제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

Kerberos 및 Kerberos 유사 인증 기법의 비교분석

황성욱*, 조경산**
* **단국대학교 정보컴퓨터학부
e-mail: * jwqueen@dku.edu
** kscho@dankook.ac.kr

Comparison and Analysis of Kerberos and Kerberos-like Authentication Mechanisms

Sung-Wook Hwang*, Kyungsan Cho**
* **Div. of Information & Computer Science,
Dankook University

요 약

본 논문에서는 Kerberos 인증 메커니즘과 Kerberos에 공개키기반의 환경을 결합시킨 PKINIT, PKTAPP, PKCROSS을 비교분석하였다. 각 인증 메커니즘에서 키 교환을 위한 비밀키 및 공개키의 암·복호화, 서명 및 서명검증 등의 횟수를 분석하고 이를 기반으로 실제 인증에 소요되는 연산시간을 비교하였다. 분석 방법 및 결과를 다양한 환경에 적용한 적절한 인증 메커니즘을 선택하는데 활용할 수 있으며, 특히 모바일과 같은 연산능력이 제한되는 환경에 적합한 인증 기법 개발에 적용시키고자 한다.

1. 서론

클라이언트-서버 분산환경 하에서 자원의 보호는 기밀성, 인증, 안전한 키교환을 위한 무결성 등을 필요로 한다. 클라이언트-서버의 인증 메커니즘으로 가장 대표적인 서비스인 Kerberos[1] 인증 프로토콜이 있다. Kerberos는 중앙집중식 인증을 제공한다.

Kerberos는 공개키 암호법을 사용하지 않고 비밀키암호법만을 사용한 관용 암호 방식에 의존하여 키 관리로 인한 확장성의 제약점을 갖는다. PKINIT[2]은 공개키 기반의 환경에 Kerberos의 제약점을 해결하기 위해 연계하였다. PKINIT은 사용자로 하여금 인증서를 사용하여 KDC에 초기인증을 할 수 있도록 하였다. PKTAPP[3]은 KDC의 필요성을 제거하면서 Kerberos 티켓의 장점을 그대로 유지한 PKDA의 아이디어를 PKINIT를 이용하여 기존 Kerberos의 최소 수정으로 구현하는 것에 목적을 두었다.

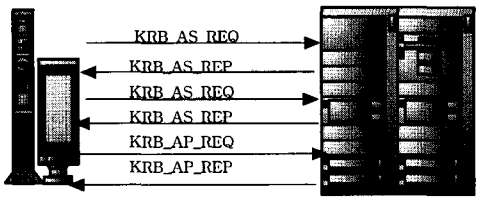
PKTAPP는 LTGS(Local Ticket Granting Server)이라는 TGS의 기능을 하는 로컬 프로세스가 클라이언트에게 PKINIT를 이용하여 인증을 받도록 서비스티켓을 발행하여 준다. PKCROSS[4]는 공개키를 사용하여 Cross-Realm 인증이 가능하도록 DNS를 사용하면서 Kerberos를 확장한다. 여기서 지원하는 방식은 신뢰된 공개키로 암호화한 Cross-Realm 비밀키를 전송한다. 본 논문의 2장에서는 개방형 분산시스템 환경하의 인증에 관한 문제점들의 해결방안 중 Kerberos와 PKINIT, PKTAPP, PKCROSS 인증과정의 프로토콜에 대해 설명한다. 3장에서는 2장에서 분석한 각 프로토콜에 대한 암·복호화, 서명 연산을 비교하고 4장에서 결론을 맺는다.

2. Kerberos, PKINIT, PKTAPP, PKCROSS 인증 프로토콜

2.1 Kerberos

Kerberos 프로토콜은 사용자의 패스워드를 중앙집중식 데이터베이스에 저장하는 AS(Authetication Server) 와 사용자의 패스워드 입력횟수를 최소화하고 티켓의 재사용 문제, 패스워드 가로채기를 방지

하고 위한 TGS(Ticket Granting Server)를 두었다. AS와 TGS는 각각 티켓을 생성하여 TGS와 AP(Application Server)에게 전송한다. 이들 티켓에는 클라이언트, 서버의 식별자, 티켓의 발행시간을 나타낸 타임스탬프, 유효시간을 나타낸 LifeTime, 인증자를 복호화 하는데 사용되는 세션키 등을 포함한다. <그림1>은 Kerberos의 동작과정을 보여주고 있다[1].

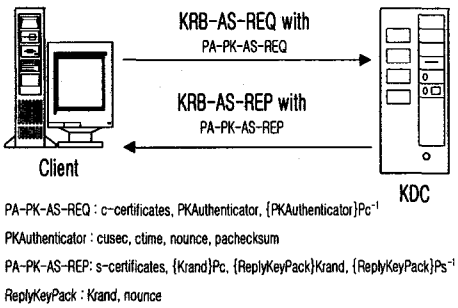


<그림1> Kerberos를 이용한 초기인증

Kerberos는 만일 AS서버가 해킹을 당할 경우, 대칭키가 공격자에게 누설될 수 있기 때문에 저장된 모든 키를 새로운 대칭키로 교체되어야 한다. 이는 대칭키를 사용하는 Kerberos의 커다란 부담이 되므로 이의 해결방안으로 PKINIT이 제시되었다.

2.2 PKINIT

PKINIT의 주된 목적은 공개키 기반의 환경과 Kerberos를 연계하는 것이다. 즉 PKINIT은 초기 인증에서 클라이언트로 하여금 공개키 인증서를 사용하여 KDC에 인증하고 티켓을 받을 수 있도록 한다. 이를 위해 PKINIT은 <그림2>와 같이 AS-REQ와 AS-REP는 동일하게 유지하면서 새로운 preauthentication data type을 사용하여 PK exchange를 수행한다[2].

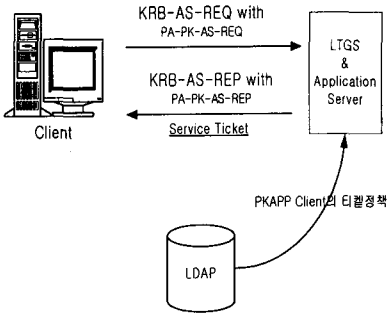


<그림2> PKINIT를 이용한 초기인증

2.3 PKTAPP

PKDA는 중앙 집중된 KDC의 필요성을 제거하면서 Kerberos 티켓의 장점을 유지하였다.

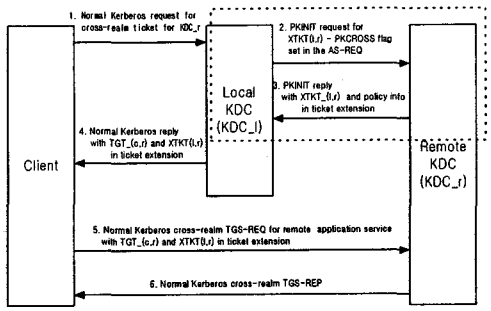
PKTAPP는 PKINIT의 개념을 KDC의 개입없이 직접 응용서버에 인증을 한다. <그림3>은 응용서버에 내장된 LTGS를 포함한 그림을 표현한 것이다[3].



<그림3> 서버에 내장된 LTGS

2.4 PKCROSS

<그림4>에 보인 PKCROSS는 cross-realm 키 관리의 부담을 최소화하기 위하여 공유키는 잠시 설정되어지며 remote realm은 클라이언트가 cross-realm 인증을 요구할 때 다시 되돌아오는 정책 정보를 발행하며 클라이언트에게는 투명하여 KDC만 수정된다[4].



<그림4> PKCROSS의 cross-realm 인증과정

3. 각 프로토콜의 비교분석

3.1 인증을 위한 각 프로토콜 키의 수와 연산 수

<표1>은 2장에서 설명된 각 프로토콜들의 실행시에 소용되는 암·복호화 연산을 나타낸 것이다. 클라이언트와 각 서버와의 사이에서 인증 및 서비스 티켓을 얻기 위해 필요로 하는 연산의 수를 나타낸

것이다. 단 인증서 검증과정에 따라 연산의 수는 늘어날 수 있다. [5]에서는 PKTAPP 및 PKCROSS를 분석하였으나 암·복호화 연산의 수는 <표1>과 약간 다른 값을 보인다.

인증 트랜잭션	공개키암호 /서명검증	공개키복호 /서명	비밀키 암·복호화	연산의 횟수
Kerberos				
C			5	5
AS			2	2
TGS			4	4
AP			3	3
TOTAL			14	**14**
PKINIT				
C	1/1	2/1	5	10
AS	2/2	0/1	2	7
TGS			4	4
AP			3	3
TOTAL	6	4	14	**24**
PKTAPP				
C	1/1	2/1	3	8
AP	2/1	1/1	4	9
TOTAL	5	5	7	**17**
PKCROSS				
C			7	7
Local KDC	1/1	2/1	5	10
Remote KDC	2/2	0/1	4	9
AP			3	3
TOTAL	6	4	19	**29**

<표1> 인증을 위해 필요한 암·복호화 연산의 수

3.2 암·복호화 시간분석

본 논문에서는 [5][7]에서 제시된 연산시간을 <표2>에서 활용하였다. <표2>는 <표1>의 각 항목의 연산 시간을 비교하기 위해 키의 길이와 암·복호화, 서명 및 검증 시간을 제시하였다. 다만 서명검증 연산시간은 명확하게 제시된 것이 없으므로 차이가 있을 수 있다.

환경	암호화 알고리즘	키의 길이	연산시간(msec)
Celeron 850Mhz Windows 2000	DES	56	0.0006
	RSA 암호화/서명검증	1024	0.320/0.254
	RSA 복호화/서명	1024	10.23/10.23

<표2> 암·복호화 연산시간

2장에서 제시한 각 프로토콜들의 분석에서 클라이언트와 각 서버들이 개인키나 공개키를 통한 실제적인 암·복호화 시간에 <표2>의 결과를 적용하였다. 예를 들어, Kerberos 프로토콜의 클라이언트의 경우 비밀키 연산만을 하므로 암·복호화 하는데 필요한 시간은 0.003msec일 것이다.

3.3 클라이언트/ 서버들의 전체 연산시간

표<1><2>를 토대로 각 인증 메커니즘의 클라이언트/서버들의 연산시간을 계산하면 <표3>과 같다. 클라이언트와 각 서버는 하나의 응용 서버에서 인증을 위한 비밀키와 공개키, 개인키의 암호화를 위해 표준 DES방식과 1024bit의 RSA 키 방식을 사용했다고 가정한다.

인증 메커니즘	C	AS	TGS	AP	Local	Remote	TOTAL
Kerberos	0.003	0.0012	0.0024	0.0018			0.0084
PKINIT	31.267	11.379	0.0024	0.0018			42.65
PKTAPP	31.266			21.356			52.622
PKCROSS	0.0042			0.0018	31.267	11.389	42.662

<표3> 클라이언트/서버들의 연산시간

<표3>의 경우 클라이언트와 서버가 1번씩 인증 프로토콜을 주고받았다는 가정 하에서 계산되었다. 위의 결과에서 보여주듯 클라이언트가 접속해야 하는 응용서버가 100개 정도를 필요로 한다면 클라이언트는 처음 인증을 받은 이후, 유효시간 이내에는 인증을 다시 받지 않아도 될 것이다. 따라서 PKINIT, PKTAPP, PKCROSS 기반 인증 메커니즘들은 서버가 많이 있다해도 연산시간이 크게 증가하지는 않는다.

4. 결론

본 논문에서는 Kerberos, PKINIT, PKTAPP, PKCROSS의 메커니즘에서 키 교환을 위한 비밀키와 공개키의 암·복호화, 서명·서명검증 등의 횟수를 분석하고 연산시간을 비교하였다. PKINIT, PKTAPP, PKCROSS 인증 메커니즘은 클라이언트가 초기인증에 필요한 공개키 복호, 서명을 하기 위한 시간이 서버의 인증 처리시간 보다 훨씬 많음을 알 수 있다. 이러한 문제점을 해결하기 위해 본 연구에서 분석한 초기 인증 과정에서 클라이언트의 전자서명용 알고리즘으로 엘립틱 커브를 사용할 수 있다. RSA 알고리즘이 보안적으로 안전하기 위해서 필요한 키의 길이가 1024bit인데 반해 엘립틱 커브는 약 160bit로 RSA 1024bit의 보안강도를 가지고 있는 엘립틱 커브(Elliptic Curve)[6]를 이용한 암호 알고리즘인 ECDSA를 적용시킬 수 있다. [8]에서는 MPKINIT라는 모바일 환경에서의 인증메커니즘을 제안하였는데 이는 클라이언트가 생성하는 세션키의 강도가 높아야 하므로 활용이 제한적이다. 추후 연구과제로는 클라이언트의 연산시간을 줄이기 위해 RSA 대신에 ECDSA 알고리즘의 사용과 인증기법의 개선을 시도하고, 적용시키는 작업에 중점을 둘 것이다.

참고문헌

[1] The Kerberos Network Authentication Service (V5)
http://www.ietf.org/rfc/rfc1510.txt?number=1510

[2] Draft-ietf-cat-kerberos-pk-init-16.txt
http://www1.ietf.org/mail-archive/ietf-anno-unce/Current/msg20286.html

[3] Draft-ietf-cat-kerberos-pk-tapp-04.txt
http://www.ietf.org/proceedings/01aug/I-D/draft-ietf-cat-kerberos-pk-tapp-04.txtRoger

[4] Draft-ietf-cat-kerberos-pk-cross-08.txt
http://www.ietf.org/mail-archive/ietf-anno-unce/Current/msg15164.html

[5] Alan harbitter and Daniel A. Menasce, "A methodology for analyzing the performance of authentication protocols"
ACM Transactions on Information and System Security, vol.5, no.4, p.458~491, 2000

[6] Draft-ietf-dnsext-ecc-key-03.txt
http://www.ietf.org/internet-drafts/draft-ietf-dnsext-ecc-key-03.txt

[7] Eric Rescorla, "Praise for SSL and TLS: Designing and Building Secure Systems"
p.27~38, 2001

[8] Alan harbitter and Daniel A. Menasce, "A methodology for analyzing the performance of authentication protocols"
ACM Special Interest Group on Security, Audit, and Control p75~85, 2001