암호방식이 관용 암호방식보다 암호화/복호화하는데 시간이 많이 소요된다는 점이다. 이러한 이유로 메세지 기밀성을 위한 암호화는 관용방식으로하고, 전자서명과 관용방식에 사용된 비밀키 분배를 위해서는 해쉬함수(Hash Function)를 이용한 공개키방식을 사용하는 것이 요즘의 추세이다.

해쉬함수는 임의의 길이를 가지고 있는 메세지를 입력으로 받아 일정한 길이의 결과로 전환해주는 함수이다. 원래의 메세지 X를 해쉬함수 f를 사용하여 나온 결과를 x라 하는 경우를 식으로 나타내면 f(X) = x 라고 할 때, 안전한 해쉬함수가 되기 위해서는 다음의 조건을 만족해야 한다[7].

조건 1 : 임의의 길이의 메세지를 입력으로 받을 수 있어야 한다.

조건 2 : 고정된 길이의 출력을 만들어야 한다.

조건 3 : 모든 X에 대해서, f(X)의 계산이 쉬워야 한다.

조건 4 : 주어진 x에 대해서 원래의 X를 구할 수 없어야 한다.

조건 5 : X, Y가 같지 않을 때, f(X), f(Y)는 같을 수 없다.

조건 6 : f(X) = f(Y)인 X, Y를 구하기가 어려워야 한다.

전자서명은 메세지를 송신자의 비밀키로 암호화함으로써 이루어지는데 공개키 암호방식은 처리시간이 많이 소요된다는 단점이 있어, 일단 메세지를 해쉬함수를 이용하여 원래보다 짧은 길이로 바꾸어 놓은 다음에 전자서명을 생성한다(보다 자세한 설명은 다음 장을 참조).

해쉬함수의 종류로는 128-bit의 출력을 생성하는 MD4, MD5와 160-bit 출력의 SHA (Secure Hash Algorithm)등이 있다.

3. PGP의 구조

3.1 PGP의 기능

표 3-1 PGP 기능과 알고리즘

Function	Algorithms used
Message encryption (confidentiality)	IDEA, RSA
Message encryption (confidentiality)	RSA, MD5
Copression	ZIP
E-mail compatibility	Radix 64 conversion
Segmentation and Reassebly	

PGP에서는 메세지의 기밀성, 전자서명, 메세지 인증 등을 모두 지원해 주고 있다. PGP가 지원해 주는 기능들과 그 기능을 지원해 주기 위해 사용되는 알고리즘을 모아 놓으면 〈표 3-1〉과 같다[7].

PGP에서는 관용 암호방식으로는 IDEA (Internatinal Data Encryption Algorithm)를 사용하는데 IDEA는 128-bit의 키가 필요하다. 공개키 암호방식으로는 RSA(Rivest Shamir Adleman)를, 해쉬함수로는 MD5를 사용한다. 다음은 앞으로 설명에 필요한 약자이다.

M : 평문
Z : 압축 알고리즘
Ks : IDEA에 사용되는 키
H : 해쉬알고리즘
ER : 공개키 암호화
EI : 관용암호화
DR : 공개키 복호화
DI : 관용복호화
KR : 비밀키
KU : 공개키

3.1.1 기밀성(Confidentiality)

메세지의 기밀성을 위해서는 RSA와 IDEA가 함께 쓰인다. IDEA에 사용될 키를 생성하여 이를 이용하여 메세지를 암호화시킨 후 사용된 키는 수신자의 공개키로 암호화하여 함께 보낸다. 수신자쪽에서는 자신의 비밀키를 이용하여 IDEA 키를 알아낸 후 다시 메세지를 복호화한다. 그림 3-1에서는 이 과정을 보여주고 있다[7]. Z는 압축 알고리즘을 나타낸다.

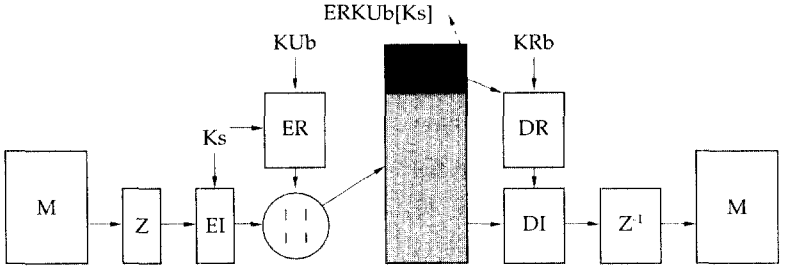


그림 3-1 기밀성을 위한 구조

3.1.2 인증(Authentication) (Digital signature)

인증과 전자서명을 위해서는 RSA와 해쉬함수 MD5가 사용되고 있다. 메세지를 해쉬함수의 입력으로 넣어 128-bit으로 표현한 다음, 이를 송신자의 비밀키로 암호화 한다. 암호화된 부분을 원래의 메세지에 붙여서 보낸다. 수신자는 송신자의 공개키로 전자서명 부분을 복호화한 부분과 원래의 메세지를 다시 해쉬함수에 적용하여 얻은 부분을 서로 비교한다. 서로 동일하지 않을 경우에는 메세지에 대한 신뢰성이 없는 경우로 간주된다. 기본적인 구조는 그림 3-2와 같다[7].

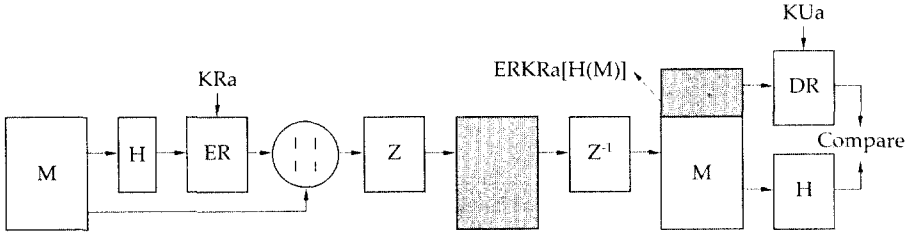


그림 3-2 인증을 위한 구조

3.1.3 기밀성 및 인증

기밀성과 인증을 모두 보장하려면 먼저 인증을 위한 동작을 수행하고 그 결과에 기밀성을 위한 동작을 해주면 된다[7](그림 3-3 참조).

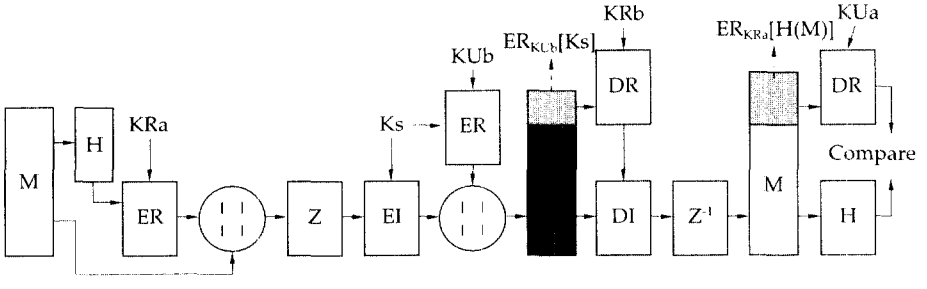


그림 3-3 기밀성과 인증을 위한 구조

3.1.4 압축 (Compression)

메세지의 압축을 위해서는 ZIP 알고리즘을 사용한다[7]. 압축 알고리즘을 사용하느냐 안하느냐 하는 것은 PGP를 사용할 때 config file에서 사용자가 지정해 줄 수 있다. 압축을 할 때에는 암호화를 하기 전에 함으로써 암호화된 결과를 가지고 평문을 추측하는 행동을 더욱 어렵게 한다.

3.1.5 전자우편 호환성 (E-mail compatibility)

PGP를 사용해 생성된 모든 화일들은 연속적인 8-bit의 흐름으로 이루어지지만 대개의 전자우편 시스템은 ASCII 문자만을 인식한다. PGP에서는 Radix-64 conversion을 통하여 세 개의 8-bit를 4개의 ASCII 문자로 변환시켜 기존의 전자우편 시스템과의 호환성 문제를 해결하고 있다[7]. 호환성을 유지하기 위하여 메세지가 약 33%정도 크기가 커지지만 압축률이 50%인 ZIP을 사용할 경우, 메세지가 결과적으로는 약 66%정도로 줄어들게 된다.

3.1.6 분할 및 재결합 (Segmentation & Reassembly)

전자우편 프로그램은 대개 50,000 byte이하의 메세지를 한번에 보낼 수 있다. 그보다 큰 메세지는 작게 나누어 보내게 되는데 PGP에서는 이러한 동작을 자동으로 행해주고 있다. 또한 분할되어 수신된 메세지를 자동으로 재결합시켜주기 때문에 사용자는 메세지 크기에 상관없이 PGP를 사용할 수 있다.

3.2 PGP 메세지의 형식

PGP에서 만들어 낸 메세지의 기본적인 형태는 그림 3-4와 같다[7]. Message component의 Timestamp는 Data가 만들어진 시각이 들어가고 Signature component의 Timestamp는 signature

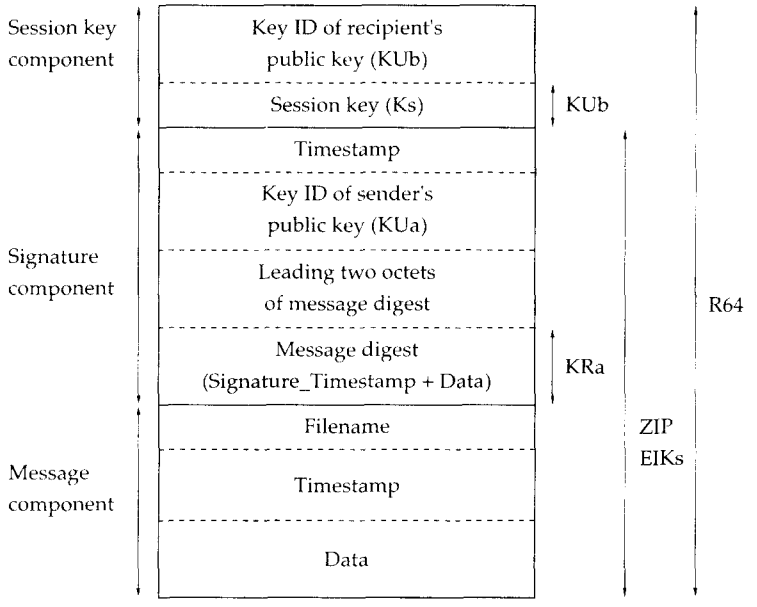


그림 3-4 PGP 메세지의 기본 형태

가 생성된 시각을 나타내고 있다. Leading two octets of message digest는 바로 밑의 암호화된 message digest를 모두 비교하면 시간이 오래 걸리므로 앞의 16bit만 비교하기 위해서 필요한 부분이다. 그림의 오른쪽에는 암호화되는 부분과 그때 사용되는 암호 알고리즘을 표시해 주고 있다.

3.3 PGP메세지의 전송

PGP를 이용하여 송신자가 수신자에게 보낼 메세지를 암호화하는 과정은 그림 3-5과 같다[7].

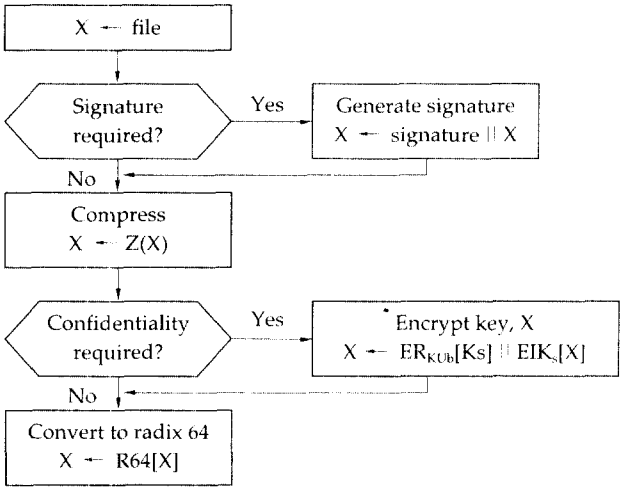


그림 3-5 송신 절차

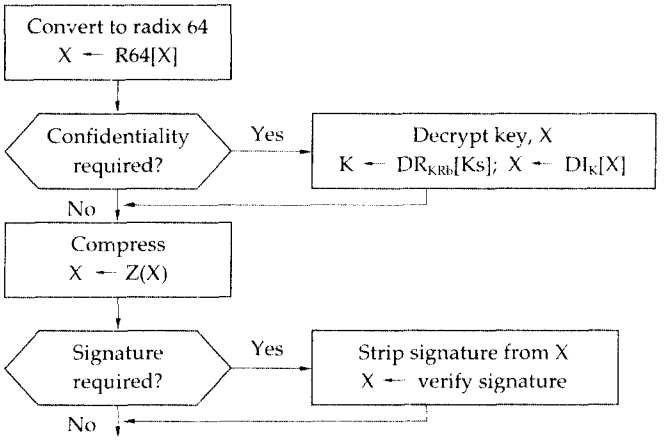


그림 3-6 수신 절차

① 메세지에 전자서명을 첨가할 것인가를 결정한다. 첨가할 경우 원래의 메세지 X에 전자서명 부분을 붙인다.
② 메세지를 압축한다.
③ 기밀성을 요구할 때는 키를 생성하여 IDEA로 메세지를 암호화하고 키를 수신자의 공개키로 암호화하여 붙인다.
④ Radix-64로 변환시킨다.

수신자가 암호화된 메세지를 받았을 때 이를 복호화하는 과정은 그림 3-6과 같다[7].

① Radix-64를 역변환시킨다.
② IDEA로 암호화되어 있을 경우에는 IDEA 키를 수신자의 비밀키로 복호화하여 메세지를 복호화한다.
③ 압축되어 있는 메세지를 풀어 준다.
④ 전자서명이 붙어 있을 경우에는 송신자의 공개키를 이용하여 확인한다.

3.4 PGP관련 화일들

PGP의 기능을 수행하기 위해서 필요로 하는 화일들을 알아보면 다음과 같다.

가. pubring.pgp

자신의 공개키와 다른 사람의 공개키를 함께 보관하는 화일로 Public key ring이라 한다. 이진형태로 저장되어 있기 때문에 일반 명령으로는 내용을 확인할 수 없고 PGP를 사용해야 한다. 공개키가 저장되는 형식은 표3-2와 같다[7].

표 3-2 Public key ring

Timestamp	Key ID	Public key	Owner trust	User ID	Key Legitimacy	Signature	Signature trust
Ti	KUi mod 2**64	KUi	trust flagi	Useri	trust flagi	ERj(H(KUi))	complete

Timestamp : 이 키가 만들어진 시각
Key ID : KUi mod 2^{64}로 계산되며 키를 식별하는데 사용된다.
Public key : 공개키가 저장된다.
User ID : 사용자의 이름을 나타낸다.
Owner trust, Key legitimacy, Signature, Signature trust는 다음 섹션에서 설명한다.

나. secring.pgp

사용자 자신의 비밀키만을 저장하며 Private key ring이라 한다. 비밀키는 남에게 보이지 않게 하기 위해서 사용자가 입력하는 pass-phrase라는 것을 IDEA의 키로 사용하여 암호화해 놓는다. pubring.pgp와 같은 이진형태로 저장되며 형식은 표 3-3과 같다[7].

표 3-3 Private key ring

Timestamp	Key ID	Public key	Encrypted private key	User ID
Ti	KUi mod 2**64	KUi	EIH(Pi)[KRi]	user i

Timestamp : 이 키가 만들어진 시각
Key ID : Kui mod 2^{64}로 계산되며 키를 식별하는데 사용된다.
Public key : 공개키가 저장된다
Encrypted private key : 비밀키가 암호화되어 저장된다.
User ID : 사용자의 이름을 나타낸다.

다. randseed.bin

공개/비밀키를 생성하는 데에 필요한 두개의 random prime number를 선택하는 데에 이용되는 seed값과 IDEA 키의 생산에 쓰일 seed값을 보관하고 있다[1]. 사용자가 공개키와 비밀키를 생성하는 과정에서 PGP는 사용자로 하여금 무작위하게 키보드를 타이핑하게 한다. 이때 눌려지는 키보드의 값과 시간이 이용되어 seed값을 생성한다.

라. config.txt

PGP를 사용할 때 사용자가 자신의 형편에 맞추어서 사용할 수 있도록 여러 값들을 변경할 수 있다[1]. 예를 들어 암호화하는 과정에서 압축 알고리즘을 사용할 것인가 안할 것인가를 결정해 주는 것과 같은 사용방식을 지정해 줄 수 있다.

3.5 Passphrase와 fingerprint

PGP를 사용할 때 가장 중요하게 여겨야 할 것이 패스워드 개념의 passphrase이다. Passphrase로 쓰일 수 있는 것에 대한 제한사항은 없으며 길이에도 제한이 없다. 문자 하나에서 책 한권의 길이까지 입력이 가능하다. 입력된 passphrase는 MD5 해쉬함수를 거치면서 128-bit으로 변환되며, 이 128-bit가 IDEA의 키로 사용된다[7]. Passphrase는 두가지의 용도로 사용되는 데 문서를 암호화할 때와 비밀키를 암호화할 때이다.

문서 M을 자신의 컴퓨터에 암호화하여 저장하려고 할 때, 사용자는 passphrase를 입력하는데 이것이 IDEA의 키로 사용되어 암호화된다 (EIH[passpharse][M]). M을 다시 복호화하기 위해서는 암호화할 때 사용한 passphrase를 똑같이 입력해야 한다. 문서를 PGP를 사용하여 암호화시킬 때 일어나는 문제가 문서 하나하나에 각각 다른 passphrase를 줄 것인가 아니면 하나의 passphrase를 가지고 모든 문서를 암호화할 것인가 하는 것이다. 전자의 경우는 보안성이 매우 좋은 방법이지만 문서가 많아지면 각각의 passphrase를 기억하고 있는 것도 어려운 일이 된다. 그러다 보면 passphrase들을 어딘가에 기록해 놓을 수가 있는 데 이렇게 되면 오히려 보안상의 취약점으로 작용된다. 후자의 경우는 하나의 passphrase를 기억하면 되기 때문에 기록해 놓을 위험은 없지만 어쩌다가 passphrase가 노출되면 모든 화일이 위험해지는 경우가 발생할 수 있다.

비밀키는 secring.pgp에 보관될 때 암호화된 형태로 저장된다. 여기서도 암호화의 키로 passphrase가 쓰인다 (EIH[passphrase][secret_key]). 그러므로 암호화나 복호화를 위해 비밀키를 사용할 때 사용자는 passphrase를 정확히 입력해야 한다.

Fingerprint는 공개키를 MD5를 이용하여 128-bit의 형태로 나타낸 형태이다. 128-bit이기 때문에 32개의 16진수로 표현된다. Fingerprint는 각 키에 대해서 고유한 값을 가지고 있기 때문에 키의 신원을 확인할 수 있는 도구로 이용된다.

3.6 키 인증

PGP 사용에 있어서 가장 큰 문제점이라 할 수 있는 것은 사용자 A의 키라고 받은 키가 정말로 A의 키인지를 확인하는 방법이 어렵다는 것이다[1]. 만약 B가 A인 것처럼 꾸며서 공개키를 보내면 사용자는 A가 아닌 B에게 메세지를 보내는 경우가 발생할 수 있다. 또는 B가 A가 주는 것처럼 하면서 거짓 정보를 줄 수도 있다. 이러한 위험요소를 PGP에서는 사용자들간의 신뢰도를 계산하여 키의 진위를 결정하게 한다.

먼저 간단하게 설명하면 아이들이 낯선 사람을 봤을 때의 행동과 같은 원리이다. 어린 아이들은 처음 보는 사람을 만나면 그 사람을 직접 평가하지 않고 자신이 믿는 부모나 친구들의 반응을 살핀다. 그래서 부모가 낯선 사람을 믿는다고 판단되면 아이도 낯선 사람을 믿게 되고, 부모가 낯선 사람을 피하면 아이도 피한다. 이와 같이 PGP에서도 사용자가 모르는 사람의 키를 받았을 때 키에 달려 있는 서명을 보고 자신이 믿고 있는 사람의 서명이 달려 있으면 지금 받은 키를 믿고 사용하고 그러한 서명이 없는 키를 사용하려 할 때에는 경고메세지를 보여 준다[1].

이러한 키의 인증과정에 필요한 값들은 Public key ring의 Owner_trust, Key_legitimacy, Signature, Signature_trust 등이다.

가. Key_legitimacy

사용자가 A의 키라고 되어 있는 키를 받았을 때 이 키가 어느 정도 A의 것인지를 나타내는 필드이다. 값은 다른 필드의 값을 참고하여 PGP가 자동으로 계산하며, 가질 수 있는 값은 다음과 같다.

- undefined : 사용자가 알고 있는 사람의 서명이 없을 때
- untrusted : 사용자가 불신하고 있는 사람의 서명이 있을 때
- marginal trust : 사용자가 어느 정도 믿는 사람의 서명이 있을 때
- complete trust : 사용자가 완전히 믿는 사람의 서명이 있을 때

complete trust를 제외한 값을 가지고 있는 키는 사용할 때 경고메세지가 보여진다.

나. Signature trust

어떤 키가 계속 쓰이다 보면 사용자들이 그 키에 자신의 서명을 붙이게 된다. 서명을 붙이는 행동은 자신의 이름을 걸고 그 키의 진위를 증명해주는 의미이다. A의 키에 자신의 서명을 붙여 놓게 되면 자신을 믿는 다른 사람이 A의 키를 받았을 때 그 서명을 보고 A의 키라는 데에 의심을 갖지 않고 사용하게 된다. 그러므로 어떤 키에 자신의 서명을 붙일 때는 키의 진위를 확실하게 확인한 후에 해야 한다[2].

키에 signature가 붙을 때 그 서명을 붙인 사람에 대한 신뢰도가 또한 필요하다. 그러한 정보는 Signature_trust 에서 표현된다. 예를 들면 A가 B의 이름이 붙은 키를 받았을 때 C의 서명이 붙어 있다고 하면 A가 C를 어느 정도 신뢰하는 가를 찾아 본다. 이는 Owner_trust 부분에서 가져 오는 데 Owner_trust 에 대한 설명은 다음에 있다. Signature_trust 에 들어가는 값은 다음과 같으며 PGP가 자동으로 계산해 준다.

- undefined trust
- untrusted to sign other keys
- usually trusted to sign other keys
- always trusted to sign other keys
- ultimate trust

어느 키의 Signature_trust에 always trusted가 하나만 있으면(사용자가 완전히 믿는 사람 1명 이상이 서명을 해놓았으면) 그 키의 Key_legitimacy는 complete_trust 값을 가지게 되고, usually trusted가 2개 이상이(config.txt의 해당 변수의 값에 따라 달라짐) 있을 때도 complete_trust 값을 가지게 된다. 1개만 있을 때에는 Key_legitimacy에 marginal_trust 값이 들어가게 된다.

다. Owner_trust

사용자가 A에 대해서 중개자로서의 평가를 하는 부분이다. 즉, A의 서명이 붙어 있는 B의 키를 받았을 때 사용자는 A의 서명을 보고 B의 키를 신뢰할 수 있느냐 하는 것을 나타낸다. 이는 사용자가 직접 입력해 주는 부분인데 가질 수 있는 값은 다음과 같다.

- undefined trust
- untrusted to sign other keys
- usually trusted to sign other keys
- always trusted to sign other keys
- ultimate trust

Signature_trust에 들어가는 값과 같은 값을 가지고 있는데 Owner_trust에 있는 값을 Signature_trust로 옮기기 때문이다. 결국 Owner_trust에 있는 값을 Signature-trust로 가져 가고 이를 참조하여 Key_legitimacy를 계산하게 된다.

3.7 키 인증의 예

앞에서 설명한 키 인증의 방법을 대표적인 다섯가지의 예제를 통해서 설명한다.

가. A(사용자 자신)가 새로운 키를 만들었을 때

사용자가 자신의 키를 만들었을때 Owner_trust, Key_legitimacy에 ultimate, complete 값이 자동으로 지정된다. 표 3-4는 이 경우의 A의 public key ring을 나타낸다.

나. Signature가 없는 B의 키를 받았을 때

Signature가 달려있지 않은 키를 받았을 때에는 Owner_trust뿐만 아니라 Key_legitimacy에도 사용자가 값을 입력해 주어야 한다.

표 3-4 사용자의 Public key ring

.....	Owner_trust	User ID	Key_legitimacy	.....
.....	ultimate	A	complete	

자동으로 지정

표 3-5 Signature가 없는 키를 수신하였을 경우

.....	Owner_trust	User ID	Key_legitimacy	.....
.....	user_defined	B	user_defined	

undefined
untrusted
marginally trusted
completely trusted

if you certify this key yourself
then complete
o.w. undefined

Key_legitimacy에 값을 입력한다는 것은 사용자가 그 키에 signature를 붙인다는 것을 의미한다 〈표 3-5〉.

다. C가 B의 키에 signature를 붙였고 A가 C의 키를 갖고 있을때

C의 owner_trust에 있는 값이 B의 signature_trust로 옮겨가고 B의Key_legitimacy를 계산하는 데 사용된다. 표 3-6에서는 C가 A로부터 완전한 신뢰를 받고 있기 때문에 C의 signature가 있는 B의 키도 A에게 받아들여진다.

표 3-6 A가 C의 키를 가지고 있을 경우

Key ID	Owner_trust	User ID	Key_legitimacy	Signature	Sig. Trust
3F1DEA81	user_defined	B	complete	C's signature	complete
A4171B2D	complete	C			

라. C가 B의 키에 signature를 붙이고 A가 C의 키를 갖고있지 않을 때

C가 B의 키에 signature를 붙였지만 A가 C가 누군지 모를 땐 B에 대한 값을 A가 직접 입력해 주어야 한다 〈표 3-7〉.

표 3-7 A가 C의 키를 가지고 있지 않을 경우

Key ID	Owner_trust	User ID	Key_legitimacy	Signature	Sig. Trust
3F1DEA81	user_defined	B	undefined	C's signature	

마. C와 D가 B의 키에 signature를 붙였을 때

두사람 이상의 signature가 붙어 있을 때에도 Key_legitimacy를 계산하는 방법은 동일하다. 만약 A가 D의 키를 가지고 있지 않을 땐 C의 값만이 이용되고 D의 키를 가지고 있을 땐 두개의 값이 모두 계산에 이용된다. 표 3-8에서는 C와 D의 Owner_trust가 각각 marginally trusted이기 때문에 B의 Key_legitimacy가 complete으로 결정되었다 (앞에서와 같이 config.txt 변수 값에 따라 달

표 3-8 두개 이상의 signature가 있을 경우

Key ID	Owner_trust	User ID	Key_legitimacy	Signature	Sig. Trust
3F1DEA81	user_defined	B	complete	C's signature B's signature	marginal margianl
A4171B2D	margianl	C			
B72F912A	marginal	D			

라짐). 만약 C의 owner_trust는 marginally trusted이고 D는 undefined이나 untrusted를 갖는다면 B의 Key_legitimacy는 marginally trusted를 갖게 된다.

4. PGP의 명령어

4.1 PGP를 구할 수 있는 곳

PGP는 최초의 버젼인 1.0에서부터 시작해 현재 2.7까지 나와 있지만 인터넷에서 구해서 사용할 수 있는 것은 2.6.i와 2.6.2.이다. 2.6.i와 2.6.2i의 가장 큰 차이는 공개키와 비밀키의 크기에 있다. 2.6.i의 경우 최대 1280bit까지 만들 수 있지만 2.6.2i는 2048bit까지 만들 수 있다. 이러한 인터넷 버젼을 구할 수 있는 곳은 국내의 다음과 같은 곳에서 구할 수 있다.

ftp: / / what.snu.ac.kr / pub / pgp /
ftp: / / venus.etri.re.kr / pub / pgp+elm /
ftp: / / juno.kaist.ac.kr / pub / security / pgp /

4.2 PGP의 옵션과 기능들

UNIX에서 PGP를 사용하는 명령어는 "pgp" 단 한가지다. 이 하나의 명령어에 하고자 하는 기능에 맞는 옵션을 주어서 사용한다.

4.2.1 문서의 암호화

가. 문서를 암호화해서 저장

%〉 pgp -c textfile

문서를 암호화해서 다른 곳으로 보내지 않고 자신의 컴퓨터에 보관할 때 사용하려면 옵션으로 "-c"를 사용한다. 사용자는 암호화에 필요한 키값을 만들기 위해 passphrase를 입력해야 한다. 암호화된 화일은 원래의 화일이름에 ".pgp"나 ".asc" 확장자가 붙는다.

나. 원래의 평문을 지우며 암호화하기

%〉 pgp -cw textfile

"-c" 옵션을 사용하여 평문을 암호화한다 하더라도 평문이 남아 있으면 보안상의 취약점이 될 수 있다. 암호문을 만든 다음 평문을 지운다고 하더라도 복구가 가능하기 때문에 평문과 암호문의 쌍을 노출시킬 수가 있다. "-w" 옵션은 암호화를 한 다음 평문에 랜덤비트를 덮어 씌운 다음 디렉토리에서 지워 버린다.

다. 암호문을 평문으로 복호화하기

%〉 pgp textfile.asc

PGP에서는 어떠한 형태로든지 암호화된 문서를 복호화할 때에는 "pgp" 명령 뒤에 아무런 옵션없이 암호문의 이름을 써준다. 만약 복호화 과정에서 passphrase가 필요할 때는 PGP는 passphrase를 입력하라는 메세지와 함께 입력을 기다린다.

4.2.2 키 관리

가. PGP 키 만들기 (Key Generation)

%〉 pgp -kg

•다른 사람들과 PGP를 이용하여 전자우편을 사용하려면 자신의 키가 필요하게 된다. 자신의 키를 새로이 만들 때는 "-kg" 옵션을 사용한다. 몇가지의 입력이 필요한 데 순서와 내용은 다음과 같다.

키의 크기를 결정 : 512bits, 768bits, 1024bits의 세가지 크기가 보여지는 데 크기가 클 수록 생성시간은 오래 걸리지만 사용하기에 안전하다. 사용자가 3종류의 크기 이외에 특정한 크기의 키가 필요하면 직접 크기를 입력해도 된

다.(2.6.2i에서는 2048bit까지 가능하다.)

User ID를 입력 : User ID로 사용되기 위한 특별한 형식은 없지만 통상적으로 사용자의 real name 뒤에 전자우편 주소를 붙여 준다.

예〉 Hyun_Dong Park 〈 hdpark @ esperosun.chungnam.ac.kr〉

Passpharse를 입력 : 비밀키를 secring.pgp에 보관할 때 암호화에 필요한 키값을 입력받는 부분이다. 나중에 이 비밀키를 사용하려면 여기서 입력하는 passphrase를 정확히 입력해 주어야 하므로 확인을 위해서 두번 입력하게 한다.

Random data를 입력 : 공개/비밀키와 IDEA키의 생성에 필요한 seed값을 얻는 과정이다. 임의의 숫자가 보여지는 데 사용자가 키보드를 칠 때마다 줄어 들어서 0이 되면 끝나게 된다. 사용자가 타이핑하는 키보드의 값과 시각이 이용되어 seed값을 생성한다.

이러한 과정들을 지나게 되면 한 쌍의 키가 생성되어 "secring.pgp"와 "pubring.pgp"에 저장된다.

```
%〉 pgp -kg

Pick your RSA key size:
1) 512 bits- Low commercial grade, fast
   but less secure
2) 768 bits- High commercial grade,
   medium speed, good security
3) 1024 bits- "Military" grade, slow,
   highest security
Choose 1, 2, or 3, or enter desired number
of bits: 3

You need a user ID for your public key.
The desired form for this user ID is your
name, followed by your E-mail address
enclosed in 〈angle brackets〉, if you have
an E-mail address.
For example : John Q. Smith 〈12345.
6789@compuserve.com〉

Enter a user ID for your public key :
Hyun_Dong Park 〈hdpark @ esperosun.
chungnam.ac.kr〉

You need a pass phrase to protect your
RSA secret key.
Your pass phrase can be any sentence or
phrase and may have many words,
spaces, punctuation, or any other printable
characters.
Enter pass phrase :
/* 화면에 안 보입니다. */
Enter same pass phrase again:
/* 역시 안 보입니다. */

We need to generate 624 random bits.
This is done by measuring the time
intervals between your key strokes.
Please enter some random text on your
keyboard until you hear the beep : 624 .
/* 아무 키나 마구 치세요... */

0 * -Enough, thank you.
Note that key generation is VERY
lengthy process.
.....+++++++.....+++++.....

Key generation completed.
/* 키 생성 끝. */
%〉
```

PGP 키에 관련한 도움이 필요하면 "pgp -

k"를 입력하면 도움말을 볼 수 있다.

나. 키 구경하기 (Key View)

%〉 pgp -kv

자신의 키 화일에 어떠한 키들이 있는 가를 보기 위해선 "kv" 옵션을 사용한다. 공개키를 볼 때는 위의 예와 같이 하면 되지만 비밀키를 보기 위해선 옵션 뒤에 secring.pgp 화일의 path를 써 주어야 한다. "-kv" 옵션을 사용했을 때 보여지는 필드들의 의미는 다음과 같다.

Type : 공개키인가 비밀키인가를 보여준다. sec면 비밀키를 나타내고, pub이면 공개키를 나타낸다.
Bits : 키의 크기를 나타낸다.
Key ID : 키를 2^{64}로 mod한 결과로 키의 확인에 이용된다.
Date : 키가 생성된 날짜가 입력된다.
User ID : 키 주인의 ID가 들어간다.PGP에서 키를 고를 때 인덱스로 사용된다.

%〉 pgp -kv

Type	pub
bits / KeyID	1024 / 903C9265
Date	1995 / 07 / 15
UserID	Hyun Dong_Park
	〈esperosun. chungnam.ac.kr〉

다. 키 속성을 변경하기 (Key Edit)

%〉 pgp -ke

하나의 키에 붙어 있는 User ID, Passphrase 등을 키 속성이라 하는데 이러한 내용을 변경시킬 수 있다. 사용자 자신의 공개키에 대해서 할 때는 Passphrase나 User ID를 변경시킬 수 있고, 다른 사람의 공개키에 대해서는 Owner_trust를 변경시킬 수 있다. User ID를 바꾸어야 하는 이유는 전자우편주소가 변경되었거나 처음 키를 생성할 때 잘못 입력했을 경우이다. Passphrase를 바꾸어야 할 때는 누군가에게 자신의 passpharse가 노출되었다는 의심이 들 때이다.

하나의 키에 여러 개의 User ID를 사용할 수가 있다. 한 사람이 여러 컴퓨터에 계정을 가지고 있을 때에 유용한 데, 여러 개의 User ID들 중에서 하나를 지우고 싶을 때는 "-kr" 옵션을 사용한다. "-kr" 옵션은 원래 키를 지울 때 사용하는 옵션이지만 여러개의 User ID를 가지고 있는 키일 때는 키는 놔두고 User ID만을 삭제할 때 쓰인다.

라. 다른 사람에게 보낼 키를 뽑아 내기 (Key Extract)

%〉 pgp -kx UserID filename

자신의 키를 다른 사람들에게 전파시키는 방법으로 키 화일 전체를 복사해서 옮기는 방법도 있지만 이 방법은 여러 종류의 위험이 뒤따른다. 그러므로 자신이 보내고자 하는 키만을 빼내어서 보내어야 한다. 이럴 때 쓰는 옵션이 "-kx" 옵션이다. 옵션 뒤에 나오는 것은 빼낼려고 하는 키의 User ID와 키를 복사해서 담을 화일의 이름이다.

키를 담고 있는 화일을 얻었으면 이 화일을 메일 프로그램을 통해 목적지로 보내야 한다. 이러한 과정을 하나의 코맨드 라인에서 행할 수 있다. "-f" 옵션을 사용한다.

%〉 pgp -kxaf HD_PARK | mail kskim @ esperosun.chungnam.ac.kr

위와 같이 하면 키가 임시화일로 옮겨져서 파이프를 통해 전자우편의 입력으로 들어간다. "a" 옵션은 Radix 64 conversion을 하는 것이다.

하나의 화일에 여러 개의 키를 담아 보내야

하는 경우가 있다. 이런 경우 위의 방법대로 하나의 화일에 User ID만을 바꾸어 가면서 계속 실행시키면 되지만 빼내야 할 키의 수가 늘어나면 번거로운 작업이 된다. 다음과 같은 script를 작성하면 한번에 여러 사람의 키를 빼낼 수가 있다.

```
%> foreach i (HD_PARK Kisu Taegap)
?pgp -kx $i rkeys.pgp
?end
%> pgp -a rkeys.pgp
```

마. 다른 사람의 키를 자신의 키 화일에 올려 놓기 (Key Add)

```
%> pgp -ka key_filename
```

누군가가 당신에게 키를 보냈을 경우이다. 키를 받았으면 자신의 키 화일에 올려 놓아야 하는 데 이때 쓰는 옵션이 "-ka" 옵션이다. 이 명령을 실행하면 들어있는 키에 대한 정보가 나오고 이 키에 직접 signature를 붙일 것이냐고 물어 보는데 yes를 하면 사용자의 signature를 붙이게 되고 no를 하면 그대로 키만 옮겨 놓게 된다.

바. 키 지우기 (Key Remove)

```
%> pgp -kr
```

사용자 자신의 공개키 화일에 있는 자신의 키나 다른 사람의 키를 지울 때 쓰는 옵션이 "-kr" 옵션이다. 자신의 키일 경우에는 매칭되는 비밀키도 지울 수 있다.

4.2.3 암호화된 문서의 송 수신

가. 메일을 암호화시켜 송신하기 (Encryption)

```
%> pgp -ea message receiver's UserID
```

메일을 암호화해서 송신하려면 몇 단계의 과정을 거쳐야 한다.

① 에디터를 이용하여 화일을 만든다.
② 수신자의 공개키를 찾는다.
③ IDEA 키를 생성하여 화일을 암호화한다. (PGP가 자동으로 하기 때문에 사용자는 이 단계를 보지 못한다.)
④ 사용된 IDEA 키를 수신자의 공개키로 암호화한다.
⑤ 메일 프로그램으로 송신한다.

수신자의 공개키로 암호화할 때에는 먼저 자신의 키 화일에 수신자의 공개키가 들어 있는 가를 "-kv" 옵션으로 확인하고 평문의 이름 뒤에 수신자의 공개키의 User ID를 써준다. 이렇게 하면 평문이 IDEA 알고리즘으로 암호화되고 IDEA 키는 수신자의 공개키로 암호화되어 함께 보내진다. 그러므로 일단 수신자의 공개키로 암호화된 문서는 암호화를 시킨 사용자도 복호화를 할 수가 없다. IDEA 키가 생성되는 과정은 사용자에게 보여지지 않는다.

```
%> pgp -ea plain Phil
Recipient's public key(s) will be used to encrypt.
Key for user ID : Phil <phil@pgp.com>
512-bit key, Key ID 43744F09, created 1994/07/07
WARNING : Because this public key is not certified with a trusted signature, it is not known with high confidence that this public key actually belongs to: "Phil <phil@pgp.com>".
/* 완전히 인증된 키가 아닐 경우 나오는 경고 메세지 */

Are you sure you want to use this public key (y/N)? y
```

```
Transport armor file name: plain.asc
%〉
```

만약 코맨드 라인에서 Phil 이라는 User ID를 명시해 놓지 않으면 사용자에게 User ID를 묻는 단계가 첨가된다.

위와 같은 방법은 여러 단계를 거치지만 앞에서 키를 빼낼 때처럼 하나의 코맨드 라인에서 문서작성, 암호화, 메일전송을 동시에 실행시키는 방법이 있다.

```
%〉 pgp -eaf Kisu | mail kskim @
esperosun.chungnam.ac.kr
Kisu!
Hi!
Bye!
^D
%〉
```

위와 같은 방법으로 하면 하나의 코맨드 라인에서 모든게 이루어지고 사용자가 입력한 내용이 따로 화일에 저장되지 않으므로 평문이 남지 않아 더욱 보안성이 좋다.

나. 암호화된 메일을 수신하여 복호화하기

```
%〉 pgp cipher.asc
```

암호화된 메일을 받았으면 화일로 저장시키고 옵션없이 pgp 명령 뒤에 화일의 이름을 써준다. 수신자의 공개키로 암호화되어 있기 때문에 복호화하는 데에는 수신자의 비밀키가 필요하므로 passphrase를 물어 본다. 수신자가 passphrase를 맞게 입력하면 암호화에 사용됐던 IDEA 키를 복호화하여 화일을 복호화한다. 복호화된 화일의 이름은 암호문화일의 이름에서 .pgp나 .asc가 생략되어 있는 형태이다.

```
%〉 pgp cipher.asc
File is encrypted. Secret key is required to
read it.
Key for user ID: Hyun Dong _ Park
〈hdpark@esperosun.chungnam.ac.kr〉
/* 수신자의 공개키 */

1024-bit key, Key ID 903C9265, created
1995/07/15

You need a pass phrase to unlock your
RSA secret key.
Enter pass phrase: .Pass phrase is
good.Just a moment....
/* 수신자의 비밀키를 쓰기 위해서 */
Plaintext filename: cipher
%〉
```

다. 화일을 복호화하여 내용만 보기

```
%〉 pgp -m cipher.asc
```

암호문을 옵션 없이 복호화하면 복호화된 평문 화일이 생기는 데 이는 평문과 암호문의 한쌍을 만드는 것이기 때문에 사용자의 유의가 필요하다. 이러한 문제점을 해결하기 위해서 "-m" 옵션을 사용하는 데 이렇게 하면 복호화된 내용이 화면에 보여질 뿐이고 다 보고난 후에 하드 디스크에 담을 것인지를 물어본다. 여기서 no를 입력하면 복호화 된 내용은 디스크에 담겨지지 않는다.

라. 여러 사람에게 암호화해서 송신하기

```
%〉 pgp -ea textfile receiver1 receiver2
```

위와 같은 형식으로 실행하면 메세지의 암호화는 한번만 이루어지고 IDEA 키는 receiver1과 receiver2 두 사람의 공개키로 각각 암호화되어 메세지 위에 첨가된다. 수신자의 공개키로 암호화를 하면 송신자도 복호화를

할 수가 없다고 했는데 위와 같은 형식으로 수신자의 리스트에 송신자의 User ID를 포함시키면 송신자도 복호화를 할 수가 있다.

4.2.4 전자서명

가. 문서에 전자서명 첨가하기 (Signature)

%〉 pgp -s textfile

문서에 signature를 달게 되면 수신자가 문서를 받아 보고 전송 도중에 불법적인 변경이 일어 났는지 확인할 수 있고 또한 송신자를 인증할 수도 있다. 또한, signature를 보관하였다가 송신자가 전송한 사실 혹은 내용자체를 부인할 경우 증거자료로 사용할 수 있다. 문서에 signature를 붙이는 단계는 다음과 같다.

① 문서를 MD5를 이용해 128-bit의 길이로 바꾼다.
② 128-bit를 송신자의 비밀키를 이용해 암호화하는데 이것이 전자서명이다.
③ 전자서명 부분을 원래의 평문에 붙인다.

%〉 pgp -s textfile
A Secret key is required to make a signature.
You specified no user ID to select your secret key, so the default user ID and key will be the most recently added key on your secret key ring.
/* 사용할 키를 지정해 주지 않을 경우에는 가장 늦게 만들어진 비밀키를 사용한다. */
You need a pass phrase to unlock your RSA secret key.
Key for user ID "Hyun Dong_Park 〈hdpark@esperosun.chungnam.ac.kr〉"

Enter pass phrase: .Pass phrase is good.
Key for user ID: Hyun Dong_Park 〈hdpark@esperosun.chungnam.ac.kr〉
1024-bit key, Key ID 903C9265, created 1995/07/15
Just a moment....
Clear signature file: textfile.asc
%〉

"-s" 옵션을 사용하면 송신자의 비밀키가 필요하기 때문에 passphrase를 입력해야 한다. 실행이 끝나면 원래의 화일 이름에 .pgp 확장자가 붙은 이름으로 암호문 화일이 생긴다.

나. 전자서명을 확인하기

수신자쪽에서 행하는 동작의 단계는 다음과 같다.

① 송신자의 비밀키로 암호화되어 있기 때문에 송신자의 공개키로 복호화 한다. (복호화의 결과는 송신자가 만들었던 128-bit의 내용이다.)
② 같이 전송된 원래의 메세지의 MD5 결과를 만든다.
③ 두개의 128-bit의 내용을 비교한다. 일치하면 전송도중 아무런 변경이 일어나지 않았다는 것과 송신자가 확실하다는 것을 인증할 수 있다.

%〉 pgp textfile.asc
File has signature. Public key is required to check signature.
Good signature from user "Hyun Dong_Park 〈hdpark @ esperosun.chungnam.ac.kr〉".
Signature made 1995/08/12 13:34 GMT

Plaintext filename: text
%〉

"-s" 옵션을 사용한 문서를 복호화하는 데에는 송신자의 공개키가 쓰이므로 수신자가 송신자의 공개키를 가지고 있지 않으면 PGP가 signature를 확인해 줄 수 없다.

다. 여러개의 비밀키 중에서 하나를 선택하여 전자서명 첨가하기

%> pgp -s textfile -u Jeff

사용자가 여러 개의 비밀키를 가지고 있다면 암호화에 어느 비밀키를 사용할 것인가를 지정해 줄 수 있다. 비밀키를 하나만 가지고 있다면 문제가 없지만 여러 개가 있다면 가장 늦게 만들어진 비밀키가 사용된다. 어느 특정한 비밀키를 사용하도록 해주는 것이 "-u" 옵션이다. "-u" 옵션 뒤에 쓰여지는 User ID를 갖는 비밀키가 암호화에 사용된다.

라. 전자서명을 첨가하여 암호화 하기 (Signature and Encryption)

%> pgp -se textfile

전자서명만을 붙이는 것은 전자서명에 사용된 비밀키에 매칭하는 공개키를 가진 사람은 복호화를 할 수 있기 때문에 기밀성을 보장해 줄 수가 없다. 그러므로 전자서명을 붙인 메세지를 IDEA 알고리즘으로 다시 암호화한다. 사용된 IDEA 키는 수신자의 공개키로 암호화된다. "-s" 옵션과 "-e" 옵션을 함께 사용하면 먼저 전자서명을 붙이고 다시 암호화한다. 송신자의 비밀키와 수신자의 공개키가 모두 쓰이기 때문에 사용되는 비밀키의 passphrase와 공개키의 User ID를 지정해 주어야 한다.

```
%> pgp -sea textfile freedom
A secret key is required to make a signature.
You need a pass phrase to unlock your RSA sceret key.
Key for user ID "Hyun Dong_Park <hdpark@esperosun.chungnam.ac.kr>".

Enter pass phrase:    .Pass phrase is good.
Key for user ID: Hyun Dong_Park <hdpark@esperosun.chungnam.ac.kr>
1024-bit key, Key ID 903C9265, created 1995/07/15
Just a moment.....

Recipient's public key(s) will be used to encrypt.
Key for user ID: Freedom Fighter <ff@freedom.net>
1024-bit key, Key ID FB704769, created 1994/08/26

Transport armor file: textfile.asc
%>
```

마. 분리된 전자서명을 만들기 (Sign By itself)

%> pgp -sb textfile

앞에서 설명한 전자서명을 만드는 것과 모든 것은 같지만 앞에서 말한 것과 틀린 것은 전자서명 부분이 원래의 메세지와 분리되어 생성된다는 것이다. 만약 두 사용자가 크기가 큰 동일한 문서를 가지고 있고 두 문서의 동일함을 확인하려 할 때 전자서명을 첨가한 문서를 다시 전송하지 않고 전자서명 부분만을 전송하여 수행할 수 있기 때문에 시간을 줄일 수 있다.

■ 기본적인 암호화 통신 단계

여기서는 지금까지 보아 온 pgp의 기능들을 이용하여 A와 B가 암호화된 통신을 하는 가장 기본적인 절차를 본다. 〈표 4-1〉

표 4-1 기본적인 암호화 통신 단계

A	B
자신의 비밀키/공개키를 생성 Pgp -kg	자신의 비밀키/공개키를 생성 Pgp -kg
B의 공개키를 요구	
	자신의 공개키를 B에게 전송 Pgp -kxaflmail A@address
B의 공개키를 키링에 첨가 Pgp -ka B_key	
자신의 공개키를 B에게 전송 pgp - kxaf AlB@address	
Fingerprint로 공개키의 진위를 확인 pgp -kvc	Fingerprint로 공개키의 진위를 확인 pgp -kvc
문서에 전자서명을 첨가하고 B만이 볼 수 있도록 암호화하여 B에게 전송 pgp -sea message B mail B@address < message. asc	
	A가 보낸 문서를 복호화하여 전자서명을 확인 pgp message.asc

4.3 PGP 키의 인증

PGP를 사용하다 보면 여러 사람들의 키를 수신하게 된다. 이러한 키들을 자신의 키 화일에 올려 놓고 사용해야 하는데 사용자들 서로간의 인증관계를 잘 알아야 잘못된 키를 사용하는 일을 막을 수 있다. 여기서는 여러 종류의 signature가 붙은 키를 받았을 때 행동해야 하는 방법을 알아 본다.

가. 키 추가 (Key Add)

case 1. 키와 주인과의 관계는 확실하지만 키의 주인은 믿을 수 없는 경우

언뜻 보면 말이 안되는 경우같이 보이지만 사실은 PGP를 사용하다 보면 흔히 겪게 되는 경우이다.

여러분이 만약 피자집에 전자우편을 통해서 주문을 한다고 가정하자. 여러분은 피자집에서 피자집의 키가 들어 있는 디스켓을 가지고 와서 여러분의 키 화일에 올려 놓는다. 디스켓의 표지에는 키의 fingerprint가 적혀 있다. 여러분은 키를 올려 놓고 fingerprint를 보아서 표지에 적혀 있는 것과 비교를 한다. 동일하다면 여러분이 방금 올린 키는 피자집의 키라는 것을 확인할 수 있다. 여기서 키와 주인과의 관계는 확실하게 보여졌기 때문에 여러분은 피자집의 키에 여러분의 signature를 붙여 놓아도 된다. 여러분의 signature가 붙으면 여러분의 친구가 피자집의 키를 원할 때 당신의 signature를 보고 다른 확인작업 없이 키를 사용할 수 있게 된다. 그러므로 여러분이 누군가의 키에 서명을 붙이는 행동은 확실한 확인작업 후에 이루어져야 한다. 이제는 피자집을 다

른 키의 중개자로서 어느 정도 믿을 것이냐가 남아 있다. 앞에서 말한 Owner_trust를 결정하는 일이다. 여러분이 만약 피자집 주인과 절친한 사이라면 피자집이 서명해 놓은 키를 믿고 받아들일 것이다. 하지만 대개의 경우는 피자집의 서명이 붙은 키를 함부로 믿으려 하지는 않을 것이다. 이럴 때는 피자집의 키에 대해서 Owner_trust를 undefined로 지정해 준다. 이러면 다른 키를 받았을 때 피자집의 signature가 붙어 있다 하더라도 키의 Key_legitimacy 계산에는 영향을 미치지 않는다. 실제 PGP 실행의 예는 다음과 같다.

%> pgp -ka phil.pgp

Looking for new keys...
pub 512 / 43744F09 1994 / 07 / 07 Phil's Pretty Good Pizza <phil@pgp.com>

Checking signatures...

Keyfile contains:
1 new key(s)
One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y / N)? y

Key for user ID : Phil's Pretty Good Pizza <phil@pgp.com>
512-bit key, Key ID 43744F09, created 1994 / 07 / 07
Key fingerprint = 24 38 1A 58 46 AD CC 2D AB C9 E0 F1 C7 3C 67 EC
/* 키의 신원을 확인 */
This key / user ID association is not certified.
Do you want to certify this key yourself (y / N) y
/* 키의 신원을 확인했으므로 signature를 붙인다. */
Looking for key for user "Phil's Pretty Good Pizza <phil@pgp.com>":

Key for user ID: Phil's Pretty Good Pizza <phil@pgp.com>
512-bit key, Key ID 43744F09, created 1994 / 07 / 07

READ CAREFULLY: Based on your own direct first-hand knowledge, are you absolutely certain that you are prepared to solemnly certify that the above public key actually belongs to the user specified by the above user ID (y / N)? y
/* 서명을 함부로 붙이지 맙시다. */
You need a pass phrase to unlock your RSA secret key.
Key for user ID "Simson L. Garfinkel <simsong@acm.org>"

Enter pass phrase: .Pass phrase is good.
Just a moment..
/* 비밀키를 써야 하니까.. */
Key signature certificate added.

Make d determination in your own mind whether this key actually belongs to the person whom you think it belongs to, based on available evidence. If you think it does, then based on your estimate of that person's integrity and competence in key management, answer the following question:

Would you trust "Phil's Pretty Good Pizza 〈phil@pgp.com〉" to act as an intriducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ? 2
/* 피자집을 중개자로 믿지 않아요. */
%〉

■ 신뢰의 정도

① "1 = don't know"는 이 사람의 signature가 붙은 키가 올 때마다 사용자에게 확인여부를 물어 보겠다는 뜻이다.
② "2 = No"는 다른 사람의 키를 소개시켜 주는 중개자로는 이 사람을 믿지 않 겠다는 뜻이다.
③ "3 = Usually" 는 확실하게 믿지는 않지만 어느 정도는 믿겠다는 뜻이다.
④ "4 = Yes, always"는 이 사람의 signature가 붙은 키는 아무 의심없이 사용하겠다는 뜻이다.

이렇게 한 사람의 키에 대해서 사용자들마다 각기 다른 Owner_trust를 줄 수 있기 때문에 PGP는 사용자 개개인의 역할이 중요하다. 여러분도 친구들에게는 믿을 수 있는 사람으로 여겨지지만 다른 곳에서는 전혀 믿지 못할 사람으로 여겨질 수 있다.

case 2. 키와 주인과의 관계도 확실하고 주인도 믿을 수 있는 경우

여러분의 친구가 여러분에게 자신의 키를 보내 왔다. Fingerprint를 확인하기 위해 친구에게 전화를 걸어 문의해본 결과 친구의 키가 확실했다. 키와 주인과의 관계는 밝혀졌고 이제 친구를 어느 정도 믿을 수 있는 사람인가를 결정해야 한다. 인간성이 좋은 친구라면 높게 줄 것이고 평소에 거짓말을 많이 한 사람이라면 낮게 주거나 못 믿을 사람으로 여길 것이다. 사용 예는 case 1의 경우와 동일하나 마지막의 신뢰의 정도를 결정해 주는 단계에서 사용자가 다른 값을 입력해줄 수 있다. 만약, "4"를 선택하면 앞으로 Terrenc의 signature가 붙은 키는 아무 의심없이 사용하겠다는 뜻이다.

%〉 pgp -ka terry.asc

Looking for new keys...
pub 512/A4171B2D 1994/05/12 Terrence Talbot 〈tjt@dtw.com〉

Checking signatures...

Keyfile contains:
1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)? y

Key for user ID : Terrence Talbot 〈tjt@dtw.com〉
512-bit key, Key ID A4171B2D, created 1994/05/12
Key fingerprint = 45 3B 25 66 3F 97 D3 CF B9 41 B4 B4 C1 99 DD 7F
/* 키의 신원을 확인 */
This key/user ID association is not certified.

Do you want to certify this key yourself (y/N) y

/* 키의 신원을 확인했으므로 signature를 붙인다. */
Looking for key for user 'Terrence Talbot <tjt@dtw.com>':

Key for user ID: Terrence Talbot <tjt@dtw.com>
512-bit key, Key ID A4171B2D, created 1994/05/12

READ CAREFULLY: Based on your own direct first-hand knowledge, are you absolutely certain that you are prepared to solemnly certify that the above public key actually belongs to the user specified by the above user ID (y/N)? y
/* 서명을 함부로 붙이지 맙시다. */
You need a pass phrase to unlock your RSA secret key.
Key for user ID "Simson L. Garfinkel <simsong@acm.org>"
Enter pass phrase: .Pass phrase is good. Just a moment..
/* 비밀키를 써야 하니까.. */
Key signature certificate added.

Make d determination in your own mind whether this key actually belongs to the person whom you think it belongs to, based on available evidence. If you think it does, then based on your estimate of that person's integrity and competence in key management, answer the following question:

Would you trust "Terrence Talbot <tjt@dtw.com>"
to act as an intriducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ? 4
/* Terrence가 소개시켜 주는 사람은 믿을만 합니다.. */
%>

case 3. 모르는 사람의 키를 받았을 경우

어느날 메일박스를 보다가 누군가가 키 화일을 보내 왔다고 하자. 키 화일에 올려 놓고 키 주인의 이름을 보니까 모르는 사람이다. 이럴 때 여러분은 어떻게 하겠는가? 먼저 signature를 살펴 본다. 만약 signature가 달려 있지 않다거나 여러분이 전혀 알지 못하는 사람들의 signature가 달려 있다면 이 키를 사용해도 되느냐 안 되느냐는 오직 여러분의 확인 작업과 판단에 달려 있게 된다. 그러나 여러분이 아는 사람의 signature가 달려 있다면 어느 정도는 키의 신원에 대해 확인할 수 있다.

%> pgp -ka sam.pgp

Looking for new keys...
pub 768/3F1DEA81 1994/08/27 Sam Spade <smokey@bureau.com>
/* 모르는 사람인데.... */

Checking signatures...
pub 768/3F1DEA81 1994/08/27 Sam Spade <smokey@bureau.com>
sig! A4171B2D 1994/05/12 Terrence Talbot <tjt@dtw.com>
/* 친구의 signature가.. */

Keyfile contains:
1 new key(s)
/* 믿는 친구의 signature가 있는 거라서

키 신원 확인작업도 생략된다. * /
Make d determination in your own mind whether this key actually belongs to the person whom you think it belongs to, based on available evidence. If you think it does, then based on your estimate of that person's integrity and competence in key management, answer the following question:

Would you trust "Terrence Talbot 〈tjt@dtw.com〉" to act as an intriducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ? 3
/* 모르는 사람이지만 친구가 믿는 사람이라서 * /
%〉

위의 예에서는 키의 확인작업과 여러분의 signature를 붙이는 작업을 행하지 않는다. 하지만 여러분이 이 키에다 signature를 붙이고 있으면 옵션을 사용해서 붙일 수 있다.

나. 키에 달린 signature 확인하기
(Key Check, Key Verbose View)

```
%〉 pgp -kc
%〉 pgp -kvv
```

PGP를 사용하다 보면 누구의 키에 누구의 signature가 붙어 있는 지를 확인해야 할 필요가 있다. 이때 사용하는 것이 "-kc" 옵션과 "-kvv" 옵션이다.

"-kc" 옵션은 키 화일에 있는 모든 키들과 키들에 붙어 있는 signature에 관련한 신뢰정도들을 볼 수 있고, "-kvv" 옵션은 누가 signature를 붙였는가 만을 보여준다.

```
%〉 pgp -kc
Type            pub
                sig!
bits / KeyID    768 / 3F1DEA81
                A4171B2D
Date            1994 / 08 / 27
                1994 / 05 / 12
UserID          Sam spade
                〈smokey@bureau.com〉
                Terrence Talbot
                〈tjt@dtw.com〉

Key ID          3F1DEA81
                c
Trust           marginal
                complete
Validity        complete
User ID         Sam spade
                〈smokey@bureau.com〉
                Terrence Talbot
                〈tjt@dtw.com〉
%〉
%〉 pgp -kvv
Type            pub
                sig!
bits / KeyID    768 / 3F1DEA81
                A4171B2D
Date            1994 / 08 / 27
                1994 / 05 / 12
User ID         Sam spade
                〈smokey@bureau.com〉
                Terrence Talbot
                〈tjt@dtw.com〉
%〉
```

다. 키의 fingerprint 보기 (Key View Check)

키의 fingerprint를 보기 위해서는 "-kvc"옵션을 사용한다.

```
%〉 pgp -kvc
Type            pub
bits / KeyID    768 / 3F1DEA81
Date            1994 / 08 / 27
User ID         Sam spade
                〈smokey@bureau.com〉
Key fingerprint = B4 42 4F A4 15 67
FA FD 84 DD 43 E7 09 45
%〉
```

라. 신뢰정도 바꾸기 (Key Edit)

```
%〉 pgp -ke
```

"-ke" 옵션을 자신의 키에 사용하면 pass phrase나 user ID를 바꿀 수 있었다. 그러나 자신의 키 화일에 있는 다른 사람의 공개키에 사용하면 사용자가 그 키에 지정해준 Owner_trust를 바꿀 수 있다. Owner_trust를 바꾸어야 하는 경우는 여러가지가 있는 데 예를 들면, 사용자가 완전히 믿고 있는 사람이 알고 보니까 아무 키에나 마구 signature를 달아 놓는다면 그 사람에 대한 신뢰정도를 떨어뜨려야 한다. 또, 별로 믿음이 가지 않던 사람인데 직접 만나서 보니까 충분히 믿을 수 있겠다고 생각이 바뀌었을 때에도 신뢰정도를 바꾸어 주어야 한다. 다음의 예에서는 완전히 믿던 신뢰정도를 떨어뜨리는 경우를 본다.

```
%〉 pgp -ke Terrence
Key for user ID: Terrence Talbot
〈tjt@dtw.com〉
512-bit key, Key ID A4171B2D, created
1994 / 05 / 12
This user is completely trusted to certify
other keys.
This key / userID association is fully
certified.
Axiomatically trusted certification from:
Simson L. Garfinkel 〈simson@acm.org〉
Current trust for this key's owner
is:complete

Make a determination in your own mind
whether this key actually
belongs to the person whom you think it
belongs to, based on available
evidence. If you think it does, then based
on your estimate of
that person's integrity and competence in
key management, answer
the following question:

Would you trust "Terrence Talbot
〈tjt@dtw.com〉"
to act as an intriducer and certify other
people's public keys to you?
(1=I don't know. 2=No. 3=Usually.
4=Yes, always.) ? 3
Public key ring updated.
%〉
```

마. 키에 signature 붙이기 (Key Sign)

```
%〉 pgp -ks
```

키를 키 화일에 올릴 때 signature를 붙이는 절차를 가지지만 그때는 signature를 붙이지 않고 나중에 signature를 붙여야 하는 경우가 생길 때 사용하는 옵션이 "-ks" 옵션이다. Signature를 붙이려고 하는 공개키를 MD5를 이용해 128-bit으로 만들고 사용자의 비밀키로 암호화한다. Pass phrase를 입력해야 하기 때문에 비밀키의 주인이 아니면 signature를 붙

일 수가 없다. 다음은 Sam의 키에 Simson이 signature를 붙이는 경우이다.

```
%〉 pgp -ks sam
A secret key is required to make a signature.
You specified no user ID to select your secret key,
so the default user ID and key will be the most recently
added key on your secret key ring.

Looking for key for user 'sam':

Key for user ID : Sam spade 〈smokey @bureau.com〉
768-bit key, Key ID 3F1DEA81, created 1994/08/27

READ CAREFULLY: Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)? y

You need a pass phrase to unlock your RSA secret key.
Key for user ID "Simson L.Garfinkel 〈simsong@acm.org〉"

Enter pass phrase: .Pass phrase is good. Just a moment....
Key signature certificate added.
%〉
```

여러 개의 비밀키가 있는 데 특정한 키를 지정해 주지 않았으므로 가장 최근에 생긴 비밀키를 사용하고 있다. 만약 특정한 키를 지정해 주고 싶으면 "-u" 옵션을 사용하면 된다. Signature를 붙이고 나서 "-kvv" 옵션으로 확인할 수 있다.

바. 키에 달린 signature 삭제하기
(Key Remove Signature)

```
%〉 pgp -krs
```

어느 키에 달려 있는 여러 개의 signature들 중에서 특정한 signature를 지우고 싶을 때에 "-krs" 옵션을 사용한다. 다음의 예는 Sam의 키에 달린 signature 들중에서 일부를 삭제하는 경우이다.

```
%〉 pgp -krs sam
Key for user ID : Sam Spade 〈smokey @bureau.com〉
768-bit key, Key ID 3F1DEA81, created 1994/08/27

Key has 2 signature(s):
sig          33681029
Nosmis Noel, Secret Agent 〈nn@nsa.gov〉
Remove this signature (y/N)? y
sig          A903C9265
Simson A.Garfinkel 〈simsong@acm.org〉
Remove this signature (y/N)? y

2 key signature(s) removed.
%〉
```

4.4 PGP 키의 취소

PGP를 사용하다가 비밀키의 pass phrase가 노출되는 등 자신의 비밀키가 남에 의해 악용되고 있다고 생각되면 그 키를 취소시켜야 한다.

가. 자신의 키를 취소하기

```
%> pgp -kd userID
```

자신의 키를 취소한다는 것은 다른 사람들에게 이 키를 사용하지 말라고 얘기해 주는 것과 같다. 옵션은 "-kd" 옵션을 사용한다. 취소하려면 자신의 비밀키로 revocation certificate를 붙여 주어야 하므로 키의 주인만이 할 수 있다. Revocation certificate를 붙인 키를 되도록이면 많은 사람들에게 전파시켜야 한다. 다음은 사용자 자신의 키를 취소하는 경우이다.

```
%> pgp -kd phil
Key for user ID: Phil's Pretty Good Pizza
<phil@pgp.com>
512-bit key, Key ID 43744F09, created
1994/07/07

Do you want to permanently revoke your
public key
by issuing a secret key compromise
certificate
for "Phil's Pretty Good Pizza <phil@
pgp.com>" (y/N)? y

You need a pass phrase to unlock your
RSA secret key.
Key for user ID "Phil's Pretty Good Pizza
<phil@pgp.com>"

Enter pass phrase:
.Pass phrase is good. Just a moment....
Key compromise certificate created.
%> pgp -kv
Type            pub
bits/KeyID      512/43744F09
Date            1994/07/07
User ID         *** KEY REVOKED ***
                Phil's Pretty Good Pizza
                <phil@pgp.com>
%>
```

키에 revocation certificate를 붙인 다음에 키를 전파시키던 방법으로 전파시킨다. 한번 취소시키면 다시는 복구시킬 수가 없으므로 사용자는 키를 취소하는 데에 신중을 기해야 한다. 다른 사용자가 revocation certificate가 붙은 키를 키 화일에 올려 놓게 되면 그 사용자도 이 키를 사용할 수 없게 된다.

나. 다른 사람의 키 취소하기

```
%> pgp -kd userID
```

앞의 경우는 자신의 키를 취소하는 방법이었고 여기서는 자신의 키 화일에 있는 다른 사람의 키를 취소시키는 것을 본다. 다른 사람의 키를 취소시키면 그 영향은 자신에게만 있고 다른 사람들에게는 아무런 영향을 미치지 않는다. 앞의 방법과 틀린 것은 남의 키를 취소시키면 나중에 다시 복구시킬 수가 있다. 가령 누구의 키가 노출되었다는 소문을 듣고 자신의 키 화일에 있는 그 사람의 키를 취소시켰다가 확인해 본 결과 헛소문으로 밝혀지면 다시 복구시킬 수 있다. 다시 복구시킬 때는 취소시킬 때와 똑같은 명령을 한번 더 실행시켜 주면 된다.

```
%> pgp -kd sam
Key for user ID: Sam Spade <smokey
@bureau.com>
768-bit key, Key ID, 3F1DEA81, created
1994/08/27

Disable this key (y/N)? y
%>
%> pgp -kd sam
```

Key for user ID: Sam Spade 〈smokey @bureau.com〉
768-bit key, Key ID, 3F1DEA81, created 1994/08/27
Key is disabled.

Key is already disabled.
Do you want to enable this key again (y/N)? y
%〉

4.5 PGP configuration file

PGP를 사용함에 있어서 사용자의 형편에 맞게 사용할 수 있도록 변수값들을 조절할 수 있다. 각 변수들을 살펴 보면 다음과 같다.

- ARMOR [-a] (default OFF) : On시키면 PGP가 만드는 모든 문서들에 대해서 radix conversion을 시켜 주므로 암호화된 문서를 전자메일을 통하여 송신하려 할때에 이용한다.
- ARMORLINES (default 720) : 크기가 큰 화일을 전자우편으로 보낸 때 PGP가 쪼개는 크기로 단위는 라인수이다.
- BAKRING (no default) : 비밀키 화일의 backup용 화일의 path
- Cert_depth (default 4) : 중개자로서의 역할을 몇 단계까지 허용하는가. 만약 A가 B를 믿고, B가 C를 믿고, C가 D를 믿고, D가 E를 믿으면 단계 수가 4이므로 A도 E를 믿을 수 있다.
- CHARSET (default noconv) : 사용자가 사용하고 있는 시스템에서 지원하는 문자체계를 지정하여 다른 문자체계를 사용하는 사용자와 통신할 수 있게 한다.noconv는 같은 체계를 사용할 때 주는 값이다.
- CLEARSIG (default On) : 메세지에 전자서명을 붙일 때 원래의 메세지는평문으로 놓고 전자서명 부분을 뒤에 붙인다. Off시에는 평문과 전자서명 부분을 함께 radix conversion하므로 화면상에서 평문의 내용이 확인이 되지 않는다.
- COMMENT (default NULL) : PGP에서 만드는 .asc 화일에 주석라인을 만든다.
- COMPLETES_NEEDED (default 1) : 키를 확인해주기 위해 필요한 completely trusted signature의 갯수.
- COMPRESS (default On) : 암호화 과정에서 압축을 실행해 준다.
- INTERACTIVE (default Off) : On이면 키를 화일에 올려 놓을 때 각 키에 대해서 확인질문을 한다.
- KEEPBINARY (default Off) : .asc 화일을 복호화하면 먼저 .pgp 화일로 만들고 이를 복호화하게 되는데 off시에는 .asc 화일과 평문만을 남기고 on 시에는 중간단계인 .pgp 화일도 남겨 놓는다.
- LANGUAGE (default en) : PGP에서 뿌려 주는 메세지의 언어를 명시.
- MARGINALS_NEEDED (default 2) : 키를 확인해 주기 위한 필요한 marginally trusted signature의 갯수
- MYNAME (no default) : 전자서명을 위해 필요한 키의 user ID를 명시.
- PAGER : "-m" 옵션을 사용할 때 복호화된 화일이 화면에 뿌려지는 형식
- PKCS_COMPAT (default 1) : 2.5 이전의 버젼에서 사용되는 variable로써 다른 형태의 메세지와 키를 사용하기

위하여 사용된다. 2.6 이후의 버젼에서는 무시된다.

- PUBRING (default $PGPPATH/pubring.pgp) : 공개키가 들어갈 화일의 path
- RANDSEED (default $PGPPATH/randseed.bin) : 난수 발생에 필요한 seed 값이 들어가는 화일의 path
- SECRING (default $PGPPATH/secring.pgp) : 비밀키가 들어갈 화일의 path
- SHOWPASS (default Off) : On이면 사용자가 pass phrase를 입력할 때 화면에 pass phrase가 보여진다.
- TMP (no default) : 화일을 복호화할 때 필요한 임시화일들이 위치할 곳의 path
- TZFIX (default 0) : GMT(Greenwich Mean Time)시간을 현지시간으로 바꾸기 위해 필요한 시간
- VERBOSE (default 1) : PGP가 실행될 때 사용자가 볼 수 있는 정보들의 양을 말한다. 0이면 프롬프트와 에러 메시지만 볼수 있고, 2이면 디버깅 메세지까지 보여 준다. 1은 중간형태를 취한다.

4.6 PGP 키 서버

PGP를 사용하다 보면 특정인의 키가 필요한 경우가 있다. 키를 가져오는 방법으로는 전자우편이나 디스켓을 사용하는 방법이 있지만 키 서버를 이용하는 방법도 있다.

가. 키 서버와 통신하기

서버와 대화하는 방법으로는 전자우편을 사용하는데 전자우편의 subject 필드에 키 서버에서 사용하는 특정한 코맨드를 넣어 주면 된다. 서버로 전자우편을 보내면 그의 대답에 해당하는 정보가 역시 전자우편으로 보내진다.

나. 키 서버 코맨드

키 서버에서 알려고 하는 정보에 따라 subject 필드에 넣어 주어야 하는 코맨드가 다르다. 중요한 코맨드를 보면 다음과 같다.

- help : 서버에 관한 정보와 실행시킬 수 있는 코맨드의 종류를 알 수 있다.

  %〉 mail -s "help" pgp-public-keys@pgp.mit.edu〈/dev/null

  Null message body; hope that's ok

  %〉

"〈/dev/null"은 전자우편의 body부분이 없다는 의미인데, PGP 키서버는 subject 의 내용만 보기 때문에 body부분은 없어도 아무 문제가 없다.

- add : 자신의 키를 키 서버에 올려 놓는다.(-ka)

  %〉 pgp -kxaf Hyun__Dong Park | mail -s "add" pgp-public-keys@pgp.mit.edu

  %〉

키 서버에 키를 등록할 때에는 하나의 서버에만 올리면 서버들끼리 서로 갱신하므로 모든 서버에 등록할 필요는 없다.

- index : 서버에 있는 모든 키들을 볼 수 있다. (-kv)

  %〉 mail -s "index" pgp-public-keys@pgp.mit.edu〈/dev/null

  Null message body; hope that's ok

  %〉

- verbose index : 서버의 모든 키들을 signature와 함께 볼 수 있다. (-kvv)
- get : 키 서버에 있는 모든 키들을 가져온다. (-kxaf)

- get userid : 특정한 사람의 키만을 가져 온다. (-kxaf userid)

  %〉 mail -s "get simson" pgp-public-keys@pgp.mit.edu〈 / dev / null
  %〉

- mget regular_expression : 정규표현에 맞는 userid를 가지고 있는 모든 키들을 가져 온다.

  %〉 mail -s "mget sim|jef" pgp-public-keys@pgp.mit.edu〈 / dev / null
  Null message body; hope that's ok
  %〉

- last days : 최근 몇일 안에 갱신된 키만을 가져 온다.

  %〉 mail -s "last 3" pgp-public-keys@pgp.mit.edu〈 / dev / null
  Null message body; hope that's ok
  %〉

최근 3일동안 갱신된 키가 온다.

다. 키 서버가 있는 곳

국 내 : pgp-public-keys@juno.kaist.ac.kr
미 국 : pgp-public-keys@pgp.iastate.edu
pgp-public-keys@pgp.mit.edu
pgp-public-keys@pgp.ai.mit.edu
pgp-public-keys@burn.ucsd.edu
pgp-public-keys@jpunix.com
영 국 : pgp-public-keys@demon.co.uk
pgp-public-keys@pgp.ox.ac.uk
네델란드 : pgp-public-keys@kub.nl
독 일 : pgp-public-keys@fbihh.informatik.uni-hamberg.de
일 본 : pgp-public-keys@ext221.sra.co.jp
호 주 : pgp-public-keys@sw.oz.au
러시아 : pgp-public-keys@kiae.su
이태리 : pgp-public-keys@dsi.unimi.it

5. 결 론

인터넷은 정보화 시대라는 사회조류와 맞물려 현재 급격한 성장을 계속하며 이미 일반인들의 생활에까지 영향을 미치고 있다. 정보화 시대에서 기대할 수 있는 정보의 공유는 자칫 정보의 탈취와 변조로 변질될 수 있다. 인터넷의 가장 인기있는 서비스인 전자우편은 개인의 문서교환뿐만 아니라 상품구매와 공공기관에서의 서류결재 등에 사용될 것으로 보이나, 보안이 보장되어 있지 않은 전자우편을 이용하는 것은 무척 위험한 일이다. 지난번 뉴질랜드 한국대사관 문서위조사건이 하나의 예라 할 수 있겠다. 이에 따라 PGP와 같은 보안도구를 이용한 전자우편 사용에 대한 관심이 고조되고 있다.

PGP는 현재 인터넷에서 가장 많이 이용되는 전자우편 보안도구로써 문서의 기밀성을 보장해 주면서 불법적 변경여부와 송신자의 신원확인, 그리고 송신자가 문서를 보낸 사실을 부인하지 못하게 하는 기능들을 제공해 주고 있기 때문에 전자우편 사용에 있어 중요한 역할을 수행하고 있다. 그러나 국내에서는 아직 초기 단계이나 차츰 사용자가 증가하고 있는 추세이다. 앞으로 일반 사용자들이 암호화/복호화의 복잡한 구조를 모르더라도 손쉽게 사용할 수 있도록 GUI(Graphic User Interface)의 구현과 한글화가 이루어지면 그 사용이 급격히 증가하리라 예상되고 있다.

참 고 문 헌

[1] Simson Garfinkel, "PGP:Pretty Good Privacy", O'Reilly & Associations, Inc., 1995

[2] William Stallings, "The PGP User's

Guide", Prentice-Hall,Inc., 1995

[3] Bruce Schneier, "E-mail Security", John Wiley & Sons,Inc., 1995

[4] 류재철, 송우길, "전자우편 보안(PEM, PGP)", Netsec-KR'95, 1995

[5] 한국전자통신연구소, "현대암호학", 1991

[6] Bruce Schneier, "Applied Cryptography", John Wiley & Sons,Inc., 1994

[7] William Stallings, "Network and Internetwork Security", Prentice-Hall, 1995

부 록 : PGP 실행 옵션

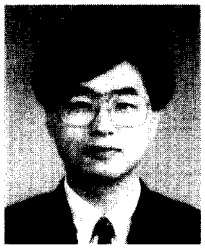
키관리에 관한 도움말 보기	pgp -k
공개키/비밀키를 생성하기	pgp -kg
키링에 키를 첨가하기	pgp -ka keyfile[keyring]
키나 사용자 ID를 지우기	pgp -kr userid[keyring]
사용자 ID나 pass phrase를 변경하기	pgp -ke your_userid[keyring]
키링에서 키를 뽑아 내기	pgp -kx userid keyfile[keyring]
키링의 내용을 보기	pgp -kv[v][userid][keyring]
공개키링에 있는 signature 확인하기	pgp -kc[userid][keyring]
공개키링에 있는 다른 사람의 키에 서명하기	pgp -ks her_userid[-u your_userid][keyring]
특정 signature 삭제하기	pgp -krs userid[keyring]
수신자의 공개키로 암호화하기	pgp -e textifile her_userid[other userids]
비밀키로 평문에 전자서명 하기	pgp -s texfile[-u your_userid]
비밀키로 서명하고, 수신자의 공개키로 암호화하기	pgp -es textifile her_userid[other_userids] [-u your_userid]
관용암호방식으로 암호화하기	pgp -c textfile
암호문을 복호화하거나 전자서명을 확인하기	pgp ciphertext[plaintext]
전자우편에 맞는 ASCII 형태로 확인하기	다른 옵션에 -a 옵션 첨가
암호화에 관한 도움말 보기	pgp -h

□ 著者紹介

박 현 동

1995년 2월 충남대학교 전산학 학사
1995년 3월 - 현재 충남대학교 컴퓨터과학과 석사과정

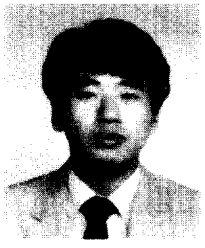
※ 관심 분야 : 컴퓨터 및 통신 보안체제



류 재 철

1985년 2월 한양대학교 산업공학 학사
1988년 5월 Iowa State Univ. 전산학 석사
1990년 12월 Northwestern Univ. 전산학 박사
1991년 2월 - 현재 충남대학교 컴퓨터과학과 조교수

※ 관심분야 : 컴퓨터 및 통신 보안체제, 네트워크 관리, 분산처리



임 채 호

1986년 홍익대학교 전자계산학과(학사)
1990년 건국대학교 대학원 전자계산학과(석사)
1986년 ~ 1992년 한국과학기술연구원 시스템공학연구소 연구원
1992년 ~ 1994년 대전실업전문대학 전자계산과 교수
1994년 - 현재 한국과학기술연구원 시스템 공학연구소 연구원

※ 관심분야 : 컴퓨터통신, 컴퓨터통신 보안, 운영체제 보안, 분산시스템

변 옥 환

1979년 한국항공대학교 정보통신공학 학사
1985년 인하대학교 대학원 전자공학 석사
1993년 경희대학교 대학원 전자공학 박사
1978년 - 현재 시스템 공학연구소 책임연구원
1983년 12월 - 1984년 12월 미국 OSM corp. 방문연구원
1995년 - 현재 한국통신정보보호학회 이사

※ 관심분야 : 정보통신망 보안, 관리 망설계 및 서비스