CORBA 환경에서 티켓을 사용하지 않은 Kerberos 인증

최 은 복 °, 이 영 록, 노 봉 남
전남대학교 전산학과

Ticketless Kerberos Authentication in CORBA Environments

Eun Bok Choi °, Young Rok Lee, Bong Nam Noh
Dept. of Computer Science, Chonnam National University

< 요 약 >

객체지향 프로그래밍의 발달과 필요성에 따라 분산된 통신망에도 객체지향 기술이 적용되는 분산객체 컴퓨팅이 새로운 버전으로 대중화되고 있다. 따라서 분산 객체간의 자원을 효율적으로 이용하고 다른 통신망의 사용자들과 원할한 통신을 하기 위해서는 자원의 공유가 필연적이다. 그러나 이러한 정보와 자원의 공유는 불완전한 통신 채널을 통해 이루어지기 때문에 불법적인 사용자들이 정보를 악용하는 보안상의 문제가 발생한다.

본 논문은 다양한 컴퓨팅 자원들이 통신망으로 연결되어 있는 개방형 분산 객체 컴퓨팅 환경에서 운영되는 컴퓨터 통신망의 정보를 보호하고 정당한 사용자에게 자원을 효율적으로 제공할 수 있는 인증 메카니즘으로 CORBA 환경에서 티켓을 사용하지 않은 Kerberos 인증을 제안하였다.

1. 서 론

정보통신의 발달로 인하여 각종 정보시스템들이 통신망으로 연결되어 다양한 형태의 정보 서비스를 제공하고 있으며 객체지향 프로그래밍의 발달과 필요성에 따라 분산된 통신망에도 객체지향 기술이 적용되는 분산객체 컴퓨팅이 새로운 버전으로 대중화되고 있다. 분산 객체 컴퓨팅은 분산 컴퓨팅의 기본단위를 객체로 한것으로 이기종 컴퓨터 시스템으로 구성된 클라이언트-서버 환경에서 분산 객체간의 상호운용을 목표로 하고 있다.

현재 관련업체들의 협의체인 OMG(Object Management Group)에서는 분산객체 컴퓨팅 시스템을 표준화하려는 작업의 하나로 분산 객체 컴퓨팅 표준 구조인 CORBA(Common Object Request Broker Architecture)를 제안하였다[8,10]. CORBA는 현재 Digital, IBM, SunSoft, HP 등에서 채택하고 있으며 많은 객체 기술 개발사들도 적극적으로 이에 참여하여 연구중에 있다.

분산 컴퓨팅 환경에서는 정보를 보다 효율적으로 활용하고 사용자의 욕구를 충족시키기 위한 정보 처리 기술이 제공되어야 한다. 분산 객체간의 자원을 효율적으로 이용하고 다른 통신망의 사용자들과 원할한 통신을 하기 위해서는 자원을 공유해야 한다. 그러나 이러한 정보와 자원의 공유는 불완전한 통신 채널을 통해 이루어지기 때문에 권한을 부여받지 못한 사용자들이 정보를 도청하거나 삽입 및 삭제하여 악용하는 보안상의 문제점이 대두되고 있다.

이러한 문제점을 해결하기 위해서는 정당한 사용자인가를 검사하는 인증문제, 사용자에게 객체에 접근할수 있는 등급을 부여하는 접근제어 기법 그리고 사용자들이 분산시스템에서 행한 행

동 등을 추적할 수 있는 보안 감사등 여러가지 보안기능이 제공되어야 한다.

보안기능 중에서 인증문제를 담당할 인증서버는 컴퓨터 통신망의 사용자들이 정보를 안전하게 전송하기 위해서 서로 사용하는 통신키를 생성, 분배, 관리하는 기능과 사용자들의 인증 문제를 효율적으로 대처할 수 있는 기능을 가지고 있어야 한다.

본 논문에서는 이러한 분산객체 컴퓨팅 환경에서 사용자간의 인증 문제를 해결할 수 있는 메카니즘을 제안하였다. 제 2장에서는 OMG에서 제안한 분산 객체 컴퓨팅 표준 구조인 CORBA의 개요에 대해 알아보고 제3장에서는 CORBA에서 발생할 수 있는 보안 위협요소와 보안서비스를 소개하였다. 제 4장에서는 분산객체 컴퓨팅 환경에서 사용자간의 인증 문제를 해결할 수 있는 메카니즘으로 CORBA 환경에서 티켓을 사용하지 않은 Kerberos 인증을 제안하였다. 마지막으로 제5장에서는 결론과 향후 연구방향을 서술하였다.

2. OMG CORBA

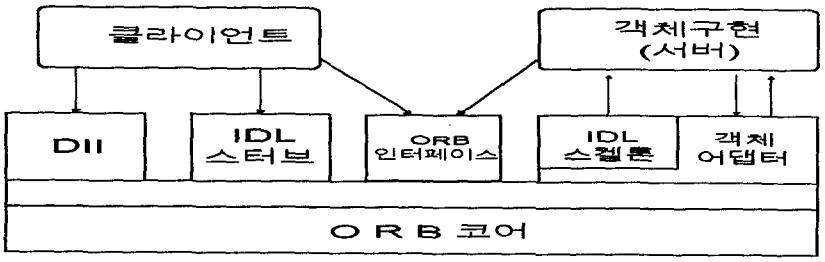
2.1 객체관리 구조(Object Management Architecture)

CORBA는 OMG에서 제시한 객체간의 통신을 중재하는 역할을 수행하기 위한 구조이다. OMG에는 수백 회원사가 있으며 SunSoft, 휴렛팩커드, DEC, HyperDesk, NCR, IBM 등이 많은 활동을 하고 있다.

CORBA에서 클라이언트와 객체들은 한개이상의 객체 요청 중개자(Object Request Broker)들을 통해서 서로 통신을 교환하는데 클라이언트에 의해 요청된 서비스는 해당 서버에 의해 제공된다. CORBA의 사양은 공통의 ORB 서비스 및 인터페이스를 마련하고 이식가능한 클라이언트 객체들의 구현을 지원하기 위한 기본 틀을 정의한다.

OMG에서 제안한 객체 관리 구조(OMA)는 개방 분산 환경에서 응용들 사이의 상호운용성을 제9공하고 이기종의 다중 객체시스템들에 대한 투명성을 제공하여 클라이언트/서버 사이의 요구 및 응답을 투명하게 제공하며 주요 구성성분들의 재사용성을 보장하는 객체 참조 모델이다.

객체 관리 구조는 다음 (그림 1)과 같이 응용객체, 객체 구현, ORB 코어 , 인터페이스 정의 언어(Interface Definition Language), 동적 호출 인터페이스(Dynamic Invocation Interface), 인터페이스 저장소, 객체 서비스, 객체 어댑터로 구성되어 있다[8,9,10].



< 그림 1 > ORB 구조

1) 응용 객체

CORBA의 응용 객체는 OMG의 기술에 의해 정의된 사용자들의 응용 프로그램으로 요청을 만들고 응답을 받는 실체로서 응용 프로그램이나 데이타베이스 관리시스템, 사용자 인터페이스 등이 객체가 될 수 있다. CORBA에서 각각의 객체는 이름 서비스나 디렉토리 서비스와 비슷하게 객체 참조에 의해서 식별된다. 객체는 연산을 호출함으로서 다른 객체에 요청을 만들 수 있는데 요청을 받은 객체는 메소드를 실행해서 연산을 수행한다. 요청은 클라이언트가 객체에게 서비스를 수행하도록 요구하는 것을 말하며, 연산을 수행하는 데 필요한 연산이름, 목적 객체, 매개변수, 그리고 연산의 내용에 관한 정보로 구성된다.

2) 객체 구현

객체구현은 IDL 스켈톤을 통하여 클라이언트의 요청을 받아들이며 요청을 처리하는 동안에 필요한 서비스를 위하여 객체 어댑터나 ORB 인터페이스를 호출한다. 요청이 완료되면 제어나 결과값이 클라이언트에 반환하게 된다. 객체구현은 클라이언트로부터 요청받은 서비스를 수행하는 데 필요한 데이타와 함수의 묶음으로 서비스를 수행하기 위한 메소드와 속성을 구현해 놓은 것이다.

CORBA는 모든 응용 프로그램들이 객체들로 정의되어 있는 하나의 분산컴퓨팅 구조로서 객체들은 클라이언트나 서버의 역할로 서로 대체될 수 있다. 객체가 서비스를 요청하는 경우 클라이언트의 역할을 수행하고 서비스를 제공하는 경우에는 서버의 역할을 하는데 이때 서버 객체를 객체구현이라 부른다. 클라이언트-서버환경에서 대부분의 객체 구현은 서버에 존재하며 객체 어댑터라는 인터페이스를 통해서 CORBA의 서비스에 접근 할 수 있다.

3) ORB 코어

ORB 코어는 객체구현과 관련된 제반 사항을 처리하는 일종의 시스템 버스의 역할을 하는 것으로 객체의 위치를 결정하고, 객체간의 요청/응답 메시지를 전달하고 그리고 메소드를 결합하는 서비스들의 집합이다.

4) 인터페이스 정의 언어(IDL)

통신망이 분산 객체들로 구현된 경우에 서비스를 원하는 한 객체가 서비스를 제공하는 분산된 다른 객체와 통신하기 위해서는 객체간에 어떻게 접근하는지를 기술한 접속방식이 정의되어야 한다. 그래서 OMG에서는 객체들 간의 접속방식인 인터페이스 정의 언어를 제시하였다.

CORBA에서 모든 객체는 IDL에서 정의한 속성과 연산의 집합으로 생성된다. IDL은 객체가 요구하는 연산범위나 이러한 연산을 수행하는 데 필요한 매개변수, 그리고 객체 인터페이스의 다른 구성요소 등을 정의한다. IDL 정의는 클라이언트 쪽의 스터브나 서버쪽의 스켈톤으로 컴파일 될 수 있으며, DII에 의해서 인터페이스 저장소에 저장될 수도 있다.

5)동적 호출 인터페이스(DII)

IDL이 대상 객체에 종속된 컴파일타임 스터브인 대신 DII는 대상 객체와 독립된 런타임 인터페이스로 컴파일시에는 서버의 접속방식에 관한 정보를 모르고 있다가 수행 시에 연산을 호출하여 서버를 수행시킨다.

6) 인터페이스 저장소

일종의 데이타베이스로 ORB 환경에 있는 객체의 인터페이스 정보, 연산 이름 그리고 매개변수의 형태 등을 저장하고 있다. 객체들은 서버에 의해 지원되는 속성이나 연산을 알기위해 인터페이스 저장소를 참조한다.

7) 객체 서비스

모든 객체가 공동으로 유용하게 사용할 수 있게 OMG에서 정의한 인터페이스이다. 예를 들면, 객체 이름 관리 서비스, 객체 사이 사건을 교환하기 위한 사건 서비스, 객체 저장을 위한 서비스 등이 있다.

8) 객체 어댑터

객체 어댑터는 객체 구현을 위해 서버가 ORB 코어로 부터 필요한 서비스를 제공받는데 이용하는 인터페이스로 객체의 활성화/비활성화, 객체 구현의 등록, 객체의 생성/삭제/질의, 객체의 구현 저장 및 접근제어 그리고 메소드 호출 등의 기능을 한다.

3. CORBA 보안

3.1 위협 행위

분산시스템에서는 몇가지 복잡한 보안 문제가 제기되고 있다. 호스트들은 네트워크를 통해 메시지를 송·수신하므로써 통신을 한다. 이러한 환경에서 호스트들 사이에 분산된 여러 자원이나 정보가 인증서버에 의해서 제공된 네트워크 서비스의 형태로 공유된다. 자원에 접근을 원하는 개개의 프로세스들은 적절한 인증서버에게 서비스 요청을 요구한다. 분산시스템은 시스템의 침입자 뿐만 아니라 정당한 사용자에 의해서도 있는 다양한 위협을 받게 된다. 일반적으로 크게 두가지 위협으로 나누어 볼 수 있다.

1) 호스트 위협

호스트 위협은 시스템 내의 호스트의 파괴와 관련되는데, 파괴의 양상은 프로세스 상태 정보에 영향을 주는 단순한 침입으로부터 한 호스트의 모든 제어를 관할하는 적극적인 침입까지 다양하다. 호스트 위협 문제는 하드웨어 기술과 소프트웨어 기술의 조합에 의해서 해결할 수 있다. 여기에서 각 호스트는 프로세스들을 적절하게 분리하기 위해 신뢰하는 참조 모니터로 구현한다.

2) 통신 위협

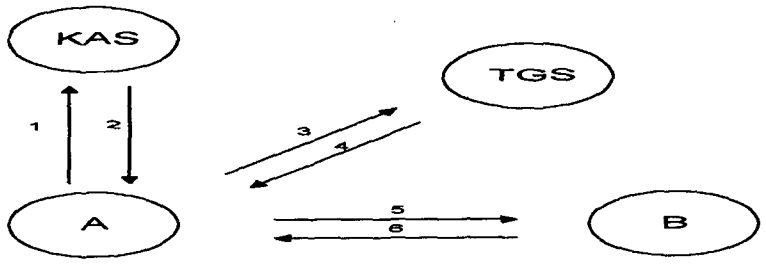
메시지 통신과 관련된 위협을 몇가지로 대별할 수 있다. 첫째는 비밀 정보를 추출하기 위해 네트워크 링크를 통해 전송되고 있는 메시지를 도청하는 경우이다. 이것은 수동적인 위협으로서 위협을 받고 있는 송·수신자에게 직접적인 영향을 미치지 않기 때문에 쉽게 탐지하기가 어렵고 예방 조치에 의해서만 위협 문제를 해결할 수 있다. 둘째는 수신자에게 위조된 메시지를 보내기 위해 네트워크 채널을 통해 전송되는 메시지를 임의로 수정, 삭제, 삽입하는 경우이다. 셋째는 이전에 전송한 메시지를 재전송하는 경우이다. 둘째와 마지막은 능동적인 위협으로 송·수신자에게 직접적인 영향을 미치며 예방, 탐지, 복구에 의해서 위협 문제를 해결할 수 있다[6].

3.2 인증 메카니즘

1) Kerberos 인증

Kerberos는 Needham and Schroeder 모델과 시간 스템프를 첨가한 Denning and Saaco 모델을 기반으로 하여 개발되었다. 다른 분산 시스템과 같이 Kerberos 인증 시스템에 관한 모델은 클라이언트에 의해서 사용되는 서비스들로 구성된다. 어떤 클라이언트가 서비스를 사용하기를 원하는 경우에 클라이언트는 티켓 제공 서버(Ticket Granting Server)로 부터 서비스를 위한 티켓을 받아야 한다. 여기에서 티켓 제공 서버는 다른 모델의 서비스와 같지만 초기 로그인 단계에서 티켓 제공 서버에게 인증받기 위한 티켓을 Kerberos 인증서버로부터 받는다는 것이 다르다.

다음은 Kerberos(그림 2)의 프로토콜과 특성들이다[1,7].



< 그림 2 > Kerberos 인증 프로토콜

<메시지 1> . A --> KAS ; < A, TGS, T_1 >

사용자 A는 KAS에 자신의 ID와 TGS ID를 타임스템프와 함께 보낸다.

<메시지 2> . KAS --> A ; < $K_{A,TGS}$ >$K_{A,KAS}$, < $TICKET_{A,TGS}$ >$K_{KAS,TGS}$

KAS는 사용자 A가 정당한 사용자인지를 판별한후 정당한 사용자이면 A와

TGS의 공유키와 티켓을 부여하여 보낸다. 여기서 $TICKET_{A,TGS}$ = < T_2, $K_{A,TGS}$, A, TGS, W_A, L>으로 구성된다.

<메시지 3> . A --> TGS ; < A, T_3, W_A >$K_{A,TGS}$, B , < $TICKET_{A,TGS}$ >$K_{KAS,TGS}$
사용자 A는 A와 TGS의 공유키를 사용하여 B와의 통신키 요청 메시지와 KAS로 부터 받은 티켓을 그대로 TGS에 보낸다.

<메시지 4> . TGS --> A ; < $K_{A,B}$ >$K_{A,TGS}$, < $TICKET_{A,B}$ >$K_{B,KAS}$
TGS는 A에게서 받은 티켓을 해독하여 A를 인증한후 자신이 생성한 티켓과 A, B간의 통신키를 A에게 보낸다. 여기서 $TICKET_{A,B}$ = < T_4, $K_{A,B}$, A, L >으로 구성된다.

<메시지 5> . A --> B ; < A, T_5 ,W_A >$K_{A,B}$, < $TICKET_{A,B}$ >$K_{B,KAS}$
사용자 A는 A,B간의 통신키를 이용하여 B와의 통신요망메시지와 TGS로 부터 받은 티켓을 B에게 보낸다.

<메시지 6> . B --> A ; < T_5+1 >$K_{A,B}$
사용자 B는 티켓으로부터 A를 확인한후 사용자 A가 자신을 인증 하도록 타임스템프에 1을 더하여 보낸다.

< Kerberos 특징 >

1) 초기에 사용자는 인증서버와 함께 공유하는 키를 통해 인증 되어지고 그후에는 인증서버로부터 부여되는 티켓을 통해 상호인증되어진다.
2) TGS, KAS에의해 공유되는 세션키를 사용한다.
3) Kerberos는 대칭 암호화 방법을 이용한다.
4) 메시지의 시간 지연은 타임스템프 방법에 의해 검사되어진다.
5) 인증서버는 온라인 상태이어야 한다.

2) SPX 인증

SPX는 공개키 암호화 기법을 기반으로 하는 인증 시스템이다. 다른 인증 메카니즘과 같이 SPX는 식별자를 증명하기위해 비밀정보를 이용한다[1,7]. 모든 개체들의 마스터 키는 CDC(Certificates Distribution Center)라는 곳에 저장하므로써 개체가 키관리를 할 필요가 없다.

신용장은 사용자 식별자와 비밀키로 구성된다. 사용자 워크스테이션에 클라이언트 증명서를 저장하는 것은 초기 신용장 처리시에 이루어진다. 클라이언트와 서버는 서로 다른 신용장을 유지한다. 중요한 차이점은 클라이언트 위임장은 장기 비밀키 대신에 단기 위임 비밀키를 유지한다.

SPX는 ISO X.509 추천에 기술된 이름 구조에 자세하게 기술되어 있다. 개체는 그의 식별자를 다른 개체에게 인증받기 전에 신용장의 등록과 설치 과정이 이루어져야 한다. SPX의 등록은 사용자가 증명서와 암호화된 비밀키, 패스워드를 CDC에 첨가하는 과정이다. 인증 교환은 토큰을 통해서 이루어지는데 클라이언트의 토큰에는 클라이언트 이름과 티켓, 서버의 공개키로 암호화된 DES 키, 암호화된 DES키의 서명, 인증자 식별자가 포함된다. 서버가 토큰을 받으면 먼저 인증자를 확인하고 티켓을 얻기 위해서 서명을 조사한다.

SPX와 같은 공개키 암호화 인증 시스템에서는 다음과 같은 증명서의 취소가 어렵고 성능저하 문제가 야기된다.

▨ 즉시 취소 불가능

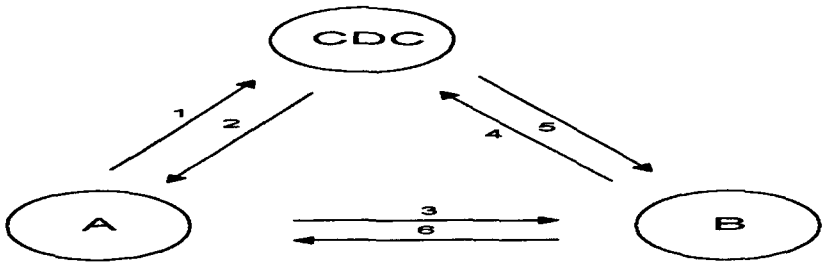
증명서 취소 리스트는 일정 간격으로 증명서 관리자에 의해서 감시된다. 만일 사용자 비밀키나 증명서가 위협받았거나 더 이상 사용할 수 없는 경우에는 증명서를 취소한다. 증명서의 취소

는 특별하거나 빈번한 연산이 아니기 때문에 키의 침입이 발견되더라도 보안 문제를 감수해야 한다.

▨ 성능저하 문제

여섯단계의 인증 프로토콜 중에서 암호화하는데 필요한 두번의 단계에서는 증명서가 사용되지 않는다. 증명서는 안전한 CDC에 저장되고 티켓은 로그인 과정에서 얻어진다. 또한 디지털 서명과 메세지 서명에는 적은 CPU 싸이클과 적은 데이타 전송이 요구되지만 메세지의 암·복호화하는데는 많은 CPU 싸이클과 데이타 전송이 요구된다. 따라서 영역간의 암·복호화과정은 중요한 성능 저하를 야기시킨다.

다음은 SPX(그림 3)의 프로토콜과 특성들이다[1].



< 그림 3 > SPX 인증 프로토콜

<메시지 1> . A --> CDC ; < B >

사용자 A는 CDC에게 B와의 통신 요망과 함께 B의 공개키를 요청한다.

<메시지 2> . CDC --> A ; < B , PK_B , L_1 >$SK_{CA,A}$

CDC는 사용자 A를 인증한후 B의 공개키를 A가 신뢰하는 CA의 비밀키로 암호화하여 보낸다.

<메시지 3> . A --> B ; < A > , < PDK_A , L_2 >SK_A , < $K_{A,B}$ >PK_B ,

{ < $K_{A,B}$ >PK_B }SDK_A , T , < T , W_A >$K_{A,B}$

사용자 A는 A와 B가 공유하는 세션키의 안전한 전송을 위해 공개 위임키를 자신의 비밀키로 암호화해서 보내며 또한 B의 공개키로 암호화한 세션키를 자신의 비밀 위임키로 암호화해서 보낸다. 그리고 자신의 주소를 A,B의 세션키로 암호화하여 함께 보낸다.

<메시지 4> . B --> CDC ; < A >

사용자 B는 CDC에게 A의 ID를 보내므로써 A의 공개키를 요청한다.

<메시지 5> . CDC --> B ; < A, PK_A , L_3 >$SK_{CA,B}$

CDC는 A의 공개키를 B가 신뢰하는 CA의 비밀키로 암호화하여 보낸다.

<메시지 6> . B --> A ; < T > , < T , W_B >$K_{A,B}$

사용자 B는 자신의 주소를 A,B의 세션키로 암호화하여 A에게 보낸다.

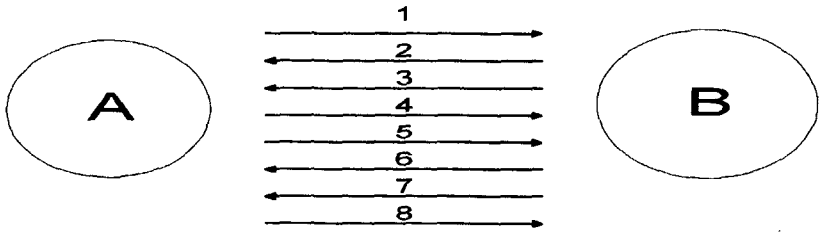
< SPX 특징 >

1) 사용자는 비밀키의 지식(knowledge)를 증명하므로써 인증되어 진다.
2) 초기 통신 사용자에 의해 형성되어지는 공유된 세션키를 사용한다.
3) SPX는 인증하는 동안 암호화와 디지탈 서명을 위하여 공개키 암호화 방법을 사용한다.
4) 신용장 제공 서버는 정당하게 신용장을 제시하므로써 신뢰되어야 한다.
5) CDC는 두 사용자 사이에 인증하는 동안 온라인 상태이어야 한다.
6) SPX는 동기화를 위해 타임스템프 방법을 사용한다.

3) SELANE

SELANE(SEcure Local Area Network Environment)는 분산 시스템을 위한 인증 서비스로써 karlsrahe에 있는 EISS에서 개발되었다. SELANE은 SKIAs(Secure Key Issuing Authorities)의 사용을 제안했는데 이는 사용자에게 단방향 또는 상호 인증을 위해 요구되는 신용장을 제공한다. 이 스키마에서 사용되는 단방향 기능은 적당한 소수 P를 사용하여 MOD연산을 수행한다. 또한 인증은 SKIA에 의해 제시되는 신용장으로 부터 유도되는 일반적인 세션키를 사용하여 이루어진다[1,7].

다음은 SELANE(그림 4)의 프로토콜과 특성들이다[1].



< 그림 4 > SELANE 인증

<메시지 1> . A --> B ; A, P_A
사용자 A는 자신의 ID와 공개키를 사용자 B에게 보낸다.

<메시지 2> . B --> A ; V_B , (단, $V_B = P_A^{TB}$ MOD P)
사용자 B는 B의 난수 TB 와 A의 공개키로 계산된 V_B값과 자신의 ID및 공개키를 사용자 A에게 보낸다.

<메시지 3> . B --> A ; B , P_B
또한, 사용자 B는 자신의 ID와 공개키를 사용자 A에게 보낸다.

<메시지 4> . A --> B ; V_A , (단, $V_A = P_B^{TA}$ MOD P)
사용자 A는 A의 난수 TA와 B의 공개키로 계산된 V_A값과 자신의 ID및 공개키를 사용자 B에게 보낸다.

<메시지 5> . A --> B ; < R_A >$K_{A,B}$
사용자 A는 사용자 B에게 $K_{A,B}$($K_{A,B} = V_B^{SA}$ MOD P)로 A의 난수 R_A를 암호화하여 보낸다.

<메시지 6> . B --> A ; [{ < R_A >$K_{A,B}$ }$K^{-1}_{A,B}$]$K_{B,A}$
사용자 B는 메시지 5에서 보낸 난수를 A,B의 비밀키를 복호화한후 다시 $K_{B,A}$($K_{B,A} = V_A^{SB}$ MOD P)의해 암호화하여 보낸다. 그러면 사용자 A는 $K_{B,A}^{-1}$로 복호화하여 자신의 난수와 비교하므로써 인증 되어진다.

<메시지 7> . A --> B ; < R_B >$K_{B,A}$

<메시지 8> . B --> A ; [{ < R_B >$K_{B,A}$ }$K^{-1}_{B,A}$]$K_{A,B}$
메시지 5,6의 역할을 바꾸어 사용자 B가 사용자 A를 인증한다.

<SELANE 특징>

1) 사용자는 공통키를 통하여 인증이 되어진다.
2) 공유키가 확립되고 양 통신 사용자는 이 세션키를 생성하기 위해 조합되어진다.
3) SELANE은 인증하는 동안 디지털 서명을 위해 공개키 암호화 방법을 사용한다.
4) 핸드세이킹 프로토콜은 재전송 위협을 방지하기 위해 사용된다.

3.3 Kerberos 인증 메카니즘의 문제점

기존의 Kerberos인증 메카니즘이 안고있는 몇가지 문제점을 나열해 보면

첫째, 사용자 A는 KAS를 인증할 수 있는 방법이 없어 무조건 믿는다는 가정이 전제된다.

둘째, 기존의 Kerberos 인증 메카니즘은 티켓이란 개념을 사용하여 이 티켓속에 공통키와 타임스템프, 자신의 ID, 유효시간 등 모든 정보를 저장하여 암호화한 후 전송한다. 그러면 사용자 A는 이 티켓을 그대로 다른 통신사용자에게 그대로 전송하므로써 실질적으로 전송시간이 길어지고 전송속도가 저하될 뿐 아니라 보내고자 하는 데이타의 양이 많아지는 단점이 있다.

셋째, 키분배와 키관리 문제로 두 영역내의 인증서버는 비밀 대화키를 공유한다. 만일 분산 환경에서 인증서버의 수가 증가함에 따라 영역간의 대화키가 증가하게 되어 대화키 관리 문제가 생긴다.

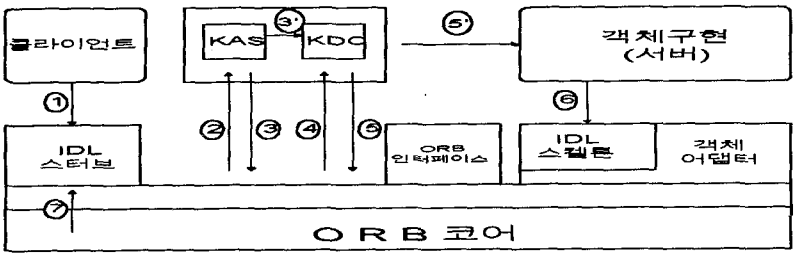
넷째, Kerberos는 클라이언트가 인증받고자 하는 인증서버나 제공받고자 하는 서버의 모든 정보를 알고 있어야 한다.

4. CORBA 환경에서 티켓을 사용하지 않은 Kerberos 인증

위와같은 Kerberos 인증 메카니즘의 몇가지 단점을 해결하기위해 CORBA에 적용될 수 있는 인증 프로토콜을 제안한다(그림 5). 본 제안된 프로토콜에서는 CORBA를 이용하므로 단지 클라이언트가 제공받고자 하는 서비스를 요청하면 CORBA에 의해 인증과 함께 원하는 서비스가 제공되므로 사용자에게 편리하다.

먼저 클라이언트는 서비스를 제공하는 객체에게 메시지를 요청한다. ORB 코어는 클라이언트를 인증서버에 의뢰하여 인증과정을 거친후 서비스를 제공하는 서버 객체의 위치를 파악하여 수행시킨다. 인증후 클라이언트의 스터브와 서버의 스켈톤은 인증서버에 의해 제공된 세션키를 통해 메시지를 송·수신하게 된다.

KAS 인증 문제는 ORB 코어가 난수를 발행하여 보내므로써 KAS를 인증할 수 있도록 하였다. 또한 티켓에 대한 과부하의 문제점은 인증서버가 각각 ORB 코어와 키분배센터에게 필요한 정보를 개별적으로 전송하고 그들간의 인증문제는 타임스템프와 난수를 사용하여 점검하도록 하였다.



< 그림 5 > CORBA 환경에서 티켓을 사용하지 않은 Kerberos 인증

<메세지 1>. Client --> Stub ; Send-Req = { target_object_referance, operation, parameter}

클라이언트는 IDL 스터브에게 목적객체의 참조, 연산의 종류, 매개변수 그리고 다른 관련된 정보를 담은 Send-Req 메세지를 전송한다.

<메세지 2>. ORB 코어 --> KAS ; KAS-Req = {Client, KDC, T_1, N_{ORB}}

Send-Req 메세지를 받은 ORB 코어는 서비스를 요청한 클라이언트가 정당한 사용자인가를 확인하기 위해 Kerberos 인증 서버에게 KAS-Req 메세지를 전송한다. 단, KAS-Req 메세지는 클라이언트와 키분배센터(KDC)의 ID, 타임스템프, 그리고 ORB 코어가 발행한 난수로 구성된다.

<메세지 3>. KAS --> ORB 코어 ; KAS-Res = $\{K_{ORB,KDC}, N_{ORB}, N_{KAS}\}K_{ORB,KAS}$
KAS는 클라이언트를 인증한 후 ORB와 KDC가 공유하는 키와 ORB의 난수, 그리고 자신이 발행한 난수를 암호화하여 보낸다.

<메시지 3'>. KAS --> KDC ; KAS-KDC = $\{K_{ORB,KDC}, T_2, N_{KAS}\}K_{KAS,KDC}$
KAS는 KDC에게 ORB와 KDC가 공유하는 키와 타임스템프, 그리고 자신이 발행한 난수를 암호화해서 보낸다.

<메시지 4>. ORB 코어 --> KDC ; KDC-Req = {Client, Server, T_3, $N_{KAS}\}K_{ORB,KDC}$, Server
ORB 코어는 자신이 발행한 난수 N_{ORB}를 보고 KAS를 인증한 후 KDC에게 클라이언트와 서버의 ID, 타임스템프, 그리고 KAS가 발행한 난수를 함께 보낸다.

<메시지 5>. KDC --> ORB 코어 ; KDC-Res = $\{K_{C,S}\}K_{ORB,KDC}$
KDC에서는 메시지 3'에서 보내온 타임스템프, 난수를 메시지 4에서 보내온 타임스템프, 난수와 서로 비교한 후 ORB 코어에게 세션키를 암호화하여 보낸다.

<메시지 5'>. KDC --> Server ; KDC-Ser = $\{K_{C,S}\}K_{S,KAS}$
클라이언트를 인증한 후 키분배센터는 서버에게도 클라이언트와 통신할 수 있는 세션키를 암호화하여 보낸다.

<메시지 6>. Server --> Skeleton ; Ser-Skel = $\{K_{C,S}\}$
서버는 실질적인 서비스 요청을 받는 스켈톤에게 세션키를 분배한다.

<메시지 7>. ORB 코어 --> Stub ; ORB-Stub = {KC,S}
ORB 코어는 실질적인 서비스 요청을 하는 스터브에게 세션키를 분배한다.

5. 결론 및 향후 연구방향

지리적으로 분산되어 있는 정보를 보다 효율적으로 활용하고 사용자의 욕구를 충족시키기 위해서는 분산된 정보에 대한 처리 기술이 제공되어야 한다. 객체지향 프로그래밍의 발달과 필요성에 따라 분산된 통신망에도 객체지향 기술이 적용되는 분산객체 컴퓨팅이 새로운 버젼으로 대중화되고 있는 요즘 관련 업체들의 협의체인 OMG에서는 분산객체 컴퓨팅 시스템을 표준화하려는 작업의 하나로 분산 객체 컴퓨팅 표준구조인 CORBA를 제안하였다. 본 논문은 다양한 컴퓨팅 자원들이 통신망으로 연결되어 있는 개방형 분산 객체 컴퓨팅 환경에서 운영되는 컴퓨터 통신망의 정보를 보호하고 정당한 사용자에게 자원을 효율적으로 제공할 수 있는 인증 메카니즘으로 CORBA환경에서 티켓을 사용하지 않은 Kerberos 인증을 제안하였다.

본 프로토콜에서는 CORBA를 이용하므로 단지 클라이언트는 제공받고자 하는 서비스를 요청하면 CORBA에 의해 인증과 함께 원하는 서비스가 제공되므로 사용자에게 편리성을 제공한다. 본 논문을 CORBA에 적용한다면 다음과 같이 두가지 방법이 있을 수 있는데 첫째, ORB 코어에 보안성을 첨가하는 안전한 ORB코어를 설계하여 구현하는 방법과 둘째, 기존의 ORB코어 위에 보안을 담당할 제어기를 설계하여 구현하는 방법이 있을 수 있겠다.

추후 연구방향으로는 본 논문에서 제안한 프로토콜의 완전성을 검증하기 위해 프로토콜 분석 도구인 BAN 논리를 적용해 보고 또한 Petri-Net을 사용하여 본 프로토콜의 타당성을 검증하고자 한다.

참 고 문 헌

[1] D. Gollmann, T. Beth, F.Damm, "Authentication Services in distributed systems", Computers & Security, Dec., 1993, pp753 - 764

[2] Thomas Y.C.Woo and Simon S. Lam "Authentication for Distributed Systems", IEEE Computer Society, Jan., 1992, pp39 - 51

[3] Michael Burrows, Martin Abadi, Roger Needham, "A Logic of Authentication", ACM Transections on Computer Systems, Vol.8, No.1, Feb., 1990, pp18 - 36

[4] John Linn, "Practical Authentication for Distributed Computing", IEEE Computer Society, May, 1990, pp31 - 40

[5] Manfred Reitecspiess, "Open System Security Standards", Computers & Security, Dec., 1993, pp341 - 361

[6] William Stallings, Networking Standards A Guide to OSI, ISDN, LAN and MAN Standards , Addison Wesley, Jan., 1993, pp499 - 540

[7] R.Yahalom, B.Kelein, T.Beth, "Trust Relationships in Secure Systems - A Distributed Authentication Perspective", IEEE Computer Society, May, 1993, pp150 - 164

[8] Susan L. Chapin, William R. Herndon, LouAnna Notargiacomo, "Security For The Common Object Request Broker Architecture(CORBA)", IEEE Computer Society, , 1994, pp21 - 30

[9] Robert H. Deng, Shailendra K. Bhonsle, Weiguo Wang, Aurel A. Lazar, "Integrating Security in CORBA Based Object Architectures", IEEE Computer Society, May., 1995, pp50 - 61

[10] The Common Object Request Broker : *Architecture and Specification*, OMG Document Number 91.12.1

[11] Belinda Fairthorne, "OMG White Paper on Security", OMG Security Working Group, April, 1994