通信情報保護學會論文誌
第 7 卷 第 3 號, 1997. 9

분산 환경에서 양방향 인증 방식

박 춘 식*, 서 창 호*, 박 상 준*

Interactive Authentications in Distributed Systems

Choonsik Park, Changho Seo and Sangjoon Park

요 약

본 고에서는 현재 분산환경에서 인증 방식인 Kerberos와 Yaksha를 소개하고, 공개키 암호를 도입한 Kerberos와 Yaksha를 개선한 효율적이 양방향 인증 방식을 제안하고자 한다. 또한 대표적인 Kerberos,Yaksha 인증 방식과 비교 분석하였다.

Abstract

In this paper, we introduce Kerberos and Yaksha of an authentication scheme and propose an effectively interactive authentication scheme which improved on Kerberos and Yaksha with the public key cryptosystem in distributed systems. Also, we compare and analyse a representative Kerberos and Yaksha authentication scheme with it.

1. 서 론

개방형 시스템인 네트워크를 통한 서버와 클라이언트 사이에서의 서비스 교환은 클라이언트가 자신의 신원을 서버에게 확인 시켜주는 과정을 필요로 한다. 암호학에서 인증(Authentication)은 사용자의 신원을 확인시켜 주는 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 있어

* 한국전자통신연구원

서 사용자가 정당한 사용자인가를 확인하기 위한 방안으로써, 현재 가장 널리 사용되고 있는 방식은 패스워드를 주고 받는 방식이다. 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)[1], 영지식 상호증명(ZKIP:Zero Knowledge Interactive Proof)[1] 등이 알려져 있다. 특히, 분산 환경에서 현재 널리 알려져 있는 대표적인 인증 방식은 역시 Kerberos[4]이다. Kerberos는 그리스 신화에 나오는 지옥의 문을 지키는 3개의 머리를 가진 개를 의미하며 패스워드 대신에 티켓이라는 개념을 이용하여 패스워드 방식의 결점을 보완한 일종의 시도 응답 방식의 인증 방식이다. 또 Yaksha[4] 인증 방식은 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 가능하다.

본 논문에서는 공개키 방식을 도입한 Yaksha와 Kerberos를 개선한 효율적인 인증 방식을 제안하고자 한다. 이 방식은 Kerberos가 사용되는 곳에도 적용 가능하며 다양한 기능을 제공하며, 다른 응용 분야에 많이 활용할 수 있다. 또한 분산 환경에서의 대표적인 Kerberos, Yaksha 인증 방식과 비교 분석하였다. 제2절에서는 Kerberos의 개선된 V.5를 다루었으며, 3절에서는 RSA 공개키 암고리즘을 이용한 Yaksha 인증방식, 그리고 4절에서는 양방향 인증 방식을 제안한다. 그리고 5절에서는 안전성 검토 및 비교 분석하고, 마지막으로 제6절에서 결론을 언급한다.

2. Kerveros 인증 방식

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 Kerberos V.4에서는 비밀키 암호 알고리즘인 DES만을 지원하는데에 비해 Kerberos V.5에서는 DES이외에도 RSA 공개키 암호 알고리즘을 사용하고 있으며 많은 시스템에서 Kerberos V.4 를 사용하고 있으나, 현재, 인터넷 Draft 표준화(RFC 1510) 상태인 Kerberos V.5의 사용이 점차 증가할 전망이다. 그림 1과 같이 Kerberos 인증 메카니즘은 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 매 서비스 마다 인증이 이루어지도록 제공하고 있다.

Kerberos의 동작은 KAS라는 인증 서버, TGS라는 티켓 승인 서버, 그리고 클라이언트와 서버간의 티켓 발행 단계, 서비스 승인 단계를 각각 거쳐서 이루어지는 것이 기본적인 것으로, Kerberos의 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_1를 KAS에 보내어 서비스를 요청한다. 여기서, ID_c는 사용자의 신분을 알리기 위한 클라이언트 C의 ID이고, ID_{tgs}는 액세스를 원하는 TGS의 ID이며, TS_1는 time-stamp 정보로 현재 서비스를 요구하는 시각을 나타낸다. TS_1

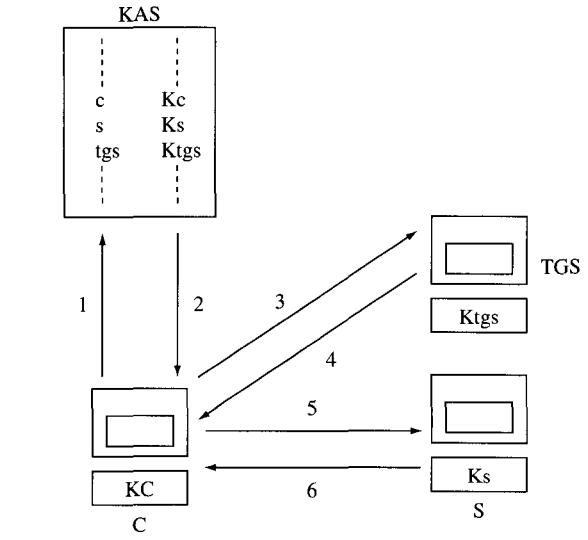


그림 1 : Kerberos 인증 방식

를 이용하여 KAS는 AR(Authenticative Request) 정보가 시기 적절한 가를 확인한다. TS_1 정보는 재전송 공격(replay attack)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 각각 생성하여 사용자의 패스워드를 일방향 해쉬 함수에 적용하여 구한 사용자 키 K_c로 암호화하여 클라이언트 C에게 전달한다. TGT는 암호화되지 않은채 전송되고 있다. $K_{c,tgs}$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 TGT가 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_c는 TGT를 사용할 클라이언트의 망 주소이다.

[단계3](메시지3)

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

클라이언트 C는 KAS로부터 수신하여 구한 $K_{c,tgs}$를 이용하여 생성한 $Authenticator$와 TGT를 TGS에게 전송한다. 여기서, 인증자, $Authenticator=E_{K_{c,tgs}}[ID_c, AD_c, TS_3]$이며, TGT의 유효성을 검증하기 위한 정보로 클라이언트 C와 TGT간의 세션키 $K_{c,tgs}$로 ID_c, AD_c, TS_3를 암호화한 암호문이다. TS_3는 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 클라이언트 C에게 보낸다. (메시지 4)에 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 클라이언트 용의 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 인증 받았음을 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)

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

클라이언트가 선택한 응용 세션을 위한 암호키인 $subkey$, 세션 동안 서버가 클라이언트에게 보낸 메시지의 순번인 Sequence(Replay 공격 방지용)가 내부적으로 사용된다. 서버 S는 SGT에서 복구된 ID_c, AD_c가 $K_{c,s}$를 이용하여 인증자로부터 복구된 ID_c, AD_c가 일치하는지 검사하여 클라이언트 C의 정당성을 확인한다. 정당한 클라이언트 C이면, 서버 S는 자신이 정당한 서버 S임을 증명하기 위해 (메시지 6)을 클라이언트 C에게 보내며, 클라이언트 C는 이를 복호화한 후, TS_5를 확인하여 서버 S의 정당성을 확인한다.

3. Yaksha

Yaksha 인증 방식은 사용자 인증 기능을 제공하는 Kerberos와의 정합성을 충분히 고려하여 설계하였지만, Kerberos가 제공하지 않는 디지털 서명을 제공하는 것이 커다란 장점이다. 그림 2 와 같이 인증 방식은 다음과 같다.

1) 클라이언트-인증 서버간의 교환

[단계1](메시지 1)

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

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

[단계2](메시지 2)

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

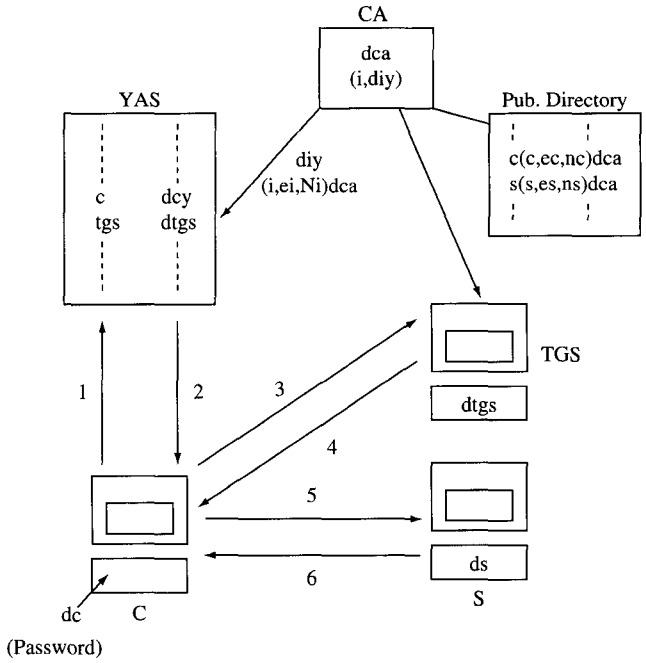


그림 2 : Yaksha 인증 방식

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)

$C \rightarrow TGS : S,T_{exp},R, \ldots, ((T_{C,TGS})^{d_{tgsy}})^{d_{c,temp}}, E_{K_{C,TGS}}(A_C), ((\text{TEMP-CERT})^{d_c})^{d_{cy}}$

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)

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

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)

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

[단계6](메시지 6)

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

단계 5는 단계 3과 유사하며 단계 6은 상호 인증을 위한 것으로 서버의 공개키를 이용해서 누구던지 검사할 수 있는 형식으로 되어있다.

4. 양방향 인증 방식

4.1 제안 동기

Kerberos 인증 방식은 첫째, 인증 서버가 각 클라이언트의 비밀키를 보관하고 있어야 하며, 티켓 발행에 의한 사용자 인증을 실현하고 있지만 디지털 서명 기능은 제공하고 있지 않다. 둘째, 하나의 패스워드가 도난당하면 이를 획득한 도청자로부터 새로운 티켓을 해독하는데 사용되고, 재전송 공격을 위하여 타임스템프에 의존하는데 이것은 동기화되어야 하고 안전한 클럭이 요구된다. 세째, 다수의 세션키를 사용하고 있으며, 패스워드에 대한 dictionary 공격에 약하다.

Yaksha 인증 방식은 첫째, 공개키 증명서를 발행할 인증 기관이 분산망내에 반드시 추가해야하며, RSA 공개키 알고리즘만이 사용되는 점, 클라이언트와 서버사이의 서비스를 재 개시할 경우 모든 단계를 수행하여야 한다. 둘째, 특정 모델에는 사용할수 없다. 이런한 문제점들을 개선하고 보다 효율적인 양방향 인증 방식 제안한다.

4.2 제안된 양방향 인증 방식

본 절에서는 공개키 알고리즘과 비밀키 알고리즘을 이용한 효율적인 양방향 인증 방식을 제안하고 이를 구체적인 인증 프로토콜을 제시한다. 한 영역내의 사용자 인증을 위한 인증 구조는 그림 3과 같이 양방향 인증 방식은 AS라는 인증 서버, 그리고 클라이언트와 서버간의 서비스 승인 단계를 각각 거쳐서 이루어지며, 단계별 동작 설명은 다음과 같다.

[단계1](메시지 1)

$C \rightarrow AS : ID_c, AD_c, ID_s, R$

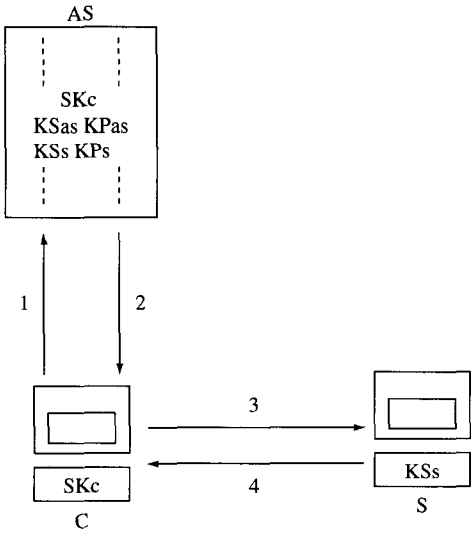


그림 3 : 양방향 인증 방식

클라이언트 C는 그림 3에서와 같이 ID_c, AD_c, ID_s, R를 AS에 보내어 서비스를 요청한다. 여기서, ID_c는 사용자의 신분을 알리기 위한 클라이언트 C의 ID이고, ID_s는 액세스를 원하는 서버 S의 ID이며, R는 dictionary 공격을 방지하기 위한 랜덤 난수이다.

[단계2](메시지2)

$$AS \rightarrow C : E_{KS_{as}}[ID_s, Texp, R, E_{SK_c}(ssk)], E_{SK_c}(Ticket,R)$$

AS는 클라이언트와 서버 S간의 세션키 ssk와 $Ticket$를 각각 생성하여 AS와 클라이언트 사이의 비밀키 SK_c로 암호화하여 클라이언트 C에게 전달한다. ssk는 먼저 SK_c로 암호화한 결과를 다시 AS의 개인키(Private key) KS_{as}로서 ID_s, $Texp$, R, $E_{SK_c}(ssk)$을 암호화한다. 클라이언트와 서버 S간의 암호 통신을 위한 세션키 ssk는 SK_c로 암호화된 메시지 내부에 있기 때문에 클라이언트 C만이 알 수 있을 뿐만 아니라 암호화된 메세지로부터 세션키 ssk을 전혀 알수 없다. 여기서, $Ticket=E_{KP_s}[ID_c, AD_c, ID_s, ssk, Texp]$이며, KP_s는 서버 S의 공개키(Public key)이며, Texp는 Ticket의 유효시간을 의미한다.

[단계3](메시지3)

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

클라이언트 C는 AS로부터 수신하여 구한 ssk를 이용하여 생성한 Authenticator와 Ticket를 서버 S에게 전송한다. 여기서, 인증자 $Authenticator=E_{ssk}[ID_c, AD_c, TS]$이며, TS는 인증자가 생성된 시간을 나타내는 정보로서, 재생 공격(replay attack)을 방지하기 위하여 매우 짧은 유효시간을 갖는다.

[단계4](메시지4)

$$S \rightarrow C : E_{ssk}[TS]$$

서버 S는 Ticket를 자신의 개인키(Private key)인 KS_s로 Ticket를 복호화한 다음 Ticket에서 복구된 ID_c, AD_c, ID_s, ssk, $Texp$와 복호화된 세션 키 ssk를 이용하여 인증자로부터 복구된 ID_c, AD_c가 서로 일치하는지 확인하므로써, 클라이언트의 정당성을 확인하게 된다. 정당한 클라이언트 C이면, 서버 S는 자신이 정당한 서버 S임을 증명하기 위해 (메시지 4)을 클라이언트 C에게 보내며, 클라이언트 C는 이를 복호화한 후, TS를 확인하여 서버 S의 정당성을 확인한다.

5. 안전성 검토 및 비교

기존의 인증 프로토콜은 인증서버에서 모든 사용자와 서버의 비밀키를 보유하고 있기 때문에 인증 서버에 대한 비밀키를 알게 되면 모든 시스템에 악영향을 준다. 그러나 제안된 양방향 인증 방식은 인증 서버와 각 서버가 비밀키를 분산하여 보관하게 되므로, 대재난(Catastrophic failure)에 대한 취약한점을 제거할 수 있으며, 인증 서버가 비밀키를 보유하는 데이타 베이스가 불필요하다. 그리고 제안된 양방향 인증 방식의 단계1 에서 난수 발생기를 사용하므로 dictionary 공격에 안전하며, 클라이언트와 서버 간의 필요한 세션키는 단계2에서 암호화된 메세지로부터 세션키를 전혀 알수 없으며, 단계3에서 타임 스템프를 사용하게 되므로 재전송 공격(replay attack)에도 안전하다. 기존의 인증 프로토콜인 Kerberos, Yaksha 등에서는 프로토콜 교환을 6단계에 의해서 수행하는데 비해 제안된 양방향 인증 방식에서는 4단계에 의해서 수행되므로 프로토콜 변화를 최소화 하였으며 가장 중요한 정보인 티켓는 변경되지 않으면서 소규모인 특정 모델에도 사용이 가능하다. 제안된 양방향 인증 방식에 기존의 인증 프로토콜 공격을 적용 하였을 경우에도 안전할 뿐만아니라 작은 사이즈의 개인 키를 사용하므로 스마트 카드 등에 적합하게 사용할 수 있다. 그리고,

Kerberos, Yaksha와 제안된 양방향 인증 방식의 비교는 이미 설명한 바와 같이, 비밀키의 사용여부, 공개키 암호의 사용여부 또는 디지털 서명의 제공 여부 등으로 나누어 볼 수 있다. 기존의 Kerberos는 대칭 키 알고리즘을 사용하고 디지탈 서명은 제공하지 않으며 Yaksha는 공개 키 알고리즘인 RSA을 사용하면서 디지탈 서명을 제공하는데 비해 제안된 양방향 인증 방식은 사용자 인증 기능을 제공하는 Kerberos와의 정합성을 충분히 고려 하였고, 영역간의 상호 인증을 위하여 공개키 알고리즘 RSA 이외의 어떠한 알고리즘도 가능하게 사용하여 키를 공유하며 디지탈 서명을 제공한다. 또한 공개키 관리를 위한 또 다른 인증 기관이 필요하지 않는 특징을 가지고 있으며, 서비스 마다 사용되는 세션키 공유를 용이하게 할 수 있는 장점이 있다. 본 장에서는 이러한 내용을 알기 쉽게 요약하여 표 1에 나타내었다.

5. 결 론

본고에서는 현재 분산환경에서 인증 방식으로 많이 검토되고 있는 Kerberos와 Yaksha의 인증 시스템의 동작 원리를 분석하고 이를 바탕으로 기존의 분산망에서의 문제점들을 제거하고 Kerberos와의 정합성을 고려하여 공개키/비밀키 암호를 적용한 효율적인 양방향 인증 프로토콜에 대하여 제안하였다. 제안된 양방향 인증 방식은 키 정보를 저장하는 또 다른 데이타 베이스가 불필요하며, 분산 컴퓨터망에서 효율적으로 사용할 수 있게 구성하였다.

표 1 : 인증 방식의 비교

구분	Kerberos	Yaksha	양방향 인증 방식
제안자	MIT Athena 프로젝트	Bell Atlantic R. Ganesan	ETRI SPP
제안년도	1989(V.5)년	1995년	1997년
기본 방식	Needham-Schroeder[2] Denning-Sacco[3]	Boyd[7] Kerberos	Kerberos
인증 모델	AS, TGS, C, S로 구성 티켓 발행, 서비스 발행	Kerberos와 동일	AS, C, S로 구성 티켓 발행
인증 방식	티켓 개념의 시도 응답방식	Kerberos와 동일	Kerberos와 동일
사용 알고리즘	비밀키	RSA+비밀키	공개키+비밀키
디지털 서명	미제공	제공	제공
상호인증	가능	가능	가능
세션키 공유	조건적 가능	용이	용이
안전성	dictionary 공격에 약	dictionary 공격에 약	dictionary 공격에 강
인증서버 의존도	모든 비밀키 보관	비밀키 분리 보관	비밀키 분리 보관
표준	인터넷 RFC1510		
확장성		Security Infrastructure	좌동
기타	비밀키 관리 필요	공개키 관리 및 인증 기관 필요	공개키 관리를 위한 인증 서버 필요

참 고 문 헌

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

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

[7] 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.

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

□ 著者紹介

박 춘 식

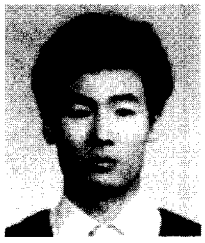
광운대학교 전자통신과 졸업(학사)
한양대학교 대학원 전자통신과 졸업(석사)
일본 동경공업대학 전기전자공학과 졸업(암호학 전공, 공학박사)
1989년 10월 ~ 1990년 9월 일본 동경공업대학 객원 연구원
1982년 ~ 현재 한국전자통신연구원 책임연구원
1997년 한국통신정보보호학회 편집이사, 종신회원

※ 주관심 분야 : 암호이론, 정보이론, 통신이론

서 창 호

1990년 2월 고려대학교 수학과 학사
1992년 8월 고려대학교 대학원 수학과 석사
1996년 8월 고려대학교 대학원 수학과 박사
1996년 ~ 현재 한국전자통신연구원 선임연구원

※ 주관심 분야 : 응용대수학 및 정수론, 암호론



박 상 준

1984년 2월 한양대학교 자연과학대학 수학과(이학사)
1986년 2월 한양대학교 대학원 수학과 (이학석사)
1986년 1월 ~ 현재 한국전자통신연구원 선임연구원
1995년 3월 ~ 현재 성균관대학교 대학원 정보공학과 박사과정