□특집□

안전한 Kerberos 인증 메커니즘 설계

조 성 아[†] 한 태 창[††] 함 호 상[†††] 이 동 훈[††††]

◈ 목 차 ◈

1 서 론
2 공개키 알고리즘을 이용한 Kerberos 설계
3. 시뮬레이션
4 결 론

1. 서 론

개방형 시스템인 네트워크를 통한 서버와 클라이언트 사이에서의 서비스 교환은 클라이언트가 자신의 신원을 서버에게 확인시켜주고 동시에 서버의 신원을 클라이언트에게 확인시켜주는 과정과 메시지의 출처를 확인해 주는 과정을 필요로 한다. 이와 같은 과정을 인증 (Authentication)이라 한다.

Kerberos는 개방형 분산 통신망에서 클라이언트와 서버간의 상호 인증을 지원하는 대표적인 인증 시스템으로 MIT에서 개발되었다.

Kerberos는 기본적으로 비밀키 암호 알고리즘인 DES를 기반으로 하는 상호 인증 시스템으로 지금까지는 Kerberos 버전 4가 많이 사용되고 있다.

가장 최근에 개발된 Kerberos 버전 5는 보다 넓은 환경에서의 지원이 가능하다는 특징을 가지고 있어서 좀더 넓은 분야의 적용이 증가할 것으로 예상된다.

Kerberos의 문제점은 안전성 문제가 대두되고 있는 비밀키 암호 알고리즘인 DES를 기반으로 하고 있다는 점이다. 또, 비밀키 암호 알고리즘을 사용함으로 인하여 수행되어져야 하는 비밀키의 생성과 관리를 Kerberos가 담당해야하고 사용자가 Kerberos에 대한 높은 신뢰도를 갖고 있어야 한다는 문제점을 갖고 있다.

본 논문에서는 Kerberos 인증 메커니즘이 갖는 문제점을 분석하고 이를 바탕으로 Kerberos 버전 5에서 대표적인 공개키 암호 알고리즘인 RSA를 이용한 인증 방식을 제안하고자 한다.

본 논문의 2장 1절에서는 Kerberos의 특징과 Kerberos 버전 5 알고리즘과 그 문제점에 대해 알아보고 2절에서는 안전성의 문제가 제기되고 있는 DES 대신 공개키 암호 알고리즘인 RSA를 이용한 Kerberos 버전 5의 알고리즘을 제안한다. 또 3장에서는 제안한 알고리즘에 대한 시뮬레이션을 통하여 올바르게 작동되는지 알아보았다.

2. 공개키 알고리즘을 이용한 Kerberos 설계

† 준회원 : 고려대 대학원 전산학과 석사과정
†† 준회원 . 고려대 대학원 전산학과 석사과정
††† 정회원 : 시스템공학연구소 전자거래연구 실장
†††† 정회원 : 고려대 자연과학대학 전산학과 교수

2.1 Kerberos 버전 5

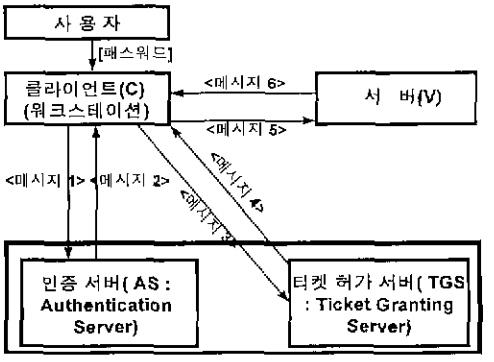
Kerberos는 MIT에서 Athena 프로젝트의 부분으로 개발된 인증시스템으로 Needham, Schroeder의 공개키를 이용한 인증 알고리즘을 바탕으로 하고 있으며 사용자들이 네트워크를 통해 서버의 서비스를 받기 위한 분산 환경을 바탕으로 한다. Kerberos 버전 1부터 3까지는 별로 실용화되지 못했으며 Kerberos 버전 4가 현재까지의 환경에서 많이 사용되었으나 분산형 환경에서의 인증 메커니즘으로는 부족하여 최근에 이에 적합한 버전 5가 개발되었고 인터넷 draft 표준(RFC 1510)으로 발표되었다.

Kerberos 인증 메커니즘은 인증 서버(AS : Authentication Server)와 티켓 허가 서버 (TGS : Ticket Granting Server)로 나누어져 있어 사용자가 여러 번 반복하여 패스워드를 입력해야 하는 불편을 없앴으며 또한 클라이언트에 대한 인증뿐 아니라 인증 서버와 티켓 허가 서버에 대한 인증도 동시에 수행되는 상호 인증(Mutual Authentication)을 제공한다.

Kerberos 버전 4는 가장 많이 쓰여졌는데도 불구하고 많은 단점이 존재한다. Kerberos 버전 4의 경우 통신 프로토콜에 의존적인 특징을 지니고 있다. 즉, TCP/IP 환경을 기반으로 하여 데이터의 크기나 주소 표현 방식 등에 있어서 많은 제한을 갖고 있다는 것이 큰 단점이었다. 이보다 향상된 버전 5의 경우 많은 가능적인 확장을 통해 버전 4가 갖고 있었던 많은 제약을 제거함으로써 암호 메커니즘과 통신 프로토콜에 대해 독립성을 갖게 하여 좀더 폭넓은 환경에서의 사용을 가능하게 하였다.

Kerberos 버전 5의 수행과정에 따른 기본적 구조는 다음과 같다.

Kerberos에서는 사용자에 대한 인증을 워크스테이션이 맡아서 하게 된다. 즉, 기본 알고리즘에서의 클라이언트는 사용자를 대신하는 워크스테이션이라고 말할 수 있다.



[Kerberos 시스템의 구조]

Kerberos의 인증과정에서 메시지 1,2의 과정 은 사용자가 워크스테이션에 로그인하는 단계에서 수행되는 과정이고 메시지 3,4 과정은 클라이언트가 어떤 종류의 서비스를 받고자 할 때 수행되며 메시지 5,6은 클라이언트가 임의의 서버 서비스를 받고자 할 때 세션 단위로 수행되게 된다. Kerberos 인증 서비스는 사용자에게 한 번의 패스워드 입력만을 요구하게 되며 따라서 패스워드에 관한 공격을 막을 수 있게 된다.

Kerberos 버전 5의 기본 서비스 과정은 다음과 같이 클라이언트-인증 서버간의 교환, 클라이언트-티켓 허가 서버간의 교환, 그리고 클라이언트와 서버간의 교환의 세 부분으로 나누어 생각할 수 있다.

1) 클라이언트-인증 서버간의 교환

< 메시지 1 >

$C \rightarrow AS$: $Options \| ID_C \| Realm_C \| ID_{TGS} \| Times_1 \| Nonce_1$

< 메시지 2 >

$AS \rightarrow C$: $Realm_C \| ID_C \| Ticket_{TGS} \| E_{K_C}[K_{C\ TGS} \| Times_1 \| Nonce_1 \| Realm_{TGS} \| ID_{TGS}]$

$$*Ticket_{TGS} = E_{K_{TGS}}[Flags||K_{C,TGS}||Realm_C || ID_C||AD_C||Times_1]$$

메시지 1에서는 서비스를 받고자 하는 클라이언트가 인증 서버에게 요청을 하게 된다.

Kerberos에서 클라이언트-서버의 마스터키는 사용자의 패스워드를 이용한다. 사용자가 자신의 워크스테이션에 패스워드를 입력하는 시간이나 워크스테이션이 사용자의 패스워드를 보유하고 있는 시간은 침입자의 공격에 대해 노출되어있다. Kerberos 버전4에서는 워크스테이션이 인증 서버에게 메시지 2를 받은 후에 사용자의 패스워드를 입력받게 되고 버전5에서는 인증 서버가 메시지 2를 워크스테이션에게 보내기 전에 워크스테이션이 사용자의 패스워드를 입력받았는지를 확인한다.

메시지 1에는 자신의 ID와 자신이 속해있는 영역(realm), 서비스를 받고자 하는 티켓 허가 서버를 명시해야 한다. 또, 티켓이 유효한 시간을 나타내는 Times, 오류를 검사하기 위해, 혹은 보낸 메시지가 여러개일 때 이를 구별해주기 위한 Nonce를 포함하여 보내게 된다.

Times와 Nonce는 Kerberos 버전5에서 새롭게 제안된 구조로 버전4에서의 Timestamp의 역할을 나누어 하게 된다. 메시지 1의 요청을 받은 인증 서버는 클라이언트의 신원을 확인하여 티켓 허가 서버와의 통신을 가능하게 하는 인증서와 같은 역할을 하는 티켓을 클라이언트에게 전달한다(메시지 2). 이 단계에서 클라이언트가 전달받는 티켓은 티켓 허가 서버의 마스터키로 암호화되어 있기 때문에 클라이언트는 그 내부를 볼 수 없다. 그리고 클라이언트가 티켓 허가 서버와의 교환에 쓸 세션 키는 클라이언트의 마스터키로 암호화하여 전달된다. 또 티켓과 동시에 클라이언트의 영역, ID를 다시 전달하여 인증 서버에게 전달되었던 내용의 변경 여부를 확인할 수 있게 된다.

2) 클라이언트-티켓 허가 서버간의 교환

< 메시지 3 >

$$C \rightarrow TGS : Options || ID_V || Times_2 || Nonce_2 || Ticket_{TGS} || Authenticator1_C$$

< 메시지 4 >

$$TGS \rightarrow C :$$

$$Realm_C||ID_C||Ticket_V||E_{K_{C,TGS}}[K_{C,V}||Times_2||Nonce_2 || Realm_V || ID_V]$$

$$* Ticket_V = E_{K_V}[Flags||K_{C,V}||Realm_C ||ID_C||AD_C||Times_2]$$

$$* Authenticator1_C = E_{K_{C,TGS}}[ID_C || Realm_C || TS_1]$$

메시지 3은 인증 서버로부터 받은 티켓을 티켓 허가 서버에게 전달한다. 또한 클라이언트가 서비스를 요청할 서버의 ID, 시간, Nonce와 클라이언트를 인증할 수 있는 인증자를 전달한다.

메시지 4단계에서는 인증을 확인한 티켓 허가 서버가 클라이언트와 서버가 사용할 세션 키를 서버에게 전달될 티켓에, 그리고 클라이언트와 티켓 허가 서버 사이의 세션 키로 암호화된 곳에 포함시켜 전달한다.

티켓 허가 서버에게 전달되는 티켓에 포함되어 있는 $Times_1$ 은 그 티켓에 대한 유효시간을 나타내고 서버에게 전달될 티켓에 포함되어 있는 $Times_2$는 서버 티켓의 유효시간을 나타낸다. 티켓 허가 서버는 두 시간을 비교하여 받은 메시지의 요청이 옳은 것인지를 판단하게 된다.

3) 클라이언트-서버 인증 교환

< 메시지 5 >

$$C \rightarrow V: Options||Ticket_V||Authenticator2_C$$

< 메시지 6 >

$V \rightarrow C : E_{Kc,v}[TS_2 \| Subkey \| Seq\#]$

$*Authenticator2_C = E_{Kc,v}[ID_C \| Realm_C \| TS_2 \| Subkey \| Seq\#]$

메시지 5단계에서는 클라이언트가 티켓 허가 서버로부터 받은 티켓을 서버에게 전달하게 된다. 또한 클라이언트를 인증할 수 있는 인증자도 전달된다. 마지막으로 메시지 6단계에서는 서버와 클라이언트사이의 세션 키로 암호화된 메시지를 전달함으로써 클라이언트가 서버를 인증하게 된다. 이 두 단계에서 보조키(Subkey)는 인증 과정이 끝난 후 클라이언트와 서버사이의 메시지 전달과정에서 암호화에 필요한 키이며 Seq#는 메시지에 대한 일련의 번호로 전달과정에서 오류가 생겼을 때 다시 보내야 하는 부분을 구별하기 위한 것이다.

티켓에 포함되어있는 Flag는 전달과정에 있어서 필요한 정보를 나타내는 역할을 한다. 이 Flag 에 대한 정보를 클라이언트가 인증서버나 티켓 허가 서버에게 알려주는 역할을 하는 것이 Option이다. 버전 5에서는 버전 4와는 달리 클라이언트가 서비스를 받을 서버들에 대한 영역의 정보를 가지고 있을 필요가 없기 때문에 인증 서버나 티켓 허가 서버가 클라이언트에게 주는 메시지에는 그 다음 단계에서 클라이언트가 접해야 하는 서버의 영역이 포함된다.

2.2 공개키를 이용한 Kerberos 버전 5 알고리즘 제안

Kerberos 버전 4에서는 비밀키 암호 알고리즘인 DES만을 지원하는 데에 비해 Kerberos 버전 5에서는 DES이외에도 RSA등과 같은 다른 공개키 암호 알고리즘들을 지원하도록 되어있다. 그러나 공개키 암호 알고리즘을 이용한 구체적인 과정이나 구현은 아직 발표되지 않았다.

Kerberos에서 비밀키 암호 알고리즘을 사용하는 데는 여러 가지 문제점이 있다.

첫째, 비밀키를 이용하는 Kerberos에서 사용하고 있는 알고리즘인 DES가 안전성에 대한 문제를 갖고 있다는 점이다. Kerberos 인증의 전 과정에서 사용하고 있는 DES가 안전성의 문제가 있다는 것은 침입자의 공격이 더 쉬워졌다는 것을 의미하며 따라서 Kerberos 전체의 안전성에 영향을 끼치게 된다.

둘째, 비밀키를 이용할 경우 인증과정에서 Kerberos는 세션 키의 역할을 하는 두 개의 비밀키를 생성하고 분배해야 한다. 즉, Kerberos는 인증과정의 수행 이외에 키분배센터(KDC : Key Distribution Center)의 역할도 해야 한다. 따라서 클라이언트나 서버의 입장에서는 Kerberos 인증과정이 상호인증임에도 불구하고 Kerberos에 대한 매우 높은 신뢰도를 가지고 있어야만 한다는 문제점이 있다. 따라서, Kerberos 버전 5에서 비밀키 암호 알고리즘보다 안정성이 높은 공개키 암호 알고리즘의 사용이 더 바람직하다. 본 논문에서는 공개키 암호 알고리즘중 RSA를 사용한 프로토콜을 제안한다.

Kerberos 버전 5에서 RSA를 사용할 때 6단계의 과정은 다음과 같다.

1) 클라이언트-인증 서버간의 교환

< 메시지 1 >

$C \rightarrow AS : Options \| ID_C \| Realm_C \| ID_{TGS} \| Times_1 \| Nonce_1$

< 메시지 2 >

$AS \rightarrow C : Realm_C \| ID_C \| Ticket_{TGS} \| E_{KP_c}[E_{KS_{as}}[Times_1 \| Nonce_1 \| Realm_{TGS} \| ID_{TGS}]]$

$$*\text{Ticket}_{TGS} = E_{KP_{TGS}}[\text{Flags}\|\text{Realm}_C\|\text{ID}_C\| \text{AD}_C \| \text{Times}_1]$$

메시지 1의 경우 이전의 알고리즘에서의 구조와 같다. 메시지 2의 경우 메시지의 내용중 시간, Nonce, 티켓 허가 서버의 영역과 ID는 인증 서버의 개인키로 암호화한 후 클라이언트의 공개키로 암호화하였다. 인증 서버의 개인키를 이용한 암호화는 서명(Signature)의 역할을 하게 되므로 상호 인증의 기능을 수행하게 된다. 인증 서버는 메시지 2 과정에서 클라이언트에게 이전의 알고리즘에서와 같이 티켓을 전달하는데 티켓 허가 서버의·공개키로 암호화하여 티켓 허가 서버만이 복호화할 수 있게 된다.

공개키 암호 알고리즘을 이용하므로 CA(Certification Authority)를 두어 키의 생성, 관리의 과정을 수행하도록 하면 Kerberos 시스템에서는 키에 대한 위의 두 과정을 수행하지 않아도 된다.

2) 클라이언트-티켓 허가 서버간의 교환

< 메시지 3 >

$$C \rightarrow TGS : \text{Options} \| \text{ID}_V \| \text{Times}_2 \| \text{Nonce}_2 \| \text{Ticket}_{TGS} \| \text{Authenticator1}_C$$

< 메시지 4 >

$$TGS \rightarrow C : \text{Realm}_C \| \text{ID}_C \| \text{Ticket}_V \| E_{KP_C}[E_{KS_{TGS}}[\text{Times}_2 \| \text{Nonce}_2 \| \text{Realm}_V \| \text{ID}_V]]$$

$$*\ \text{Ticket}_V = E_{KP_V}[\text{Flags} \| \text{Realm}_C \| \text{ID}_C \| \text{AD}_C \| \text{Times}_2]$$

$$*\ \text{Authenticator1}_C = E_{KS_C}[\text{ID}_C \| \text{Realm}_C \| \text{TS}_1]$$

클라이언트는 티켓 허가 서버에게 이전의 알고리즘과 같은 구조의 메시지 3을 전달하게 된다. 여기서 티켓은 서버의 공개키로 암호화되어 있으며 인증서는 클라이언트의 비밀키로 암호화 되어 있다. 클라이언트의 비밀키로 암호화되어 있는 인증서는 클라이언트가 보낸 것인지에 대한 여부를 확인할 수가 있으므로 서명의 과정을 거친 것이라고 할 수 있다. 인증서와 티켓을 받아서 클라이언트에 대한 인증이 수행된 후 티켓 허가 서버는 클라이언트에게 메시지 2와 같이 부분적으로는 자신의 개인키로 암호화된 후 다시 클라이언트의 공개키로 암호화되어 있는 메시지를 서버가 받을 티켓과 함께 보내게 된다.

3) 클라이언트-서버간의 교환

< 메시지 5 >

$$C \rightarrow V : \text{Options} \| \text{Ticket}_V \| \text{Authenticator2}_C$$

< 메시지 6 >

$$V \rightarrow C : E_{KP_C}[\text{TS}_2 \| \text{Seq\#} \| \text{Subkey}]$$

$$*\ \text{Authenticator2}_C = E_{KS_C}[\text{ID}_C \| \text{Realm}_C \| E_{KP_V}[\text{TS}_2 \| \text{Seq\#} \| \text{Subkey}]]$$

(단, KP : 공개키, KS : 개인키)

Authenticator2의 TS_2의 경우 TS_1과 다른 역할을 수행한다. TS_1는 인증서 1의 발행시간만을 나타내지만 TS_2의 경우는 인증서 2의 발행시간을 나타냄과 동시에 메시지 6에서 다시 보냄으로서 메시지 6이 서버로부터 온 것임을 확인하게 된다. 그런데 Suq#와 Subkey의 경우 옵션 사항이므로 이 두 정보가 생략된다면 서버 이외의 다른 공격자가 TS_2를 알고 공격을 할 수 있다. 따라서 TS_2는

Suq#, Subkey와 같이 서버의 공개키로 암호화하여 서버만이 볼 수 있도록 하는 것이 안전하다.

SubKey의 경우는 인증 후 클라이언트와 서버사이의 메시지 전달을 위한 비밀키를 나타내는 것으로 메시지 전달과정은 비밀키 암호 알고리즘인 DES를 이용하도록 한다.

3. 시뮬레이션 결과

본 논문에서 제안한 공개키 암호 알고리즘을 이용한 Kerberos 버전 5를 시뮬레이션 하였다. 시뮬레이션은 기본 요소만을 이용하였다.

RSA 생성 알고리즘을 이용하여 각각 클라이언트(C), 인증 서버(AS), 티켓 허가 서버(TGS), 서버(V)의 공개키나 개인키를 생성하였다. 각각의 공개키와 개인키의 형태는 아래와 같다 (단, n : modulus, e : public key, d : private key)

[클라이언트의 공개키와 개인키]

n e078 c8f9 2ac3 d7b1 6aab 6600 15ed 5428
9181 359f 6f8a b36a ac9d 0a75 12df 9e77
f0d3 3c30 f4ad 336a 9a65 2420 898b f795
9dbe 9abd 3177 80ac 140f a490 e60d 0a29

e 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0001 0001

d 9367 3db2 419e f159 1439 1876 1df0 073f
ccac e8a5 95fd a2eb fe05 f204 072c c946
066b f547 6edc f415 f427 05b7 7915 e2dc
b39e 2925 bdaa 331c 67d4 711b f68c 8531

[인증 서버에 대한 공개키와 개인키]

n ae53 c047 bc86 f74f 01af 4598 99ac d318
5d26 028c 62d9 ff68 e0f9 5b6c 0abf 2caa
1c08 443d c3e9 24ce aab7 67ad 81d0 c0cc
647e 5d4b 8e86 0706 8057 05f0 2c32 8edb

e 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0001 0001

d aa9a 69fd 8282 273b 59b1 b711 8dab 75f7
4d09 aaca cade ff37 1b9c cd1c a233 cfb7
98d3 a663 9709 90e1 e7d6 37bf 4eca d4bc
a003 2aaa b993 9379 df1f 1663 57f9 63f9

[티켓 허가 서버에 대한 공개키와 개인키]

n c2cf 0a76 16a9 3475 4102 35e2 f0d6 7641
e35a e448 9163 3e25 a949 f39f c312 780b
dc10 cdb1 98f1 40cd 9f0f 39f6 2d61 8793
2b50 e357 e3a7 5a94 9d7f 62bc 5802 e1e3

e 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0001 0001

d 07e4 5959 255b df7b 2c92 095a d344 ecef
8865 6340 295b aae5 ddb0 d844 cda3 4ee7
a1f8 75b7 fc24 2d14 aee1 fa1d 22cb f619
37f7 5e8a 80a8 aff6 155d 24d9 3daf 5061

[서버에 대한 공개키와 개인키]

n ed7c b9e5 36d7 678f 7a80 7a0c 086b 8ed9
59f5 3222 3ea1 72f2 6080 24ec 2e0a 1231
1053 6814 3095 5ea4 41c1 95e1 1b40 776d
6c64 2e28 5ad7 6ca0 17de bf44 9412 1015

e 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0001 0001

d 3bdb 41e7 59c9 34da dedf 5ee3 8583 c67f
55c3 0d1d c077 64c7 5e37 2e17 310b 6cc5
e0b1 5444 1990 730d 5c14 f1c0 affb f2cf
124e d107 b6b1 f667 dd1c 5de8 1071 b181

알고리즘의 각 메시지에 대한 정보는 다음과 같다.

< 메시지 1 >

$C \rightarrow AS : ID_C \| Realm_C \| ID_{TGS}$

sacho,r-korea,tgs_3

< 메시지 2 >

$AS \rightarrow C : Realm_C \| ID_C \| Ticket_{TGS}$

$\| E_{KP_C}[E_{KS_{AS}}[Realm_{TGS} \| ID_{TGS}]]$

$*Ticket_{TGS} = E_{KP_{TGS}}[Realm_C \| ID_C \| AD_C]$

r-korea,sacho,

5a61 0f3a 3f4a 0678 9612 f8cc 710e 0e8b
5ffd b103 ae14 a386 88bf da1f aff5 cc4d
6db1 4c81 ad9b 0167 6bde 81e3 d039 d254
e91b 3984 7eb8 13da 8c0f 0764 77cd 8aa9
5b15 97c4 4869 8512 1dbc 6e08 1b9f 091a
c09a 33ae 3601 1e62 daba e6c6 02e7 f4cf
1842 dd51 9004 6d96 b021 7bea 0f81 4b24
3912 fbe5 2a7a 2a58 79e5 e7cf 219e b7e5

< 메시지 3 >

$C \rightarrow TGS: ID_V \| Ticket_{TGS} \| Authenticator1_C$

$*Authenticator1_C = E_{KS_C}[ID_C \| Realm_C]$

csjch,

5a61 0f3a 3f4a 0678 9612 f8cc 710e 0e8b
5ffd b103 ae14 a386 88bf da1f aff5 cc4d
6db1 4c81 ad9b 0167 6bde 81e3 d039 d254
e91b 3984 7eb8 13da 8c0f 0764 77cd 8aa9
07f0 15ea db0d 8f2f 79ad 58be df0c 3c39
c18a 2730 17b2 de8f 95ab dd29 f9d5 cc9c
a4ee f54e 057a ba3f a4dc 53e2 1757 7b6d
081c 9ccd c64f d004 9925 a2cf 81c5 fc68

< 메시지 4 >

$TGS \rightarrow C : Realm_C \| ID_C \| Ticket_V \|$

$E_{KP_C}[E_{KS_{TGS}}[Realm_V \| ID_V]]$

$*Ticket_V = E_{KP_V}[Realm_C \| ID_C \| AD_C]$

sacho,r-korea,

148b e7a6 6464 1335 81be ab2b 1b3e 092d
44a8 824b 002f a7b7 128c 75df 3175 4955
f9b6 1f41 35b7 16b4 2aef bff2 1705 3c0f
17d9 88a8 f6f9 9397 bffe 4b3b 2772 d679
a25f efed c1b8 774c 230b bb23 9bdc 8cd2
c313 5e92 7d70 1172 8812 d31d 1c5d 6b18
2564 fca3 d9f8 dc3a c261 ad87 46b5 1386
7f7e d6a8 9e33 5bd6 68c2 3bb6 e8c9 c752

< 메시지 5 >

$C \rightarrow V : Ticket_V \| Authenticator2_C$

$*Authenticator2_C = E_{KS_C}[ID_C \| Realm_C \|$

$E_{KP_V}[Seq\# \| Subkey]]$

148b e7a6 6464 1335 81be ab2b 1b3e 092d
44a8 824b 002f a7b7 128c 75df 3175 4955
f9b6 1f41 35b7 16b4 2aef bff2 1705 3c0f
17d9 88a8 f6f9 9397 bffe 4b3b 2772 d679
07f0 15ea db0d 8f2f 79ad 58be df0c 3c39
c18a 2730 17b2 de8f 95ab dd29 f9d5 cc9c
a4ee f54e 057a ba3f a4dc 53e2 1757 7b6d
081c 9ccd c64f d004 9925 a2cf 81c5 fc68

< 메시지 6 >

$V \rightarrow C : E_{KP_C}[Seq\# \| Subkey]$

3c42 2591 4336 d9b4 daee 2e29 8ab1 0fb6
c499 11eb 7d95 6444 1103 4614 d5cf b10a
aa30 9f60 60e9 b034 64fd 7489 bbc2 a73d
45b7 a928 152a 82ea eba3 a5b3 9597 5898

(단, KP : 공개키, KS : 개인키)

수행 결과, 본 논문에서 제안한 Kerberos 버전 5에 공개키 암호 알고리즘중 RSA를 이용한 인증 방식이 올바로 수행됨을 알 수 있었다.

4. 결 론

개방형 시스템인 네트워크 상에서의 서비스 교

환은 사용자와 서버간의 인증을 필요로 한다. Kerberos는 대표적인 인증 메커니즘으로 특히 버전 5는 분산형 환경에 적합하도록 설계되었다.

일반적인 Kerberos 인증 시스템에서 사용하고 있는 비밀키 암호 알고리즘인 DES는 안전성에 대한 문제와 비밀키 생성과 분배에 대한 문제를 갖고 있다

따라서, 본 논문에서는 대표적인 비밀키 암호 알고리즘인 DES대신 공개키 암호 알고리즘인 RSA를 이용한 Kerberos 버전 5 알고리즘을 제안하였다. 제안된 알고리즘은 RSA를 이용하였으므로 DES에 관련된 안전성에 대한 문제를 해결하였으며 공개키를 이용함으로써 이전의 알고리즘에서 Kerberos 시스템이 수행해야 했던 비밀키(세션키) 생성 및 분배 과정을 없앴다. 그리고 시뮬레이션을 통해 제안된 알고리즘이 올바르게 수행됨이 확인되었다

앞으로의 연구 방향은 제안된 알고리즘의 구현을 통하여 분산환경에서의 적용 여부를 알아보고자 한다.

참고문헌

[1] Kaufman, Perlman and Speciner, Network Security, Prentice Hall, 1995.

[2] William Stallings, Network and Internetwork Security, Prentice Hall, 1995.

[3] J. Kohl and C. Neuman, The Kerberos Network Authentication Service, RFC 1510, 1993.

[4] D. W. Davies and W. L. Price, Security For Computer Nerwork, A Wiley-Interscience Publication, 1984.

조 성 아

1996년 고려대학교 전산학과 학사
1996년-현재 고려대학교 대학원 전산학과 석사과정
관심분야 · 알고리즘, 암호학, 네트웍 보안, 스케줄링

한 태 창

1996년 고려대학교 전산학과 학사
1996년-현재 고려대학교 대학원 전산학과 석사과정
관심분야 알고리즘, 그래프 이론 네트웍 보안, 프로그래밍 언어, 스케줄링

함 호 상

1981년 고려대학교 산업공학과 학사
1981년 고려대학교 산업공학과 석사
1995년 고려대학교 산업공학과 박사
1982년-현재 시스템공학연구소 시스템통합연구부 전자거래 연구실장
관심분야 : CALS/EC 및 인터넷 보안기술

이 동 훈

1984년 고려대학교 경제학과 학사
1988년 오클라호마 대학 전산학과 석사
1988년 오클라호마 대학 전산학과 박사
1993-현재 고려대학교 자연과학대학 전산학과 조교수
관심분야 · 컴퓨터 이론, 알고리즘, 네트웍 보안