# 필터와 트랜스포머를 이용한 투명한 보안기반의 설계 및 구현

김 용 민† · 이 도 헌†† · 노 봉 남†† ·
최 락 만††† · 인 소 란†††

## 요 약

최근의 정보통신 환경은 다양한 시스템 및 응용서비스를 지원하는 개방형 분산처리 환경으로 변모하고 있으며, 또한 상속 및 캡슐화, 재사용 등의 다양한 장점을 제공하는 객체지향 기법을 이용한 응용의 개발이 이루어지고 있다. 이러한 이용의 증대는 외부의 보안 위협에 취약한 네트워크 환경에 안전한 정보 교환을 위한 대책을 필요로 한다.

본 논문에서는 분산 객체환경에서 응용의 안전성을 위하여 CORBA의 보안서비스 규격에 기반하여 인증, 보안설정, 접근제어, 보안정보관리의 기능을 갖는 투명한 보안 기반구조를 설계 및 구현하였다. 보안 기반구조는 사용자 인증정보, 데이터 암호화 및 무결성을 위한 키 분배 등을 지원하기 위한 외부 보안서비스로서 SESAME V4를 이용하였다. 또한, 응용의 요청에 대한 투명한 보안 서비스를 지원하기 위하여 객체요청중개자(Object Request Broker: ORB)와 인터페이스를 지원하는 필터(filter)와 트랜스포머(transformer)의 기능을 이용하였다. 필터는 메시지의 전송 및 수신 전후에 파라미터 및 메소드를 삽입 또는 제거 할 수 있으며, 트랜스포머는 메시지의 전송전 및 수신후에 암호화 및 복호화를 위해 바이트 스트림에 직접 접근할 수 있다. 이것은 CORBA에서 정의한 안전한 객체요청중개자(secure ORB)의 접근제어 및 안전한 호출 인터셉터를 필터와 트랜스포머를 이용하여 구현한 것이다.

# Design and Implementation of a Transparent Security Infrastructure using Filter and Transformer

Yong-Min Kim† · Do-Heon Lee†† · Bong-Nam Noh††
· Rak-Man Choi††† · So-Ran Ine†††

## ABSTRACT

In these days, information communication systems are based on both open distributed computing technologies and object-oriented techniques like inheritance, encapsulation and object reuse to support various system configuration and application. As information systems are interconnected through unsecure networks, the need for the secure information exchange is more critical than before.

※ 본 연구는 '97년 한국전자통신연구원 "초고속 정보통신기반 안정성 기술개발 사업"의 지원에 의해 수행되었슴.
† 준 회 원 : 전남대학교 전산학과
†† 종신회원 : 전남대학교 전산학과
††† 정 회 원 : 한국전자통신연구원 소프트웨어공학연구실
논문접수 : 1997년 12월 1일, 심사완료 : 1998년 2월 17일

In this paper, we have designed and implemented a transparent CORBA-based security infrastructure with authentication, security context association, access control and security information management to support a secure applications in distributed object environment. SESAME Ver. 4 was adopted as an external security service to manage user privilege attributes and to distribute keys for data encryption, decryption and integrity. Using filter and transformer with an interface to Object Request Broker, it provides a transparent security service to applications. The filter objects are special classes that allow additional parameters to be inserted into messages before they are sent and removed just after they are received. The transformer objects are special classes that allow direct access to the byte stream of every messages for encryption and decryption before it is sent and just after it is received. This study is to implement the access control interceptor(ACI) and the secure invocation interceptor(SII) of secure ORB defined in CORBA using filter and transformer.

## 1. 서 론

최근 응용의 개발은 개방형 분산처리와 객체지향 기법을 사용하는 분산객체 환경에서 이루어지고 있다. 이러한 분산객체 환경에서의 이용 증대는 보호하고자 하는 데이터, 응용 및 시스템에 불법적인 접근, 도청, 위장의 가능성이 함께 증대되어 이러한 위협에 대한 수동적 또는 능동적인 보안 대책의 수립과 보안 위협에 대처할 보안기반의 구축을 요구한다.

개방형 분산처리 환경에서 보안을 지원하기 위한 관련 기술은 대표되는 것으로 인증 서비스를 위해 사용되는 Kerberos[1] 및 신뢰성 있는 보안 서비스를 제공하기 위하여 분산 플랫폼을 제공하는 OSF/DCE(Open Software Foundation /Distributed Computing Environment)[7], 응용계층 중심의 안전성을 지원하는 시스템인 SESAME (Secure European System for Application in a Multi-vendor Environments)[8]가 있다. Kerberos는 대칭형(symmetric) 암호 서비스에 기반한 인증 시스템이며, OSF/DCE는 분산처리 환경에서 등록, 인증, 권한부여 및 접근제어의 보안 서비스를 제공한다. SESAME는 안전한 개방형 분산처리 환경에서 응용계층의 서비스에 중점을 갖는 보안 시스템으로서 인증, 권한, 데이터 비밀성 및 무결성, 부인봉쇄 등을 지원한다. SESAME는 OSF/ DCE와 마찬가지로 외부 인터페이스를 위해 GSS-API (Generic Security Service-Application Programming Interface)를 제공하며 전체 구조는 믿을수 있는 제3자(Trusted Third Party: TTP), 서버, 클라이언트의 세부분으로 구성되어 있다.

그러나 분산 환경에서의 보안을 지원하는 위의 시스템들은 분산 객체지향 환경에서의 응용서비스에 대하여는 고려하지 않는다. 따라서 최근 그 유용성이 입증되고 있는 객체지향 응용의 개발 및 서비스를 지원하기 위한 분산처리 기반구조를 위해 OMG(Object Management Group)는 CORBA(Common Object Request Broker Architecture)를 제안하였다. CORBA는 클라이언트 및 서버 응용의 요청 및 응답에 대하여 객체요청중개자(Object Request Broker: ORB)를 정의하여 객체 기반의 투명한(transparent) 분산처리를 지원하며, 응용서비스에 안전성을 지원하기 위한 보안 기능 및 인터페이스 규격을 포함한 안전한 객체요청중개자(Secure ORB)를 정의하였다[5,6].

본 연구에서는 투명한 보안 서비스를 위한 인터셉터(interceptor) 기능을 현재의 객체요청중개자 제품들은 제공하고 있지 않기 때문에 객체요청 중개자 외부에 라이브러리 형태로 이의 기능을 지원하는 IONA Orbix의 필터(filter)와 트랜스포머(transformer)를 이용하여 인증, 보안문맥연결설정, 접근제어, 보안정보관리로 구성되는 CORBA 상의 보안기반(security infrastructure)을 설계하고 구현한다. 또한 사용자 인증정보, 암호화 및 무결성을 위한 키 분배 등을 지원하기 위하여 외부 보안서비스로서 SESAME V4를 이용하였다.

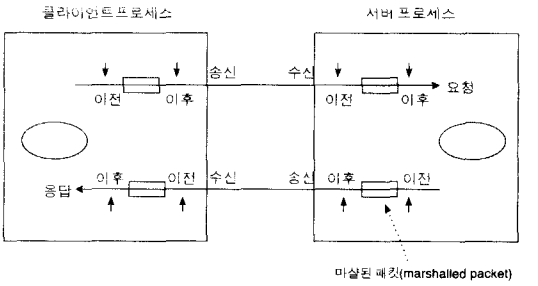
본 논문의 구성은 2장에서는 투명한 보안서비스를 지원하는 방안에 대하여 살펴보고 3장에서는 설계 고려사항 및 투명한 분산객체 환경을 위한 보안기반 시스템의 구조 및 동작을 보인다. 4장에서는 구현된 보안기반의 필터 및 트랜스포머의 이용을 보이고 5장에서는 결론 및 향후 연구방향을 기술한다.

## 2. 투명한 보안을 지원하기 위한 방안

분산 객체환경에서 보안 서비스를 제공하고자 하는 보안기반 구조는 기본적으로 위치 투명의 분산처리를

서원하며 이에 제공되는 보안 서비스는 요청한 메세지를 가로채어 이의 파라미터를 확인하고 해당되는 요청처리 및 전송을 하며 전송되는 메세지에 대하여 암호화 및 복호화를 지원하여야 한다. 이러한 작업은 요청메세지의 사건(event) 발생시 이를 감지하고 자동으로 처리하는 메카니즘을 필요로 한다. CORBA의 안전한 객체요청중개자 규격에서는 응용의 요청 및 응답을 처리하기 위하여 요청 수준(request-level) 및 메세지 수준(message-level) 인터셉터를 정의하였다. 그리고 보안서비스를 위하여 요청 및 메세지 수준 인터셉터에 각각 해당하는 접근제어 인터셉터(Access Control Interceptor: ACI)와 안전한 호출 인터셉터(Secure Invocation Interceptor: SII)를 정의하였다[5]. 그러나 안전한 객체요청중개자를 위한 이러한 기능을 포함한 원시코드를 획득하거나 자체 개발하는 것은 어렵다.

Orbix의 필터 객체는 요청 메시지에 대해 클라이언트로부터 서버로 전송 및 수신 전후에 관심있는 파라미터 및 메소드를 삽입 또는 제거 할 수 있으며, 또한 감시, 감사로그, 디버깅, 그 외의 인증과 같은 보안관련 사항의 기능을 추가할 수 있다. 필터링 방법은 클라이언트에서 요청한 서비스의 인터페이스 정의 언어를 서버에서 제공하는 응용으로 연결하기 위하여 객체 클래스 및 메소드에서 상속하여 제공하는 기본객체적응자(Basic Object Adaptor: BOA)로 할 것인지, 또는 프로그래머가 구현한 특별한 객체 클래스로 위임(delegation)하여 연결 처리하는 TIE(tie together) 방법으로 할 것인지에 의해 필터링 방법을 결정한다. 프로세스 수준 및 객체 수준의 필터링 서비스에서는 필터의 연속적인 연결(filter-chain)이 가능하며 응용의 요구에 따라 각 필터에 임의의 연산을 부가하여 새로운 메세지 패킷으로 만드는 마샬링(marshaling) 시점을 고려한다. 각 필터의 연결은 메세지의 송신 및 수신, 요청 및 응답, 마샬링 전후의 형태를 조합하여 8개의 사건 필터링 시점으로 구성되며, (그림 1)은 프로세스 수준의 필터링 시점을 보인 것이다[3]. 또한 트랜스포머 객체는 클라이언트로부터 메세지의 전송 전 및 서버에서 수신 후 각 요청 메세지에 암호화 및 복호화 그리고 무결성 검사를 위해 전송 및 수신되는 바이트 스트림에 접근 가능하도록 한다[2].




(그림 1) 프로세스 수준의 필터링 시점
(Fig.1) Filtering monitor points of process level

## 3. 보안기반 구조 및 동작

### 3.1 설계 고려사항

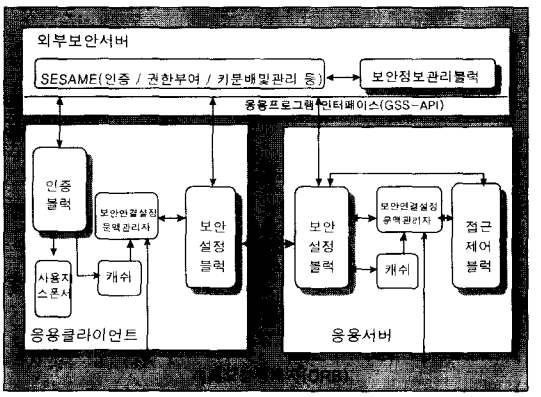
CORBA를 지원하는 다양한 제품중에서 객체지향 프로그래밍 언어인 C++로 인터페이스 정의 언어(Interface Definition Language: IDL)를 사상(mapping)하는 컴파일러를 제공하며 객체요청중개자 표준을 지원하여 널리 사용되고 있는 것을 선택한다. 또한 투명한 보안기반 시스템을 위하여 객체요청중개자 내에서 응용 프로그램의 요청을 가로채어(intercept) 개발자에게 지원할 수 있는 메커니즘을 이용할 수 있어야 한다. 따라서 객체요청중개자 제품중에서 응용의 요청 메세지를 가로채어 해석할 수 있는 기능 및 메시지를 암호화 및 복호화하기 위하여 객체요청중개자의 외부에 라이브러리 형태로 제공하는 분산객체 플랫폼을 이용한다[3,4].

투명한 보안 서비스를 제공하는 보안기반 시스템은 CORBA 보안 규격에 따르는 다양한 보안 서비스중에서 다음과 같이 제한하여 개발하였다. 응용은 제공하는 보안 서비스 이용 여부를 선택적으로 지원할 수 있으나 객체요청 중개자를 통하여 반드시 보안 검사를 거치도록 하였으며, 다중 영역(multi-domain)에서의 보안정책을 지원하지만 본 연구에서는 단일 영역(single-domain)에서 보안정책을 지원하도록 구현한다. 접근제어 기능의 수행은 개시자(initiator) 및 목표(target) 시스템에서 수행 가능하나 목표 시스템에서만 수행하도록 하며, 사용자의 인증정보, 키 분배, 암호 및 무결성 지원, 보안정책 등의 지원을 위하여 CORBA에서 채택 가능한 외부 보안서비스로서 지정된 것 중의 하나인

SESAME V4를 이용한다. 현재의 시스템은 상호인증, 권한위임, 부인봉쇄 및 보안 감사의 기능은 지원하지 않는다. 이와 같은 사항을 고려하여 응용에 투명한 보안서비스를 제공하는 보안기반 구조는 안전한 객체요청 중개자를 중심으로 인증 블럭, 보안설정 블럭, 접근제어 블럭, 보안정보관리 블럭과 같은 요소들로 구성되며, 각 블럭들은 필요에 따라 외부 보안 서비스를 이용할 수 있도록 하였다.

### 3.2 구조 및 동작

(그림 2)는 CORBA 환경에서 보안기반을 위한 전체 구조를 블럭 다이어그램의 형태로 보인 것이다.




(그림 2) CORBA 응용을 위한 보안기반 구조
(Fig. 2) Security Infrastructure for CORBA-based Applications

사용자는 인증 블럭의 사용자 프로그램을 이용하여 외부 보안 서비스로부터 인증 절차를 수행하여 신임장(credential)을 획득하고 이를 보안캐쉬에 저장한다. 이후 수행되는 응용은 클라이언트의 보안설정 블럭에 요구를 전달하며 객체요청중개자를 통하여 서버의 보안설정 블럭에 연결 요청한다. 서버의 보안설정 블럭은 신임장의 내용 및 보안정책을 고려하여 보안문맥을 생성하며, 보안문맥의 생성후 안전한 연결을 위한 보안연결채널을 생성하여 이후에 전송되어지는 데이터에 대해 비밀성 및 무결성 서비스를 제공한다. 보안 연결 채널의 설정 이후에 요청되는 서버 객체로의 접근은 보안정보관리 블록에서 제공되는 정책에 따라 접근제어 블럭에 의해 접근 허용여부가 결정된다. 각 블럭별 기능의 개요는 다음과 같다[9].

#### 3.2.1 인증블럭

인증블럭은 서버 객체에 접근하려고 하는 사용자 및 응용과 같은 주체(principals)의 인증 및 권한 부여를 위한 인증 서비스를 위해 로그인 프로그램 또는 사용자 스폰서(user sponsor)를 통하여 외부 보안서비스를 이용하며, 만약 사용자 식별자등의 관련 정보가 있는 경우에 그들이 상호 인증하는데 필요한 신임장을 인증블럭에 반환한다. 이를 위하여 인증블럭은 외부 보안서비스로 SESAME 인증 서비스 (Authentication Service: AS)와 권한부여 서비스(Privilege Attribute Service: PAS)를 이용하여 주체에 대한 인증을 수행하며 그 결과로 인증서 및 세션 키를 받는다. 인증블럭을 구성하는 객체 클래스는 외부 보안서비스에 인증 요청을 하는 *PrincipalAuthenticator*, 사용자의 식별자 및 권한속성 등에 관한 내용을 가지고 있어 연계설정 요청을 위한 정보로 이용하는 *Credentials*, 보안문맥과 관련하여 신임장 객체의 권한속성 데이터에 접근 기능을 갖는 *Current*가 있으며, 그리고 사용자가 인증시 사용하는 사용자 스폰서 프로그램이 있다.

SESAME의 인증 서비스를 이용하는 *PrincipalAuthenticator* 객체의 멤버함수 *authenticate*()의 구현은 (그림 3)과 같다.

```
Security::Authentication StatusSecurityLevel2
        ::PrincipalAuthenticator::authenticate() {
. . . :
apa_return=apa_auth(); //SESAME APA(Authentica-
                       //tion and PrivilegeAttribute)
                       //의 API
ret_val=auth_result_ok();
switch(ret_val)        {
case 0 : // Errors
. . . :
case 1 : // Success
. . . :
} }
```

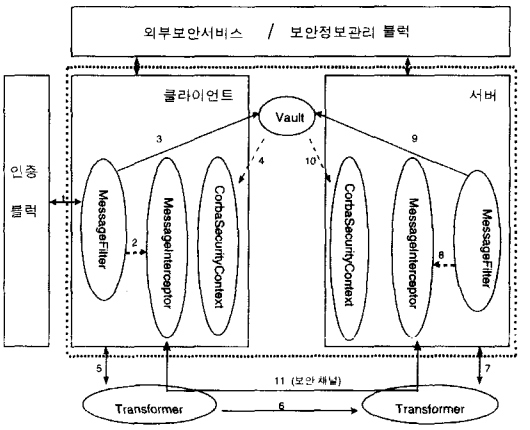
(그림 3) 멤버함수 *authenticate()*
(Fig. 3) Member function *authenticate()*

#### 3.2.2 보안설정 블럭

보안설정 블럭은 보안문맥(security context)의 생성 및 관리와 인터셉터의 기능으로 구분된다. 보안문맥의 생성 및 관리 그리고 안전한 연결의 유지는 *Vault* 객체 및 보안 연결설정 문맥 관리자(Security Association Context Manager: SACM)에 의해 수행되며, 인터셉터의 기능은 IONA's Orbix에서 제공하

는 필터와 트랜스포머를 이용하였으며 이의 구현내용은 4장에서 상세히 보인다.

응용 클라이언트로 부터 새로이 요청된 메세지는 클라이언트의 보안설정 블럭에 전달되며 보안설정 블럭은 요청 인터셉터를 대행하는 필터를 사용하여 메세지에 호스트 주소, 프로세스 식별자 정보를 추가하며 *MessageInterceptor* 객체를 생성한다. 또한 필터는 *Vault* 객체를 통하여 사용자의 신임장 및 보안정보관리 블럭에서 제공하는 연계설정 요구사항을 고려한 클라이언트 보안문맥을 생성 및 초기화하고 서버에 전송할 메세지에 클라이언트의 신임장과 동일한 내용을 생성할 수 있는 정보를 갖는 보안 토큰(security token)을 추가한다. 클라이언트 보안설정 블럭은 트랜스포머를 통하여 보안 토큰을 포함한 메세지를 서버의 트랜스포머로 전달하고 서버의 필터는 메세지를 가로채어 서버의 *MessageInterceptor* 객체를 생성한다. 그리고 *Vault* 객체를 이용하여 클라이언트에서 전달되는 보안 토큰의 정보에 기반한 서버의 보안문맥을 초기화하며 클라이언트와 서버간의 안전한 연결을 위한 보안 채널을 설정한다. 계속되어지는 클라이언트에서 서버로의 메세지 전송은 트랜스포머를 통하여 *MessageInterceptor* 객체를 호출함으로써 이루어진다. 보안설정 블럭은 (그림 4)와 같이 구성되며 외부 인터페이스와의 관계를 보인 것이다.




(그림 4) 보안설정 블럭의 구조
(Fig. 4) Structure of security association block

*Vault*와 *CorbaSecurityContext* 객체 클래스의 정의 및 *Vault* 객체 멤버함수의 알고리즘은 (그림 5)와 같다.

```
Class Vault {
 Security::AssociationStatus init_security_context(
  in CredentialsList     &cred_list,
  in SecurityName        target_security_name,
  in CORBA::Object_ptr   target,
  in DelegationMode      delegation_mode,
  in OptionDirectionPairList association_options,
  in MechanismType       mechanism,
  in Opaque              mech_data,
  in Opaque              chan_binding,
  out gss_buffer_t       out_token,
  out CorbaSecurityContext_ptr &security_context
 );
 Security::AssociationStatus accept_security_context(
  in CredentialsList   &cred_list,
  in Opaque            &chan_bindings,
  in gss_buffer_t      in_token,
  out Opaque           &out_token,
  out CorbaSecurityContext_ptr &security_context
 );
}

Class CorbaSecurityContext {
 Security::AssociationStatus
                    continuous security context(

  in Opaque                 in_token,
  out Opaque                out_token,
  );
 void protect_message(
  in gss_buffer_t           msg_buf,
  in QOP          qop,
  out gss_buffer_t  text_buf,
  out Opaque                token
  );
 boolean reclaim_message(
  in gss_buffer_t           text_buf,
  in Opaque                 token,
  in QOP          qop,
  out gss_buffer_t  msg_buf
  );
}
```

(a)

```
Security::AssociationStatus
 Vault::init_security_con t ext() {
  //association_option을 검사
  //SESAME의 gss_init_sec_context 호출
  //CorbaSecurityContext를 생성하여 SESAME
  //에서 리턴한 securitycontext의 포인터(핸들) 유지
}
Security::AssociationStatus
  Vault::accept_security_context() {
  //SESAME의 gss_accept_sec_context 호출
  //CorbaSecurityContext를 생성하여 SESAME에
  //서 리턴한 securitycontext의 포인터(핸들) 유지
}
```

(b)

(그림 5) (a) *Vault*와 *CorbaSecurityContext* 객체 클래스
(b) *Vault* 의 멤버함수
(Fig. 5) (a) Class of Vault and *CorbaSecurity- Context*
(b) Member function of *Vault*

3.2.3 접근제어 블럭

접근제어 기능의 수행은 안전한 연결설정 후 응용 클라이언트에서 요청된 서버 객체로의 접근 메세지는 *RequestInterceptorFiler* 객체에 의해 필터링 되며 *RequestInterceptor* 객체의 유무를 확인하여 없을 시에는 새롭게 생성한다. 서버의 보안설정 블럭에 의해 생성된 보안문맥으로 부터 응용 사용자의 권한속성을 획득하여 *AccessDecision* 객체에 전달하고 접근 허용 여부를 제공된 접근제어 정책에 따라 결정한다. 접근제어 정책은 CORBA 보안규격에 따라 보안정보관리 블록에 기술된 권한기반 정책(Right-based Policy)을 이용한다. 권한기반 정책은 영역접근 정책(Domain Access Policy)과 요구권한 정책(Required Rights Policy)을 기반으로 응용 서버에 대한 사용자의 접근 권한 유무를 표시한 것이다.

접근제어 블록의 객체 클래스는 요청 인터셉터의 역할을 하며 RequestInterceptor 객체를 생성하는 *RequestInterceptorFilter*, 요청된 메세지의 접근제어를 위해 필요한 속성정보를 보안문맥으로 부터 가져오는 *RequestInterceptor*, 요청 메세지에 대한 접근제어 허용 여부를 수행하는 *AccessDecision* 으로 구성한다.

접근제어 블럭에서의 *AccessDecision* 객체 클래스의 정의 및 *access_allowed()* 멤버함수 알고리즘은 (그림 6)과 같다.

```
Class AccesDecision {
  AccessDecision::access_allowed(
     in SecurityLevel2::CredentialList  *cred_list,
     in CORBA::Object                   *target,
     in Identifier      operatio_name,
     in Identifier      target_interface_name
  );
AccessDecision::access_allowed() {
   // 입력 매개변수를 검사하여 권한 속성을 획득하고
   // 보안정보관리 블록의 DomainAccessPolicy 접근
   // 하며 granted_right 리스트를 획득하여 접근
   // 허용여부를 결정한다.
}
```

(그림 6) *AccessDecision* 클래스 및 멤버함수 *access_allowed()*

(Fig. 6) Class of *AccessDecision* and Member function *access_allowed()*

3.2.4 보안정보관리 블럭

보안정보관리 블럭은 사용자 인증 및 권한속성 정보, 접근제어를 위한 보안정책 정보, 보안 설정을 위한 데이터베이스 및 각 관리 모듈로 구성되며, SESAME로부터 정보의 획득 및 설정을 위한 유틸리티 모듈로 구성된다. 보안정보관리 블럭은 인증, 보안설정, 접근제어 기능의 동작을 위한 관련 정보를 제공하며 유지하는 기능을 한다. 관리 유지하는 기능은 관리 정보의 안전성을 위해 SESAME 보안 서버가 있는 시스템에서만 수행되도록 하였다. 보안정보 관리 블럭의 보안정책을 위한 자료구조는 (그림 7)과 같다.

```
//요구권한 정책(RequiredRights Policy)
struct required_rights_rec {
   char                header;
   char                interface_name[20];
   char                operation_name[20];
   Security::RightsCombinator
          rights_combinator;
   char                required_rights[20];
};
//영역접근정책(Domain Access Policy)
struct do_acc_po_rec {
   char                header;
   char                priv_attr[20];
   Security::DelegationState   del_state;
   char                granted_rights[20];
};
//보안 연결설정 정책
//(Security Association Policy)
struct sec_Invocation_po_rec {
   char                header;
   char                object_type[20];
   Security::RequiresSupports  requires_supports;
   Security::CommunicationDirection    direction;
   Securty::AssociationOptions         options;
};
```

(그림 7) 보안정책을 위한 자료구조

(Fig. 7) Data Structure for Security Policy

## 4. 투명성을 지원하는 필터와 트랜스포머

CORBA 표준 규격에서는 클라이언트와 서버의 객체 사이에 원격 메소드 호출과 같은 상호 동작을 위해 객체요청중개자를 통한 서비스 요청에 대하여 요청 및 메세지 인터셉터의 개념으로 메세지의 투명한 처리를 정의하였다. 그러나, 현재 이러한 인터셉터를 지원하는 안전한 객체요청중개자 제품의 확보는 어렵다. 따라서 본 연구의 투명한 보안기반의 설계 및 구현을 위하여 객체요청중개자 외부에서 보안 서비스를 지원하도록 하

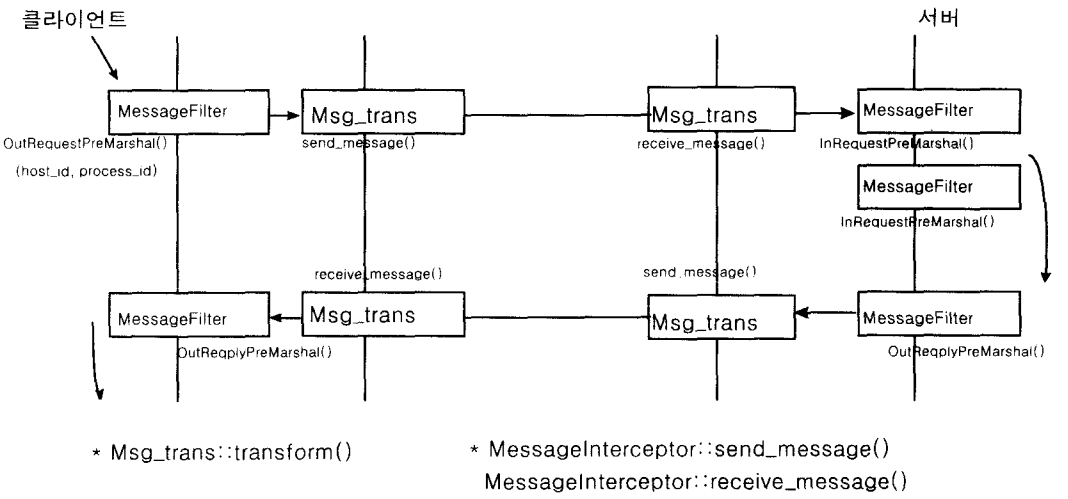
였으며 분산 객체 플랫폼으로 사용한 IONA Orbix의 객체요청 중개자와 투명한 인터페이스를 위하여 필터 및 트랜스포머 객체를 이용하여 설계 및 구현하였다.

응용 클라이언트의 서버로의 요청에 대하여요청된 객체를 응용 서버의 객체 클래스에서 상속하여 클라이언트의 인터페이스 정의 언어에 바인드(bind)하기 위하여 기본객체적응자를 이용하는 접근 방법을 사용하였다. 이에 따라 메세지의 투명한 처리를 위한 필터링의 방법은 2장에서 소개한 프로세스 수준의 필터링 방법을 이용하여 설계 및 구현하였다. CORBA 환경의 응용을 위한 보안서비스를 제공하기 위하여 3장에서 설계 및 구현된 블럭에서 요청 메세지를 필터링하기 위해 관련된 블럭은 보안설정 및 접근제어 블럭이며, (그림 8)은 투명한 보안 기반을 지원하는 필터와 트랜스포머의 연결 구성을 보인다.

클라이언트는 요청 메세지를 전송하기 전에 보안설정을 위해 *MessageFilter* 객체의 멤버함수 *OutRequestPreMarshal()*은 호스트 주소 및 프로세스 식별자 정보를 필터링 한다. Orbix의 트랜스포머 객체인 *Msg_trans*는 *transform()*을 수행하여 *MessageInterceptor* 객체의 *send_ message()*를 호출하며, *send_message()*는 메세지에 대해 SESAME 암호화 서비스를 받기 위하여 *protect_message()*를 호출한다. 요청된 메세지가 서버측에 수신후 트랜스포머 객체인 *Msg trans*는 *transform()*를 수행하여 *receive_ message()*를 호출하고, *receive_message()*는 메시지에 대한 SESAME 복호화 서비스를 위하여 *reclaim_message()*를 호출한다. 서버측 *MessageFilter* 객체에서는 응용 객체에 전달하기 전에 *InRequestPreMarshal()*을 통하여 클라이언트의 호스트 주소 및 프로세스 식별자를 필터링 한다.

클라이언트와 서버의 보안연결 설정을 위한 작업 처리 후, 클라이언트에서 전달된 메세지의 요청에 대하여 서버의 접근제어 블럭에서는 클라이언트에서 요청된 메세지가 요구한 서버의 객체 또는 연산에의 접근에 대한 처리를 위하여 접근제어 블럭의 *RequestInterceptorFilter* 객체의 멤버함수 InRequest PreMarshal()을 통하여 접근제어를 위한 필터링을 한다. 이 때 접근제어 블럭의 *RequestInterceptorFilter*인 *InRquestPreMarshal()*는 보안설정 블럭의 *InRquestPreMarshal()*과 함께 사용한다. 즉, 보안설정의 *InRequestPreMarshal()* 내에 접근제어를 위한 기능을 부가하여 구현하였다. 처리된 응용 프로세스는 접근제어의 허용 여부를 서버에서 클라이언트로 *OutReplyPreMarshal()*, *transform()*, 그리고 *InReplyPreMarshal()* 등을 호출하여 전송한다.

이와 같은 절차는 프로세스 수준의 필터 체인을 보인 것이며, (그림 9)는 필터와 트랜스포머를 이용한 구현을 보인 것이다.




(그림 8) 필터와 트랜스포머의 연결 구성
(Fig. 8) Configuration of filter-chain and transformers

```
MessageFilter::OutRequestPreMarshal() {
  if (first_request) {
    . . . .
    message_interceptor =
          new MessageInterceptor();
    . . . .
    ret = init_vault(request);
    . . . .
  }
  else {
    current_target = target;
    gethostname();
    request << getpid();
    request << hostname;
  }
}
```

(a)

```
Msg_Trans::transform() {
  if (sending) {
    if(!first_data) {
      send_message();
    }
    else {
      receive_message();
    }
  }
}
```

(b)

```
MessageFilter::InRequestPreMarshal() {
  if (first_request) {
    . . . .
    message_interceptor =
          new MessageInterceptor();
    . . . .
    ret = init_vault(request);
    . . . .
  }
  else {
    gethostname();
    request << getpid();
    request << hostname;
  }
  // 접근제어 모듈의 필터
    . . . .
  if(request_interceptor == NULL)
    new RequestInterceptor();
    . . . .
    if(!access_allowed){
    . . . .
    throw CORBA::NO_PERMISSION();
  }
}
```

(c)

(그림 9) (a) 클라이언트 필터 (b) 클라이언트 및 서버 트랜스포머 (c) 서버 필터

(Fig. 9) (a) Filter of Client (b) Transformer of Client and Server (c) Filter of Server

## 5. 결론 및 향후 연구방향

본 논문에서는 CORBA 환경에서 응용에 보안 서비스를 지원하는 보안 기반구조를 설계 및 구현하고, 투명한 서비스를 제공하기 위하여 IONA Orbix의 필터와 트랜스포머를 이용하여 안전한 객체요청중개자를 이용하도록 하였다.

필터는 안전한 객체요청 중개자에서의 응용의 접근제어 인터셉터의 기능을 대신하며 트랜스포머는 안전한 호출 인터셉터의 역할을 한다. 필터링은 기본객체적응자의 접근 방법에 따라 프로세스 수준의 연결체인으로 구성하였다. 안전성을 지원하기 위한 보안기반은 인증, 보안설정, 접근제어, 보안정보관리의 기능을 지원하며, 또한 사용자 인증 정보, 암호화 및 무결성을 위한 키 분배 등을 지원하기 위하여 외부 보안서비스로서 SESAME V4를 이용하였다. 구현 환경은 Solaris 2.5 운영체제와 INOA Orbix 2.0.1, 외부보안서비스 SESAME V4에 기반한 환경에서 C++ 프로그래밍 언어를 이용하였다.

향후 상이한 객체요청중개자 간의 상호 작용이나 서로 다른 보안 영역간의 상호 운용에 따른 보안 기능 개발과 보안감사, 부인봉쇄, 상호 인증, 권한 위임 등의 보안 서비스에 대한 추가 기능 개발이 요구된다.

## 참 고 문 헌


[1] B. C. Neuman and T. Ts'o, "Kerberos : An Authentication Service for Computer", IEEE Communications, Vol.32, No.9, pp.33-38, 1994.

[2] IONA, "OrbixSecurity White Paper Part 2 of 2", http://www-usa.iona.com/Developer/epa-per/osecurity/whitepaper2.html", 1997.

[3] IONA, "Programming guide orbix2 distributed object technology ", Release 2.0 Vol. 1, Nov. 1995.

[4] IONA, "Programming guide orbix2 distributed object technology ", Release 2.0 Vol. 2, Nov. 1995.

[5] Object Management Group, Framingham, "Common Object Services Specification-Securi-ty Services Specification", Volume 1", 1992.

[6] Object Management Group, "CORBA Security", Document No. 95.12.1, Dec., 1995.
[7] OSF, "Open Software Foundation Training Course", OSF DCE System Aministration Course Student Guide, Vol.1, Dec. 1992.
[8] "SESAME Technology Version4-Overview ", http:// www.esat.kuleuven.ac.be, 1997.
[9] 김용민외 5, "분산객체 환경을 위한 투명한 보안 기반의 설계 및 구현", 1st OSTA, 한국통신정보보호학회, pp. 135-141, 1997. 11.

## 김 용 민

1989년 전남대학교 전산통계학과(학사)
1991년 전남대학교 대학원 전산통계학과(석사)
1996년~현재 전남대학교 대학원 전산통계학과 박사과정
관심분야 : 통신망 관리 및 보안망 서비스 프로토콜

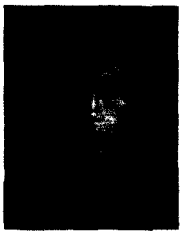
## 이 도 헌

1990년 한국과학기술원 전산학과(학사)
1992년 한국과학기술원 전산학과(석사)
1995년 한국과학기술원 전산학과(박사)
1996년~현재 전남대학교 전산학과 전임강사
관심분야 : 데이타 마이닝, 데이타 웨어하우징, 퍼지데이타베이스

## 노 봉 남

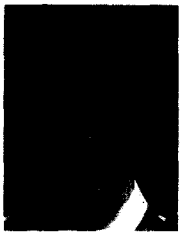
1978년 전남대학교 수학교육과 졸업(학사)
1982년 한국과학기술원 전산학과(공학석사)
1994년 전북대학교, 대학원 전산통계학과(이학박사)
1983년~현재 전남대학교 전산학과 교수
관심분야 : 객체지향 시스템, 통신망관리, 정보 보안, 컴퓨터와 정보사회 등

## 최 락 만



1977년 한양대학교 전자공학과(학사)
1987년 한양대학교 산업대학원 전산학과(석사)
1977~현재 한국전자통신연구원 책임연구원
관심분야 : 정보통신 시큐리티, 분산정보처리, 멀티미디어

## 인 소 란



1978년 홍익대학교 전자계산학과 졸업
1982년 홍익대학교 전자계산학과(석사)
1987년 정보처리기술사 취득(전자계산기 조직 응용분야)
1991년 홍익대학교 전자계산학과(박사)
1978년~현재 한국전자통신연구원 S/W공학연구실장(책임연구원)