역할 위임을 위한 ERBAC 설계

오 석 균† · 김 성 열††

요 약

본 논문은 분산 서버 환경에서 다양한 업무를 운영하려고 할 때에 발생되는 보안상의 문제를 해결하기 위하여 RBAC(Role Based Access Control) 기법을 이용하여 분산 환경에서 운영 가능한 역할 위임을 위한 확장된 RBAC(Extended RBAC : ERBAC) 모형을 설계하였다. ERBAC는 기본적으로 Sandhu 등이 제안한 RBAC96 모형에 역할 위임부분을 추가하였다. 따라서, 역할을 위임하기 위해 ERBAC를 이용하면 사용자 수준의 역할 위임이 가능하여 업무의 중단없이 영속성을 보장할 수 있다. 또한 분산 서버의 소스코드 수정 없이 구현 가능하고, 이식성이 높으며, 보안 관리가 단순하고 용이하다는 장점을 갖는다.

Design of the ERBAC for Role Delegations

Sug Kyun Oh† · Seong Ryeol Kim††

ABSTRACT

This paper applies RBAC policy for solving on the security problems when it will be operated several business on the distributed environments and designed Extended RBAC (ERBAC) model that it is possible to manage security systems on the distributed environments. The designed ERBAC model is based on RBAC96 model due to Sandu et al and added role delegations. Therefor, the designed ERBAC model have the advantage of the following : it can be processed of business without interrupts and implemented server system without modifying its source code, high migration, easy and simple of secure managing.

키워드 : RBAC, 역할기반, 역할위임, 보안

1. 서 론

정보시스템들이 분산 환경에서 개발됨으로 인하여 네트워크를 이용하는 업무들이 증가되고 있다. 따라서 누구나 쉽게 네트워크를 통해 정보에 접근할 수 있기 때문에 보안 문제가 대두되고 있다. 이러한 문제를 해결하기 위해 역할 기반접근제어(RBAC)[1]를 이용한다.

RBAC의 기본 개념은 허가(permission)가 역할(role)과 관련되고, 사용자에게 적절한 역할을 할당[1, 3]한다는 것이다. 이러한 개념은 역할 허가 관리를 매우 단순화 시켰으며, 역할이 조직 내에서 다양한 작업의 기능에 따라 생성되고 사용자의 책임과 자격을 근거로 사용자에 할당된다. 역할은 특정 태스크에서 하는 능력을 나타낼 수도 있고, 권한과 책임을 구체적으로 명기할 수 있어 여러 사용자에게 특정 임무를 순환 시켜가며 할당[1, 2]할 수도 있다. 그러나 이러한 RBAC 모형은 사용자 부재시 할당받은 역할을 계속적으로 수행할 수 있다는 근거가 없다. 따라서 본 논문에서는 역할 수행의 영속성을 보장할 수 있도록 역할을 사용자 수준에서 위임이 가능하도록 역할 위임을 위한 확장된 ERBAC (Extended RBAC)를 설계 제안하고자 한다.

† 정 회 원 : 충청대학 컴퓨터학부 교수
†† 정 회 원 : 청주대학교 컴퓨터정보공학과 교수
논문접수 : 2003년 2월 26일, 심사완료 : 2003년 10월 15일

2. 역할위임 기반 ERBAC 설계

2.1 RBAC 모형군의 역할 위임

Sandhu 등이 정의한 초기의 모형은 RBAC96[1, 3]이며, 역할의 관리를 위해 관리 역할 계층을 두고 상위역할이 하위역할보다 더 많은 권한을 갖도록 설계한 ARBAC97(Administrative RBAC97)[2, 4, 5]를 발표되었으며, 기본적인 기능과 특징은 [1, 3]에 잘 정의되어 있다. 이러한 RBAC 모형들을 RBAC 모형군이라고 한다.

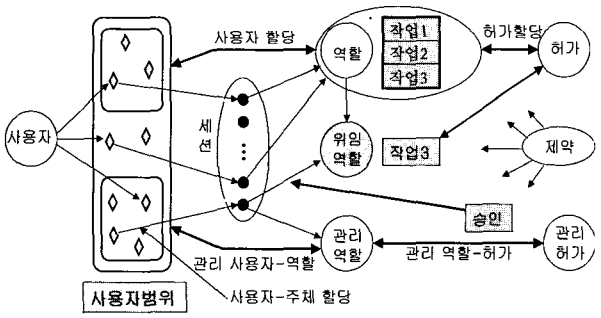
이러한 RBAC 모형군에서 제시된 역할 계층의 상속 개념과 실제 기업 조직의 관리 규칙이 잘 조화되지 않는 점이 존재한다. 역할 계층상의 상위 역할과 하위 역할은 서로 직무분리(separation of duty)의 관계를 가질 수는 없어, 하위 역할의 허가를 상위 역할이 모두 상속하기 때문에 관리

규칙[6]의 오류가 있다. 따라서 감독(supervision)의 경우에도 동일한 문제가 발생한다. 상속을 통해 하위 역할이 가진 허가를 수행할 수 있는 상위 역할이 자신이 수행할 수 있는 허가에 대한 감독을 한다는 것은 정당하지 못하다. 그리고 위임의 경우에도 허가의 분배 입장에서 허가를 위임하였어도, 역할 상속에 의해 계속 위임한 허가를 유지하게 되는 문제가 존재한다.

따라서 기존의 RBAC 모형군은 이러한 허가위임 정책이 실질적으로 현실에 적합하지 않다는 것이다. 이러한 문제를 해결하기 위해서 허가위임 정책을 상위 역할에 종속시키는 것이 아니라 필요에 따라 하위 역할 소유자가 해당 역할을 적합하게 수행할 수 있도록 자기 역할을 위임 할당할 수 있도록 역할 위임기반의 확장된 RBAC(Extended RBAC : ERBAC)를 설계 제안하고자 한다.

2.2 ERBAC 역할 위임 설계

기존의 RBAC 모형을 사용해서는 위임을 제대로 반영할 수 없다. 역할 위임이란 어떤 객체가 가진 역할의 일부 또는 전체를 다른 객체에 부여하여 그 객체가 위임된 역할을 수행할 수 있도록 하는 것을 의미[6]한다. 위임을 위한 여러 가지 조건이나 제약사항을 ARBAC97과 같이 정규역할의 외부에 놓는 것[2]이 아니라, 역할 자체와 관련된 속성으로 정의 설계함으로써 역할 위임을 구현할 수 있도록 설계하여 제안하는 모형은 (그림 1)과 같다.



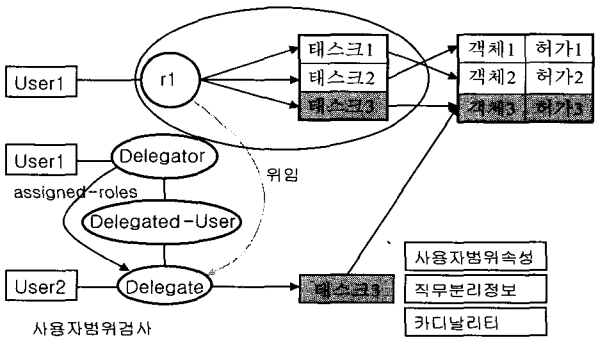
(그림 1) 역할 위임 기반 ERBAC 모형

그림과 같이 제안한 모형에서 사용자 수준의 위임 역할은 사용자가 위임할 허가의 집합으로 구성된 새로운 역할을 생성하며, 위임 역할은 생성한 사용자가 소유권을 가지게 되고, 생성된 위임 역할에 DAC 기법[6]을 적용시켜 다른 사용자에게 허가를 위임할 수 있도록 위임 역할 생성과 위임감독을 다음과 같이 정의하여 설계한다.

2.3 위임 역할 생성

역할을 위임하려는 사용자는 위임할 역할 허가를 포함하는 새로운 역할을 생성할 수 있으며, 새로운 역할은 사용자가 이미 할당되어 있는 역할의 부분집합이 되며, 이렇게 생성된 역할은 역할을 생성한 사용자의 소유가 된다.

(그림 2)는 본 논문에서 제안한 역할위임 기반 ERBAC 모형에서의 사용자 수준의 위임관계를 음영처리부분과 점선으로 관계를 나타내고 있다. 그림에서 사용자 $User_1$이 자신에 할당된 역할 r1의 일부분을 위임하려고 한다면, r1의 부분집합으로 구성된 역할 Delegate를 생성하며, 이 Delegate의 소유자는 $User_1$이 되고 Delegate는 r1이 가진 허가의 일부를 수행할 수 있다. 이렇게 역할 Delegate가 생성되면 자동적으로 Delegate에 대한 Delegator, Delegated_User 역할이 생성되도록 설계한다.



(그림 2) 역할 r1의 일부를 위임하는 과정

이러한 사용자 수준의 역할 위임을 수행하기 위하여 사용자 위임역할 생성권한, 할당된 역할의 일부 위임역할 할당, 역할 생성함수, 위임역할 허가, 위임역할 사용자 할당과 제거에 대한 정의는 [정의 1]에서 [정의 5]와 같이 정의하여 ERBAC에 적용 설계한다.

[정의 1] 사용자 위임 역할 생성 권한

$\forall r_1$, Delegate_Role $\in$ ROLES, $\forall u \in$ USERS,
$\forall u \in$ *assigned-users*(r_1) $\Rightarrow$ *create_delegate_role*(r_1)
$\rightarrow$ Delegate_Role, Delegate_Role $\in$ *assigned-roles*(u)

[정의 1]에서 Delegate_Role는 역할 r_1로부터 생성된 위임 역할이다. 사용자는 자신이 할당받은 역할 중의 일부분을 위임하기 위해 새로운 역할로 위임 역할을 생성할 수 있다. 역할의 일부분이란 각 역할을 이루는 여러 태스크 중에서 필요한 태스크만 선택하여 형성된 것을 의미한다. 그리고 이렇게 생성된 위임 역할을 자신에게 할당할 수 있어야 한다. 이를 형식 명세로 표기하면 [정의 2]와 같다.

[정의 2] 할당된 역할의 일부 위임역할에 할당

$\forall r \in$ ROLES, $\forall u \in$ USERS, $\forall r \in$ *assigned-roles*(u),

$t \in$ *assigned-tasks*(r) $\Rightarrow$ *assign_delegated_role* (Delegate_Role) $\rightarrow$ t

새로 생성된 위임 역할을 관리하기 위해서는 시스템에서 자동으로 위임역할을 위임한 위임자 역할인 Delegator_Role과 위임역할을 위임받은 사용자 역할인 Delegated_User_Role 역할을 생성하여야 하며 이들 역할에 해당하는 허가도 생성해야 한다. 따라서 이들을 생성할 수 있는 함수가 있어야 한다. 이와 관계된 함수들은 [정의 3]과 같이 정의하여 사용한다.

[정의 3] Delegator_Role과 Delegated_User_ Role 역할 생성 함수

- 위임자 역할 Delegator_Role를 생성 함수

  create_delegator_role(Delegate_Role) $\rightarrow$ Delegator_Role

- 위임받은 사용자 역할 Delegated_User_ Role를 생성 함수

  create_delegated_user_role(Delegate_Role) $\rightarrow$ Delegated_User_Role

여기서 Delegate_Role은 [정의 1]의 함수 create_delegate_role(r)를 통해 생성된 위임 역할로 각각의 역할은 [정의 4]와 같이 Delegator_Role 역할의 허가를 갖는다.

[정의 4] Delegator_Role 역할의 허가

Delegate_Role $\in$ ROLES

destroy_delegate_role(Delegate_Role)

[정의 3]에서 자동 생성된 역할 Delegator_Role은 위임역할을 생성한 사용자에게 자동적으로 할당되어 역할을 위임하거나 회수할 수 있는 허가를 갖는다.

[정리 1] 생성된 Delegator_Role 역할은 위임역할을 생성한 사용자에게 할당된다.

$\exists u \in$ USERS, Delegator_Role $\in$ ROLES

assigned-users(Delegator_Role) = u,

cardinality(Delegator_Role) = 1

[증명] Delegator_Role은 [정의 3]에 의해 생성된 것으로 Delegate_Role 위임 역할을 관리하기 위해서 시스템에서 자동으로 생성한 위임자 역할이다. Delegate_Role 역할은 [정의 1]에 의해 생성된 위임 역할로 역할 r에서 위임된 역할이다. 따라서 [정의 1]에서 Delegate_Role $\in$ *assigned-roles*(u)이므로 *assigned-users*(Delegator_Role) = u이다. 따라서 *cardinality*(Delegator_Role) = 1이다.■

[정의 5] Delegate_Role, Delegated_User_Role에 사용자의 할당과 제거

- Delegate_Role에 사용자를 할당

  AssignUser_Delegate_role(u, Delegate_Role)

- Delegate_Role에서 사용자를 제거

  RevokeUser_Delegate_role(u, Delegate_Role)

- Delegated_User_Role 역할에 사용자를 할당

  AssignUser_Delegated_User_Role(u, Delegate_Role)

- Delegated_User_Role 역할에서 사용자를 제거

  RevokeUser_Delegated_User_Role(u, Delegate_Role)

각 역할 Delegate_Role, Delegated_User_Role, Delegator_Role은 계층 관계와 허가 상속을 통하여 역할위임과의 위임 정도를 관리한다.

[정리 2] 위임 역할의 역할 계층에 의한 허가 상속

Delegator_Role, Delegated_User_Role,
Delegate_Role $\in$ ROLES, u_1, u_2, $u_3 \in$ USERS
($u_1 \in$ *assigned-users*(Delegate_Role) $\rightarrow$ $^*u_2 \in$ *assigned-users*(Delegated_User_Role)) $\cap$
($u_2 \in$ *assigned-users*(Delegated_User_Role) $\rightarrow$ * $u_3 \in$ *assigned-users*(Delegator_Role))

[증명] 각 위임 역할 Delegate_Role, Delegated_User_Role, Delegator_Role은

Delegate_Role$\ll$Delegated_User_Role$\ll$Delegator_Role

관계를 가지며, RBAC96 모형의 기본정리의 *inherits* 함수에 의해

$u_1 \in$ *assigned-users*(Delegate_Role)

인 사용자 u_1의 허가를

$u_2 \in$ *assigned-users*(Delegated_User_Role)

인 사용자 u_2에 상속할 수 있다. 즉,

$u_1 \in$ *assigned-users*(Delegate_Role) $\rightarrow$ $^*u_2 \in$ *assigned-users*(Delegated_User_Role) (1)

이다. 그리고,

$u_2 \in$ *assigned-users*(Delegated_User_Role)

인 사용자 u_2의 허가를

$u_3 \in$ *assigned-users*(Delegator_Role)

인 사용자 u_3에 상속할 수 있다. 즉,

$u_2 \in$ *assigned-users*(Delegated_User_Role) $\rightarrow {}^{*}u_3 \in$ *assigned-users*(Delegator_Role) (2)

이다. 따라서, 위임 역할 Delegator_Role이 위임 역할 Delegate_Role로부터 상속받을 수 있는 허가는 u_1이 u_2에 상속한 허가 중에서 일부 또는 전체를 u_2가 u_3에 허가를 상속하므로 식 (1)과 식 (2)의 공통집합인

$(u_1 \in$ *assigned-users*(Delegate_Role) $\rightarrow {}^{*}$ $u_2 \in$ *assigned-users*(Delegated_User_Role)) $\cap$ $(u_2 \in$ *assigned-users*(Delegated_User_Role) $\rightarrow {}^{*}$ $u_3 \in$ *assigned-users*(Delegator_Role))

이다.■

[정리 3] Delegated_User_Role 역할의 사용자 수(cardinality)를 조정하여 위임받는 사용자의 수를 제한한다.

Delegated_User_Role $\in$ ROLES, $\forall u \in$ USERS
$\forall u \in$ *assigned-users*(Delegated_User_Role) $\leq$ *cardinality*(Delegated_User_Role)

[증명] RBAC96 모형의 기본정리[1, 3]에 의해 *assigned-users*(Delegated_User_Role)는 위임역할 Delegated_User_Role에 할당된 사용자 집합이 되고, *cardinality*(Delegated_User_Role)는 위임역할 Delegated_User_Role을 사용할 수 있게 권한이 부여된 최대 사용자수가 된다. 그러므로 Delegated_User_Role에 할당된 사용자는 최대 사용자 수를 초과하지 못한다. 즉,

$\forall u \in$ *assigned-users*(Delegated_User_Role) $\leq$ *cardinality*(Delegated_User_Role)

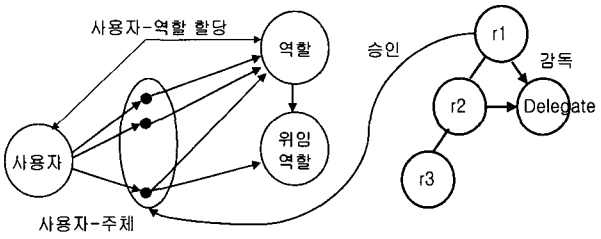
이다.

따라서 Delegated_User_Role 역할의 cardinality를 변화시키게 되면 최대 사용자 수를 초과하지 않게 위임받을 수 있는 사용자수가 조정되어야 된다. 고로, Delegated_User_Role 역할의 사용자 수를 조정하게 되면 위임받는 사용자의 수는 그에 따라 제한을 받는다.■

따라서 [정의 1]~[정의 5]의 위임에 대한 기본적인 정의는 최초 권한을 위임한 사용자에 의해서 모든 위임된 허가가 회수되고, 허가는 위임한 후에도 계속 수행할 수 있도록 설계한다. 이러한 위임된 허가의 회수는 허가-독립 회수(Grant-Independent Revocation)를 기본으로 하여 설계하였다.

2.4 위임 감독

역할 위임의 적합성을 판단하기 위하여, 상위 역할 또는 보안 관리자가 사용자 범위 내에서 이루어지는 위임에 대한 적합성 여부를 승인하도록 하는 방법으로 위임 감독을 제시한다. 위임역할은 새로 생성된 역할이기 때문에 다른 역할과의 계층관계를 갖지 않고 위임역할을 감독할 만한 역할이 존재하지 않는다. 이러한 점을 보완하기 위해서 다음 그림과 같이 위임역할과 감독허가를 상속받도록 설계한다.



(그림 3) 위임 역할에 대한 감독 역할 및 위임 역할의 활성화

그림에서 위임 역할의 모태가 되는 역할이 위치하는 역할 계층에서 상위 역할이 가지는 감독허가에 대한 정보를 상속받는다. 따라서 새로 생성된 위임역할에 대한 감독은 기존 역할에 대한 감독을 하는 역할 계층상의 상위 역할이 위임역할에 대한 감독을 수행할 수 있게 된다. 따라서, 위임역할에 대해 사용자 할당이 일어난 경우, 위임을 받은 사용자가 위임역할을 활성화하기 위해 상위 감독역할의 사용자에 의해서 정적 역할-사용자 할당이 적합한 위임이라는 것을 승인한 후에야 수행할 수 있도록 정의 적용해야 하는데 관련된 정의는 [정의 6]에서 [정의 8]과 같이 정의하여 적용한다.

[정의 6] 주체는 관련된 사용자에게 허가된 역할만을 활성화한다.

$\forall r \in$ ROLES, $\forall s \in$SUBJECTS
$r \in$ *subjects-active-roles*(s) $\Rightarrow$ *assigned-subjects-users*(s)$\in$*assigned-users*(r)

여기에 상위 역할에 의한 감독이 추가된다.

[정의 7] r의 위임역할 Delegate_Role가 생성되면, 위임역할 Delegate_Role은 다음과 같은 허가를 갖는다.

$\forall r$, Delegate_Role $\in$ ROLES, $r \rightarrow {}^{*}$Delegate_Role,
$\exists u \in$ *assigned-users*(Delegate_Role)
u = delegate_permit(Delegate_Role)

[정의 8] 위임역할에 할당된 사용자가 아래 조건을 만족해야 해당되는 위임역할을 활성화한다.

$\forall s \in SUBJECT, Delegate\_Role \in ROLES$
$Delegate\_Role \in assigned\text{-}roles(s) \Rightarrow$
$assigned\text{-}subjects\text{-}users(s) \in assigned\text{-}users$
$(Delegate\_Role) \wedge delegate\_permit(Delegate\_Role)$

3. ERBAC 모형구성과 동작과정

3.1 구성 요소

제안된 역할위임 기반의 ERBAC 모형을 구현하려면 기본적인 RBAC 모형의 구성요소[1-5]를 사용할 수 있으며, ERBAC 서버, 데이터베이스, API 라이브러리에 Visual C, C++ 라이브러리 등을 추가하고 CGI, Perl, JAVA 뿐만 아니라 PHP 기법을 사용할 수 있어 구현시 분산환경에서 서버의 소스코드 수정 없이 구현 가능하고, 이식성이 높다. 또한 세션 관리자와 관리 도구도 RBAC96 모형에서 제안된 내용에 역할 위임부분을 추가 확장하여 구성함으로 보안 관리가 단순하고 용이하다는 특성을 가질 수 있다. 이러한 구성요소의 기능은 RBAC96 모형군과 동일하다.

3.2 관리자와 사용자 접근과정

관리자는 관리 도구를 이용하여 ERBAC 데이터베이스에 접근하여 사용자/역할과 역할/역할 관계를 관리하며, 사용자/역할 할당, 역할 계층, 정적 직무분리 제약, 상호 배타적 직무분리 제약, 개방적 직무분리 제약, 역할 카디날리티 명세를 관리한다.

사용자의 접근 과정은 ERBAC 모형이 사용자 접근은 접근이 허가되기 전에, 사용자는 ERBAC 세션을 확립해야만 사용자가 요구한 접근을 허가한다. 이러한 ERBAC 세션은 사용자가 작업할 수 있는 ARS(Active Role Set) 중에서 필요한 ARS를 선택함으로써 설정되고, 사용자에 선택한 ARS가 할당된다. 이 ARS는 사용자가 ERBAC에 의해 통제된 접근 방법을 결정하며, 사용자가 새로운 ARS를 확립할 때까지 영향 내에 있다. ERBAC 세션 관리자는 사용자들의 ARS를 변경하여 그 사용자를 허용하게 한다.

3.3 사용자 인증

설계 제안한 ERBAC 모형은 패스워드, SSL, Secure HTTP 등이 포함하고 있는 인터넷 인증과 기밀 서비스를 결합하여 사용할 수 있는 접근통제 메커니즘을 이용[1-5] 할 수 있다.

서버에 인증된 사용자만이 ERBAC 세션을 설정할 수 있다. 따라서, 서버에서 사용자의 인증이 삭제되면 사용자의 접근을 거부한다. 또한 사용자 인증과 ERBAC 세션의 설정이 완벽하게 분리되어 동작하므로 설계 제안된 ERBAC 모형은 서버에서 제공되는 인증 메커니즘 내에서 작업이 이루어지게 된다.

4. 결 론

본 논문은 분산 서버 환경에서 다양한 업무를 운영하려고 할 때에 발생되는 보안상의 문제를 해결하기 위하여 RBAC 기법을 이용하여 분산 환경에서 운영 가능한 역할 위임을 위한 확장된 RBAC 모형을 설계 제안하였다.

설계 제안한 역할 위임을 위한 ERBAC 모형은 기존의 RBAC가 제공하고 있는 역할 기반의 접근 통제방식에 사용자 부재시 역할을 위임받아 역할 수행의 중단 없이 계속적으로 수행할 수 있도록 역할 수행의 영속성을 보장할 수 있도록 위임역할 생성과 위임감독부분을 정의 설계하여, Sandu가 제안한 RBAC96 모형군에 역할 위임부분을 추가 확장하여 설계 제안하였다.

설계 제안한 역할 위임을 위한 ERBAC를 이용하면 사용자 수준의 역할 위임이 가능하여 업무의 중단없이 영속성을 보장할 수 있다. 또한 RBAC96 모형군과 같이 분산환경에서 서버의 소스코드 수정 없이 구현 가능하고, 이식성이 높으며, 보안 관리가 단순하고 용이하다는 특성을 갖고 있다.

참 고 문 헌

[1] D. Ferraiolo, J. Cugini and D. R. Kuhn, Role Based Access Control : Features and Motivations, In Annual Computer Security Applications Conference, 1995.

[2] R. Sandhu and V. Bhamidipati, The URA97 Model for Role-Based User-Role Assignment, Proc. of IFIP WG 11.3 Workshop on Database Security, Aug., 1997.

[3] R. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, Role-Based Access Control Models, IEEE Computer, Vol. 29, No.2, pp.38-47, Feb., 1996.

[4] R. Sandhu, V. Bhamidipati, E. Coyne, S. Ganta and C. Youman, "The ARBAC97 Model for Role-Based Administration of Roles : Preliminary Description and Outline," Proceedings of Second ACM Workshop on RBAC, Fairfax, Virginia, November, 1997.

[5] R. Sandhu and Q. Munawer, "The RRA97 Model for Role-Based Administration of Roles Hierarchies," ACSAC, 1998.

[6] J. Linn and M. Nystrom, "Attribute certification : an enabling technology for delegation and role-based controls in distributed environments," Proceedings of the fourth ACM workshop on Role-based access control, Fairfax, VA USA, pp.121-130, October, 1999.

오 석 균

e-mail : osk58@ok.ac.kr

1981년 숭실대학교 전자계산학과 공학사
1983년 숭실대학교 대학원 전자계산학과 공학석사
2002년 청주대학교 대학원 전산정보공학 공학박사
1986년~1991년 전주기전여자대학 전자계산과 조교수
1999년~2001년 충청대학 컴퓨터학부 학부장
1991년~현재 충청대학 컴퓨터학부 부교수
관심분야 : 컴퓨터네트워크, 컴퓨터 보안, 운영체제, 멀티미디어

김 성 열

e-mail : srkim@chongju.ac.kr

1982년 숭실대학교 전자계산학과 공학사
1987년 숭실대학교 대학원 전자계산학과 공학석사
1992년 숭실대학교 대학원 전자계산학과 공학박사
1982년~1984년 한국전력공사 전자계산소 근무
1984년~1990년 오산대학 전자계산과 교수
1997년~1998년 호주 QUT ISRC 객원교수
1990년~현재 청주대학교 컴퓨터정보공학과 교수
관심분야 : 컴퓨터네트워크, 컴퓨터 보안, 분산객체시스템