通信情報保護學會論文誌
第 8 卷 第 4 號, 1998. 12

분산 객체 환경에서의 보안 서비스 구현

이 권 일 *, 김 명 준*, 류 재 철**

Implementation of Security Service in Distributed Object-Oriented Environment

Kwon-il Lee*, Myoung-Joon Kim*, Jae-Cheol Ryou**

요 약

OMG (Object Management Group)에서는 CORBA (Common Object Request Broker Architecture) 환경에서의 보안 문제를 해결하기 위해서 CORBA 보안 서비스[8]를 정의하였다. CORBA 보안 서비스는 다양한 보안 기술을 허용하는 보안 구조를 제안하고 있으며, 사용자 인증, 접근제어, 보안 통신 등에 필요한 보안 객체를 정의하고 있다. 또한 CORBA 환경에서 수행되는 응용들에게 투명한 보안 통신을 제공하는 것을 기본으로 한다.

본 논문에서는 ECMA (European Computer Manufacturers Association) SESAME (a Secure European System for Application in Multi-vendor Environment) Ver. 4[7]를 사용하여 CORBA 환경에서 수행되는 응용들에게 보안 통신을 제공하는 것에 중점을 둔 CORBA 보안 서비스의 설계 및 구현에 관하여 기술하였다. 접근제어, 통신 무결성, 통신 기밀성, 적절한 보안 수준을 보장하기 보안 정책 관리 기능 등을 제공하는 보안 소프트웨어인 SESAME Ver. 4에서 제공하는 GSS-API (Generic Security Service Application Programming Interface)[5, 10, 11]를 사용하여 CORBA 환경에서 보안 통신에 필요한 보안 객체들을 설계 구현하였고, CORBA 환경을 위한 전체 보안 구조를 제시하였다. 본 논문에서 제시한 보안 구조는 보안 통신을 제공하기 위해 구현된 보안 객체와 SESAME Ver. 4에서 제공하는 인증, 접근제어, 보안 정책 관리 기능을 통합한 형태이다.

Abstract

OMG (Object Management Group) announced CORBA (Common Object Request Broker Architecture) Security specification[8] to resolve security problem on CORBA environment. It suggests

* 한국전자통신연구원 컴퓨터 소프트웨어연구소 데이타공학연구부부
** 충남대학교 컴퓨터 과학과

an architecture that allows various security technologies to be used and specifies security objects to supply user authentication, access control, secure communication, and so on. And an application object running on CORBA environment which want to secure communication should not need to be aware of security.

In this paper, we design and implement security objects to support secure communication using ECMA (European Computer Manufacturers Association) SESAME (a Secure European System for Application in Multi-vendor Environment) Ver. 4[7]. The GSS-API (Generic Security Service Application Programming Interface)[9, 10, 11] of SESAME Ver. 4 was used to implement security objects for supporting secure communication. SESAME Ver. 4 supports access control, communication integrity and communication confidentiality, and offers the means to ensure that access to services is policed to appropriate level of security. This paper also proposes overall security architecture for CORBA environment. This architecture is composed of SESAME and security objects to be implemented for supporting secure communication on ORB (Object Request Broker). In this architecture, security objects, which are implemented in this paper, provide communication integrity and communication confidentiality, and SESAME provides user authentication, access control, security audit and security policy management.

Keyword : CORBA, CORBA Security Service, Security, GSS-API

I. 서 론

분산 처리 기술은 분산 처리 환경을 구성하는 단일 시스템의 동작 정지가 전체 시스템에 미치는 영향을 적게 함으로써 신뢰도를 높일 수 있고, 여러 시스템을 동시에 활용할 수 있으므로 한 개의 대형 시스템을 활용하는 것보다 저렴한 비용으로 보다 나은 성능을 얻을 수 있다. 또한 시스템 확장이 필요한 경우 단순히 새로운 시스템을 분산 처리 환경에 추가하고 분산 처리 환경을 간단히 재구성함으로써 쉽게 목적을 달성할 수 있다[1].

1990년대 후반에 이르러서 떠오르는 분산 객체 처리 기술은 분산 처리 기술에 객체 지향 개념을 도입하여 분산된 자원들을 객체화 시켜 객체들 사이의 상호 운용성을 제공하는 것으로 분산 처리의 장점에 상속, 다양화, 캡슐화, 재사용 등의 객체 지향 기술의 장점을 동시에 지닌 새로운 기술로 주목 받고 있다.

분산 객체 처리 기술과 지식을 가진 SunSoft, HP, IBM, DEC 등의 회사와 사용자, 그리고 S/W 제공자들로 구성된 표준화 기구인 OMG (Object Management Group)는 분산 객체 처리를 위한 기본 구조로 응용 객체, 공용 기능 (Common Facilities), 객체 서비스 (Object Service), 그리고 객체 요구 중개자 (ORB: Object Request Broker)로 구성된 객체 관리 구조 (OMA: Object Management Architecture)[2, 3]를 정립하여 제안하였다.

CORBA는 이기종 환경에서 응용 프로그램간의 상호 운용성과 다중 객체 시스템의 상호 연결성을 제공하기 위해 규정된 객체 요구 중개자의 표준 구조로 분산된 객체들 사이의 투명한 요청과 응답을 제공해 주는 분산 객체 처리를 지원해 준다.

그러나 분산 객체 환경은 분산 처리 환경이 가지고 있는 보안 상의 문제점을 그대로 가지고 있다. 분산 처리 환경에서의 보안 문제를 해결하기 위해서는 Kerberos[4, 5], OSF/DCE (Open Software Foundation/Distributed Computing Environment) Security[6], ECMA (European

Computer Manufacturers Assocaition) SESAME (a Secure European System for Application in a Multivendor Environment)[7] 등과 같은 보안 기술이 개발되었다.

분산 객체 환경을 위협하는 요소들로는 인가받지 않은 자료로의 정보 접근, 합법적인 사용자로의 위장, 우회되어지는 보안 통제, 통신망 상의 자료의 불법적 도청, 객체들 사이에 전송되는 자료의 변조 등이 있다. 응용 서비스들은 위의 위협 요소들로부터 보호되어야만 신뢰성 있고 가용한 서비스를 제공할 수 있다. 특히 금융 서비스 등과 같은 보안에 민감한 서비스의 경우에는 더욱 그러하다.

OMG에서는 OMA를 만족시키며 기밀성, 무결성, 유용성 등과 같은 보안 특성을 제공하고 응용 객체들에게 보안 투명성을 제공하는 것을 기본으로 하는 CORBA 보안 서비스[8]를 제시하였다. CORBA 보안 서비스를 구현함으로써 분산 객체 응용 개발자들이 보안에 대한 부담 없이 응용 서비스를 개발할 수 있고 기존에 개발된 응용들도 수정 없이 보안 서비스를 적용 받을 수 있다.

CORBA 보안 서비스 규격은 보안 객체들이 GSS-API (Generic Security Service Application Programming Interface)[9, 10, 11]와 같은 표준 인터페이스를 사용하여 특정 보안 기술에 독립적으로 보안 서비스를 제공하도록 하고 있다. 즉 GSS-API를 제공하는 SESAME, DCE Security, Kerberos 등과 같은 제 3 자 보안 소프트웨어를 사용하여 CORBA 보안 서비스를 구현할 수 있다는 것이다.

DCE Security, SESAME 등과 같이 이미 개발 검증된 보안 기술을 이용하여 CORBA 보안 서비스를 구현하면 빠르고 효과적으로 CORBA 보안 서비스를 제공할 수 있다. CORBA 보안 서비스 규격도 보안 서비스를 제공하는 각 보안 객체들이 사용하는 보안 메커니즘과 같은 보안 기술에 관해서는 언급하지 않고 단지 각 보안 객체가 제공하는 인터페이스만 명시하고 있다.

본 논문에서는 ORB 환경에서 수행하는 응용들 사이에 전송되는 메시지들의 기밀성과 무결성을 제공하기 위해 SESAME에서 제공하는 GSS-API를 이용하여 CORBA 보안 서비스를 구현하였다.

SESAME는 ECMA에서 제안한 보안 플랫폼으로 분산 응용 표준 인터페이스인 GSS-API와 Kerberos에 기반을 둔 사용자 인증을 제공하고 있다. SESAME에 관한 자세한 내용은 II장에서 기술하였다.

본 논문의 구성은 II장에서 CORBA 보안 서비스와 SESAME를 살펴보았다. 그리고 III장에서 CORBA 보안 서비스 설계 및 구현에 관해 설명하고 IV장에 시험 항목과 결과를 그리고 V장에 결론을 기술하였다.

II. 관련 연구

CORBA 보안 서비스를 보안 기술에 독립적으로 구현하기 위해서는 분산 환경에서의 표준 보안 인터페이스인 GSS-API를 사용하여야 한다. GSS-API를 제공하는 여러 보안 기술들이 존재하지만 II 장에서는 본 논문에서 사용한 보안 기술인 SESAME에 대해 살펴본다.

1. CORBA 보안 서비스

CORBA 환경에서 수행되는 응용 객체들에게 보안 서비스를 제공하기 위해서 CORBA 보안 서비스는 (그림 1)과 같은 보안 객체 모델을 제시하였다.

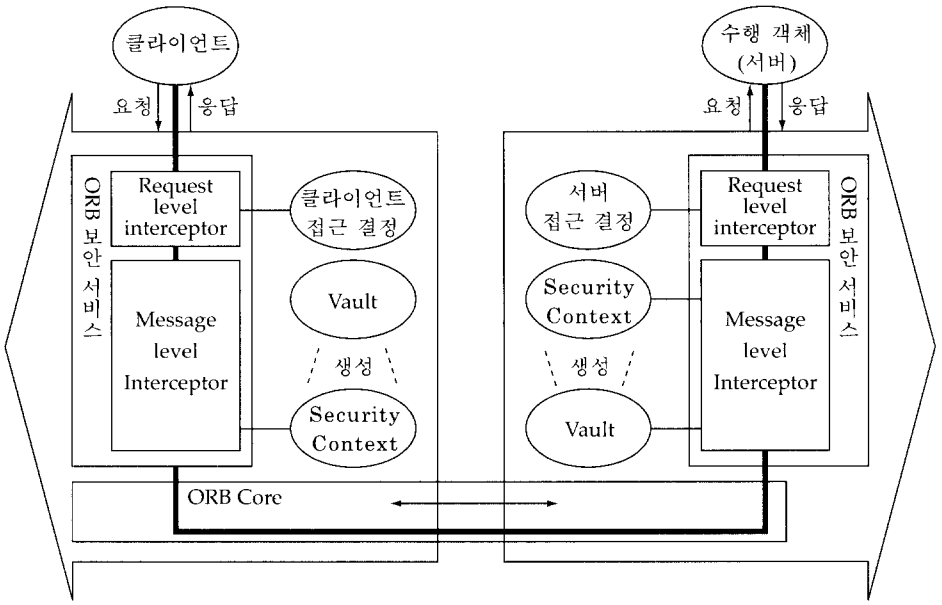


그림 1. 구현 측면의 ORB 보안 객체 모델
Fig. 1. ORB Security Object Model

객체 호출 시 수행되는 보안 기능은 ORB core 위에 ORB 보안 서비스와 보안 서비스 객체들로 존재하며 접근제어와 보안 컨텍스트 (Security Context)를 통해 보안 호출이 수행된다. 보안 서비스를 제공하기 위해 ORB에 추가로 필요한 구현 측면의 보안 객체는 클라이언트와 서버 접근 결정, 보안 컨텍스트를 초기화 하는 Vault, 전송되는 메시지에 보안 매커니즘을 적용하는 보안 컨텍스트 등이 있다. 접근 결정 객체는 클라이언트가 서버로 접근할 수 있는지를 판단하는 역할을 한다. Vault 객체는 클라이언트와 서버 사이에 보안 세션을 초기화 하는 일을 수행한다. 보안 컨텍스트 객체는 클라이언트와 서버 사이의 보안을 책임진다.

(그림 1) 같이 시스템에서 제공하는 ORB 보안 서비스는 클라이언트 또는 서버와 ORB core 사이에 위치하여 모든 응용들에게 기본적인 접근제어와 보안 세션 설정 및 메시지 보호에 대한 서비스를 제공한다. 따라서 ORB 환경의 응용 객체들에게 보안 서비스를 제공하기 위해서는 접근 결정, Vault, 보안 컨텍스트 객체 등이 기본적으로 구현 제공되어야 한다.

CORBA 보안 서비스 규격은 보안 기능 수준을 기능 수준 1과 기능 수준 2로 구별하였다. 보안 기능 수준 1은 보안 시스템을 구성하기 위해 필요한 기본적인 구성을 설명하고 있으며, 보안 기능 수준 2는 보안 영역과 보안 정책을 관리하고 지정할 수 있는 다양한 인터페이스를 제공하도록 확장된 것이다.

2. ECMA SESAME

ECMA는 1987년에 개방형 시스템에서의 보안에 관해 작업하기 시작하였으며 이 작업의 결과물로 SESAME를 발표하였다. SESAME Ver. 2와 Ver. 3이 시험 프로젝트로 개발되었으며 Ver. 4가 정식으로 공표되었다. 본 논문에서는 SESAME Ver. 4를 이용하였다.

SESAME는 보안이 보장된 개방형 분산 시스템을 구축하기 위해 응용 계층에 중점을 둔 보안 시스템으로 인증, 접근제어, 데이터 무결성, 데이터 기밀성, 부인 봉쇄 등을 지원하며 암호화를 최소로 사용하도록 하였다. 즉 클라이언트와 서버 사이에 전달되는 보안 관련 제어 데이터에 대해서만 암호화하도록 하였으며 사용자 데이터에 대해서는 선택적으로 암호화가 가능하도록 하였다. 또한 Kerberos에서 제시한 인증 기법을 확장하여 사용하였고 분산 환경에서의 표준 보안 인터페이스인 GSS-API를 제공한다. SESAME에서는 데이터를 보호하기 위한 기법으로 Kerberos 키 분배 규약을 이용하였으며, 공개키 암호 방식도 사용할 수 있도록 하였다. 또한 X.509 디렉토리 사용자 신임장을 지원한다.

2.1. GSS-API

보안 서비스를 위한 일반적인 인터페이스인 GSS-API는 IETF (Internet Engineering Task Force) 표준 규격으로 응용이 보안 메커니즘과 통신 프로토콜에 독립적으로 보안 서비스를 제공받을 수 있도록 보안에 관한 일반적인 인터페이스를 제공한다. 즉 GSS-API는 무결성, 기밀성, 인증 등과 같은 보안 서비스를 위한 인터페이스를 제공함으로써 보안 세션을 요구하는 클라이언트/서버 구조의 분산 응용 프로그램을 지원한다.

SESAME 에서 제공하는 GSS-API 는 <표 1>과 같다.

〈표 1〉 SESAME에서 제공하는 GSS-API
<Table 1> SESAME GSS-API

	인터페이스	설 명
인증서 handing	Gss_acquire_cred	사용을 위해 인증서를 얻는다.
	Gss_inquire_cred	인증서에 관한 정보를 display 한다.
	Gss_release_cred	사용한 인증서를 discard 한다.
보안 컨텍스트 handling	Gss_init_sec_context	상대 응용에게 전송될 보안 컨텍스트를 수용한다.
	Gss_accept_context	상대 응용에게서 전송받은 보안 컨텍스트를 수용한다.
	Gss_delete_sec_context	보안 컨텍스트를 지운다.
	Gss_process_context_token	상대 응용으로부터 전해진 보안 턴텍스트에 있는 제어토큰을 처리한다.
	Gss_context_time	컨텍스트 유효 시간을 reset한다.
응용 사이에 전송되는메시지 보호	Gss_get_mic, Gss_sign	전송할 메시지에 분리된 무결성 check value를 계산한다.
	Gss_verify_mic, Gss_verify	메시지를 따라온 check value를 가지고 무결성을 점검한다.
	Gss_wrap, Gss_seal	메시지 무결성을 위해 메시지에 sign한다. 메시지 기밀성을 위해 선택적으로 암호화한다.
	Gss_unwrap, Gss_unseal	메시지의 무결성을 점검하고 필요하다고 복호화한다.
기타 인터페이스	Gss-display_status	API 상태 코드를 텍스트 형태로 바꿔준다.
	Gss-indicate_mechs	사용된 인증 기법을 알려준다.
	Gss-compare_name	내부 형식의 이름을 비교한다.
	Gss_display_name	내부 형식의 이름을 text형태로 바꾸어 준다.

	인터페이스	설 명
기타 인터페이스	Gss_display_name	내부 형식의 이름을 text형태로 바꾸어 준다.
	Gss_import_name	텍스트 형태의 이름을 내부 형식의 이름으로 바꾼다.
	Gss_release_name	내부 형식의 이름을 지운다.
	Gss_release_buffer	사용된 버퍼를 지운다.
	Gss_release_oid_set	객체 식별자의 집합을 놓아준다.
SESAME에서 추가한 GSS_API	Gss_modify_cred	인증서와 관련된 권한 특성과 제어 정보를 변경한다.
	Gss_get_attributes	보안 컨텍스트 또는 인증서의 특성을 얻어온다.
	Gss_compound_cred	두개의 인증서를 하나로 만든다.
	Gss_get_delegate_creds	보안 컨텍스트 또는 인증서 리스트에서 위임 인증서를 찾아온다.
	Gss_set_default_cred	Alternative 인증서를 default 인증서로 설정한다.

III. 설계 및 구현

III 장에서는 본 논문에서 설계 구현한 CORBA 보안 서비스에 대해 기술하였다. 1절에서 시스템 동작 및 구조를 설명하였다. 그리고 2 절에 구현 내용을 기술하였다.

1. 시스템 동작 및 구조

1.1. 기본 구조

본 논문에서는 CORBA 보안 서비스 규격에 명시된 보안 기능 수준 1을 만족하는 보안 객체들을 구현하는 것을 목적으로 하였다.

CORBA 보안 서비스는 보안 기능을 제공하는 보안 객체들로 구성되어 있으며 이러한 보안 객체 들은 GSS-API를 제공하는 제 3자 보안 소프트웨어를 이용하여 구현할 수 있다. 이러한 구현 방법은 이미 검증된 보안 소프트웨어를 사용하기 때문에 빠르고 효과적으로 보안 서비스를 제공할 수 있다.

GSS-API를 제공하는 보안 소프트웨어로는 Kerberos, DCE Security, SESAME 등이 있다. 이들 중 특히 SESAME는 공개키 기반 보안 소프트웨어로 공유키 기반 보안 소프트웨어인 Kerberos나 DCE Securiy에 비해 고도의 보안 기술을 제공한다. 따라서 SESAME에서 제공하는 GSS-API를 사용하여 CORBA 보안 객체를 구현하고 SESAME의 보안 서버를 이용하여 CORBA 보안 환경을 구축하면 Kerberos나 DCE Security를 사용하는 것보다 고도의 보안 기술을 취할 수 있다.

이에 따라 본 논문에서는 제3자 보안 소프트웨어로 SESAME를 사용하였다. 그러나 GSS-API를 이용하여 보안 서비스를 구현하였기 때문에 필요에 따라 Kerberos나 DCE Security로 보안 소프트웨어를 대치할 수도 있다.

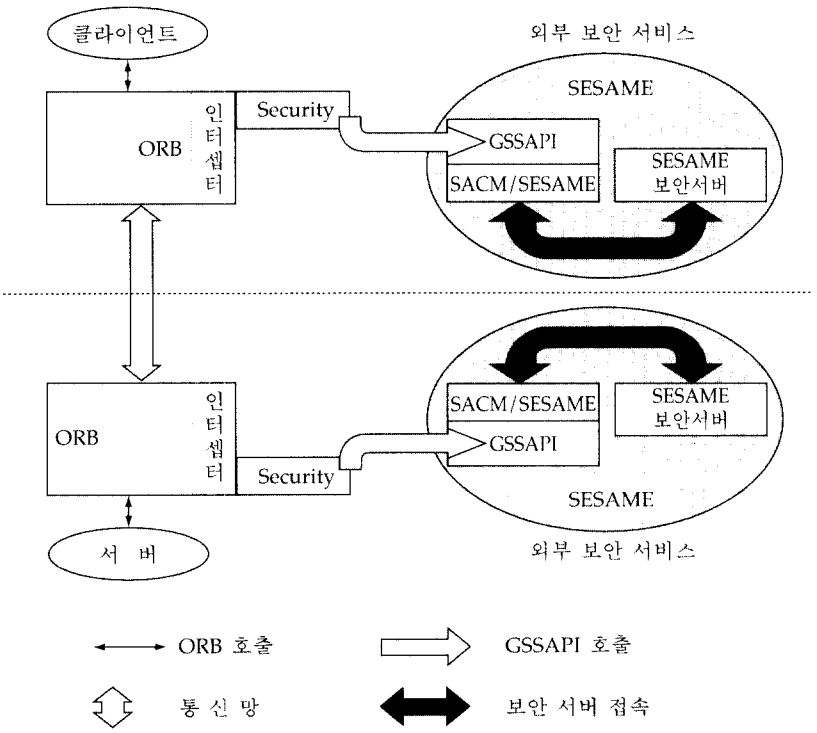
(그림 2)는 본 논문에서 구현한 CORBA 보안 서비스의 개략 구조를 보여 준다. GSS-API를 이용한 보안 서비스 구현은 ORB core에 존재하는 인터셉터 계층을 통해 이루어진다. 인터셉터 계층은 ORB core에 존재하는 것으로 ORB로 전송되는 응용의 요청, 응답 메시지들을 가로채어 보안 서비스를 제공 받을 수 있게 해주는 계층이다. GSS-API를 이용한 보안 서비스를 인터셉터 계층 뒤에 두는 구조는 보안 서비스의 대치 가능성을 최대한 보장한다.

1.2. 시스템 구조 및 동작

CORBA 보안 서비스 규격은 보안 서비스로 사

용자 인증, 접근제어, 보안 감사, 보안 정책 관리, 데이터 기밀성, 데이터 무결성, 신임장 위임, 부인봉쇄 서비스를 정의하고 이들을 제공하기 위해 보안 객체들을 정의하고 있다.

사용자 인증, 접근제어, 보안 감사, 보안 정책 관리 등의 서비스는 제 3자 보안 소프트웨어에서 제공하는 기능을 그대로 사용하여 CORBA 보안 서비스 환경을 구축할 수 있다. 그러나 SESAME와 같은 제 3 자 보안 소프트웨어들은 응용 개발자들이 응용을 작성할 때 보안 서비스 인터페이스를 사용하여 데이터를 보호한 후 이를 소켓과 같은 통신 인터페이스를 사용하여 상대방에게 전송하는 구조를 지원한다. 따라서 ORB를 통신 채널로 사용하지 않는 제 3자 소프트웨어는 ORB를 통해 전송되는 메시지들에게 기밀성과 무결성을 제공할 수 없다. ORB를 통해 전송되는 메시지들의 투명한 보안 통신을 보장하기 위해서는 CORBA 보안 서비스를 위한 보안 객체들이 제공되어야만 한다.



SACM:Secure Association Context Manager

그림 2. 전체 구조
Fig. 2. Overall Architecture

본 논문에서 제시한 CORBA 보안 서비스의 전체 동작 구조는 (그림 3)과 같다.

보안 통신은 보안 세션 설정 단계와 보안 데이타 교환의 단계로 이루어진다. CORBA 보안 서비스에서 보안 통신을 제공하는 객체로는 Vault 객체와 보안 컨텍스트 (Security Context) 객체가 있다. Vault 객체는 보안 세션 설정을 수행하며 보안 컨텍스트 객체는 메시지에 보안 기술을 적용하여 메시지의 보안 통신을 보장한다.

응용들 사이의 보안 통신을 적용받고자 하는

사용자는 우선 사용자 인증 절차를 거친다. 인증 절차를 거친 사용자는 신임장 캐쉬에 인증 증명서인 신임장을 저장하고 있다. 신임장은 클라이언트 응용과 서버 응용 사이의 보안 세션 설정에 이용되는 것으로 사용자 식별자, 이용 권한 등의 정보를 가지고 있다. 클라이언트 응용은 신임장 캐쉬에 있는 신임장을 이용하여 서버 응용에게 통신 세션 설정을 요구한다. 이 요구는 보안 세션 설정 초기화를 담당하는 Vault 객체를 통해 이루어진다. 보안 세션 설정을 요구받은 Vault 객체는 보안 통신을 위해 필요한 보안 컨텍스트 객체를 생성하여 보안 컨텍스트 풀에 저장한다. 그리고 생성된 보안 컨텍스트 객체 정보를 서버에게 전송한다. 서버 ORB는 클라이언트로부터 전달받은 정보를 자신의 Vault 객체를 통하여 해석하여 클라이언트 쪽 보안 컨텍스트 객체와 쌍을 이루는 보안 컨텍스트 객체를 생성한다. 생성된 보안 컨텍스트 객체는 보안 컨텍스트 풀에 저장된다.

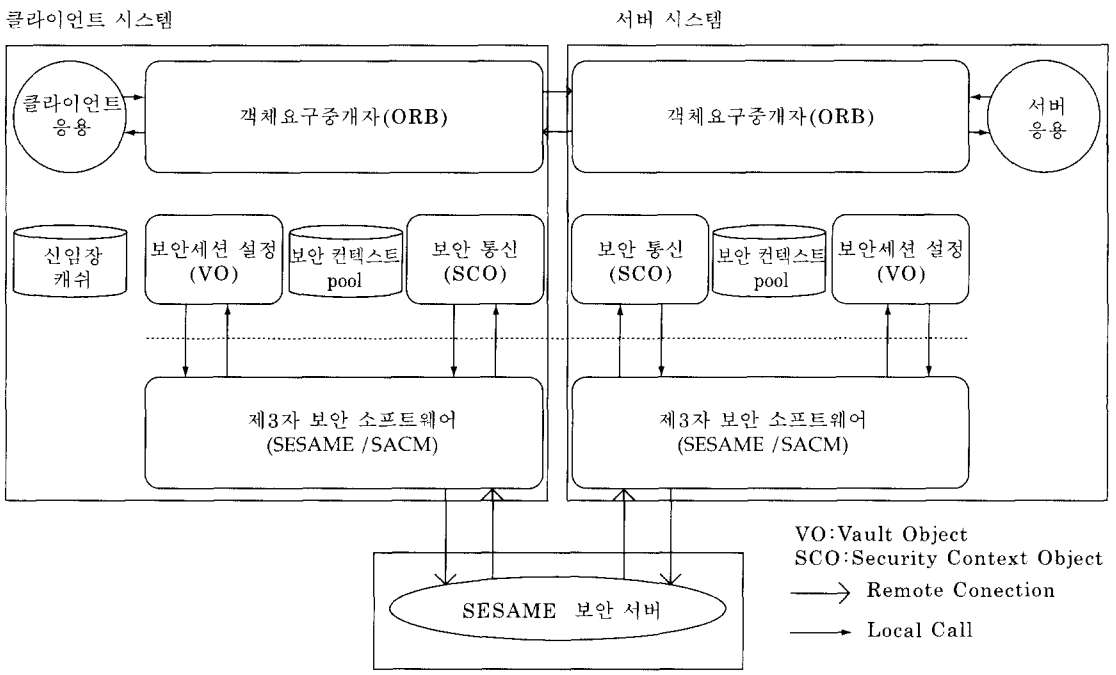


그림 3. 전체 동작 구조
Fig. 3. Operational Archutecture

클라이언트와 서버 양 쪽에 동일한 보안 컨텍스트 객체가 설치되면 보안 세션 설정이 완료된다. 보안 세션이 설정되면 클라이언트 응용과 서버 응용 사이에서 전달되는 메시지는 양쪽에 설치된 보안 컨텍스트 객체에 의해 보안 기술을 적용 받는다.

본 논문에서는 Vault 객체, 보안 컨텍스트 객체를 SESAME에서 제공하는 GSS-API를 사용하여 구현하였다. 이 방법은 SESAME에서 제공하는 보안 기법을 이용하여 CORBA 보안 서비스를 제공하는 것이다.

보안 세션 설정 시 필요한 정보인 사용자 신임장은 SESAME에서 제공하는 인증 방법에 따라 발급되는 신임장을 그대로 사용하였다. 사용자 인증의 결과인 신임장은 PAC (Privilege Attribute Cerificate) 정보를 유지한다. PAC의 구

조는 (그림 4)와 같다.

접근제어는 SESAME에서 제공하는 push 방법을 그대로 사용하였다. 보안 세션 설정을 요구 받은 서버쪽 SESAME는 접수한 보안 컨텍스트 객체 정보를 해석하여 서버 응용으로의 접근 권한이 있는지를 확인한다. 서버 응용으로의 접근이 허용된 경우에 한해 보안 세션 설정을 허락하는 것이다. 또한 보안 감사, 보안 정책 관리 등의 보안 서비스는 SESAME에서 제공하는 기능을 그대로 이용한다.

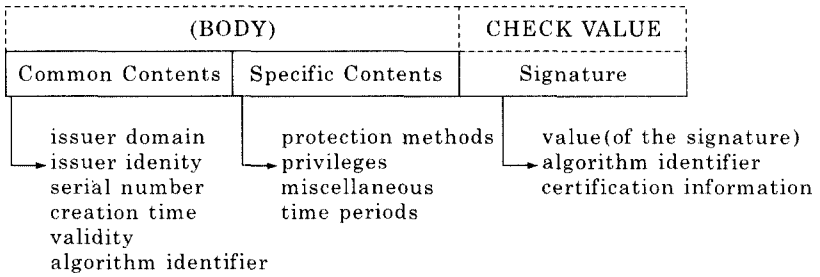


그림 4. Sesame PAC 구조
Fig 4. Structure of PAC (Privilege Attribute Certificate) of SESAME

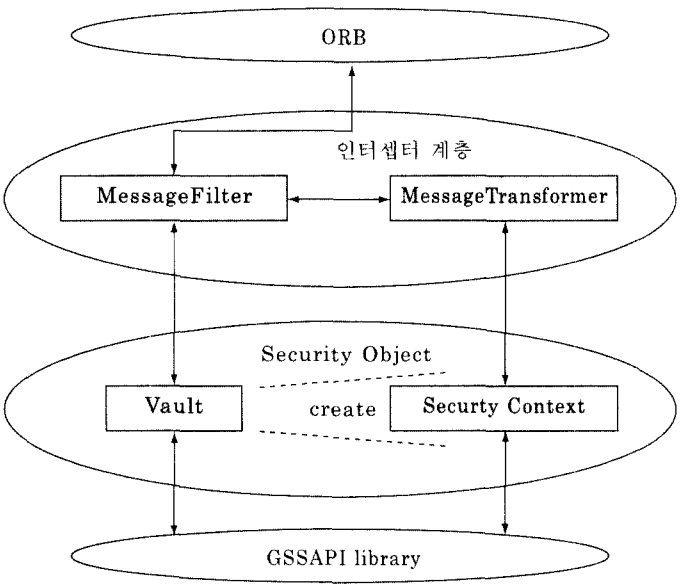


그림 5. 보안 서비스 제공 Class들 사이의 관계도
Fig. 5. Relationship between Security Service Class

2. 구현

CORBA 보안 서비스 구현 환경은 아래와 같다.

- 운영체제 : Solaris 2.5
- ORB : IONA Orbix 2.1 MT
- 제 3 자 보안 소프트웨어 : SESAME Ver.4

보안 기능 수준 1을 만족하는 응용에게 투명한 보안 서비스를 제공하기 위해서는 ORB에서 인터

셉터를 제공하여야 한다. 그러나 구현 환경으로 택한 IONA Orbix 2.1 MT[12]는 CORBA ver 2.0[13]을 기반으로 한 ORB 제품으로 인터셉터를 제공하지 않는 대신에 Filter와 Transformer[12]라는 인터셉터와 유사한 기법을 제공하고 있다. Filter은 CORBA의 Request level 인터셉터와 유사하며 Transformer는 Message level 인터셉터와 유사한 기능을 제공한다.

본 논문에서는 Orbix의 Filter와 Transformer를 상속받아 MessageFilter와 Message Transformer라는 보안 인터셉터를 구현하였다.

CORBA 보안 서비스를 제공하기 위해 구현된 Class들은 다음과 같으며 (그림 5)는 이들 사이의 관계를 보여준다.

- MessageFilter : 통신 주체간에 보안 세션을 초기화하기 위해 전송되는 Request를 가로채어 Vault객체에게 넘겨준다.
- MessageTransformer : 전송되는 메시지를 가로채어 보안 통신을 보장하기 위해 SecurityContext 객체에게 전달한다.
- Vault : 통신 양 주체에 보안 컨텍스트를 설치하여 보안 세션 설정을 초기화 한다.
- SecurityContext : 보안 정책을 반영하여 설치된 보안 컨텍스트를 반영하여 전송되는 메시지들에 보안 서비스를 적용한다.

2.1. 수행 절차 및 흐름

응용들 사이의 보안 통신을 제공하기 위해 필요한 보안 세션을 설정하는 과정은 (그림 6)과 같고 설정된 보안 세션을 이용하여 보안 통신을 수행하는 과정은 (그림 7)과 같다.

보안 세션을 통하여 응용들 사이에서 보안 통신은 우선 응용 클라이언트와 응용 서버간에 보

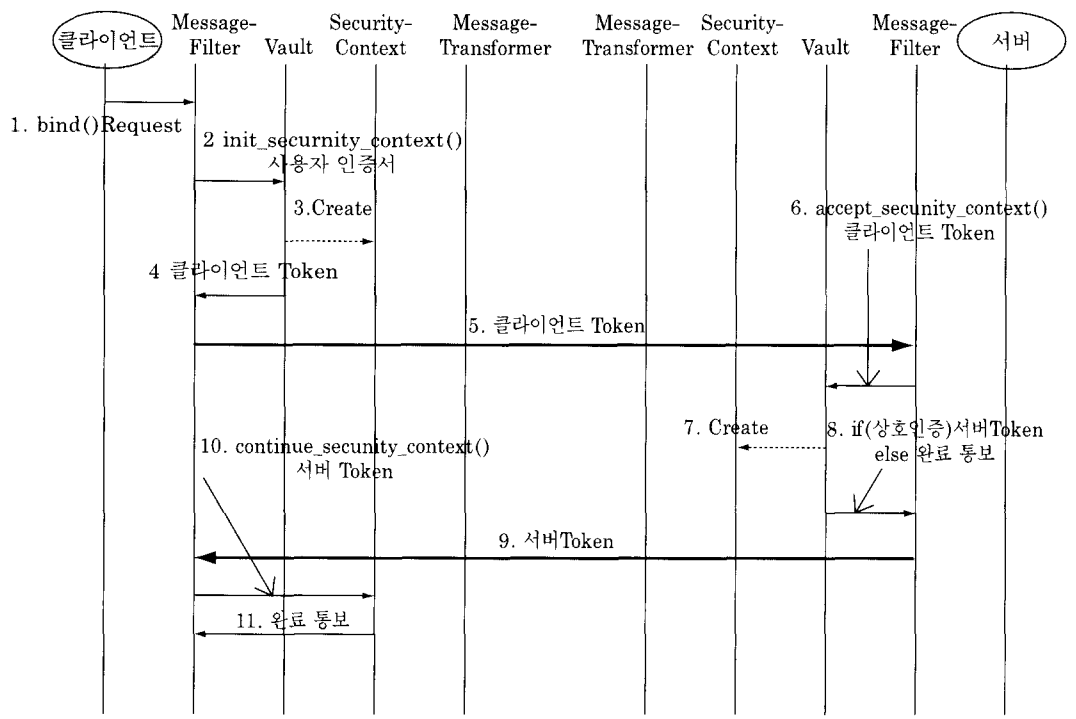


그림 6. 보안 세션 설정 절차
Fig. 6. Procedure of Secure Session Establishment

안 세션을 설정하는 것부터 시작한다. 보안 세션 설정 과정은 다음과 같다.

1. bind() request 를 ORB를 통해 응용서버에게 전송한다.
2. MessageFilter가 bind() request를 가로챈다. MessageFilter는 사용자 신임장을 가지고 Vault 의 init_security_context()를 호출하여 SecurityContext 객체를 초기화를 요구한다. 이 때 Vault 객체는 GSS-API 라이브러리의 Gss_init_sec_context()를 호출하여 보안 컨텍스트를 초기화하고 보안 세션에 필요한 보안 정보를 가진 클라이언트 토큰을 생성한다.
3. GSS-API 라이브러리에 의해 초기화된 보안 컨텍스트를 반영하여 SecurityContext 객체를 생성한다.
4. MessageFilter에게 클라이언트 토큰을 전달한다.
5. MessageFilter은 ORB를 통하여 서버쪽의 MessageFilter에게 클라이언트 토큰을 전송한다.
6. MessageFilter가 보안 토큰을 가지고 Vault의 accept_security_context()를 호출하여 서버쪽의 SecurityContext를 초기화 한다. Vault 객체는 GSS-API 라이브러리의 Gss_accept_sec_context()를 호출하여 클라이언트 토큰을 해석하여 이 정보를 가지고 보안 컨텍스트를 초기화한다.
7. GSS-API 라이브러리에 의해 초기화된 보안 컨텍스트를 참조하여 SecurityContext 객체를 생성한다.
8. 클라이언트와 서버사이의 상호 인증인 경우 Vault 객체는 서버 토큰을 MessageFilter에게 전송한다. 그렇지 않은 경우에는 완료 통보를 MessageFilter에게 전송하여 보안 세션 설정을 끝낸다.
9. 상호 인증인 경우 MessageFilter는 서버 토큰을 ORB를 통해 클라이언트 MessageFilter에게 전달한다.

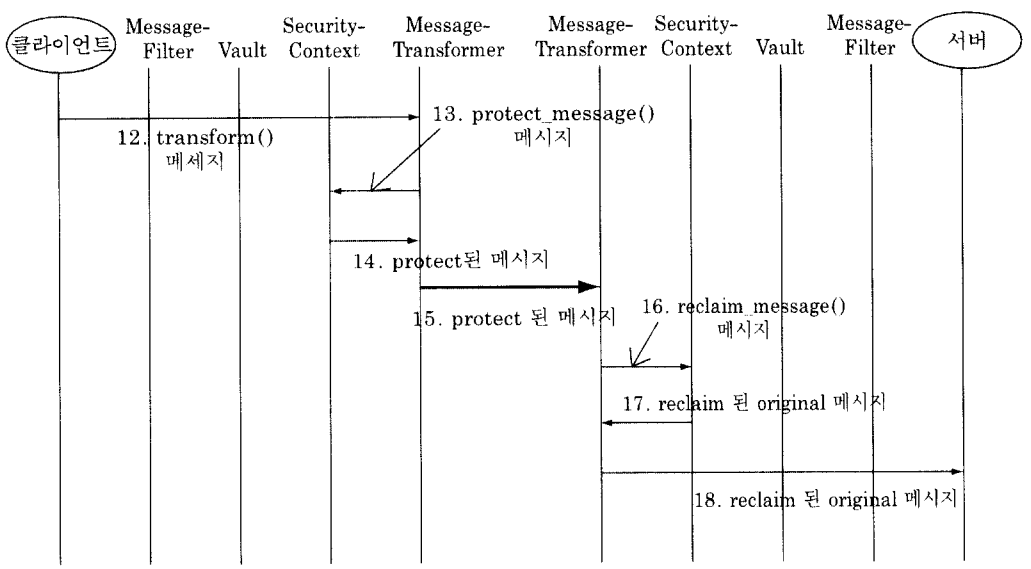


그림 7. 보안 통신 절차
Fig. 7. Procedure of Secure Communication

10. 클라이언트 MessageFilter은 서버 토큰을 가지고 SecurityContext 객체의 continue_security_context()를 호출하여 서버를 인증한다. Vault 객체는 GSS-API 라이브러리의 Gss_init_sec_context()를 호출하여 서버 토큰을 해석하여 서버를 인증한다.
11. MessageFilter에게 완료 통보를 보내 보안 세션 설정을 완료한다.

보안 세션 설정이 완료되면 두 응용 사이에 전송되는 자료들은 보안 통신을 적용 받게 된다. 보안 통신을 수행하는 과정은 다음과 같다.

12. 응용 클라이언트는 ORB를 통해 응용 서버에게 자료를 전송한다.
13. ORB는 응용 서버로 전달되는 자료를 가지고 MessageTransformer의 transform()을 호출하여 자료의 변형을 요구한다.
14. MessageTransformer는 SecurityContext 객체의 protect_message()를 호출하여 자료의 보호를 요청한다. SecurityContext 객체는 SecurityContext에 반영된 보안 호출 정책에 따라 자료를 보호한다. 무결성 보장 정책인 경우 GSS-API 라이브러리의 Gss_sign()을 호출하여 자료에 서명한다. 기밀성 보장 정책인 경우는 GSS-API 라이브러리의 Gss_seal()을 호출하여 자료를 암호화한다. 기밀성, 무결성 모두를 보장하는 보안 정책인 경우는 Gss_sign()을 호출하여 서명된 자료를 다시 Gss_seal()로 암호화한다.
15. 14단계에서 변조된 자료를 응용 서버의 MessageTransformer에게 전달한다.
16. MessageTransformer는 SecurityContext 객체의 reclaim_message()를 호출하여 자료의 원상 복구를 요청한다. SecurityContext 객체는 SecurityContext에 반영된 보안 정책에 따라 자료의 원상 복구를 수행한다. 무결성 보장 정책인 경우 GSS-API라이브러리의 Gss_verify()를 호출하여 자료의 무결성을 점검한다. 기밀성 보장 정책인 경우는 GSS-API 라이브러리의 Gss_unseal()을 호출하여 자료를 복호화한다. 기밀성, 무결성 모두를 보장하는 보안 정책인 경우는 Gss_unseal()을 호출하여 자료를 복호화한 후 복호화된 자료를 Gss_verify()로 무결성 점검을 한다.
17. 원상 복구된 자료를 응용 서버에게 전달한다.
18. 응용 서버에서 응용 클라이언트로의 응답 자료는 응용 클라이언트에서 응용 서버로의 자료 전송의 역으로 수행된다.

2.2. 보안 컨텍스트 관리

응용 클라이언트와 응용 서버 사이에서는 다자간 보안 통신이 가능하여야 한다. 이를 위해 보안 컨텍스트 풀을 유지 관리한다. 보안 통신을 요구하는 응용은 상대 응용과 대응되는 보안 컨텍스트를 보안 풀에서 찾아 보안 통신에 이용한다. 보안 컨텍스트 풀의 구조는 (그림 8)과 같다.

보안 컨텍스트 풀은 보안 통신의 주체인 응용별로 하나씩 생성 관리되며 각 풀은 보안 컨텍스트 리스트를 유지한다. 보안 컨텍스트 풀의 소유주가 응용 클라이언트인 경우 보안 컨텍스트의 세션 Id로 통신 상대 응용 서버의 이름을 사용하며, 응용 서버의 경우에는 보안 통신 상대의 process Id와 호스트 이름(또는 호스트 Id)를 조합하여 보안 컨텍스트의 세션 Id로 사용한다. 예를 들어 응용 클라이언트 I이 응용 서버 I, II와 다자간 보안 통신을 하고 있다고 가정하자. 응용 클라이언트 I은 자신의 보안 컨텍스트 풀을 가지고 있으며 이 풀에는 응용 서버 I, II와의 보안 통신을 위해 필요한 2 개의 보안 컨텍스트를 유지하고 있을 것이다.

이 경우 응용 클라이언트I은 응용 서버 I과의 통신을 위해 생성한 보안 컨텍스트의 세션 Id는

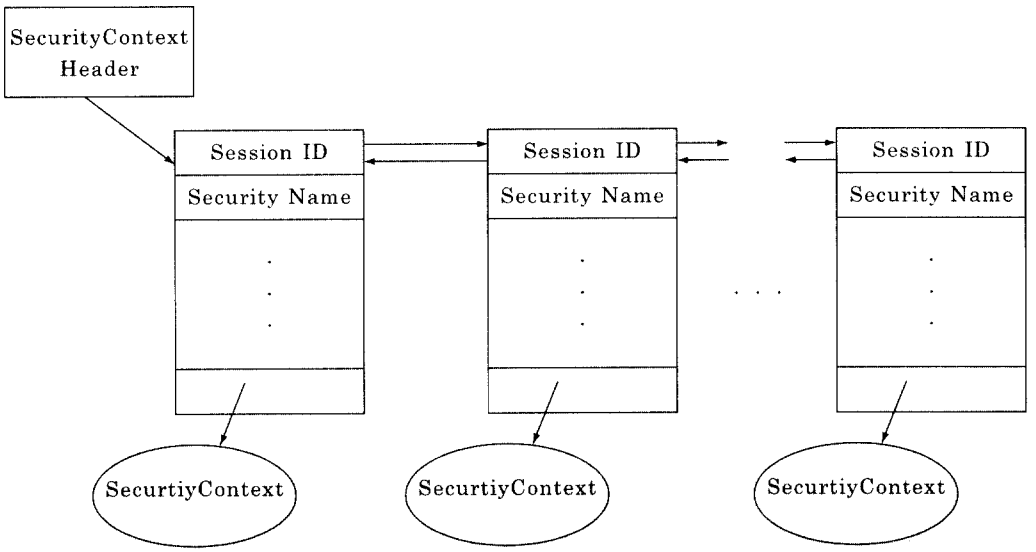


그림 8. 보안 컨텍스트 풀 구조
Fig. 8. Structure of Security Context Pool

응용 서버 I의 이름을 사용하고, 응용 서버 II와의 통신을 위해 생성한 보안 컨텍스트의 세션 Id는 응용 서버 II의 이름을 사용한다. 응용 서버 I이 유지하고 있는 응용 클라이언트 I과의 보안 통신을 위한 보안 컨텍스트의 세션 Id는 응용 클라이언트가 수행 중인 호스트 이름과 응용 클라이언트의 process Id의 조합이며, 응용 서버 II의 경우도 동일하다.

IV. 시 험

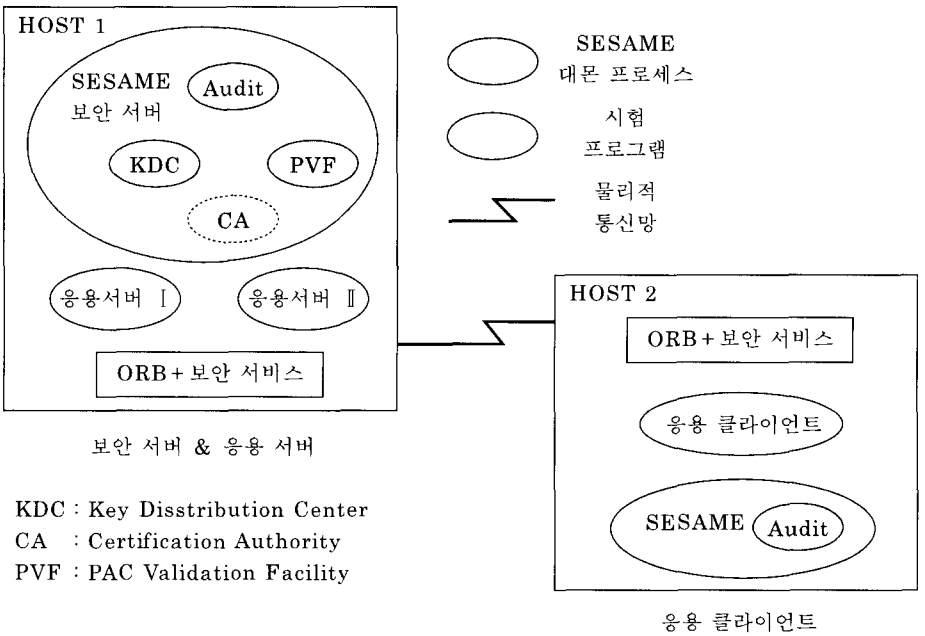
사용자 인증 기능, 응용에 대한 사용자 접근제어 기능, 그리고 응용의 보안 정책 정보 및 사용자 정보 관리 기능 등은 제 3자 보안 소프트웨어인 SESAME에서 제공하고 있다. 따라서 본 절에서는 클라이언트 응용과 서버 응용 사이에 적절한 보안 세션이 설정되며 설정된 보안 세션을 통해 유효한 보안 통신이 이루어지는지에 대한 시험을 소개한다.

1. 시험 항목

- 클라이언트와 응용 서버 사이에 보안 세션 설정 확인 : 클라이언트와 응용 서버 각각에 보안 컨텍스트가 설치되었는지 확인
- 설정된 보안 세션을 통해서 상호 교환되는 데이터의 비밀성 및 무결성이 보장되는지 확인
- 다수의 클라이언트와 다수의 응용 서버들 사이에서 정확하게 각각의 보안 호출 정책에 따라서 보안 컨텍스트가 생성되는지 확인

2. 시험 환경

시험을 수행하기 위해 통신망으로 두 대의 워크스테이션을 연결하여 물리적으로 기본적인 분산 시스템 환경을 구성하였다. 시험 환경의 논리적 구성은 HOST 2에 응용 클라이언트, HOST 1에 응용 서버와 SESAME가 제공하는 보안 서버가 위치하도록 구성하였다. 또한 응용 클라이언트와



(그림 9) 시험 구성도
(Fig. 9) Configuration of Test Environment

서버 시스템은 각기 기본 ORB로 IONA Orbix 2.1cMT 버전을 설치하였다.

3. 시험 시나리오

시험을 위한 응용 프로그램은 (그림9)와 같이 두 개의 응용 서버와 하나의 응용 클라이언트로 이루어진다. 시험을 수행할 사용자의 ID는 이미 SESAME의 사용자로 등록되어 있고, 응용 서버 I, II는 SESAME와 Orbix의 응용 서버로 등록되어 있어야만 시험이 가능하다.

시험 절차는 우선 HOST 1에서 응용 서버 I, II를 수행 시킨다. 그리고 사용자 ID를 가지고 HOST 2에서 사용자 인증 과정을 거친 후에 응용 클라이언트를 수행시킨다. 응용 클라이언트는 HOST 1에 존재하는 응용 서버 I, II가 가지고 있는 자료를 요청하여 그 결과를 가져와 클라이언트 시스템 화면에 출력한다.

하나의 응용 클라이언트와 하나의 응용 서버, 하나의 응용 클라이언트와 두 개의 응용 서버를 가지고 다자간 보안 통신을 시험한다. 그리고 클라이언트와 서버 사이의 보안 통신을 위한 보안 호출 정책을 무결성으로 설정하고 시험하고, 무결성과 기밀성을 모두 설정한 후 시험한다.

4. 시험 수행 및 결과

기능 시험 수행 항목과 수행 결과는 <표 2>와 같다.

〈표 2〉 기능 시험 수행 항목 및 수행 결과
<Table 2> Test Item and Result

시험 방법	예상 결과	수행 결과
응용 서버 I 과 클라이언트 응용 프로그램 간의 1:1 보안 컨텍스트 설치 확인 응용 클라이언트와 서버 각각에 생성된 보안 컨텍스트 객체의 식별자로 확인	보안 컨텍스트 설치 성공	보안 컨텍스트 설치 성공
보안 호출 정책에 무결성을 설정하고 응용 클라이언트와 응용 서버에서 각기 메시지를 송부하기 전에 보낼 메시지의 내용을 표준 출력 장치에 입력하고 그 내용을 확인	출력되는 문자열에 무결성 보장을 자료가 첨부되어 전송	출력되는 문자열에 무결성 보장을 위한 자료가 첨부되어 전송(시험완료)
보안 호출 정책에 무결성을 설정하고 응용 클라이언트와 응용 서버에서 각기 메시지를 송부하기 전에 보낼 메시지의 내용을 표준 출력 장치에 입력하고 그 내용을 확인	출력된 문자열 판독 불가	출력된 문자열 판독 불가(시험 완료)
응용 클라이언트에 두개의 보안 컨텍스트 객체가 생성되었는가 확인 실제로 응용 서버들과 통신할 때 적절한 보안 컨텍스느를 통해 보안 서비스를 제공는지 확인	다중 보안 컨텍스트 설치 성공	다중 보안 컨텍스트 설치 성공(시험 완료)

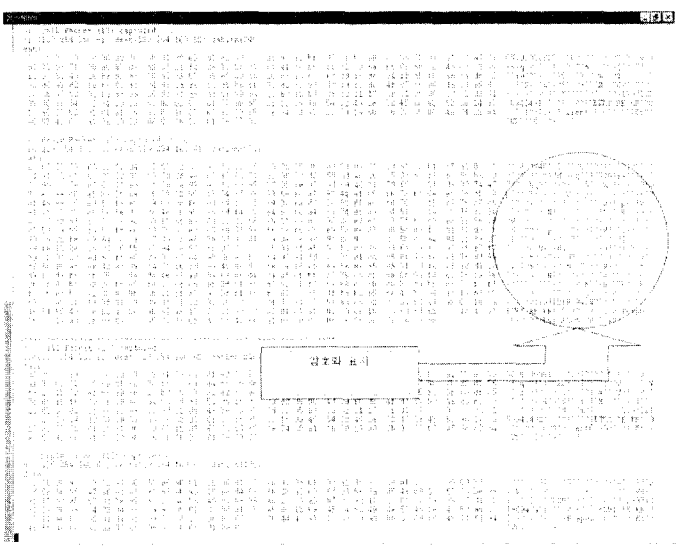


그림 10. 암호화된 메시지 모니터링 화면
Fig. 10. Result of message protection

(그림 10)은 보안 호출 정책에 기밀성과 무결성을 설정한 경우 응용 클라이언트와 응용 서버 사이에 전송되는 메시지를 모니터링한 결과를 보여준다.

V. 결 론

본 논문에서는 CORBA 보안 서비스를 구현하여 분산 객체 환경에서 수행되는 응용들에게 보안 기능을 제공하였다. 특히 통신망을 통해 전송되는 응용들의 데이터 보안에 중점을 두고 있다.

본 논문에서 설계 구현한 보안 서비스는 ORB를 사용하는 분산 객체 환경의 응용 서비스들에게 투명한 보안 기능을 제공한다.

CORBA 보안 서비스는 보안 메커니즘에 독립적인 보안 객체들이 보안 기능을 제공하여 응용의 필요에 따라 최적의 보안 메커니즘을 선택할 수 있도록 하고 있다. 이를 위해서는 GSS-API와 같은 보안 표준 인터페이스를 사용하여 보안 객체를 구현하여야 한다.

SESAME는 분산 환경에서 보안 표준 인터페이스인 GSS-API를 제공하는 보안 시스템으로 CORBA 보안 서비스 구현에 필요한 보안 메커니즘을 제공한다. 특히 ORB를 통해 전송되는 데이터들과 관계없는 사용자 인증, 접근제어, 보안 감사, 보안 관리 기능 등은 SESAME에서 제공하는 기능들을 그대로 CORBA 환경에 적용 가능하다. 그러나 SESAME는 TCP/IP 소켓 인터페이스를 기본으로 하여 전송되는 응용들 사이의 데이터에 대한 보안을 보장하고 있어 ORB를 통해 전송되는 데이터들에게 보안을 제공하는 것은 불가능하다.

본 논문에서는 SESAME를 이용하여 CORBA 환경에서 수행 중인 응용들에게 사용자 인증, 접근제어, 보안 감사 등의 보안 기능을 제공하는 동시에 ORB를 통해 전송되는 데이터들의 기밀성과 무결성을 보장하기에 필요한 CORBA 보안 객체인 Vault 객체와 Security Context 객체를 GSS-API를 사용하여 구현하여 분산 객체 환경에서 수행되는 응용들에게 투명하게 보안 서비스를 제공하였다.

GSS-API를 사용하여 구현된 보안 객체들은 GSS-API를 제공하는 하부 보안 메커니즘에 독립적으로 보안 기능을 제공한다. 이는 GSS-API를 응용의 필요에 따라 새로운 보안 메커니즘으로 대치 가능한 구조를 제공한다.

향후 연구 방향은 Secure IIOP (Internet Inter-Operable Protocol)[14]를 구현하여 서로 상이한 보안 정책과 보안 메커니즘을 가지는 ORB들 사이에서 상호 연동 가능한 보안 서비스를 제공하여야 한다.

참고 문헌

[1] George Coulouris, Jean Dollimore and Tim Kindberg, "Distributed Systems Concepts and Design", Addison-Wesley Publishing Company, 1994.

[2] Richard Mark Soley and Christopher M Stone, Object Management Architecture Guide Revision 3.0, OMG, June 13, 1995.

[3] Thomas J. Mowbray and Ron Zahavi, "The Essiential CORBA: Systems Integration Using Distributed Objects", John Wiley &Sons, Inc., 1995.

[4] B. C. Neuman and T. Ts'o, Kerberos: An Authentication Service for Computer, IEEE Communications, Vol.32, No.9, pp.33~38, http://nii.isi.edu/publications/kerberos-neuman-tso.html, 1994

[5] J. Linn, The Kerberos Version 5 GSS-API Mechanism, IETF, RFC 1964, http://www.roxen.com/rfc/rfc1964.html, June 1996.

[6] OSF, Open Software Foundation Training Course, OSF DCE System

Administration Course Student Guide Vol.1.0, Dec 1992

[7] T. Parker and D. Pinkas, SESAME V4 Overview, SESAME Issue 1, 1995.

[8] OMG, CORBAservices: Common Object Security Specification, pp.15-1 ~ 15-294, Nov., 1996

[9] J. Linn, "Generic Security Service Application Program Interface", IETF, RFC 2078, Jan. 1997.

[10] J. Linn, "Generic Security Service Application Program Interface", IETF, RFC 1508, Nov. 1993.

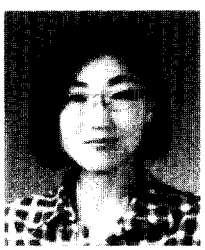
[11] J. Wray, "Generic Security Service Application Program Interface: C-binding", IETF, RFC 1509, Nov. 1993.

[12] IONA Ltd, Orbix Reference Guide, 1997.

[13] OMG, The Common Object Broker Architecture and Specification, Revision 2.0, July 1995.

[14] OMG, CORBA Secure Interoperability, OMG Document Number: orbos/96-06-20, July 1996.

□ 著者紹介



이 권 일

1988년 2월 충남대학교 계산통계학과 졸업(이학사)
1998년 9월 충남대학교 대학원 컴퓨터학과 졸업(이학석사)
1996년 정보처리기술사(전자계산기 조직응용 분야)
1988년 ~ 현재 한국전자통신연구원 컴퓨터.소프트웨어 기술연구소 인터넷서비스연구부 선임연구원

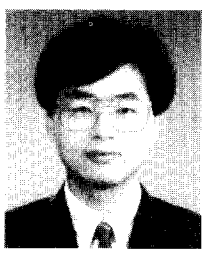
※ 주관심 분야 : 분산시스템, 정보 보호, 분산객체 기술

□ 著者紹介

김 명 준

1978년 서울대학교 계산통계학과(학사)
1980년 한국과학기술원 전산학과(석사)
1986년 5월 프랑스 Nancy 제 1대학교 응용수학 및 전산학과(박사)
1980년 2월 ~ 1981년 6월 아주대학교 종합연구소 연구원
1981년 10월 ~ 1986년 5월 프랑스 Nancy 전산학 연구소 연구원
인터넷서비스연구부장, 책임연구원
1993년 프랑스 Nice Sopia-Antipolis 대학 초빙교수

※ 주관심 분야 : 암호이론, 컴퓨터 보안



류 재 철

1985년 2월 한양대학교 산업공학(학사)
1988년 5월 Iowa State Univ. 전산학석사
1990년 12월 Northwestern Univ. 전산학박사
1991년 2월 ~ 현재 충남대학교 컴퓨터과학부 부교수

※ 주관심 분야 : 컴퓨터 및 통신 보안체제, 네트워크 관리, 분산 처리