通信情報保護學會誌
第7卷 第3號, 1997. 9

분산 환경에서의 인증 방식; Kerberos와 Yaksha

User Authentications in Distributed Systems ; Kerberos and Yaksha

박 춘 식*

요 약

본 고에서는 현재 분산환경에서의 인증 메커니즘으로 널리 고려되고 있는 Kerberos 인증 방식의 초기 버전인 Kerberos V.4 와 개선된 버전인 V.5 그리고 공개키 암호를 도입하여 Kerberos를 개선한 Yaksha 인증 메카니즘을 소개하고자 한다. 또한 분산 환경에서의 대표적인 두 방식을 비교 분석하여 보았다.

1. 서 론

개방형 시스템인 네트워크를 통한 서버와 클라이언트 사이에서의 서비스 교환은 클라이언트가 자신의 신원을 서버에게 확인시켜주는 과정을 필요로 한다. 암호학에서 인증(Authentication)은 사용자의 신원을 확인시켜주는 것을 말하며 인증은 개체의 신분을 확인하는 사용자 인증과 사용자가 생성한 데이타의 무결성을 확인하는 메시지 인증으로 나누어 볼 수 있다. 인증 방식에서 이용되는 기밀성 서비스는 불법의 도청자로부터 정보를 보호하는데 이용된다. 액세스 제어는 동작 수행이 허용되었는가를 판단하는 과정이고, 일반적으로 인증 후에 수행되며 또는 다른 인증 정보에 기초를 두고 수행한다.

네트워크에 있어서 사용자가 정당한 사용자인가를 확인하기 위한 방안으로써, 현재 가장 널리 사용되고 있는 방식은 패스워드를 주고 받는 방식일 것이다. 그러나 아무런 대책없이 개방된 네트워크를 통하여 패스워드를 주고 받는 방식은 패스워드 자체의 도청에 의한 위험이 존재하고 있다.

패스워드를 이용하여 정당한 사용자인가를 확인하는 방법외에도 최근에는 암호를 이용하는 시도 응답 방법(Challenge-Response), 영지식상호증명(ZKIP:Zero Knowledge Interactive Proof) 등이 알려져 있다. 특히, 분산 환경에서 현재 널리 알려져 있는 대표적인 인증 방식은 역시 Kerberos이다. Kerberos는 그리스 신화에 나오는 지옥의 문을 지키는 3개의 머리를 가진 개를 의미하며 패스워드 대신에 티켓이라는 개념을 이

* 한국전자통신연구원 책임연구원

용하여 패스워드 방식의 결점을 보완한 일종의 시도 응답 방식의 인증 방식이다.

또다른 인증 방식은 Yaksha로 힌두교 신화에 나오는 천국의 문을 지키는 반은 신이며 반은 사람으로 새나 호랑이나 다른 어떠한 모습으로 변신이 가능하다. 의미가 나타내는 바와같이 Yaksha는 Kerberos가 사용되는 곳에도 적용 가능하며 다양한 기능을 제공하므로, 다른 응용 분야에 많이 활용될 것으로 예상된다.

본 논문에서는 인증 메커니즘인 Kerberos 및 Yaksha를 살펴보고 이에 대한 문제점을 분석 비교하였다.

제2장에서는 Kerberos의 초기 버전인 V.4와 개선된 V.5를 다루었으며, 공개키 암호를 이용한 Yaksha는 제3장에서 그리고 이들을 비교 분석한 내용은 제4장에서 다루었다. 마지막으로 결론부를 제5장에 언급하였다.

2. Kerberos 인증 방식

2.1 개요

Kerberos 인증은 MIT가 대학내의 네트워크를 구축하기 위해, 1983년 IBM, DEC과 공동으로 시작한 Athena 프로젝트에서 개발된 분산 환경에 적합한 인증시스템으로 패스워드 도청에 의한 누설과 사용자가 매번 패스워드를 입력해야 하는 문제점을 해결한 방식이다. 게다가 경우에 따라서는, 클라이언트와 서버간 데이터의 기밀성과 무결성을 제공할 수 도 있다. Kerberos는 분산망에서의 인증 시스템으로서 프로세스인 클라이언트가 특정 사용자를 대신하여 검증자에게 사용자의 신분을 확인시켜 주기 위한 일련의 암호화된 메시지를 교환하는 과정이다. 많은 시스템에서 Kerberos V.4를 사용하고 있으나, 현재, 인터넷 Draft 표준화(RFC 1510) 상태인 Kerberos V.5의 사용이 점차 증가할 전망이다.

Kerberos 인증 메카니즘은 인증 서버(AS : Authentication Server)와 티켓 허가 서버(TGS : Ticket Granting Server)로 나누어져 있어 사용자가 여러번 반복하여 패스워드를 입력해야 하는 불편이 없으며 또한 클라이언트에 대한 인증뿐아니라 서버에 대한 인증도 동시에 수행되는 상호 인증(Mutual Authentication)을 제공하고 있다.

Kerberos 인증 방식은 Needham-Schroeder[2]의 기본 프로토콜에 Denning-Sacco[3]가 지적한 time-stamp 정보를 추가하고, 티켓 개념을 도입한 방식으로 분산 환경에서의 인증과 키 교환을 TTP(Trusted Third Party)를 이용하여 수행하는 방식으로 초기 버전인 V.4와 개선된 버전인 V.5가 알려져 있다. Kerberos V.4에서는 비밀키 암호 알고리즘인 DES만을 지원하는데에 비해 Kerberos V.5에서는 DES이외에도 RSA등과 같은 공개키 암호 알고리즘들을 지원하도록 되어있다. 그러나 공개키 암호 알고리즘을 이용한 구체적인 과정이나 구현은 아직 알려져 있지 않다.

Kerberos에서 비밀키 암호 알고리즘을 사용하는데는 여러 가지 문제점이 있다. 첫째, 비밀키를 이용하는 Kerberos에서 사용하고 있는 알고리즘인 DES가 안전성에 대한 문제를 갖고 있다는 점이다. Kerberos 인증 전과정에서 사용하고 있는 DES가 안전성에 문제가 있다는 것은 침입자의 공격이 더 쉬워졌다는 것을 의미하여 따라서 Kerberos 전체의 안전성에 영향을 끼치게 된다. 둘째, 비밀키를 이용할 경우 인증과정에서 Kerberos는 세션키의 역할을 하는 두 개의 비밀키를 생성하고 보유하고 있어야 한다. 즉, Kerberos는 키분배센터(KDC : Key Distribution Center)의 역할을 하게 된다. 따라서 클라이언트나 서버의 입장에서는 Kerberos 인증과정이 상호인증임에도 불구하고 Kerberos에 대한 매우 높은 신뢰도를 가지고 있어야만 한다는 문제점이 있다. 세째, 많은

서비스를 동시에 받고자 할 때 클라이언트는 많은 세션키를 관리하고 있어야 한다. 세션키는 클라이언트와 해당되는 서버만이 보유하고 있는 키이므로 많은 세션키를 동시에 클라이언트가 관리하는 것은 심각한 위험이 따르게 된다. 따라서, Kerberos V.5에서는 비밀키 암호 알고리즘보다 안전성이 높은 공개키 암호 알고리즘이 사용된다.

다음은 Kerberos 인증 방식에 요구되는 조건들을 정리하였다.

- 네트워크 도청자가 사용자를 가정해서 필요한 정보를 얻을 수 없어야 한다.
- 신뢰성 있게 동작해야 한다.
- 인증 과정의 부가로 인해 사용자에게 패스워드 입력 외에 별도의 정보를 요구하지 않아야 하는 투명성이 제공되어야 한다.
- 많은 클라이언트와 서버를 지원할 수 있는 규모의 가변성에 대한 요구이다.

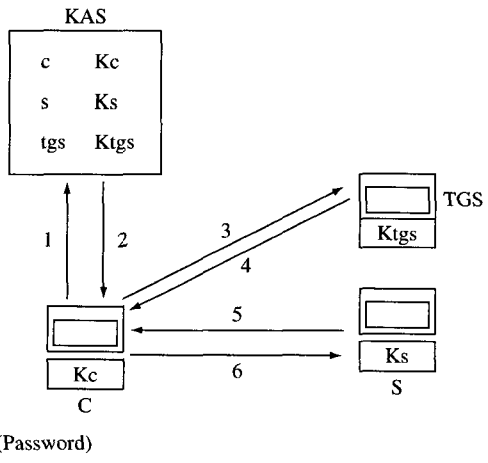


그림 1 : Kerberos 인증 방식

Kerberos의 인증 방식은 그림1과 같으며, KAS(Kerberos Authentication Server)와 TGS(Ticket Granting Server), 서버와 클라이언트로 구성된다. KAS는 모든 사용자의 패스워드나 관련 비밀키를 모두 관리하고 있으며, TGT(Ticket Granting Ticket)를 생성한다. 그리고, TGS는 SGT(Service Granting Ticket)를 발생하여 서비스마다의 인증이 이루어지도록 제공하고 있다.

2.2 Kerberos V.4 인증 방식

Kerberos의 동작은 KAS라는 인증 서버, TGS라는 티켓 승인 서버, 그리고 클라이언트와 서버간의 티켓 발행 단계, 서비스 승인 단계를 각각 거쳐서 이루어지는 것이 기본적인 것으로, Kerberos V.4의 단계별 동작 설명은 다음과 같다.

[단계1] (메시지 1)

$$C \rightarrow KAS : ID_c, ID_{tgs}, TS_1$$

클라이언트 C는 그림 1에서와 같이 ID_c, ID_{tgs}, TS_1 를 KAS에 보내어 서비스를 요청한다. 여기서, ID_c는 사용자의 신분을 알리기 위한 클라이언트 C의 ID이고, ID_{tgs}는 액세스를 원하는 TGS 의 ID이며, TS_1 는 Timestamp 정보로 현재 서비스를 요구하는 시각을 나타낸다. TS_1를 이용하여 KAS는 AR(Authenticative Request) 정보가 시기 적절한가를 확인한다. TS_1 정보는 replay 공격의 대책으로 사용된다.

[단계2](메시지2)

$$KAS \rightarrow C : E_{K_c}[K_{c,tgs}, ID_{tgs}, TS_2, lifetime_1, TGT]$$

KAS는 클라이언트와 TGS간의 세션키와 TGT를 각각 생성하여 사용자의 패스워드를

일방향 해쉬 함수에 적용하여 구한 사용자 키 K_C로 암호화하여 클라이언트 C에게 전달한다. $K_{c,tgs}$는 클라이언트와 TGS간의 암호 통신을 위한 세션키로 K_c로 암호화된 메시지 내부에 있기 때문에 클라이언트 C만이 $K_{c,tgs}$를 알 수 있다. K_c는 사용자의 패스워드로부터 유도된 암호키로서 패스워드를 일방향 해쉬 함수를 이용하여 구한 비밀키이다. TS_2는 TGT 가 발행된 시간이며, $lifetime_1$는 TGT의 유효시간을 의미한다. 여기서, $TGT = E_{K_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, lifetime_1]$ 이며, AD_c 는 TGT를 사용할 클라이언트의 망 주소이다.

[단계3](메시지3)

$$C \rightarrow TGS : ID_s, TGT, Authenticator_c$$

클라이언트 C는 KAS로부터 수신하여 구한 $K_{c,tgs}$를 이용하여 생성한 Authenticator와 TGT를 TGS에게 전송한다. 여기서, 인증자, $Authenticator_c = E_{K_{c,tgs}}[ID_c, AD_c, TS_3]$이며, TGT의 유효성을 검증하기 위한 정보로 클라이언트 C와 TGT간의 세션키 $K_{c,tgs}$로 ID_c, AD_c, TS_3를 암호화한 암호문이다. TS_3 는 인증자가 생성된 시간을 나타내는 정보로서, 재생 공격을 방지하기 위하여 매우 짧은 유효시간을 갖는다.

TGS는 자신의 비밀키인 K_{tgs}로 TGT를 복호화하여 $K_{c,tgs}$, ID_c, AD_c, ID_{tgs}, TS_2, $lifetime_1$ 를 복구한다. 그 다음 $K_{c,tgs}$를 사용하여 인증자를 복호화 한 후, TGS는 TGT에서 복구된 ID_c, AD_c와 인증자로부터 복구된 ID_c, AD_c가 서로 일치하는지 확인하므로써, 클라이언트의 정당성을 확인하게 된다.

[단계4](메시지4)

$$TGS \rightarrow C : E_{K_{c,tgs}}[K_{c,s}, ID_s, TS_4, SGT]$$

TGS는 TGT로부터 구한 $K_{c,tgs}$를 이용하여, 서버 S와 클라이언트와의 세션키 $K_{c,s}$ 그리고 SGT를 암호화하여 클라이언트 C에게 보낸다. 메시지 4에 포함된 ID_s는 서버 S의 ID, TS_4는 SGT가 발행된 시간을 나타내며, SGT는 $SGT = E_{K_s}[K_{c,s}, ID_c, AD_c, ID_s, TS_4, lifetime_3]$이다. SGT내의 $lifetime_3$는 SGT의 유효시간을 의미하며, $K_{c,s}$, ID_c, AD_c, ID_s, TS_4, $lifetime_3$을 TGS와 서버 S가 사전에 공유하고 있는 비밀키인 K_s로 암호화한 암호문이다.

[단계5] (메시지5)

$$C \rightarrow S : SGT, Authenticator$$

클라이언트 C는 TGS와의 세션키를 이용하여 암호문으로부터 서버와 클라이언트용의 세션키 $K_{c,s}$를 복구한 후, 이를 이용하여 구한 인증자를 SGT와 함께 서버 S에 전송한다. $Authenticator$는 $K_{c,s}$로 ID_c, AD_c, TS_5를 암호화한 결과로, $Authenticator = E_{K_{c,s}}[ID_c, AD_c, TS_5]$이다. SGT는 클라이언트 C가 KAS에 의해 인증받았음을 확인하기 위한 정보로, 재사용이 가능하다. ID_c는 TGT의 정당한 소유자의 ID이고, AD_c는 클라이언트의 주소로서 다른 주소를 갖는 터미널로부터의 TGT사용을 방지하기 위한 것이다. 인증자는 SGT를 제시하고 있는 클라이언트가 SGT를 발행받은 정당한 클라이언트임을 입증하기 위한 정보이며, TS_5는 인증자가 생성된 시간이다.

[단계6](메시지6)

$$S \rightarrow C : E_{K_{c,s}}[TS_5+1]$$

서버 S는 SGT에서 복구된 ID_c AD_c가 $K_{c,s}$를 이용하여 인증자로부터 복구된 ID_c, AD_c가 일치하는 지 검사하여 클라이언트 C의 정당성을 확인한다. 정당한 클라이언트 C이면, 서버 S는 자신이 정당한 서버 S임을 증명하기 위해 메시지 6을 클라이언트 C에게 보내며, 클라이언트 C는 이를 복호화한 후, TS_5+1 를 확인하여

서버 S의 정당성을 확인한다.

Kerberos V.4는 Athena 프로젝트 환경내에서 사용하기 위해 개발되었으므로 일반적인 요구를 완전하게 충족시키지 못했다. 버전4의 주요 환경적인 단점은 다음과 같다.

- 특정 암호시스템에 의존한다(DES).
- 인터넷 프로토콜에 의존한다.
- 메시지를 보내는 사람이 바이트의 순서를 자신이 선택하고, 가장 왼쪽에서 있는 바이트가 가장 낮은 주소를 갖거나 가장 오른쪽에서 있는 바이트가 가장 낮은 주소에 있도록 메시지에 표시를 한다.
- 티켓의 유효 기간에 문제가 있다.유효시간 값은 5 분 단위로 8비트 분량으로 부호화된다(21 시간을 조금 넘는 길이다.).
- 다른 어떤 호스트로 전송되는 신임장을 한 클라이언트가 발행하거나 발행된 신임장을 다른 클라이언트가 사용하는 것을 허락하지 않는다.
- N 영역사이의 상호 동작을 위하여 N^2의 관계를 요구한다.

Kerberos V. 4의 주요 기술적인 단점은 다음과 같다.

- 티켓에 이중 암호화를 수행한다.
- 암호는 PCBC(Plain and Cipher Block Chaining)이라고 알려진 DES의 비표준 모드를 사용하고 있다.
- 각 티켓에는 티켓과 관련된 인증자를 생성하기 위하여 클라이언트가 사용하는 세션키를 가지고 있다. 또한 세션키는 인증과정 이외에 세션 동안 클라이언트와 통신 상대간에 메시지를 보호하기 위한 암호키로 계속해서 사용될 수 있다.

2.3 Kerberos V.5 인증 방식

기본적으로 Kerveros V.5는 V.4의 내용과 유사하다. 메시지 1, 3, 5는 동일하며 메시지 2, 4에서의 메시지의 이중 암호가 삭제되고 단한 번의 암호로 나뉘어져 수행되는 부분이 크게 다른 점이다.

내부적으로 Kerberos V.5는 Kerberos V.4의 부분들을 개선 보완하였다. 버전 4가 효율적이며 단순하나, TCP/IP 네트워크에만 적용될 수 있는 반면, 버전 5는 보다 기능화하여 다양한 곳에 적용될 수 있도록 한 점이다. 또한, 버전 4에서 메시지의 형태는 고정된 variable-length field를 취하고 있으나, Basic Encoding Rule (BER)인 ASN.1(Abstract Syntax Notation 1) syntax를 버전 5가 취하여, 앞으로의 확장성을 더욱 용이하게 개선하였다. 버전 4가 사용한 암호 알고리즘은 DES에 국한되어 있으나, 버전 5에서는 DES에 국한하지 않고 어떠한 암호 알고리즘도 사용할 수 있도록 하였다. 이것은 현재 DES가 갖고 있는 문제점인 안전성 문제나 미국에서의 수출 규제 문제를 해결할 수 있으며 동시에 비밀키의 크기도 가변할 수 있어 더욱 사용의 범위를 확대할 수 있도록 하였다. 이외에도 메시지 포멧의 간단화, 티켓 lifetime의 변경(버전 4에서는 약 21시간정도이나 버전 5에서는 무한대의 시간까지 가능하도록 함)등이 있다. Kerberos V.4와의 다른 부분을 중점으로 Kerberos V.5의 동작을 단계별로 설명하면 다음과 같다.

[단계1](메시지 1)

$$C \rightarrow KAS : ID_C, ID_{tgs}, TS$$

Kerberos V.4와 내부적으로는 약간의 차이는 있으나 거의 유사한 형태이므로 간략화하여 동일하게 표시하였다.

[단계2](메시지2)

$$KAS \rightarrow C : E_{K_c}[K_{c,tgs}, ID_{tgs}, TS_2, lifetime_1], TGT$$

단계2는 Kerberos V.4와의 외형적인 차이점이 존재하는 곳으로, TGT가 이중 암호되지 않은 채, 전송되고 있다. 여기서 TGT는 Kerberos V.4의 내용과 동일하며 다음과 같다.

$$TGT = E_{K_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, lifetim_{e_1}]$$

[단계3](메시지3)

$$C \rightarrow TGS : ID_s, TGT, Authenticator_c$$

단계3도 Kerberos V.4의 부분과동일하다.

[단계4](메시지4)

$$TGS \rightarrow C : E_{K_{c,tgs}}[K_{c,s}, IDs, TS_4], SGT$$

단계 4는 Kerberos V.4의 단계 4와는 다른 부분으로, TGS가 메시지 3을 수신하여 TGT의 유효성을 검증한후, TGT에서 구한 세션키를 이용하여 인증자의 유효성을 검증한다. 유효성 검증이 완료되면 해당 서버를 확인한 후 , 서버와 클라이언트간의 세션키 $K_{c,s}$를 클라이언트와 TGS간의 세션키 $K_{c,tgs}$로 암호화하여 SGT와 함께 클라이언트 C에게 전송한다.

[단계5] (메시지5)

$$C \rightarrow S : SGT, Authenticator_C$$

Kerberos V.4와 동일한 메시지로 클라이언트 C는 클라이언트 C와 서버 S간의 세션키를 이용하여 구한 인증자를 SGT와 함께 서버로 전송한다.

[단계6](메시지6)

$$S \rightarrow C : E_{K_{c,s}}[TS_5, subkey, Seq]$$

Kerberos V.4와 거의 동일한 것으로, 클라이언트가 선택한 응용 세션을 위한 암호키인 subkey, 세션 동안 서버가 클라이언트에게 보낸 메시지의 순번인 Sequence(Replay 공격 방지용)가 내부적으로 사용되는 점외에는 버전 4에서 제공하는 상호 인증을 그대로 제공하고 있다. Kerberos V.4와는 달리 클라이언트가 보낸 인증자의 타임스템프가 그대로 포함되어 있으므로 서버는 타임스템프를 1 증가한 값을 되돌려 주지 않는다. 이는 메시지 특성상 침입자가 암호키를 모르고서는 절대로 메시지 6을 만들수 없기 때문이다.

앞에서도 설명한 바와 같이 Kerberos V.5는 버전 4의 단점들을 많이 개선하였다. 버전 5는 버전 4의 환경적인 측면과 기술적인 측면에서의 문제점들을 보완 개선하였는데 주요 내용을 정리하면 다음과 같다.

- 임의의 암호 기술을 사용할수 있도록 암호화 형식 식별자를 이용한다.
- 모든 메시지 구조가 ASN.1과 명확한 바이트 순서를 제공하는 기본 부호 규칙을 사용한다.
- 티켓은 불규칙한 유효 시간을 허용하면서 명백한 시작과 끝 시간을 가지고 있다.
- 클라이언트가 서버에 접속하게 하면 서비스를 제공하는 서버는 클라이언트 입장에서 또 다른 서버에 접속할수 있는 기능을 제공한다.
- N 영역간의 상호 동작을 가능하게 한다.
- 티켓의 이중 암호에서 단일 암호로 바꾸었다.
- 표준 CBC(Cipher Block Chaining)모드를 사용한다.

- 클라이언트와 서버가 오직 한번의 접속을 위해서만 사용할수 있는 서브 세션키를 서로 협상하여 구할수 있다.

3. Yaksha

3.1 개요

Yaksha는 사용자 인증 기능을 제공하는 Kerberos와의 정합성을 충분히 고려하여 설계하였지만, Kerberos가 제공하지 않는 디지털 서명을 제공하는 것이 커다란 장점이다. 또한, 세션 설정시마다 사용자간의 키공유도 가능하여, 암호 통신도 제공할 수가 있다. 더욱이, 최근 화제가 되고 있는 키 위탁 제도에 대한 기능도 고려하고 있어 앞으로 많은 활용이 예상되는 인증 방식이다.

Yaksha는 Kerberos가 가지는 다음과 같은 3가지 문제점을 가능하면 Kerberos의 형태를 유지하면서 해결한 인증 방식이다.

- Kerberos 인증 서버가 클라이언트의 비밀키를 보관하고 있다.
- 티켓 발행에 의한 사용자 인증을 실현하고 있지만 디지털 서명 기능은 제공하고 있지 않다.
- 패스워드에 대한 dictionary 공격 가능성이 있다.

3.2 Yaksha

Yaksha 인증 방식은 다음과 같다.

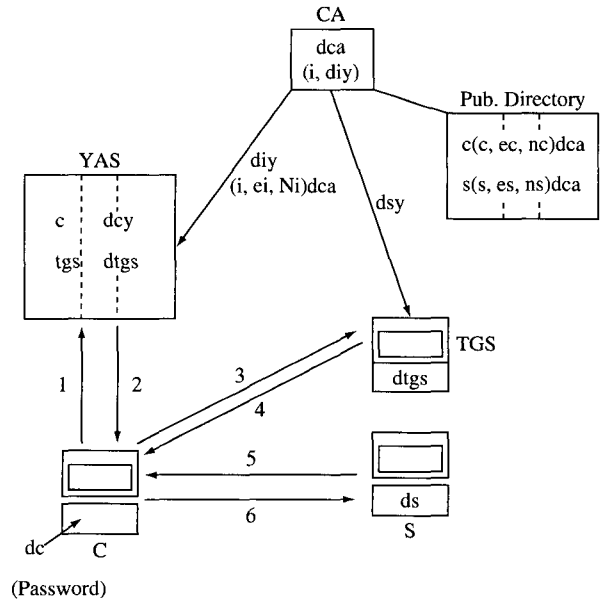


그림 : 2 Yaksha 인증방식

1) 클라이언트-인증 서버간의 교환

[단계1](메시지 1)

$C \rightarrow YAS : C, TGS, T_{exp}, ((TEMP - CERT)^{dc}, R)^{dc}$

클라이언트 C는 그림 2에서와 같이 일시적인 공개키를 포함한 TEMP-CERT를 작성하여, $((TEMP\text{-}CERT)^{dc} \| R)^{dc}$의 형식으로 Yaksha 인증 서버에 전송한다. TEMP-CERT는 $TEMP\text{-}CERT = C \| e_{c,temp} \| n_{c,temp} \| T_{exp} \| \cdots$로 뒤의 단계에서 복호가 되므로, $(TEMP\text{-}CERT, (TEMP\text{-}CERT)^{dc})$를 이용한 dc에 관한 Dictionary 공격을 방지하기 위하여, 랜덤수 R을 첨부하여 $(TEMP\text{-}CERT)^{d_c}$를 숨기고 있다. Yaksha 인증 서버는 $(TEMP\text{-}CERT)^{dc} \| R$를 복호한 후, 다시 그 결과를 복호한 후, TEMP-CERT를 최종적으로 가지게 된다.

[단계2](메시지 2)

$$YAS \rightarrow C : (K_{C.TGS}, T_{exp}, R, \cdots)^{e_{c.temp}}, (T_{C.TGS})^{d_{tgsy}}, ((TEMP\text{-}CERT)^{d_c})^{d_{sy}}$$

단계 2에서는, Kerberos에서의 DES 사용 대신에 RSA 암호를 사용하는 점이 다르다. Yaksha 인증 서버가 보내는 첫번째 메시지는 생성한 키인 $K_{C,TGS}$ 등을 평문으로 하여 일시적인 공개키 $(e_{c,temp}, n_{c,temp})$를 이용하여 암호화한 암호문이다. 세번째 메시지는 클라이언트 C의 공개키 (e_c, n_c)를 이용하여 복원될 수 있도록 되어있다. 여기서, $T_{C,TGS} = C, TGS, Addr, TS, K_{C.TGS.}$

2) 클라이언트-티켓 허가 서버간의 교환

[단계3](메시지 3)

$$C \rightarrow TGS : S, T_{exp,R}, \cdots, ((T_{C,TGS})^{d_{tgsy}})^{dc.TEMP}, E_{Kc,TGS}(A_C), ((TEMP\text{-}CERT)^{dc})^{d_{cy}}$$

단계 3에서는 $((T_{C,TGS})^{d_{tgsy}})$의 정보를 클라이언트의 일시적 비밀키인 $d_{C,TEMP}$를 이용하여 디지털 서명된 정보가 보내어지고 있다. 이것은 Yaksha 인증 서버로부터 d_{tgsy}가 도난당한다 하더라도, 정당한 클라이언트를 경유하지 않고서는 티켓 발행 요구를 할 수 없도록 하기 위한 수단이다. 디지털 서명의 최종 확인은 $T_{C,TGS}$ 내의 정보와 A_C내의 정보와의 비교로 가능하다.

[단계4](메시지 4)

$$TGS \rightarrow C : E_{KC,TGS}(K_{C,S}, T_{exp}, R, S), (T_{C,S})^{d_{ty}}, ((T_{C,TGS})^{d_{tgsy}})^{d_{tgs}}$$

단계 4에서는 클라이언트가 희망하지 않는 티켓 발행 서비스를 요청하는 부정에 대한 방지책으로, 티켓 발행 서버의 비밀키 d_{tgs}를 이용해서, $(T_{C,TGS})^{d_{tgsy}}$를 서명하고 있다.

3) 클라이언트-서버간의 교환

[단계5](메시지 5)

$$C \rightarrow S : E_{Kc,s}(A'_C), ((T_{C,S})^{d_{sy}})^{dc.TEMP}, ((TEMP\text{-}CERT)^{d_c})^{d_{cy}}$$

[단계6](메시지 6)

$$S \rightarrow C : ((T_{C,S})^{d_{sy}})^{d_s}$$

단계 5는 단계 3과 유사하며 단계 6은 상호인증을 위한 것으로 서버의 공개키를 이용해서 누구든지 검사할 수 있는 형식으로 되어있다.

3.3 Yaksha에서의 디지털 서명과 키 위탁 방식

Yaksha 인증 방식에서는 Kerberos 인증 방식에서 제공되지 못한 디지털 서명과 키 위탁 방식을 제공할 수 있다.

3.3.1 디지털 서명

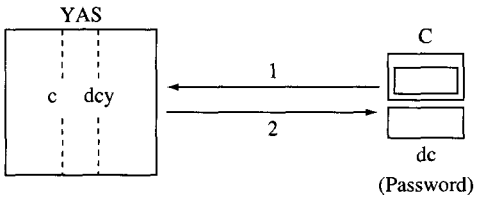


그림 3 : Yaksha 디지털 서명 방식

[단계1](메시지 1)

$$C \rightarrow YAS : C, ((h(m) \| ts)^{d_c} \| R)^{d_{C,TEMP}}, (TEMP\text{-}CERT)^{d_c} \| R')^{d_c})$$

단계 1은 Dictionary 공격을 고려한 암호문의 생성과 일시적인 비밀키 $d_{C,TEMP}$를 사용하는 형태로 Yaksha 인증 서버와 클라이언트 인증을 행하고 있다. 먼저,

$$(TEMP\text{-}CERT)^{d_c} \| R')^{d_c})$$

로부터 올바른 일시적 공개키인 $(e_{C,TEMP}, n_{C,TEMP})$를 계산해 낼 수 있는 것은 d_{cy}를 알고 있는 Yaksha 인증 서버뿐이다. 또, 일시적인 공개키를 계산하여, $(h(m)\ ts)^{d_c}$를 얻은 Yaksha 인증 서버는 $h(m) \| ts$를 복원하여 ts가 정확한지 여부를 판단하여 클라이언트 C가 서명 작성을 요구하였는 지를 확인할 수가 있다.

[단계2](메시지 2)

$$YAS \rightarrow C : ((h(m) \| ts)^{d_c})^{d_{cy}} \| R'')e_{C,TEMP}$$

단계 2에서는 $(h(m) \| ts)^{d_c}$가 누구라도 확인할 수 있는 $(h(m) \| ts)^{d_c})^{d_{cy}}$로 변환되어, d에 대한 Dictionary 공격을 방지하기 위해 난수 R''를 사용하여 암호화하였다. 메시지 m에 대한 디지털 서명은 CA(Certificate Authority)에 의한 공개키 증명서를 이용하면 $(m, ((h(m) \| ts)^{d_c})^{d_{cy}}, (c \| e_c \| n_c)^{d_{ca}})$가 된다.

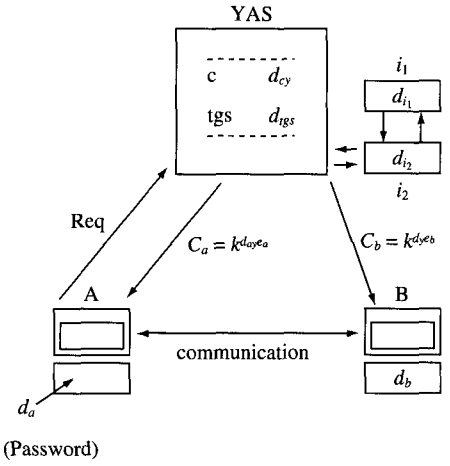


그림 4 : Yaksha 키 공유와 위탁방식

3.3.2 키 위탁 방식

Yaksha 인증 방식에서는 키 공유 방식과 이에 대한 키 위탁 방식을 그림 4와 같이 제공할 수 있다. 암호 통신에 앞서서, Yaksha 인증 서버는 온라인으로 세션키 k를 클라이언트 A와 B의 공개키를 이용하여 암호화하여 보내어준다. 이때 클라이언트 A에게는 $C_a = k^{d_{cy}e_a}$를 B에게는 $C_b = k^{d_{cy}e_b}$가 주어진다. 이때 클라이언트 A와 B는 자신의 비밀키 d_a와 d_b를 각각 이용하여 복호화하면 세션키를 공유할 수가 있다.

키 위탁 방식은 Yaksha 인증 서버가 세션키 k를 제출함으로써 실현될 수 있다. 수사기관이 2개인 경우를 고려할 때, 각각의 비밀키를 d_{i1}과 d_{i2}라 하면, 법원 등으로부터 발부 받은 영장 Z를 메시지로 하여 먼저 수사기관 i_1이 디지털 서명을 한 결과, $Q_{i_1} = Z^{d_{i1}}$를 수사기관 i_2에 보낸다. 수사기관 i_2는 이 정보를 다시 자신의 비밀키로 디지털 서명하여 Yaksha 인증 서버에게 제출하면 된다. 이때, 인증 서버

는 정당한 수사기관으로부터의 요구인지 그리고 법원등으로부터의 영장 내용 Z를 확인한 후, 수사기관들이 복호할 수 있는 형태로 즉, $k^{d_{i1},e_{i1}}$와 $k^{d_{i2},e_{i2}}$로 보내어주면 된다.

4. Kerberos와 Yaksha의 비교

Kerberos와 Yaksha의 비교는 이미 설명한 바와 같이, 비밀키의 사용이냐 공개키 암호의 사용이냐 또는 디지털 서명의 제공 여부 등으로 나누어 볼 수 있다. 본 장에서는 이러한 내용을 알기 쉽게 요약하여 표 1에 나타내었다.

표 1 Kerberos와 Yaksha의 비교

구 분	Kerbers	Yaksha
의 미	지옥문을 지키는 머리3개의 개	천국문을 지키는 반은 신, 반은 인간
제 안 자	MIT Athena 프로젝트	Bell Atantic R. Ganesan
제안 년도	1989년(V.5)년	1995년
기본 방식	Needhan-Schroeder[2] Denning-Sacco[3]	Boyd[6] Kerberos
인증 모델	AS, TGS, C, S로 구성 티켓 발행, 서비스 발행	Kerberos와 동일 Kerberos와 동일
인증 방식	티켓 개념의 시도 응답방식	Kerberos와 동일
사용 알고리즘	비밀키 암호	RSA + 비밀키 암호
디지털 서명	미제공	제공
상호 인증	가능	가능
세션키 공유	조건적 가능	용이
키 위탁	기능 없음	가능
안 전 성	dictionary 공격에 약함	dictinary 공격에 견딤
인증서버 의존도	모든 비밀키를 보관	비밀키를 분리보관
표 준	인터넷 RFC1510	미정
확 장 성	?	Security Infrastructure 가능
기타 사항	비밀키 관리 필요	공개키 관리 및 인증기관 필요

5. 결 론

본고에서는 현재 분산 환경에서의 인증 시스템으로 많이 검토되고 있는 Kerberos 인증 방식의 초기 버전인 V.4와 그리고 개선된 V.5에 대해서 검토하여 보았다. 또한 최근 Kerberos와의 정합성을 고려하여 공개키 암호를 적용한 Yaksha 인증 방식에 대해서도 살펴 보았으며 두 방식의 비교도 아울러 수행하였다.

참 고 문헌

[1] R. Ganesan, "Yaksha:Augmenting Kerberos with Public Key Cryptography," Proc. of ISOC Symposium on Network and Distributed System Security, pp. 132 -- 143, 1995.

[2] R.M. Needham and M.D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM, Vol. 21, No. 12, pp. 993 -- 999, 1978.

[3] D.E.Denning and G.Sacco, "Timestamps in Key distribution Protocols", Comm. of ACM, Vol. 24, pp. 533 -- 536, 1981.

[4] B.C.Neuman and T. Ts'0, "Kerberos: An Authentication Service for Computer Networks", IEEE Comm. Magazine, Vol. 32, No. 9, pp. 33 -- 38, 1994.

[5] J.Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)",Internet RFC 1510, Sept, 1993.

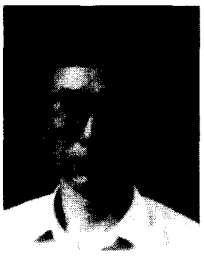
[6] C. Boyd, "A New Multiple Key Cipher and an improved Voting Scheme", Advances in Cryptology, Proceedings of EUROCRYPT'89, Springer-Verlag, pp. 617 -- 625, 1990.

[7] C. Kaufman, R. Perlman and M. Speciner, Network Security, Prentice Hall, 1995.

[8] B. Schneier, Applied Cryptography, John Wiley and Sons, 1996.

[9] R. Oppliger, Authentication Systems for Secure Networks, Artech House, 1996.

□ 著者紹介



박 춘 식

광운대학교 전자통신과 졸업(학사)
한양대학교 대학원 전자통신과 졸업(석사)
일본 동경공업대학 전기전자공학과 졸업(암호학 전송, 공학박사)
1989년 10월 ~ 1990년 9월 일본공업대학 객원 연구원
1982년 ~ 현재 한국전자통신연구원 책임연구원
1997년 한국정보보호학회 편집이사, 종신회원

※ 주관심분야 : 암호이론, 정보이론, 통신이론