Step by Step 스텝 바이 스텝 4

네트워크 보안 기술의 원리와 응용

보안 애플리케이션, Kerberos와 PGP

운영체제와 같은 시스템 소프트웨어에서 사용자 계정을 관리하는 메커니즘은 어떤 것이 있을까? 지난 시간에 살펴봤던 DES를 기반으로 하는 Kerberos라는 것이 있다. 그리스 신화에 나오는 머리 셋 달린 괴물에서 유래된 뜻으로 시스템을 지킨다는 의미를 내포한다. 대표적으로 마이크로소프트 윈도우의 계정관리, IIS(Internet Information Server), NFS, 유닉스의 네트워크 인증서비스 등에 활용된다.

연 재 가 이 드
운영체제 | 윈도우, 리눅스, 유닉스, 매킨토시 등 모든 플랫폼
기초지식 | 네트워크 개론
응용분야 | 네트워크 보안 애플리케이션, 프로토콜 구현

서상원 smiler@ssm.samsung.co.kr | 필자는 2005년부터 삼성소프트웨어멤버십 15기로 활동 중이다. 최적의 개발을 위한 플랫폼솔루션과 개발 프레임워크에 큰 관심을 두고 있으며, 이에 기반 되는 아키텍처 설계에 고심을 하고 있다. 현재는 삼성전자에서 미디어 셋톱박스 관련 프로젝트를 수행하고 있다.

인증을 필요로 하는 시스템에서 활용도가 높은 Kerberos와 인터넷의 영원한 킬러 애플리케이션인 전자메일 서비스의 보안을 담당하는 PGP에 대해 자세히 살펴보자. Kerberos는 MIT의 Athena 프로젝트의 일부로 개발된 인증 서비스다. 워크스테이션을 이용할 때 분산된 네트워크 환경에서 사용자 인증을 목적으로 쓰였다. 인증된 사용자만 서버에 접속할 경우, 인터넷과 같은 개방형 네트워크 환경에서는 다음과 같은 위험요소가 존재할 수 있다. 첫 번째 위험요소는 마치 본인이 인증된 사용자인 것처럼 가장하는 것이다. 보안용어로는 'Spoofing'이라고 한다. IPv4 체계에서 헤더정보의 주소정보(Source IP Address)를 임의로 변경해 악용하는 사례를 흔히 접한다. 이와 마찬가지로 남의 가면을 쓰고 가장하는 것은 큰 위협요소다. 두 번째로 워크스테이션과 인증된 사용자간의 네트워크 패킷을 가로채서 내용이 도청될 수 있다. 보안용어로는 'Eavesdrop'이라고 한다. 마지막으로 재전송 공격을 통해 패스워드를 알아낼 수 있는 위협요소가 존재한다.

이런 위협요소를 방지하기 위해 등장한 애플리케이션이 바로 Kerberos이다. 모든 인증시스템에 안정성 있는 프로토콜을 제공해 시스템간의 호환성을 제공한다. Kerberos 이전에는 시스템마다 인증방식이 다르고 그에 따른 프로토콜도 모두 달랐다. 점점 복잡도가 높은 시스템이 등장하면서 표준이 필요했다. Kerberos는 티켓이라는 인증매체를 이용하고, 모든 종류의 암호화 기법을 지원한다(물론 Kerberos 버전 4는 DES로 제한된다). 패킷이 누군가에 의해 도청돼도 내용을 확인하기란 거의 불가능해졌다. 버전 5에서는 세션이라는 개념을 둬 사용자를 한번 인증하면 세션키라는 것을 제공하므로, 여러 번 요청을 시도해 재전송 공격을 하는 악의적인 사용자를 사전에 차단할 수 있게 됐다.

PGP는 Phil Zimmermann이 고안 했고, 인터넷을 통해 오픈소스 솔루션 패키지로 배포했다. 주로 전자메일과 파일 송수신에 많이 쓰이며, 기밀(Confidentiality)과 인증(Authentication)이 동시에 가능하다. 대부분의 보안 솔루션은 기밀과 인증에 따라 구분될 수 있음을 지난 호에서 언급했다. PGP는 Pretty Good Privacy의 약자인데, 그 이름에 걸맞게 전 세계적으로 널리 쓰인다. 널리 쓰이는 가장 큰 이유는 운영체제나 프로세서와 상관없

필자 메모

지난 시간에 메시지 인증방법과 공개키 암호화에 대해 살펴봤다. 이번 호에서는 이런 방법들을 응용한 Kerberos와 PGP에 대해 다뤄 볼 것이다. 특히 윈도우 서버 관리자라면, 반드시 Kerberos에 대한 이해가 필요하다. 단순히 툴을 사용해 도메인을 추가하고, 계정 권한을 부여하는 것이 내부적으로는 규격화된 보안 프로토콜이 존재한다는 사실을 알게 될 것이다. S/MIME과 함께 전자메일 보안에 가장 널리 쓰이는 PGP를 통해 알려진 암호화 알고리즘을 응용하는 사례에 대해 알아보자.

이 사용되는 플랫폼에서 독립적으로 동작한다는 것이다. RSA나 SHA와 같은 검증된 알고리즘을 사용해서 선호도를 높였다.

Kerberos 버전 4

버전4를 먼저 살펴보면 Kerberos에 대한 이해를 높일 수 있다. Kerberos 버전 4는 최근 많이 사용하는 공개키 암호화를 쓰지 않고, 대칭키 알고리즘 같은 이전부터 많이 쓰여 진 알고리즘을 사용한다. 실제로는 DES를 사용한다. 보안의 중요도와 시스템의 성능에 따라 3중 DES를 사용할 수도 있다. 물론 요즘은 버전 4를 이용하는 애플리케이션은 거의 없다.

단순화한 인증절차

〈리스트 1〉은 Kerberos의 복잡해 보이는 인증절차를 간략화 한 것이다. 용어를 먼저 숙지한 후 (1)번 과정부터 (3)번 까지 살펴보자. 〈리스트 1〉은 클라이언트가 서버의 서비스를 이용하기 위해 인증티켓을 얻는 과정으로 요약할 수 있다. 이때 인증 티켓을 발급하는 서버를 인증서버(AS)라고 한다. (1)번 과정에서, 클라이언트(C)는 인증서버로(AS)로 본인의 아이디와 패스워드를 입력하고 접속할 서버를 선택한다. 원격을 이용해 윈도우(mstsc)나 리눅스(telnet, ssh)에서 사용자 로그인 화면과 유사함을 눈치 챘을 것이다.

〈리스트 1〉 단순한 인증 절차

```
1) C -> AS : IDc || Pc || IDv
2) AS -> C : Ticket
3) C -> V : IDc || Ticket

[용어 정리]
- C : 클라이언트
- AS : 인증서버
- V : 서버
- IDc (Identifier of user on client) : 사용자의 아이디 정보
- Pc (Password of user on client) : 사용자의 패스워드 정보
- IDv (Identifier of server) : 서버의 아이디 정보(서버마다 구분되는 아이디가 존재한다)
- Kv : 인증서버(AS)와 서버(V)간의 공유하는 비밀 키
- Ticket = EKv[ IDC || Pc || IDv ]
```

(2)번 과정에서는, 인증서버에서 클라이언트 정보를 확인 한 후, 유효한 경우에 한해 다시 클라이언트에게 서버 이용 티켓(Ticket)을 발급한다. 티켓을 발급 받은 클라이언트는 (3)번 과정과 같이 서버의 서비스를 이용할 수 있다. 이때 티켓은 (1)번 과정에서 클라이언트가 전송한 정보들을 미리 정해진 비밀 키로 암호화 한 것이며, (3)번 과정에서 서버는 복호화(Decryption)해 인증여부를 확인할 수 있다. 인증서버가 암호화시켜 발급한 티켓을 서버가 복호화 하는 것이 가능한 이유가 바로 비밀 키를 공유하는 것이다. DES와 같은 대칭키 알고리즘을 사용하므로 키를 공유하면 암호화와 복호화가 가능하다.

위의 인증절차는 얼핏 보기에 어느 정도 보안 문제를 해결한 것 같아 보이지만, 다음과 같은 문제가 있을 수 있다. 첫째는 서버를 이용할 때마다 티켓을 발급받아야 하므로 패스워드를 매번 입력해야 하는 번거로움이 있다. 둘째는 패스워드 정보를 보낼 때도 다른 메시지와 구분 없이 평문형태로 보낸다는 것이다. 도중에 패킷을 도청당할 위험성을 가지고 있는 셈이다.

보다 안전한 인증절차

앞에서 단순화한 인증절차의 문제점을 보완한 것이 바로 여기서 설명할 내용이다. 버전 4의 실질적인 인증절차이기도 하다. 즉 단한번의 패스워드 입력으로 사용자 패스워드 보호가 가능한 시나리오를 가진다. 〈그림 1〉이 전체 과정을 플로우 다이어그램으로 보여준다. 자세히 살펴보면, TGS(Ticket Granting Server)라고 하는 서버가 추가 된 것을 확인할 수 있다. 이는 인증서버(AS)를 인증해주는 서버라고 이해하면 된다. 기존에 인증서버에서 서버 사용티켓을 발급했던 반면, 〈그림 1〉에서는 이중의 과정이 필요하다.

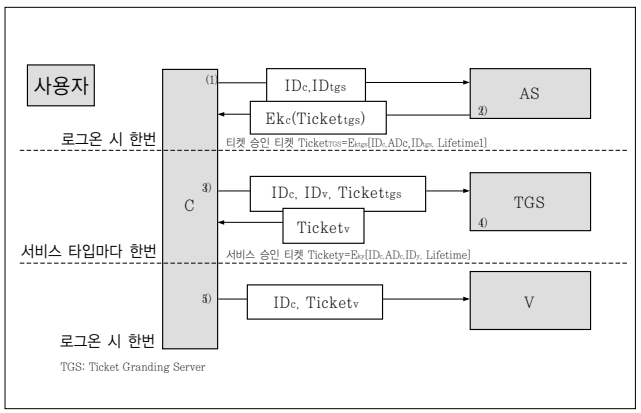
클라이언트가 인증서버로부터 TGS 티켓을 발급 받고, TGS는 또 다시 서버의 티켓을 발급하는 절차이다. 결과적으로 매번 로그인하는 과정을 생략하고, 서비스 타입마다 한 번씩만 티켓을 발급받도록 한다. 또한 티켓을 암호화하므로 결과적으로 도청이 돼도 내용을 알아내기가 어렵다. 〈그림 1〉을 (1)번 과정부터 살펴보자. 클라이언트가 인증서버로 본인의 아이디를 입력하고 이용할 TGS 서버를 선택한다. 이때 타임스탬프를 같이 전송한다.

〈리스트 2〉는 〈그림 1〉을 정리한 식으로, 과정을 따라가며 함께 참고하기 바란다. 타임스탬프는 〈리스트 2〉에서 TS1으로 표기됐다. 인증과정의 순서를 기억함으로써 악의적인 요청을 사전에 방지하고, 타임아웃을 적용할 수 있다. (2)번 과정에서 인증서버는 TGS를 사용하도록 승인티켓을 발급하면서 〈그림 1〉에는 생략돼 있지만, 사용자에게 패스워드 입력을 요구한다.

패스워드가 확인되면, 티켓이 발급된다. 또한 'Lifetime'이라고 하는 티켓 유효기간을 기록한다. 이제 사용자 인증과정은 모두 마쳤다. 사용자 인증과는 별개로 서비스마다 티켓이 존재하기 때문에 이 과정은 더 이상 필요하지 않다. 이제 TGS를 이용할 수 있는 티켓을 발급받았으므로 어떠한 서비스를 이용할 것인지에 대해 서버티켓을 발급받는 절차가 필요하다. (3)번 과정에서 클라이언트는 어떠한 서버의 서비스를 이용할 것인가에 대한 정보를 TGS티켓과 함께 TGS에 요청한다. 이때 TGS는 티켓의 유

효기간과 타임스탬프를 확인해 문제가 없는 경우 서버티켓을 발급해준다. 이것이 (4)번 과정이다.

〈그림 1〉과 〈리스트 2〉는 약간 다르게 보일 수도 있지만(이해를 돕기 위해 간략화 시켜 〈그림 1〉에는 생략한 부분들이 있다) 실질적으로 동일한 내용이다. 서버티켓까지 발급받은 클라이언트는 드디어 서버의 서비스를 이용할 수 있는 사용자로 인증을 받게 된 것이다. (5)번 과정처럼 클라이언트는 서버티켓을 이용해 서비스를 이용할 수 있다. 지금까지 설명한 처리과정 중에 가장 중요한 내용은 서비스마다 TGS인증을 받으므로 로그인 인증은 한번이면 된다는 것과 티켓이 암호화된다는 것, 그리고 타임스탬프의 개념(티켓 유효기간)이 있다는 것이다. 이를 이해했다면, Kerberos 버전 4에 대해 이해한 것이나 마찬가지다.



〈그림 1〉 버전 4의 인증절차(플로우 다이어그램)

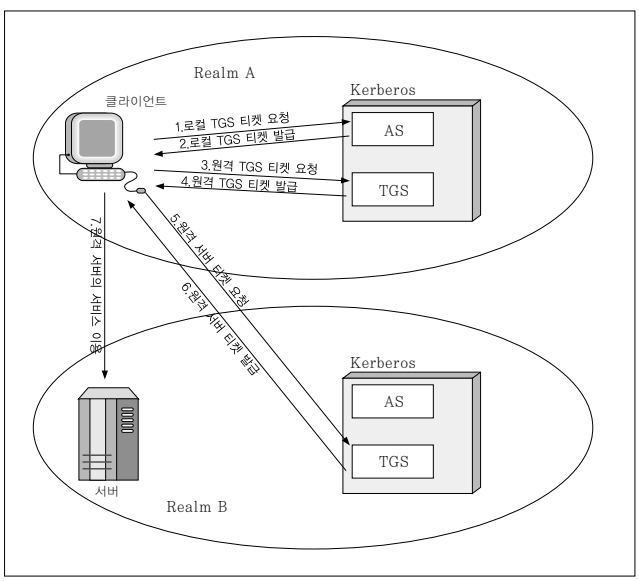
〈리스트 2〉 버전 4의 인증절차 (정리식)

```
티켓승인 티켓을 얻기 위한 인증 절차 (C <-> AS)
(1) C -> AS :   IDc || IDtgs ||TS1
(2) AS -> C :   EKc [Kc,tgs|| IDtgs || TS2 || Lifetime2
||Tickettgs ]

서비스승인 티켓을 얻기 위한 인증 절차 (C <-> TGS)
(3) C -> TGS :  IDv ||Tickettgs ||Authenticatorc
(4) TGS -> C :  EKc,tgs [Kc,¨v|| IDv || TS4 || Ticketv]
 * Authenticatorc = EKc,tgs[IDc || ADc || TS3]

실제로 서비스를 받기 위한 인증 절차 (C <-> V)
(5) C -> V : Ticketv || IDc
(6) V -> C : EKc,v[TS5+1]
```

지금까지 살펴본 방식은 단일 Kerberos 시스템에 국한된 것이다. 그럼, Kerberos 간의 인증은 어떻게 할 수 있을까? 이를 위해 Realm(공동체)이라고 부르는 개념을 도입하고 있다. 서로 다른 관리조직 하에 있는 클라이언트와 서버의 네트워크는 서로 다른 조직체를 가진다. 〈그림 2〉와 같이 Realm A에 있는 클라이언트가 Realm B에 있는 서버를 이용하려면 Realm B에도 사용자로 등록을 해야 할까? 그렇게 되면, 클라이언트는 Realm A에도 사용자가 등록되고 Realm B에도 등록되어 중복등록이 되는 결과를 가져온다. 클라이언트가 만 명이라면, 만 명의 클라이언트 계정이 중복되는 결과를 가져온다. 이는 비효율적이고, 사실상 불가능한 시나리오다.

이런 문제를 해결하기 위해 Realm A와 Realm B에서 Kerberos 간의 인증만 됐다면, 다른 Realm에 클라이언트가 등록되지 않더라도 Realm 간의 서비스가 가능하도록 고안했다. 이때 상호 교류하는 각 Realm에 속한 Kerberos 들은 비밀 키를 공유한다. 키를 공유해야만 Kerberos간의 상호 인증이 가능하기 때문이다. 〈그림 2〉를 보며 인증 과정을 정리해보자. Realm A의 클라이언트는 Realm B의 서버를 이용하기 위해, Realm B의 인증서버(AS)를 거치지 않고, 본인이 속한 Realm의 인증서버에 티켓을 요청하는 것을 확인할 수 있다. 이것이 1번 과정이며, 주의할 점은 Realm B의 TGS티켓을 직접 요청하는 것이 아니라 우선 로컬 TGS 티켓을 발급 받고(2번 과정), 로컬 TGS를 이용해 Realm B의 TGS티켓을 발급 받는 다는 점이다(3, 4번 과정).

그 다음 로컬 TGS로부터 발급받은 원격 TGS 티켓을 이용해 Realm B의 TGS 서버에 전달해 Realm B의 서버 티켓을 요청할 수 있다(5번 과정). Realm B의 TGS는 유효한 사용자 티켓의 여부를 판단해, 서버티켓을 발급한다(6번 과정). 서버 티켓을 발급 받은 후 부터는 인증된 사용자로 서버의 서비스를 이용할 수 있다(7번 과정). 이런 과정을 살펴보면, Realm이라는 공동체 영역을 하나의 단일 테두리로 보기 때문에, Realm 간의 인증도 〈그림 1〉과 같은 단일 Kerberos 시스템과 다를 바가 없다는 것을 이해할 수 있다.



〈그림 2〉 Realm 간의 인증(원격 서버 이용)

Kerberos 버전 5

사실 버전 4는 여러 문제점을 갖고 있어 요즘에는 거의 사용되지 않는다. 버전 4를 먼저 다룬 이유는 버전 5를 이해하기 위함이다. 버전 4와 개념은 동일하고, 문제점을 보완한 것이 버전 5이다. 결과적으로, 복잡해 보이는 버전 5를 가지고 먼저 설명한다면, 말 그대로 어렵다고 느꼈을 것이다.

버전 4의 한계와 버전 5의 등장

버전 4에 어떤 문제점이 있고 버전 5에서는 어떻게 보완했는지 살펴보자. 첫째, 버전 4는 DES라고 하는 암호화 알고리즘으로 제한되어 있다. DES는 수출제한도 걸려있고, 최근 암호화 기법들에 비해 보안강도도 다소 떨어진다.

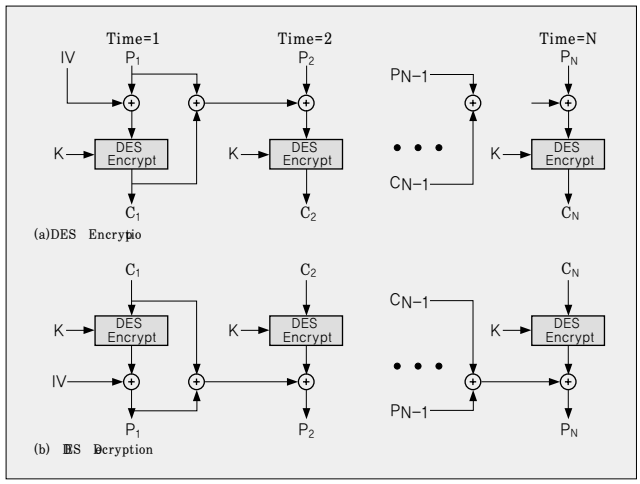
이에 대해 버전 5에서는 클라이언트와 서버가 통신할 때, 알고리즘을 선택하는 프로토콜이 추가돼 모든 알고리즘의 사용이 가능하다. 둘째, 버전 4는 IP 프로토콜에서만 가능하다는 것이다. ATM이나 ISO와 같은 다른 유형의 프로토콜을 고려하지 않아 이기종 네트워크간의 호환이 불가능하다. 버전 5에서는 네트워크 유형도 정할 수가 있으므로, 대부분의 네트워크 간의 호환성을 보장한다. 셋째, 버전 4는 티켓 유효기간에 제한이 있다. 유효기간(Lifetime)은 5분 단위로 8비트를 사용해 부호화를 한다.

결국 표시할 수 있는 최대 유효기간은 28 × 5 = 1280분에 불과하다. 대략 21시간 정도가 된다. 장시간을 필요로 하는 시스템에는 부적합하다. 반면, 버전 5는 티켓의 유효시작시간과 만료시간을 정함으로써, 표시할 수 있는 코드한계에 제한받지 않는다. 넷째, PCBC(Propagating Cipher Block Chaining)라고 하는 비 표준모드의 암호문 블록을 교환하는 기법을 사용함으로써, 공격에 취약하다는 것이 증명됐다. 버전 5에서는 표준모드인 CBC 기법을 사용하도록 보완했다.

PCBC 모드와 CBC 모드

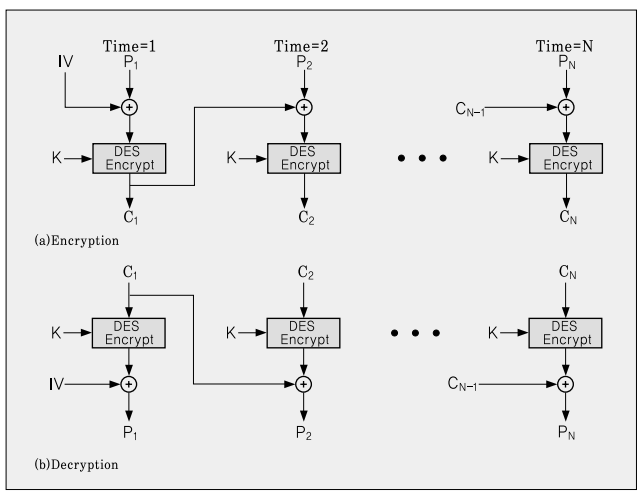
〈그림 3〉은 버전 4에서 사용하는 PCBC 암호화블록 체인모드를 보여준다. 예를 들어, 아이디 정보와 패스워드 정보 문자열이 있다고 하자. 이 정보들은 이어져서 하나의 문자열을 만들게 될 것이다. 문자열을 통째로 암호화 알고리즘에 입력할 수 있는 것은 아니다. 그렇기 때문에 입력의 비트수에 맞도록 블록으로 나누어야 하는데, 그러한 일련의 방법들을 암호화블록 체인모드라고 한다. 〈그림 3〉을 보면, IV라는 것이 있는데, 지난 시간에 설명했던 초기 값을 의미한다.

이 값은 임의로 정의할 수 있는 시드(seed)값과 같은 것이라고 했었다. 이 값과 첫 번째 블록인 P1이 XOR연산을 해서 암호화 알고리즘의 출력으로 C1을 생성한다. 여기서 CBC라고 하는 표준모드와의 차이점은 P1을 다음 블록 연산에 또 사용한다는 것이다. 즉 P1은 평문(Plain Text)인데, 이것이 반복된다고 하면, 보안상 문제가 생길 것이다. 그래서 보안적으로 더 안전한 〈그림 4〉의 CBC 모드를 표준모드라고 하고, 이를 더 많이 사용한다. 버전 5에서 바로 이 CBC 모드를 사용한다.



〈그림 3〉 PCBC 비표준 모드

CBC 모드를 이용한 Kerberos의 암호화 방식



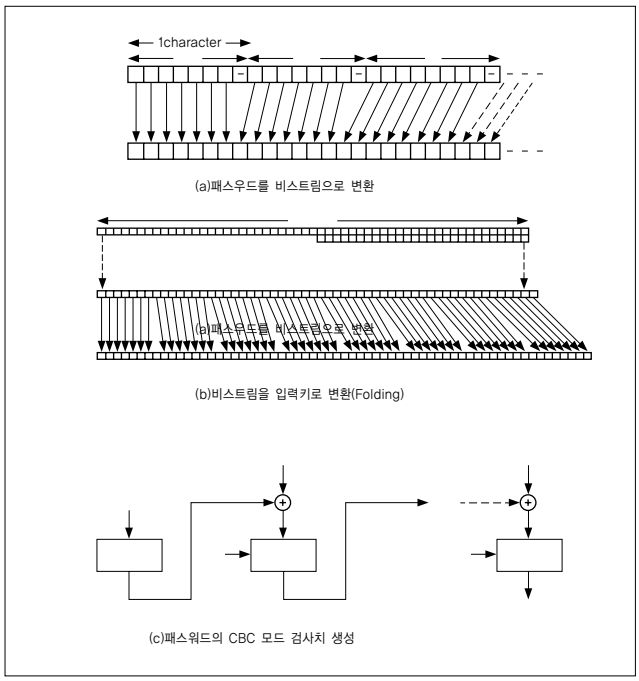
〈그림 4〉 CBC 표준 모드

사용자 패스워드를 CBC 모드를 사용해 암호화하는 과정을 살펴보는 것으로 Kerberos에 대한 전반적인 설명을 정리해보자. 이 과정을 〈그림 5〉에 (a), (b), (c)로 나누어 자세히 정리했다. 먼저 (a)단계에서는, 사용자로부터 입력받은 패스워드 문자열을 비트스트림으로 변환한다. 일반적으로 문자열은 7비트 아스키코드를 사용하는데, 이때 8비트를 할당한다. 즉 1비트가 비워져 있다. 이를 제거한 코드를 비트스트림이라고 한다. 생성된 비트스트림은 사용자가 입력한 패스워드의 길이에 비례해 길이가 제각

각이다. 그러나 일반적인 알고리즘은 입력과 출력 시의 비트 크기를 고정하고 있으므로 (b)단계에서처럼, 비트스트림을 56비트씩 끊어서 겹치도록 접어준다. 이를 폴딩(folding)한다고 한다.

예를 들어, 비트스트림의 길이가 100비트라고 하면, 56비트로 한 세트를 만들고, 나머지 46비트로 한 세트를 만든 후 순서대로 나열한다. (b)의 그림을 보면, 쉽게 이해할 수 있다. 이렇게 층층이 나열한 비트스트림을 열에 맞추어 XOR 연산을 수행한다. 결과적으로 아무리 비트스트림이 길어도 56비트로 정해된다. 그 다음 원래의 7비트 아스키코드 형태로 바꿔주면 64비트 입력키가 완성된다. 여기까지가 (b)단계의 내용이다. 〈그림 5〉에서는 이때의 키를 Kpw라고 한다. 이 키가 〈그림 4〉의 CBC모드에서 K 값이 되는 것이다. 〈그림 4〉의 C1은 〈그림 5〉의 s[0]에서 s[7]까지의 문자열이 되는 것임을 확인하도록 하자.

Kerberos 응용 사례



〈그림 5〉 패스워드에서 암호키 생성하기

대표적인 응용사례로, 마이크로소프트의 윈도우에서 사용자 계정을 관리할 때 Kerberos를 이용한다. KDC라고 하는 키 배포 센터에서 서비스 티켓을 발행한다. 윈도우 사용자라면, 액티브 디렉토리(Active Directory)에 대해 어느 정도 알고 있을 것이다. 액티브 디렉토리에는 도메인 컨트롤러라는 것이 있는데, 인증서버(AS)와 TGS가 이것에 포함되어 설치된다. 액티브 디렉토리에 등록된 사용자가 윈도우에 로그온 하는 과정을 살펴보면, 가장 먼저 도메인을 선택하고 사용자 계정을 입력한다. 이때, UDP 88번 포트를 이용해서 도메인 컨트롤러의 KDC에 해당 도메인에 대한 TGS 티켓을 요청하고, 승인되면, 다시 도메인 컨트롤러의 TGS에 원하는 서버의 티켓을 요청하는 절차가 이뤄진다.

〈그림 1〉의 절차와 동일하다는 것을 알 수 있다. 윈도우 2000이나 2003 서버의 경우, 〔관리도구〕의 〔도메인보안〕에서 Kerberos 정책을 설정하거나 변경할 수 있다. 또한 IIS(Internet Information Server)에서도 〔인터넷 정보서비스〕->〔등록정보〕->〔디렉토리보안〕->〔편집〕으로 들어가서, 인증방법을 윈도우 통합 인증으로 선택하면, Kerberos 인증절차에 의해 웹페이지를 실행할 수 있다. 윈도우 서버를 관리할 때, 액티브 디렉토리는 반드시 필요한데, Kerberos의 기본 동작원리를 이해하고, 기능을 이용한다면, 관리하는데 더욱 도움이 된다. 보안과 관련된 애플리케이션이나 프로토콜은 무척 복잡해보이지만, 내부 동작원리를 살펴보면, 그렇지만은 않다는 것을 이해했으면 좋겠다.

PGP

PGP(Pretty Good Privacy)는 Phil Zimmermann에 의해 고안된 보안 모델로, 다양한 플랫폼에서 사용할 수 있다는 장점을 갖고 있다. 가장 많이 활용되는 응용사례가 전자메일이다. PGP는 디지털 서명을 통해 인증(Authentication) 서비스를 제공하고, 동시에 메시지 암호화를 통해 기밀성(Confidentiality)을 제공한다. 이 두 가지 서비스는 PGP의 핵심이라고 할 수 있으며, 주안점을 두어 살펴 볼 것이다. 또한 메시지의 크기를 줄이기 위해 압축 알고리즘을 사용하고, 전자메일의 호환성을 위한 방법도 제공하는데, 이에 대해서도 정리할 것이다. 본론으로 들어가기에 앞서, 〈리스트 3〉의 용어부터 살펴보자.

PGP 인증 서비스

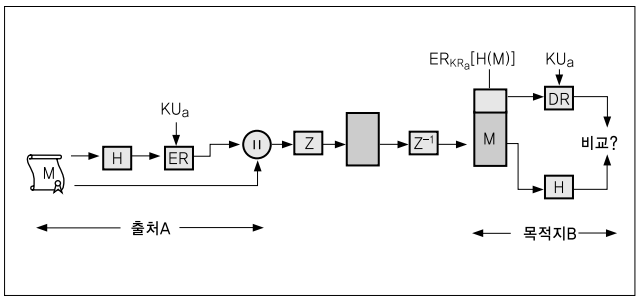
〈리스트 3〉 PGP 관련 용어 정리

```
KS : 세션키
KRa : A의 개인키
KUa : A의 공개키
ER : RSA를 이용한 암호화
DR : RSA를 이용한 복호화
EI : IDEA를 이용한 암호화
DI : IDEA를 이용한 복호화
H  : 해쉬 코드화(MAC)
|| : 이어 붙여 쓰기
Z : ZIP 알고리즘을 이용한 압축
R64 : radix 64 ASCII 형식 (=Base 64)
```

〈그림 6〉은 A라는 출발지에서 B라는 목적지로 PGP의 인증서비스를 이용해서 메시지가 전송되는 흐름을 보여준다. 인증이란, 메시지를 보낸 사람이 본인과 일치하는 가를 증명하는 일련의 과정이다. 도서관에서 책을 빌릴 때, 신분증을 제시하는 것도 인증

의 한 예이다. 즉 메시지를 보낼 때, 신분증을 붙여서 보내는 것이 PGP의 인증 서비스의 핵심이다. 이전 시간에 다뤘던, 해쉬함수가 이때 사용된다. MD5와 같은 알고리즘을 이용해 128비트의 해쉬코드를 생성한다.

우선, 출처 A에서 원문 메시지인 M을 H로 만든다. 즉, 128비

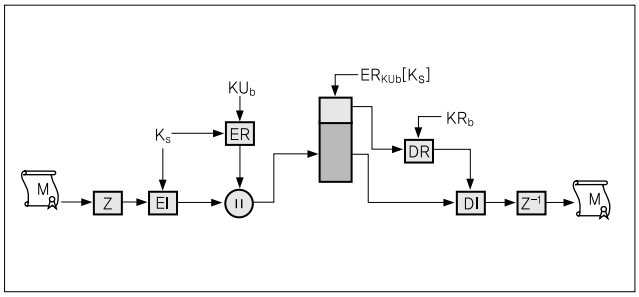


〈그림 6〉 PGP 인증 서비스 (전자서명)

트 해쉬코드가 생성되었다. 그 다음, A의 개인키로 해쉬코드를 RSA로 암호화시켜 암호문으로 만든다. 암호문이 곧, 전자 서명이다. 서명과 원문 메시지 M을 붙인 것을 Zip으로 압축하면 인증을 포함한 메시지가 생성된다. 여기서는 메시지 자체를 암호화한 것은 아님을 유의하기 바란다. 단순히 인증만을 위한 준비를 마친 것이다. 목적지에서는 전송받은 인증 메시지의 압축을 풀면(Unzip), 메시지와 전자 서명을 분리할 수 있다. 전자 서명은 A가 배포한 공개키로 복호화 할 수 있는데, 복호화하면 128비트 해쉬코드가 복원된다. 인증은 여기서 부터다. 만일 인증된 사용자의 메시지가 일치한다면, 복원된 해쉬코드와 전송받은 원문 메시지를 H로 만들어 생성한 코드와 같아야 한다. 같다면, 인증된 메시지로 판단할 수 있으며, 다르면 도중에 변질된 메시지로 판단할 수 있다.

PGP 기밀성(Confidentiality) 서비스

〈그림 7〉은 〈그림 6〉과 다르게 PGP의 기밀성(Confidentiality) 서비스를 이용해서 메시지가 전송되는 흐름을 보여준다. 인증과는 다른 개념으로, 메시지가 도중에 유출되더라도 내용을 알 수 없도록 하는데 그 목적이 있다. 즉 암호화를 한다는 것인



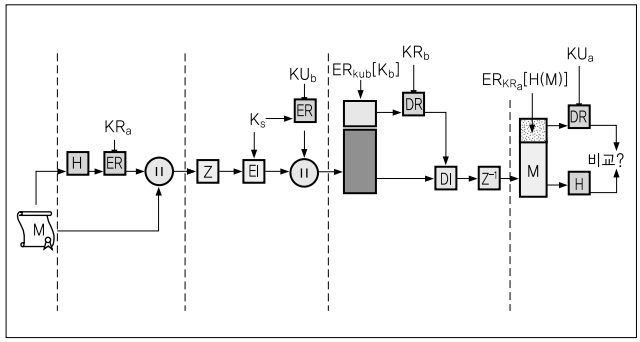
〈그림 7〉 PGP 기밀성 서비스(암호화를 이용한 메시지 보호)

데, PGP에서는 좀 더 체계적인 방식으로 암호화가 이뤄짐을 확인할 수 있다.

여기서 인증 서비스에는 없었던 세션키(Ks)라는 것이 등장하는데, 암호화를 시켜 송신측에서 수신측으로 전송한다. 〈그림 7〉을 보면, 가장 먼저 원문메시지를 Zip으로 압축을 한다. 압축을 한 후에, 세션키를 이용해서 IDEA 암호화 한다. 여기까지 과정에서 메시지는 암호화 된 상태로 유의해서 볼 부분이다. 메시지를 암호화할 때 사용했던 세션키를 수신측(b)의 공개키를 이용해서 RSA 암호화를 한다. 암호화된 세션키와 암호화된 메시지를 붙이면 전송할 준비가 완료된 것이다. 여기서 인증서비스와 구별해야 할 점은 인증서비스는 RSA 암호화를 개인키로 하고 공개키로 복호화하는 반면, 기밀성 서비스에서는 공개키로 암호화하고 개인키로 복호화 한다는 점이다. 〈그림 6〉과 〈그림 7〉을 비교하면서 살펴보기 바란다.

인증과 기밀성은 보안에서 중요한 부분이니 숙지하는 것이 좋다. 그럼, 수신측에서 복호화를 할 때는 어떻게 하면 될까? 복호화를 할 때는 전송 받은 기밀 메시지를 암호화된 공개키와 암호화된 메시지로 분리한다. 다음으로 메시지를 복호화하려면 우선, 세션키를 복호화 해야 한다. 수신측이 b라고 했으므로, 당연이 개인키를 가지고 있을 것이다. 이 개인키를 이용해서 RSA 복호화를 하면 세션키를 얻는다. 세션키를 얻었기 때문에 IDEA 복호화가 가능하다. 이제는 압축만 풀면, 원문 메시지를 얻을 수 있다.

지금까지 인증 서비스와 기밀성 서비스에 대해 살펴봤다. 실제로 PGP는 이 둘을 혼합해서 사용한다. 일상 생활에서 전자메일을 사용할 때를 떠올려 보자. 보낸 이가 본인임을 입증해야 하는 경우를 종종 접한다. 또한 중요한 메일인 경우, 도중에 도청을 방지해야 할 필요성을 느꼈을 것이다. 이런 문제들을 모두 해결해 주는 솔루션이 PGP이다. 〈그림 8〉에서 인증과 기밀성 서비스가 혼합된 형태를 확인할 수 있다. 원문 메시지를 인증하는 절차를 먼저 수행하고, 기밀성 처리를 한 후 결과를 전송하는 형태이다. 반대로 수신 측에서는 기밀성 처리된 메시지를 복호화하고, 전자서명을 확인해 인증하는 형태다.



〈그림 8〉 PGP 인증 & 기밀성 서비스

전자메일 호환성

Radix-64(Base-64) 포맷변환을 써서 기존의 전자 우편 시스템과 호환성 문제를 해결했다. 〈그림 9〉와 같이 8비트 3세트를, 4세트의 Radix-64 아스키코드로 변환한다. 아스키코드는 대부분의 전자메일에서 호환이 된다. 〈그림 10〉은 Radix-64의 테이블을 의미하는데, 알파벳과 숫자를 64개의 테이블로 갖고 있다. 즉 64개를 표시하기 위해 6비트가 필요하며, 〈그림 9〉처럼 4개의 아스키코드는 6비트로 이뤄졌음을 확인할 수 있다. 예를 들어, 'A'라는 문자열은 Radix-64 변환을 거치면, '000000'이 된다.

```
+--first octet--+-second octe--+--third octet--+
|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
+---------------+---------------+---------------+
|5 4 3 2 1 0|5 4 3 2 1 0|5 4 3 2 1 0|5 4 3 2 1 0|
+--1.index--+--2.index--+--3.index--+-- 4.index--+
```

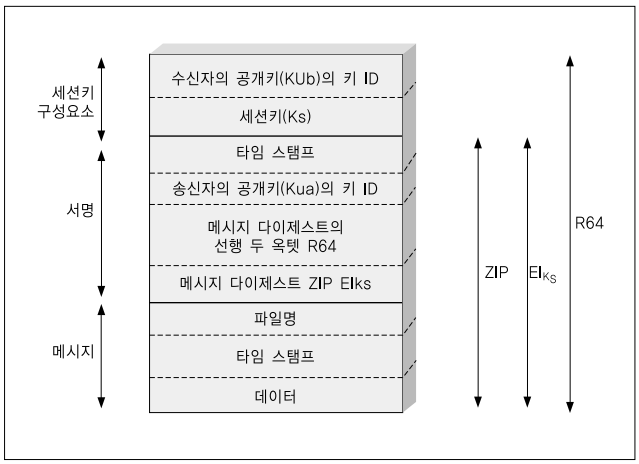
〈그림 9〉 바이트의 Radix-64 변환

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	Z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

〈그림 10〉 Radix-64 인코딩 테이블

PGP 메시지의 형식

〈그림 11〉은 일반적인 PGP 메시지의 형식을 보여준다. 이 메시지 형태는 인증 서비스가 완료된 상태의 메시지이다. 즉, 서명과 메시지가 붙여진 메시지임을 파악할 수 있다. 〈그림 8〉을 다시 확인해 보자. 인증을 했다면 그 다음은 기밀성을 보장할 차례이다. 그 부분이 〈그림 11〉의 메시지 오른쪽 부분에 표시한 서명과 메시지를 Zip으로 압축(ZIP)한 부분과 세션키를 이용해서 IDEA 암호화(EIks)를 하는 부분이다. 세션키 구성요소를 보면, 수신자의 공개키와 세션키 항목이 있는데, 모두 기밀성 서비스에서 이용되었던 것을 기억할 것이다. 세션키는 수신자의 공개키로 RSA 암호화를 하여 암호화된 세션키를 만들어 IDEA 암호화된 메시지에 붙여서 최종적으로 전송할 메시지 형태를 갖춘다. R64는 Radix-64 형식으로 변환하는 것을 의미한다. 앞서 언급했듯이, Radix-64 아스키코드 형식으로 변환해야 전자메일에서 호환성을 보장한다.



〈그림 11〉 메시지 형식

PGP에서 사용되는 암호화키를 정리해보면 〈표 1〉와 같다. 특히 공개키 암호화 시스템에 대해서는 지난 시간의 내용을 바탕으로 한다면, 이해하는데 무리가 없을 것이다.

세션키	IDEA 알고리즘, 각 세션키는 한 번만 사용된다.
공개키	RSA 알고리즘, 세션키를 암호화할 때 이용된다. 송/수신자는 서로의 공개키 배포본의 관리가 필요하다 (공개키 링).
개인키	RSA 알고리즘, 디지털 서명을 위한 메시지 암호화에 이용된다.

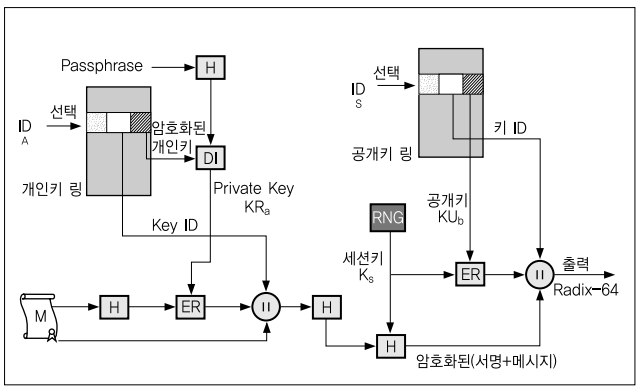
〈표 1〉 PGP에서 사용되는 암호화키 정리

전자메일을 A와 B만 주고받는다면, 공개키와 개인키는 한 쌍씩이면 되겠지만, 여러 사람이 이용한다면 공개키나 개인키를 관리할 수 있는 체계적인 방법이 필요하다. 일상생활에서 자동차 키를 포함해서 여러 개의 열쇠를 가지고 다니는데, 이때 효율적인 관리를 위해 열쇠고리를 이용한다는 것과 같은 의미이다. 보안용어로는 이를 키 링(Key Ring)이라고 부른다. 즉 공개키 링과 개인키 링이 있으며, 이 둘은 약간의 차이가 있다. 먼저, 개인키 링은 사용자 ID나 키 ID로 색인화를 하고 본인의 시스템에만 저장한다. 주로 CAST-128이나 IDEA를 이용해 암호화를 한다.

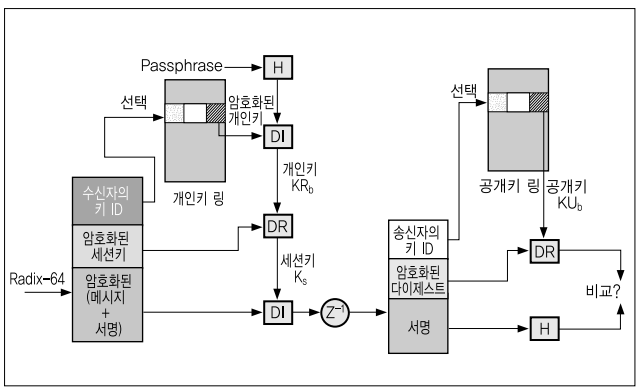
사용자가 개인키를 얻기 위해서는 개인키 링에 패스워드를 입력해야 하며, 패스워드의 해쉬코드를 이용해 CAST-128이나 IDEA로 암호화된 개인키를 복호화 시켜 사용한다. 공개키 링에는 다른 사용자의 공개키가 저장되며, 개인키와 마찬가지로 사용자의 ID와 키 ID로 색인화 한다. 그러나 개인키와는 달리 복호화하는 방법이 복잡해서 주로 신뢰되는 인증기관(CA)의 발급절차를 따른다. 지난 시간에 공인인증서에 대해 다뤘는데, 공인 인증서가 바로 은행의 공개키인 셈이다. 이제, 〈그림 12〉와 〈그림 13〉에서 〈그림 8〉을 더해 키 링 시스템이 추가된 최종적인 PGP 서비스를 정리해보자.

〈그림 12〉는 송신측, 〈그림 13〉은 수신측을 나타낸다. 〈그림

12〉에서 보면, 위에서 설명했던 봐와 같이, 개인키를 개인키 링에서 꺼내어 사용하려면 패스워드(Passphrase)가 반드시 필요하다는 것을 알 수 있다. 개인키는 암호화된 상태로 키 링에 저장이 되어 있으며, 이를 복호화하려면 패스워드의 해쉬코드가 필요하다. RNG 라고 하는 것은 'Random Number Generator' 라고 해서 세션키를 랜덤하게 생성하는 모듈이다. 나머지 부분은 〈그림 8〉의 내용과 같으므로 비교해보길 바란다.



〈그림 12〉 키 링 시스템이 추가된 PGP 서비스(송신)



〈그림 13〉 키 링 시스템이 추가된 PGP 서비스(수신)

PGP 패키지

PGP 배포 사이트(http://www.pgpi.org)에 방문하면 프리웨어 버전을 다운로드 받아볼 수 있다. 그러나 2002년에 배포한 PGP 8.0이 최신버전이다. 2002년 이후로는 대부분 상업용으로 배포되기 시작했기 때문이다. 대표적으로 PGP 닷컴(http://www.pgp.com)을 예로 들 수 있다. PGP 닷컴에서 최신의 PGP 패키지를 다운로드 받아서 설치해보자. PGP 패키지에서 살펴볼 만한 파일로는 pubring.pgp, secring.pgp, randseed.bin 정도를 들 수 있다. pubring.pgp는 자신의 공개키와 다른 사람의 공개키를 함께 저장하는 파일이며, 바이너리 형태로 저장이 된다.

즉 공개키 링을 관리하는 파일이다. secring.pgp는 자신의 개인키만을 저장하며 패스워드를 해쉬코드로 변환한 값이 함께 저장되어 있다. 즉 개인키 링을 관리하는 파일이다. randseed.bin는 공개키/개인키를 생성할 때 필요한, 랜덤한 소수 값을 생성하기 위한 시드값을 저장한 파일이다. 이해가 어렵다면, 지난 시간에 다뤘던 Diffe-Hellman의 키 생성 방법을 참고하기 바란다. 〈화면 1〉은 필자의 PGP Fingerprint를 캡쳐 한 화면이다.

〈화면 1〉 PGP Fingerprint

여기까지해서 Kerberos와 PGP에 대해 살펴봤다. 모두 지난 시간에 다뤘던 암호화 기법들이 기본적인 구성요소가 된다. 다음 호에서는 IPSec이라고 하는 인터넷 보안 프로토콜에 대해 다룰 것이다.

참고 자료

1. Network Security Essentials 2 nd edition (William Stalling)
2. Cryptography and Network Security 4th edition (William Stalling)
3. Network Security :Private Communication in a Public World (PHPTR)
4. 네트워크 보안 프로토콜(윤종호)
5. PGP 공식 사이트 (http://www.pgpi.org)
6. PGP 닷컴 (http://www.pgp.com)