# 3장. 접근제어
## Access Control

박종혁
서울과학기술대학교 컴퓨터공학과
jhpark1@seoultech.ac.kr

# 목차

# 3-1. 접근 제어 원리

# 1. 접근 제어 원리

- **접근제어 원리(Access Control Principles)**

  - **정의:** 접근 제어를 특정 요청에 대해 허가 혹은 거부하는 프로세스 (NIST IR 7298, Glossary of Key Information Security Terms)
    1. 정보의 획득과 사용, 관련된 정보 처리 서비스
    2. 특정 물리적 장비에 진입

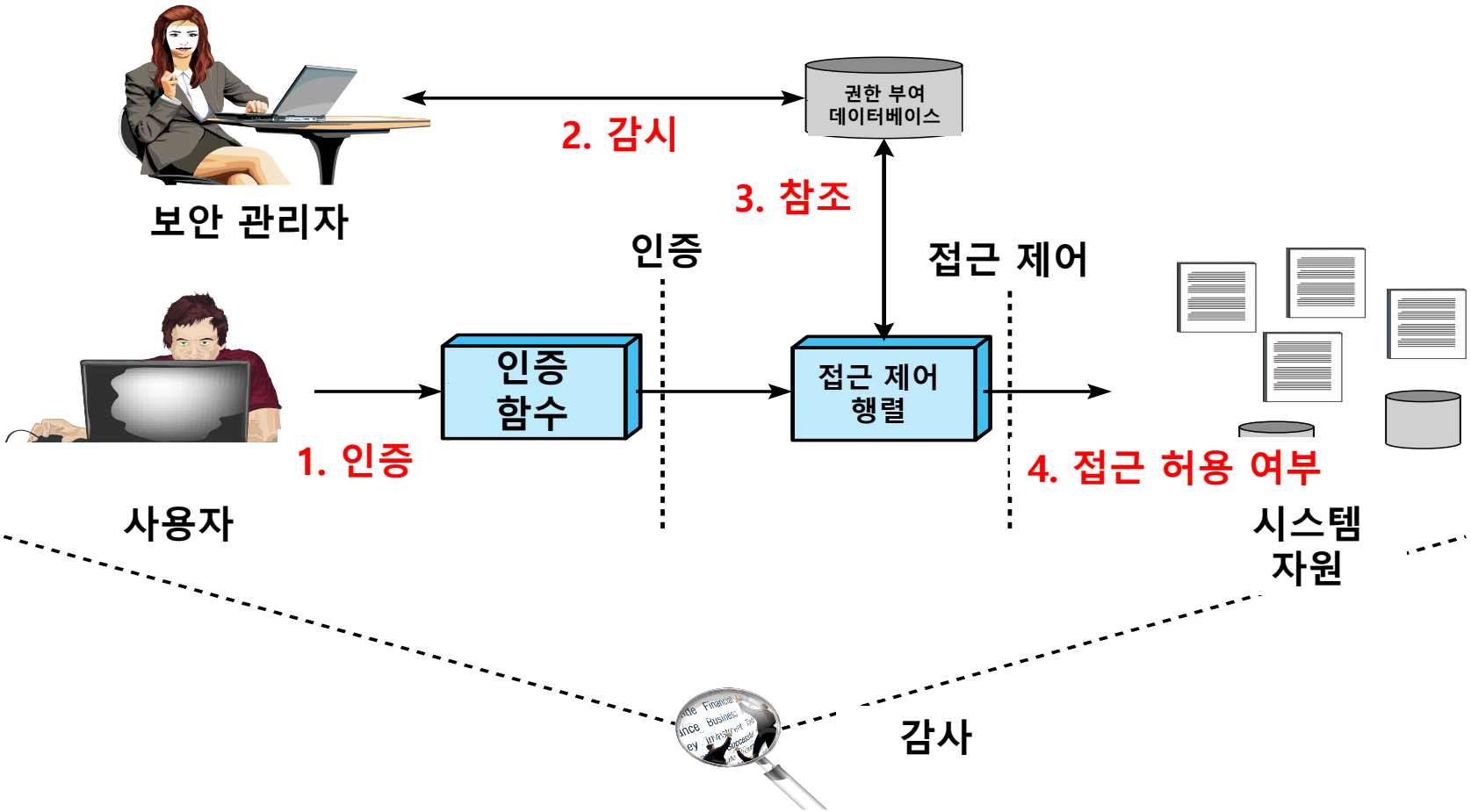
  - RFC 4949
    - 접근 제어를 보안 정책에 따라 시스템 자원 사용을 규제하고 정책에 따라 인가된 존재(사용자, 프로그램 등)에게만 사용을 허가

- **접근제어 상황 – 1 (Access Control Situation)**
  - **인증 :** 사용자 또는 다른 시스템 존재의 자격이 유효한지 검증
  - **허가 :** 특정 시스템 자원에 접근할 수 있는 권한이나 허가를 다른 시스템 존재에게 승인
  - **감사 :** 접근제어에서 나타난 모든 변화를 알리기 위해서 시스템 레코드와 활동에 대한 독립된 검토와 조사
    - 시스템 컨트롤 적절성 테스트
    - 시스템내 발행된 정책 검사
    - 시스템 운영상의 절차에 맞게 운용되는지 확인
    - 보안 위반사항 및 정책, 컨트롤, 절차 등에서 취약점 발견

- **접근제어 상황 – 2 (Access Control Situation)**

  - 접근제어 시스템은 가장 처음에 접근하려고 하는 존재를 인증 해야 함
    - 인증 함수는 사용자가 시스템에 접근할 수 있는 지의 승인 여부를 결정
  - 보안 관리자는 이 사용자에게 접근이 승인된 자원에 어떻게 접근하는지를 명시하는 권한 데이터베이스를 관리
  - 접근제어 함수는 접근을 허용해줄지 여부를 결정하기 위해서 이 데이터베이스를 참조
  - 접근제어 함수는 이 사용자에 의해서 요청된 접근이 허용되었는지의 여부를 결정

# 제어와 다른 보안함수들 사이에서의 관계



# 접근 제어 정책(Access Control Policies)

- **임의 접근 제어(Discretionary Access Control, DAC)**
  - 접근을 요청하는 자의 신원, 어떤 사람이 접근 승인이 되는지를 말해주는 접근 규칙들에 기반하는 접근 제어
- **강제 접근 제어(Mandatory Access Control, MAC)**
  - 보안레벨과 허가를 비교 기반 접근 제어
    - 보안레벨:시스템 자원이 얼마나 민감하고 중요한지를 나타냄
    - 보안 허가: 어떤 시스템 개체가 특정 자원에 접근할 수 있는지를 나타냄
- **역할 기반 접근 제어(Role-based Access Control, RBAC)**
  - 시스템 내에 사용자가 가지는 역할들, 그 역할을 맡은 사용자에게 허용되는 접근 규칙들 기반 접근 제어
- **속성 기반 접근 제어(Attribute-based Access Control, ABAC)**
  - 사용자, 접근되는 자원, 현재 환경 조건의 속성에 기반한 접근 제어
- **자격 기반 접근 제어(Capability Based Access Control, CapBAC)**
  - 최소 권한 원칙과 권한 위임 기능을 부여, 주체에게 자신의 서비스 및 정보에 대한 접근 제어 관리

# 3-2. 주체, 객체, 접근권한

# 2. 주체, 객체, 접근 권한

- **주체(Subject)**
  - 객체에 접근할 수 있는 존재, 모든 사용자나 응용프로그램의 대표 하는 프로세스의 도움으로 객체에 접근
    - **소유자** :
      - 파일과 같은 자원을 만든 사람
      - 시스템 자원: 소유권은 시스템 관리자에 속할 수 있음
      - 프로젝트 자원: 프로젝트 관리자나 지휘자에 소유권이 할당
    - **그룹** : 소유자에게 할당된 특권뿐만 아니라, 지정된 그룹 사용자들 또한 그룹의 회원들이 충분히 행사할 수 있는 접근 권한 승인 받을 수 있음
    - **전체** : 시스템 접근과 자원에 대해 소유자나 그룹의 범주에 속해 있지 않은 사용자에게 접근 승인
- **객체(Object)**
  - 접근이 제어되는 자원, 정보를 포함 또는 받기 위해서 쓰이는 존재
    - 예) 레코드, 블록, 페이지, 세그먼트 파일
    - 몇몇 접근 제어 시스템은 비트, 바이트,워드 등 또한 포함할 수 있음

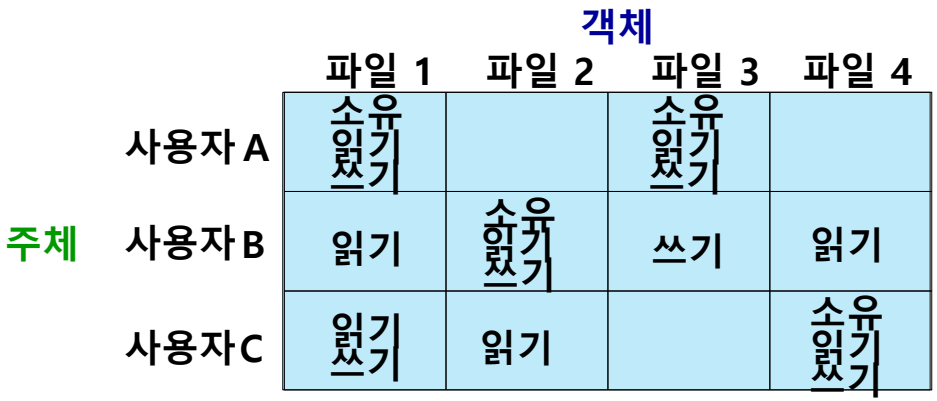
- **접근 권한 (Access Rights)**
  - 주체가 객체에 접근하는 방법
    - 읽기 : 사용자는 시스템 자원에 대한 정보를 볼 수 있음
      - ➢ 읽기 권한은 복사와 프린트 읽기 권한 포함
    - 쓰기 : 사용자는 시스템 자원(예 - 파일,레코드, 프로그램)의 데이터를 추가, 수정, 삭제할 수 있음
      - ➢ 쓰기 권한은 읽기 권한을 포함
    - 실행 : 사용자는 특정 프로그램을 실행할 수 있음
    - 삭제 : 사용자는 파일, 레코드 등 특정 시스템 자원을 삭제할 수 있음
    - 생성 : 사용자는 새로운 파일이나 레코드,필드 생성할 수 있음
    - 검색 : 사용자는 디렉터리 안에 있는 파일 목록 만들 수 있거나 디렉터리를 검색할 수 있음

# 3-3. 임의 접근 제어

# 3. 임의 접근 제어(DAC)

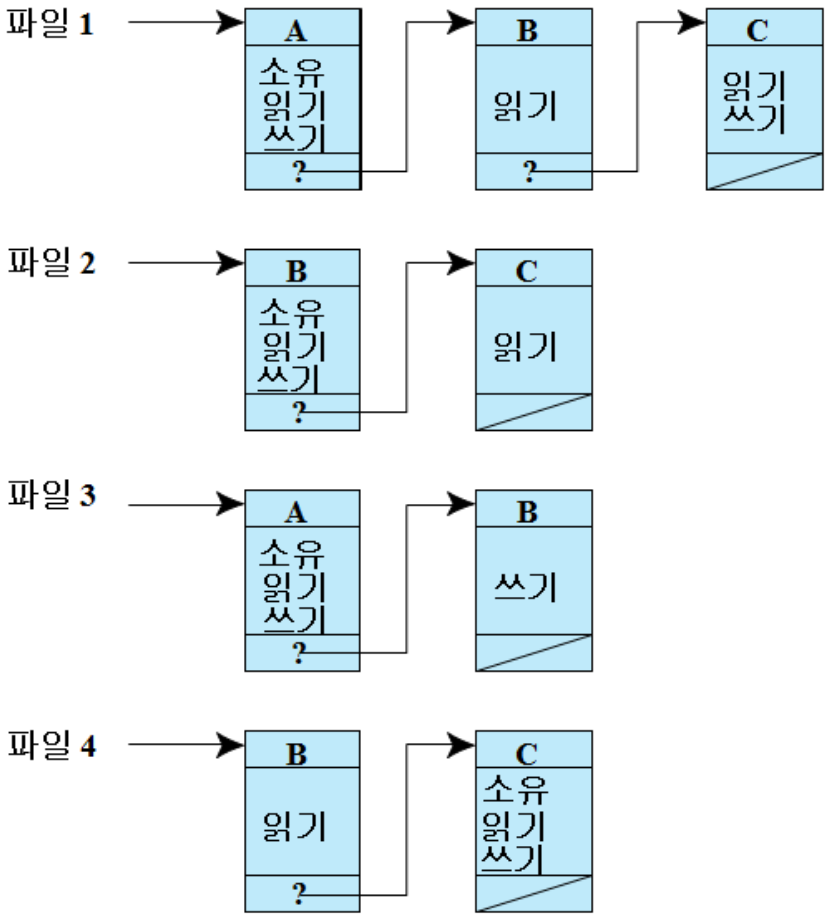
- **DAC(Discretionary Access Control)**
  – 한 존재가 자신의 의지대로 다른 존재가 자원에 접근 할 수 있도록 허용할 수 있는 접근 권한을 승인 받을 수 있음

- **접근 제어 행렬(Access control Matrix)**
  - 접근 행렬의 한쪽 차원은 자원에 접근을 시도하는 확인된 주체(사용자, 터미널,네트워크 장비, 호스트 등)로 구성
  - 다른 쪽의 차원은 접근이 되는 객체(레코드, 파일 등) 이루어져 있음
  - 사용자 A는 파일 1과 파일3을 소유 그 파일들에 대해서 읽기와 쓰기 권한을 가짐
  - 사용자 B는 파일 1과 파일4에 대해서 읽기 권한만 가지고 있음



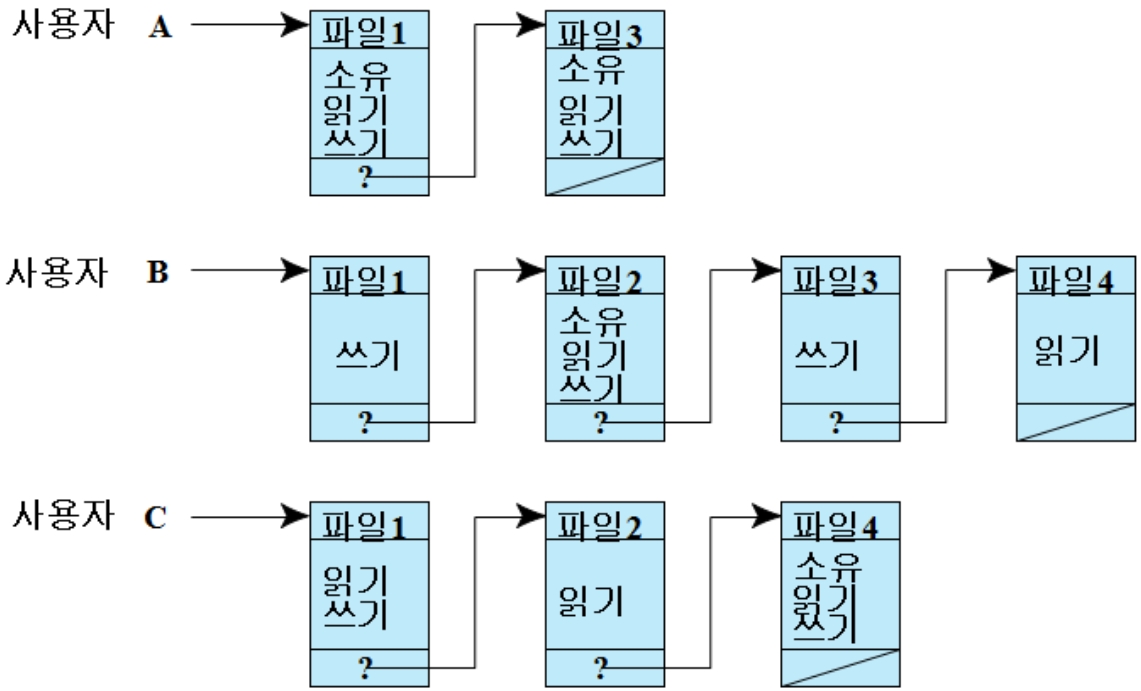

[ 접근 제어 구조의 예]

- **접근제어 목록**
  **(Access control lists: ACLs)**
  - 접근 행렬은 행으로 분리될 수 있음
  - 각 행렬은 객체 중점으로 객체에 접근 가능한 주체들의 리스트임
  - 특정 자원에 대해 특정 접근 권한을 가진 주체를 결정할 때는 ACL이 편리함
  - 특정 사용자가 어떤 접근 권한을 이용할 수 있는지를 결정하는 데에는 불편함



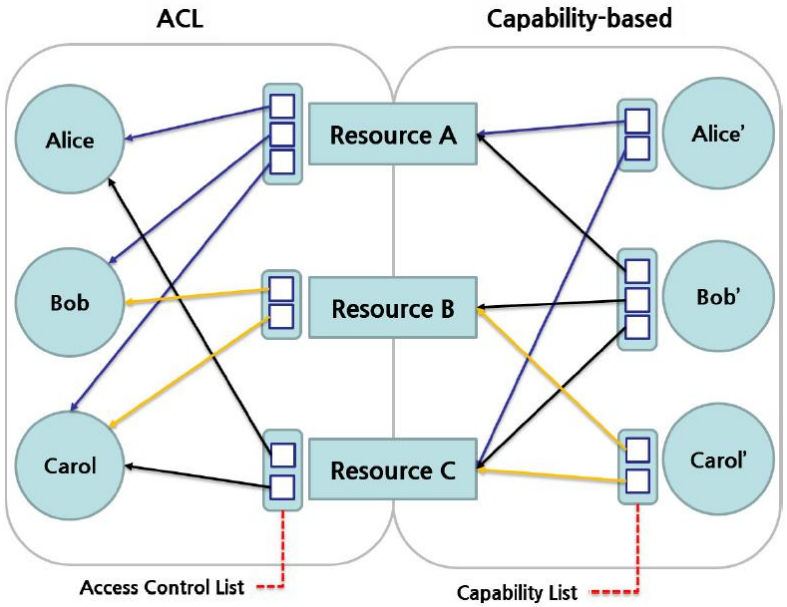

**[접근제어 목록의 파일 파트]**

- **가용성 티켓 (Capability tickets)**
  - 특정 사용자에 대한 승인된 객체와 기능을 말함, 주체의 중점으로 주체에 접근 가능한 객체들의 리스트
  - 티켓의 무결성은 반드시 보호 되어야 하고, 보장되어야 함
  - 콘텐츠의 보안성을 보장받지 못하는 분산 환경에 적합




**[능력 목록의 파일 파트]**

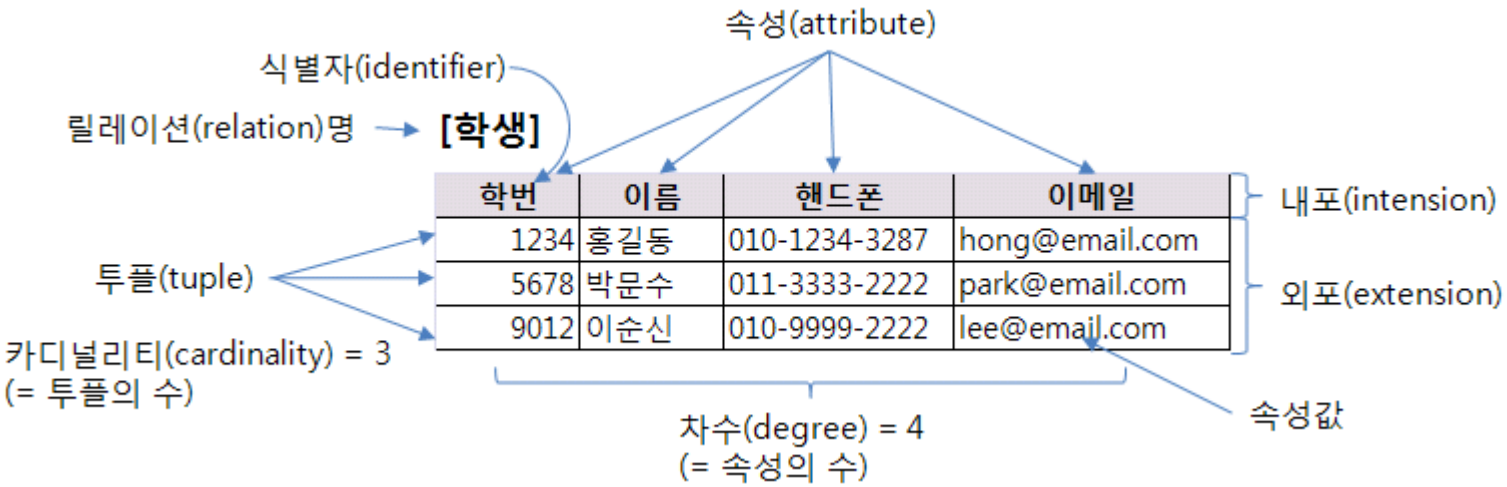
# ACL기반과 CL기반 접근제어 비교




- **ACL 기반 접근제어**
  - 사용자가 자원에 대한 연산을 요청할 경우
  - 서비스 제공자 : 사용자가 직접/간접적으로 객체에 대한 권한 여부 확인 후, 요청한 자원/연산을 수행하도록 인가
  - 권한에 대한 검증 : 접근대상이 되는 객체가 갖는 ACL을 통해 객체가 검증

- **CL 기반 접근제어**
  - 사용자가 자신이 갖고 있는 Capability를 서비스 제공자에게 제시
  - 서비스 제공자 : Capability를 확인 후 요청한 자원/연산을 수행하도록 인가
  - 권한에 대한 검증 : 객체 에 접근하고자 하는 주체가 CL을 갖고 Capability를 제시함으로써 각 객체에 접근

- **SAND94 -** 권한 테이블
  - 테이블의 한 열에 하나의 자원에 대한 한 주체의 한가지 접근 권한 포함
  - 접근 제어 목록(ACLs)과 가용성 티켓보다 더 편리한 자료 구조를 제안
  - 주체에 의해 정리되거나 접근되는 것은 가용성 티켓과 동일
  - 객체에 의해서 정리되거나 접근되는 것은 ACL과 동일
  - 관계형 데이터베이스는 이러한 종류의 승인 테이블을 쉽게 구현할 수 있음




**[관계형 데이터베이스 모델]**

# 접근 제어 모델(Access Control Model)

- 주체 집합, 객체 집합, 객체에 대한 주체의 접근을 통제하는 규칙 집합을 가정

- 시스템의 보호 상태를 특정 시점에서 각각의 객체에 대한 주체의 접근 권한을 명시하는 정보의 집합
  - 보호 상태 나타내기, 접근 권한 실행하기, 주체가 보호 상태를 특정 방법으로 변경하도록 허용

- 보호 상태를 나타내기 위해 접근 행렬에서 객체분야를 확장
  - **프로세스** : 삭제,정지 혹은 깨우는 권한
  - **디바이스** : 장비 읽기/쓰기 권한, 기능 제어 (예- 디스크 탐색), 디바이스 사용 블로킹/블로킹 해제 등의 권한
  - **메모리 지역** : 기본 권한에서는 접근 불가, 보호된 메모리의 특정 지역 읽기/쓰기 권한
  - **주체** : 다른 객체에 대한 그 주체의 접근 권한을 승인 또는 삭제하는 권한

# 확장 접근제어 행렬

- 접근행렬 A[S,X]
  - 속성이라고 불리는 문자열을 포함
  - 객체 X에 대한 주체 S의 접근 권한
  - 예) S1은 F1을 읽을 수 있음
    ➔ '읽기' 문자열 A[S1,F1]에 있음

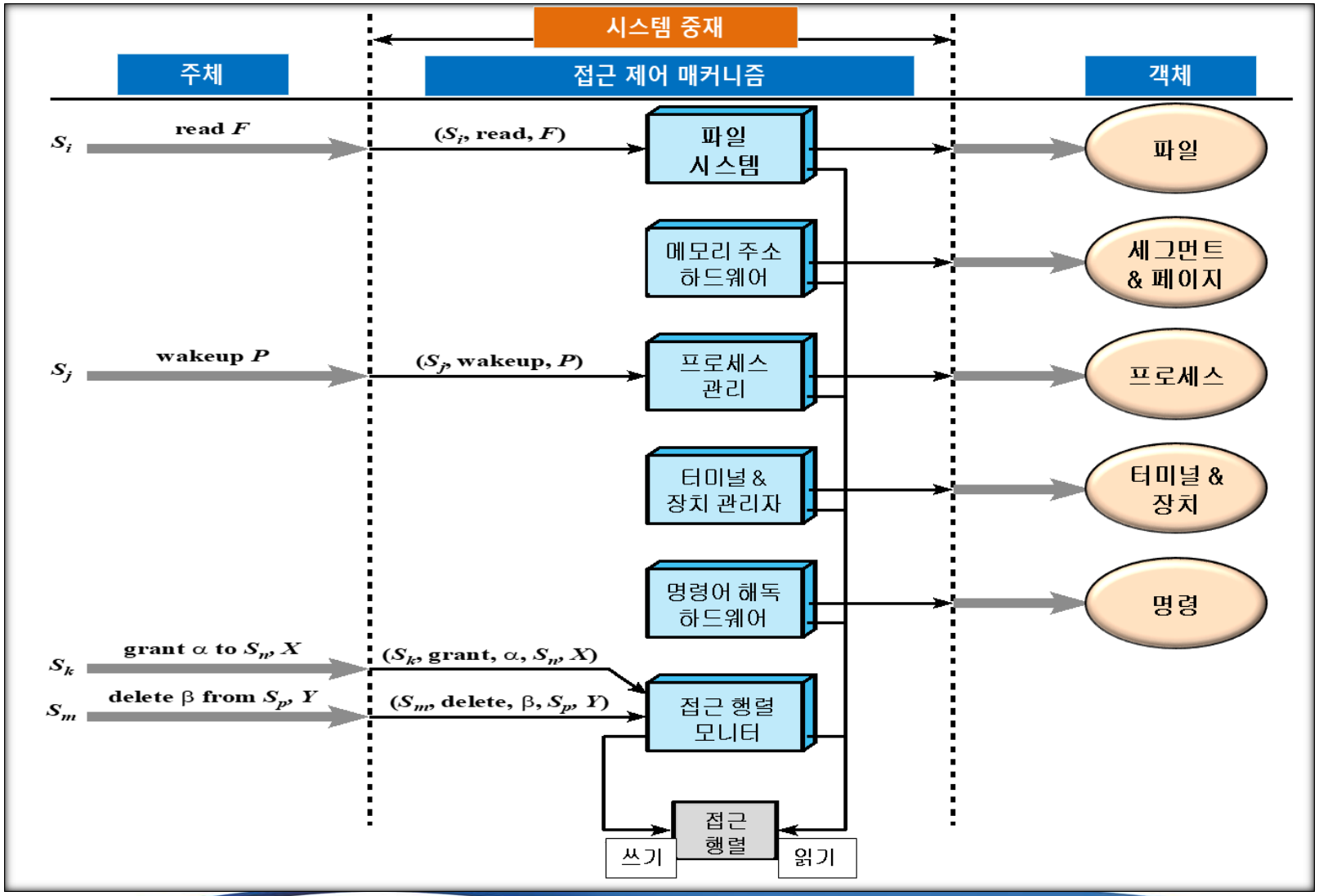
| | | 객체 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 주체 | | | 파일 | | 프로세스 | | 디스크 드라이브 | |
| | | $S_1$ | $S_2$ | $S_3$ | $F_1$ | $F_2$ | $P_1$ | $P_2$ | $D_1$ | $D_2$ |
| 주체 | $S_1$ | 제어 | 소유 | 소유<br>제어 | 읽기* | 읽기<br>소유 | 실행<br>준비 | 실행<br>준비 | 탐색 | 소유자 |
| | $S_2$ | | 제어 | | 쓰기* | 실행 | | | 소유자 | 탐색* |
| | $S_3$ | | | 제어 | | 쓰기 | 정지 | | | |

* : 카피 플래그 셋 – 객체을 복사할 때, 복사된 객체에 대해 해당 권한을 부여할지 제어 가능함

[확장된 접근 제어 행렬]

# 접근제어 행렬의 접근 시도

- 접근 제어 행렬의 구조



- **접근제어 행렬의 접근 시도**
  1. 주체 S0은 객체 X에 α 타입에 대한 요청 발행
  2. 요청 시스템(운영체제, 특정 종류의 접근 제어 인터페이스 모듈)으로 X를 위한 컨트롤러 전해질 메시지 (S0,α,X) 작성
  3. 컨트롤러는 α가 A[S0,X]에 있는지를 결정하기 위해 접근 행렬 A에 질문
     - 있으면 접근 허용, 없으면 접근 허용되지 않고 보안 위반 발생
     - 이 위반은 경고나 적당한 행동이 취해지도록 해야 함
- **접근 행렬 수정을 통제하는 규칙 집합 포함**
- **세 개의 규칙**
  - 접근 권한의 양도, 승인, 삭제
  - 예) 접근 제어 시스템에 정의 될 수 있는 규칙 집합
    - 추가적인 규칙 또는 대안적인 규칙이 포함될 수 있는 예 (부록 상세 설명 참조)

# 보호 도메인(Protection Domains)

- 해당 객체에 대한 접근 권한과 함께 존재하는 객체 집합
- 사용자는 사용자의 접근 권한의 부분 집합으로 프로세스를 생성할 수 있음
  - 프로세스와 도메인간의 연관성은 정적 또는 동적 일 수 있음
- **사용자 모드** - 특정 영역의 메모리가 사용되지 못하도록 보호되며 특정 명령어가 실행되지 않을 수 있음
- **커널 모드** - 권한이 부여된 명령이 실행될 수 있으며 메모리의 보호된 영역에서는 접근 할 수 있음

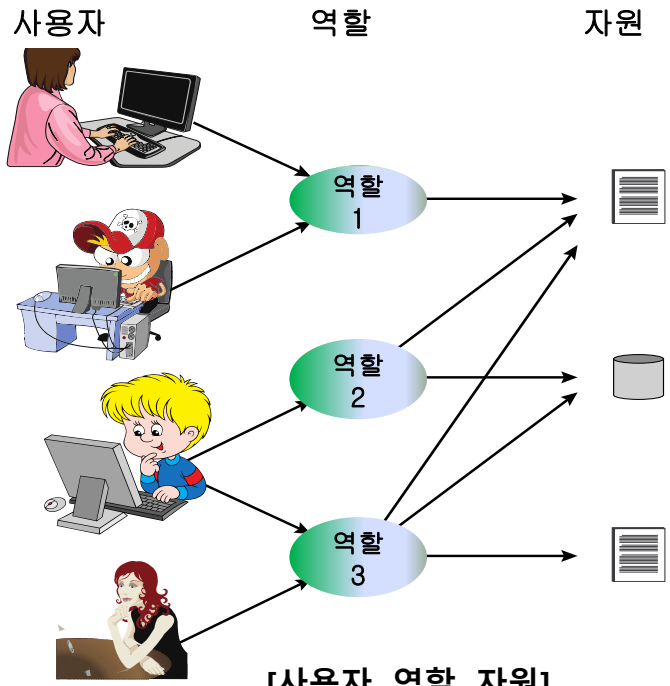
# 3-4. 역할 기반 접근 제어

# 4. 역할 기반 접근 제어

- **Role-based Access Control (RBAC)**
  - 전통적인 DAC 시스템
  - 개별 사용자 및 그룹의 접근 권한을 정의하는 RBAC모델
  - 각각의 사용자 대신에 역할에 접근 권한을 할당
  - 사용자는 정적 / 동적으로 각자 책임에 따라 다른 역할에 할당됨
  - 상업적으로 널리 쓰이고 연구가 활발히 진행 중

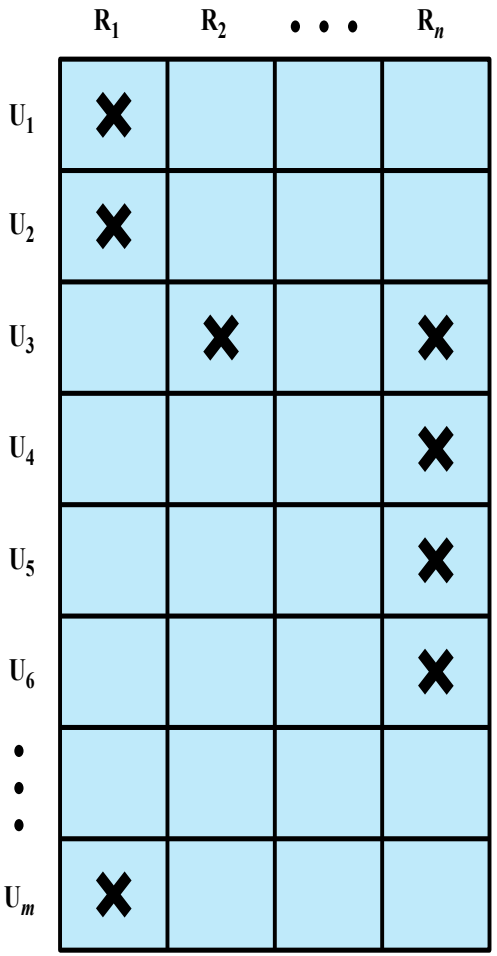
  - 국립 표준 기술 연구소 (NIST)느 FIPS PUB 140-3이라는 표준 발표
    - 역할을 통한 접근 제어와 관리 지원

- 사용자(Users)와 역할(Roles)의 관계는 역할과 자원(Resources) 또는 시스템 객체의 관계와 마찬가지로 다 대 다 관계임
  - 사용자 집합은 특정환경에서 자주 변하기 때문에 사용자와 1개 이상의 역할과의 할당 또한 동적으로 작용함
  - 대부분 환경의 시스템 내의 역할
    - 아주 가끔씩 추가,삭제만 존재
    - 상대적으로 정적으로 작용
  - 각 역할에는 하나 이상의 자원에 대한 특정 접근 권한이 부여
  - 자원 집합과 특정한 역할에 맞는 특정한 접근권한도 또한 자주 변하지 않음



[사용자, 역할, 자원]

- RBAC 시스템의 중요한 요소를 간단한 용어로 묘사하기 위해 접근 행렬 사용
  - 사용자와 역할의 관련성
  - 전형적으로 역할보다는 사용자가 많은 편임
  - 각 행렬 항목은 공백이거나 표시되며,
    - 표시: 역할에 사용자가 할당되어 있는 것
  - 한 사용자가 여러 개의 역할(ROW(행)에 여러 개의 마크)에 할당
  - 여러 명의 사용자가 한 역할(Column(열)에 여러 개의 마크)에 할당
  - 역할과 객체의 관계




[RBAC를 나타내는 접근 제어 행렬]

객체

| 역할 | 주체 | | | 파일 | | 프로세스 | | 디스크 드라이브 | |
|---|---|---|---|---|---|---|---|---|---|
| | $R_1$ | $R_2$ | $R_3$ | $F_1$ | $F_2$ | $P_1$ | $P_2$ | $D_1$ | $D_2$ |
| $R_1$ | 제어 | 소유 | 소유<br>제어 | 읽기* | 읽기<br>소유 | 실행<br>준비 | 실행<br>준비 | 탐색 | 소유자 |
| $R_2$ | | 제어 | | 쓰기* | 실행 | | | 소유자 | 탐색* |
| . | | | | | | | | | |
| . | | | | | | | | | |
| . | | | | | | | | | |
| $R_m$ | | | 제어 | | 쓰기 | 정지 | | | |

**[RBAC를 나타내는 접근 제어 행렬]**

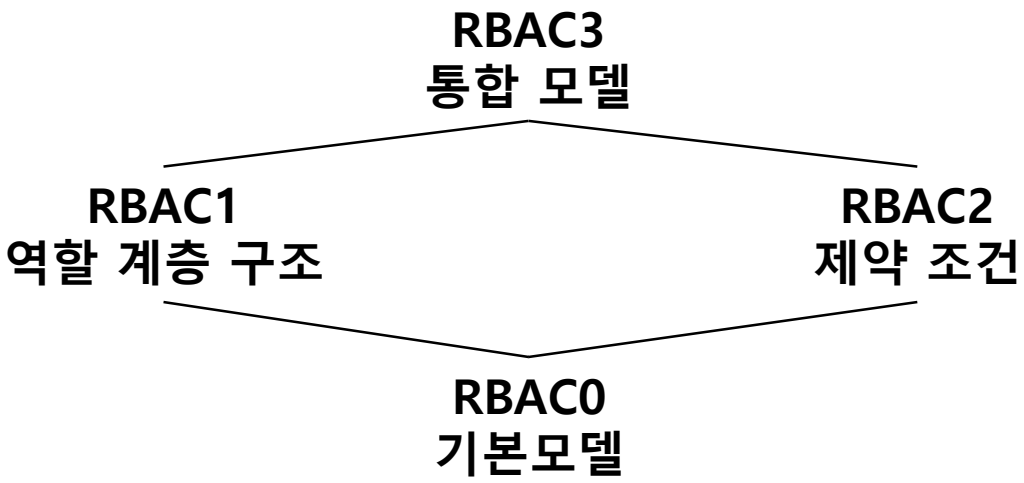
- 행렬은 주체로서 역할을 하는 DAC 접근 제어 행렬과 동일한 구조를 가지고 있음
- 전형적으로 역할의 수는 적고, 객체의 수나 자원의 수는 많음
- 항목 : 역할이 가질 수 있는 접근 권한
- 역할은 역할에 대한 역할 계층 구조라고 정의하면서 역할 자체가 하나의 객체로 취급될 수 있음
  - 각각의 역할은 그 역할에 필요한 최소한의 접근 권한의 집합을 포함해야 하고, 한 역할에 할당된 사용자는 그 역할에 딱 필요한 작업만 실행할 수 있음
  - 여러 명의 사용자가 같은 역할에 할당되면, 최소한의 같은 접근 권한을 가질 수 있음

# RBAC 참고 모델(RBAC Constraints Model)

- RBAC의 다양한 측면을 분류 하기 위해 RBAC을 기능별로 추상 모델 집합을 정의
- [SAND96]
  - 계속되는 표준화 노력의 기초로 수행하는 참조 모델 집단을 정의
  - 4가지 모델

| 모델 | 계층 구조 | 제약 |
|---|---|---|
| RBAC0 | NO | NO |
| RBAC1 | YES | NO |
| RBAC2 | NO | YES |
| RBAC3 | YES | YES |

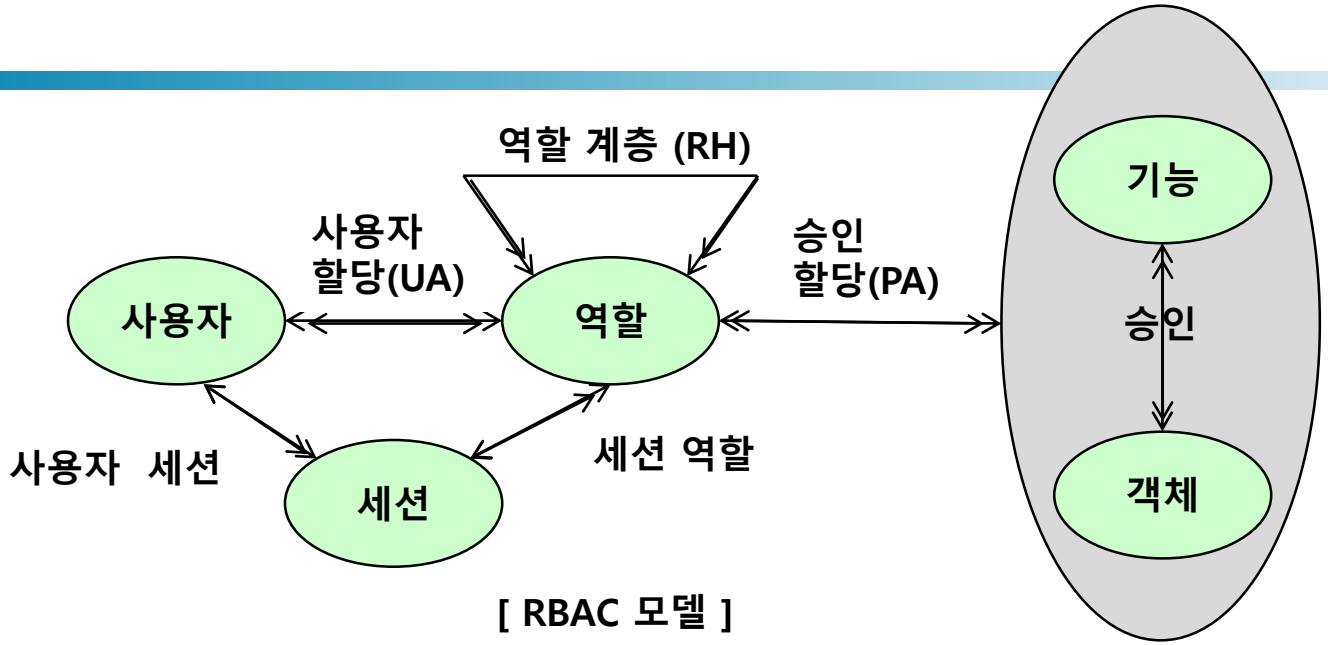
- **RBAC0** – RBAC 시스템의 최소한의 기능만 포함
- **RBAC1** - RBAC0의 기능과 하나의 역할이 다른 역할에 승인 정보를 상속할 수 있는 역할 계층구조 추가
- **RBAC2** – RBAC0의 기능과 RBAC 시스템의 구성 요소를 수정될 수도 있는 방법에 대한 제약 조건 추가
- **RBAC3** – RBAC0, RBAC1, RBAC2 의 기능을 모두 포함




**[ RBAC 모델 간의 관계 ]**

# 기본 모델(RBAC0)

- RBAC0 :역할 계층 및 제약에 조건이 없음
- 시스템의 네 가지 유형의 항목
  - 사용자(Users) : 컴퓨터 시스템에 접근할 수 있는 사람
    - 각각의 사용자는 관련된 사용자 ID 를 가짐
  - 역할(Role) : 컴퓨터 시스템을 제어하는 조직 내에서 지명된 직무
  - 허가(Permission) : 하나 이상의 객체에 대한 특정한 접근 권한의 승인
  - 세션(Session) : 사용자가 할당된 역할의 집합에서 활성화된 부분 집합과의 매핑



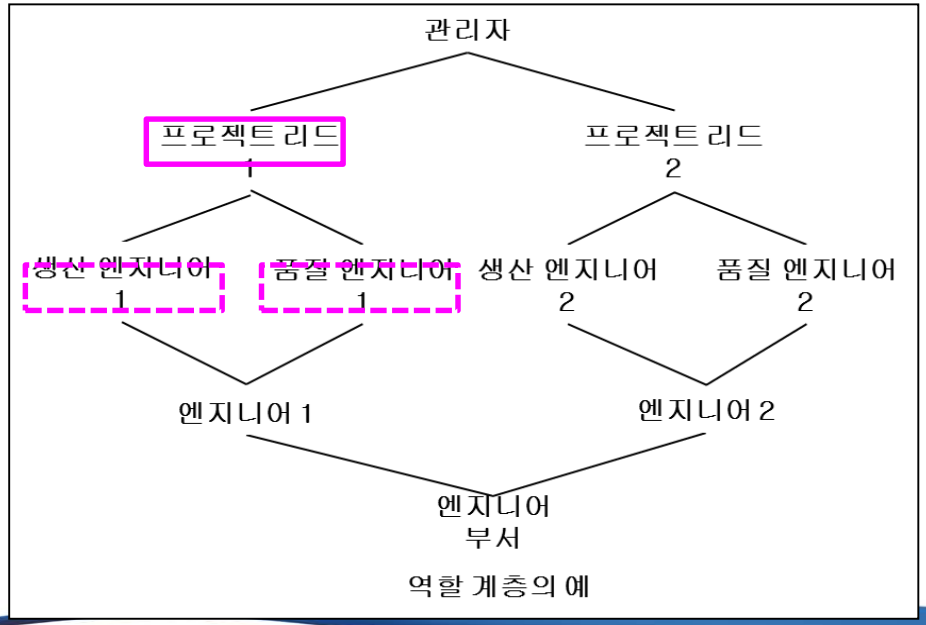

[ RBAC 모델 ]

- 실선 : 관계 또는 매핑을 의미, 하나의 화살촉: 1개, 두 개의 화살촉: 다수
- 세션 : 사용자와 사용자에게 할당되어 있는 1개 또는 그 이상의 역할 사이의 일시적 일 대 다 관계
- 사용자 : 특정한 작업에 필요한 역할에 대해서만 세션을 만듦
- 융통성(flexibility)과 정교성(granularity) 제공
  - 사용자: 역할과 역할-승인(Permission)과의 다 대 다 관계 (전통적인 DAC 구조에서 찾아 볼 수 없음)
  - 위험성: 접근 종류의 제한적인 통제가 허용
    - 사용자가 자원에 대해서 필요 이상의 접근 권한을 승인 받을 수 있음

# 역할 계층 구조(RBAC1)

- 조직에서 역할의 계층 구조를 반영하는 수단을 제공
- 보통보다 큰 책임을 지닌 직무는 자원에 접근 할 수 있는 더 큰 권한을 가짐
  - 2개 역할 사이에 있는 한 개의 줄은 상위 역할이 하위 역할에 허용되지 않는 다른 접근 권한뿐만 아니라 하위 역할의 모든 접근 권한을 포함을 의미
  - 예) 프로젝트 리드 역할

    생산 엔지니어 역할과 품질 엔지니어의 역할에 대한 모든 접근 권한 포함
  - 여러 개의 역할이 하나의 같은 하위의 역할로부터 상속할 수 있음



역할 계층의 예

# 제약 (RBAC2)

- RBAC을 조직 내의 관리 및 보안 정책의 세부 사항에 적응하기 위한 수단으로 제공
  - 역할과의 관계 또는 역할과 관련된 상태의 관계로 정의
  - 제약 조건 [SAND96]
    - 상호 배타적 역할(Mutually exclusive roles)
    - 카디널리티(Cardinality)
    - 필수 역할(prerequisite roles)

- 상호 배타적인 역할(Mutually exclusive roles)
  - 사용자가 역할 집합에 있는 하나의 역할에만 할당
  - 한 기관 내 직무와 능력을 분리
  - 추가 제한 조건을 사용한 상호 배타적인 역할 집합의 특성
    1. 한 사용자는 오직 하나의 역할에만 할당될 수 있음
       - 세션이 존재하는 동안 또는 정적으로 할당하는 경우
    2. 모든 허가(접근 권한)는 오직 하나의 역할에만 승인
       - 서로 배타적인 역할 집합에는 중복되는 접근권한이 없음

- **개수 (Cardinality)**
  - 역할에 대한 최대 수를 설정
  - 예) 프로젝트 리더 역할 또는 부서장 역할은 한 명의 사용자로 제한

- **필요 조건 (prerequisite roles)**
  - 최소 권한 개념의 구현을 구조화하는데 사용
  - 계층 구조에서는 사용자가 이미 하위 역할을 할당 받은 경우에만 상위역할에 사용자를 할당 할 수 있음

    예) 프로젝트 리드 역할
    - 하위 생산엔지니어와 품질엔지니어의 역할에 할당되어야 함
    - 만약 사용자가 특정 작업에 프로젝트 리드 역할의 모든 권한을 필요로 하지 않으면 사용자는 필요한 하위 역할만 사용하여 세션을 호출 할 수 있음

  - 필요조건을 사용하는 것은 RBAC3 모델이 필요한 계층구조의 개념에 연결

# 3-5. 속성 기반 접근 제어

# 5. 속성 기반 접근 제어(ABAC)

- **Attribute-Based Access Control Model (ABAC)**
- 자원과 주체의 속성에 대한 조건을 표현하는 허가를 정의
- 장점
  - 유연성(flexibility)과 표현력(expressive power)
- 우려사항
  - 실제 시스템 적용시 각 접근에 대한 자원과 사용자 특성에 미치는 성능 평가의 영향력이 있음
  - 웹 서비스 및 클라우드 컴퓨팅의 응용프로그램에서는 이미 상대적 높은 성능 비용 때문에 중요하지 않게 여겨짐
- 웹 서비스 : XAMCL (eXtensible Access Control Markup Language)을 도입

- 클라우드 서비스에 ABAC 모델 적용하는 데 많은 관심을 보이고 있음
- 객체에 대한 접근을 존재(주체와 객체)의 속성, 동작, 요청에 관계된 환경에 대해 규칙을 평가하여 제어
- 주체 속성, 객체 속성, 공식적인 관계, 주어진 환경에서 주체-객체 속성 조합에 허가된 동작을 정의하는 접근 제어 규칙의 평가에 의존
- 어떤 접근 제어 규칙이라도 만족하는 무제한의 조합되는 속성을 허가
- 세 가지 주요 요소: 속성, 정책모델, 구조모델

# 속성(Attributes)

- 주체, 객체, 환경 조건, 권한에 의해 미리 정의되고 할당된 요구되는 동작의 특정 측면을 정의

  예) 속성에 의해 제공되는 정보의 클래스, 이름 및 값을 나타내는 정보

  Class = Hospital Records Access, 이름 = Patient Information Access,

  값 = MF Business Hours Only

- **ABAC 모델의 세 가지 유형의 속성**
  - **주체** : 정보가 객체 사이를 이동하게 하거나 시스템 상태를 변경하도록 하는 능동적 존재 (사용자, 프로세스 등)
    - 각 주체는 주체의 특성과 신원을 정의하는 속성과 연관됨
    - 예) 주체의 신원, 이름 등 포함할 수 있음
  - **객체** : 정보를 포함하거나 받는 수동적인 정보 시스템 관련 존재 (파일,레코드,프로세스 등)
    - 접근 제어 결정을 돕는 속성
      예) MS 워드 문서는 이름, 주체 등 속성을 가질 수 있음
  - **환경** : 운영, 기술, 상황 혹은 정보 접근이 발행하는 환경
    - 대부분의 접근 제어 정책에 많이 무시되어왔음

      예) 현재 날짜와 시간, 현재 바이러스/해커 동작 등 같은 속성은 특정 주체나 자원과 연관되지 않지만, 접근 제어 정책을 적용하는 데 관련 있음

# ABAC 논리 구조

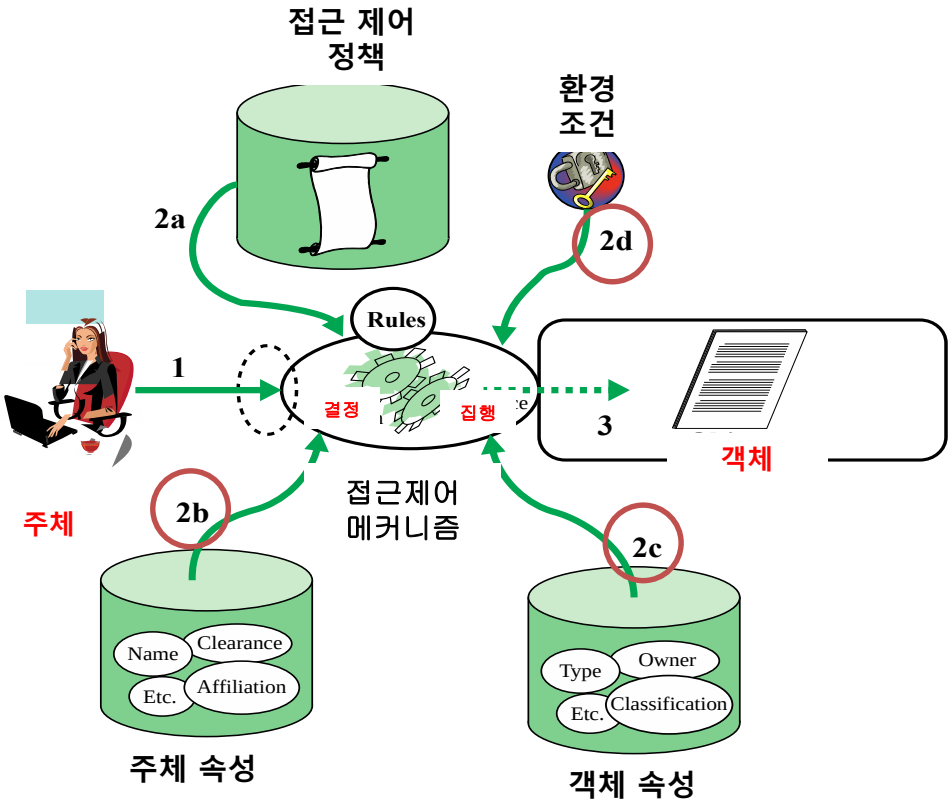
## 객체에 대한 주체의 접근 처리 단계

1. 주체는 객체 접근 요청을 하고 접근 제어 메커니즘으로 전송됨

2. 2a) 접근 제어 메커니즘은 미리 구성된 접근 제어 정책이 정의하는 규칙 집합 적용

3. 이 규칙에 기반하여, 접근 제어 메커니즘은 주체(2b), 객체(2c), 현재 환경 조건(2d) 의 속성들에 접근하여 허가를 결정

4. 접근이 허가되었다면 주체가 객체에 접근을 허가하고, 허가되지 않으면 거부




[간단한 ABAC 시나리오]

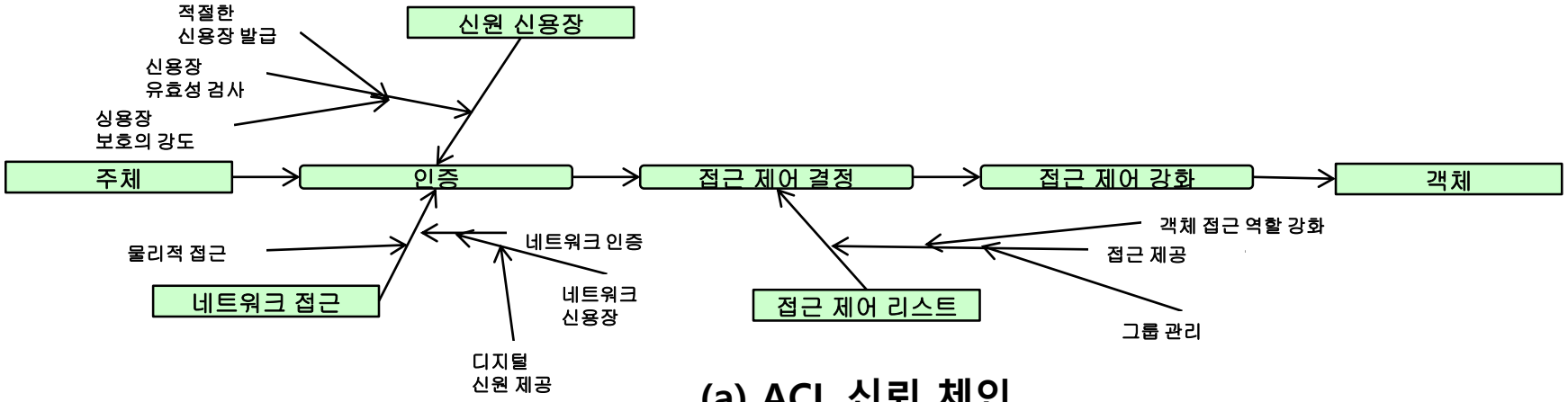
# ABAC과 DAC-ACL의 비교

- ACL을 사용하는 DAC 모델과 비교되는 ABAC 모델의 범위를 파악하기 유용함

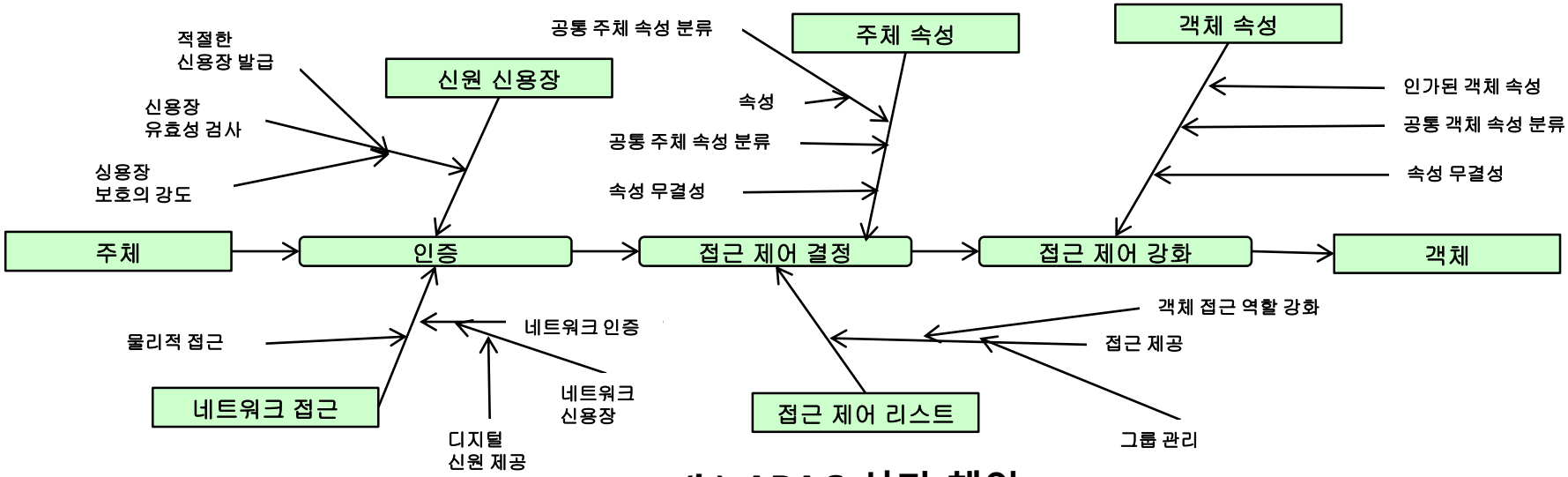
- 두 모델의 상대적 복잡도와 신뢰 요구 사항을 보여줌
  - 화살표로 표시되는 ACL 사용과 ABAC 사용의 대표적인 신뢰 관계의 비교는 ABAC가 제대로 동작하려면 훨씬 복잡한 신뢰 관계가 요구됨
  - ACL 신뢰의 루트 - 객체 소유자 (객체 속성 권한, 정책 개발자, 신용장 발급자)와 함께 ACL에 사용자를 추가하여 객체에 접근을 제공함으로써 객체 접근 규칙 적용
  - ABAC 신뢰의 루트 - 객체 소유자가 제어하지 못하는 많은 소스들에서 파생

- 기업 이사회는 모든 신원,신용장, 접근 관리 능력 배치와 동작을 관리하도록 구성
  - 각 하위 조직은 배치 관리와 기업 ABAC 구현과 연관된 패러다임 변환에 일관성이 보장되도록 유사하게 유지되도록 권고 (SP 800-162)

# ACL과 ABAC 의 복잡도 및 신뢰 관계




(a) ACL 신뢰 체인




(b) ABAC 신뢰 체인

# ABAC 정책(ABAC Policies)

- 정책은 주체의 권한과 환경 조건에서 자원 또는 객체들이 보호되는 것에 기반한, 허가된 행위를 관리하는 규칙과 관계의 집합

- 보호가 필요한 객체와 주체에 사용 가능한 권한의 관점에서  사용

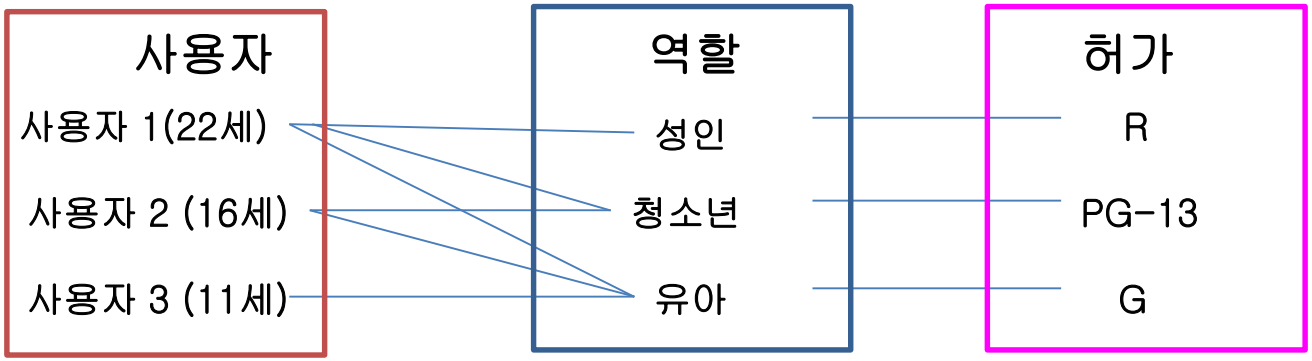
- 권한 - 주체의 허가된 행위
  - 권한에 의해 정의되면 정책으로 구체화됨
  - 권한 = 권리, 위임 ,인가

# RBAC와 ABAC 방법 비교 예시

- 사용자 연령과 영화 등급에 기반한 접근 제어 정책

| 영화 등급 | 사용자 연령 |
|---|---|
| R | 17세 이상(성인) |
| PG-13 | 13세 이상(청소년) |
| G | 모든 연령(유아) |

- RBAC 모델
  - 모든 사용자는 등록 중에 성인, 청소년, 유아의 세가지 중 한 가지가 부여되어 세가지 허가(R, PG-13, G)가 생성됨
  - 사용자-역할 할당과 허가-역할 할당은 수동 관리 작업



- **ABAC 모델** : 역할 정의 필요 없음
  - 사용자(u)가 영화(m)을 시청할 수 있는지 정책 규칙으로 평가
    - R1 : can_access(u, m, e) <- (Age(u) ≥ 17 ∧ Rating(m) ∈ {R, PG-13, G}) ∨ (Age(u) ≥ 13 ∧ Age(u) < 17 ∧ Rating(m) ∈ {PG-13, G}) ∨ (Age(u) ≤ 13 ∧ Rating(m) ∈ {G})
  - 연령과 등급: 주체 속성과 개체 속성
  - 정적 역할 관리와 정의가 없으므로, 사용자-역할 할당과 허가-역할 할당의 관리적 작업 필요 없음

- **RBAC모델** : 각 사용자를 연령과 가격에 따라 역할과 허가의 수가 2배가 되어야 함
  - 추가적인 속성을 효율적으로 다룸 (환경 속성 추가가 용이)
    - R2 : can_access(u, m, e) <-(MembershipType(u)= Premium)∧ (MembershipType(u)= Regular ∧ MovieType(m) = OldRelease)
    - R3 : can_access(u, m, e) <- R3 ∧ R4
- 새 정책 규칙 추가를 원할 경우,
  - 일반 사용자는 프로모션 기간에 신작 시청이 가능
  - RBAC 모델 : 표현되기 어려움
  - ABAC 모델 : 현재 날짜가 프로모션 기간에 해당하는지 AND 규칙을 추가하기만 하면 됨

# 3-6. 자격 기반 접근 제어

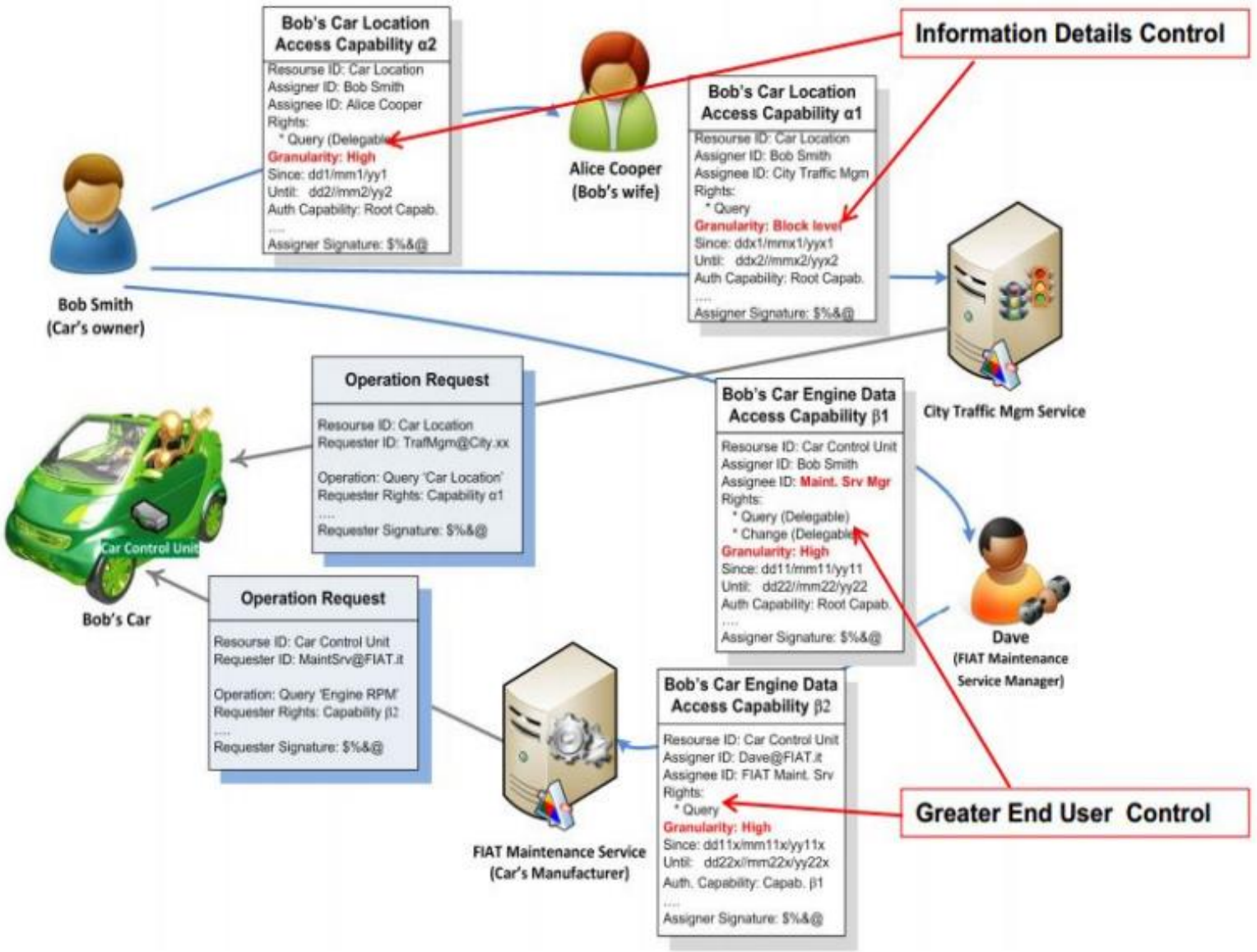
# 6. 자격기반 접근제어 (IoT환경을 위한)

- **Capability Based Access Control (CapBAC)**
- 필요성
  - RBAC은 디바이스에 적용될 경우 규칙의 폭발적인 증가를 수용하기 어려움
  - ABAC은 다수의 디바이스가 연동되는 사물인터넷 환경의 속성들을 동일하게 일치시키기 어려움
  - 두 방법은 권한 위임의 어려움 존재
- S. Gusmeroli 는 사물인터넷 시스템의 접근제어를 위해 자격기반 접근제어 기법이 제안됨
- 최소권한 원칙과 권한 위임기능을 부여, 주체에게 자신의 서비스 및 정보에 대한 접근제어 관리
- 특정 리소스를 사용할 수 있는 권한 요청 시 토큰을 발급하여 사용할 수 있게 함

  -권한 위임, 자격 철회 등 지원

  -자격을 매우 자세하게 기술할 수 있음

# CapBAC 적용 시나리오

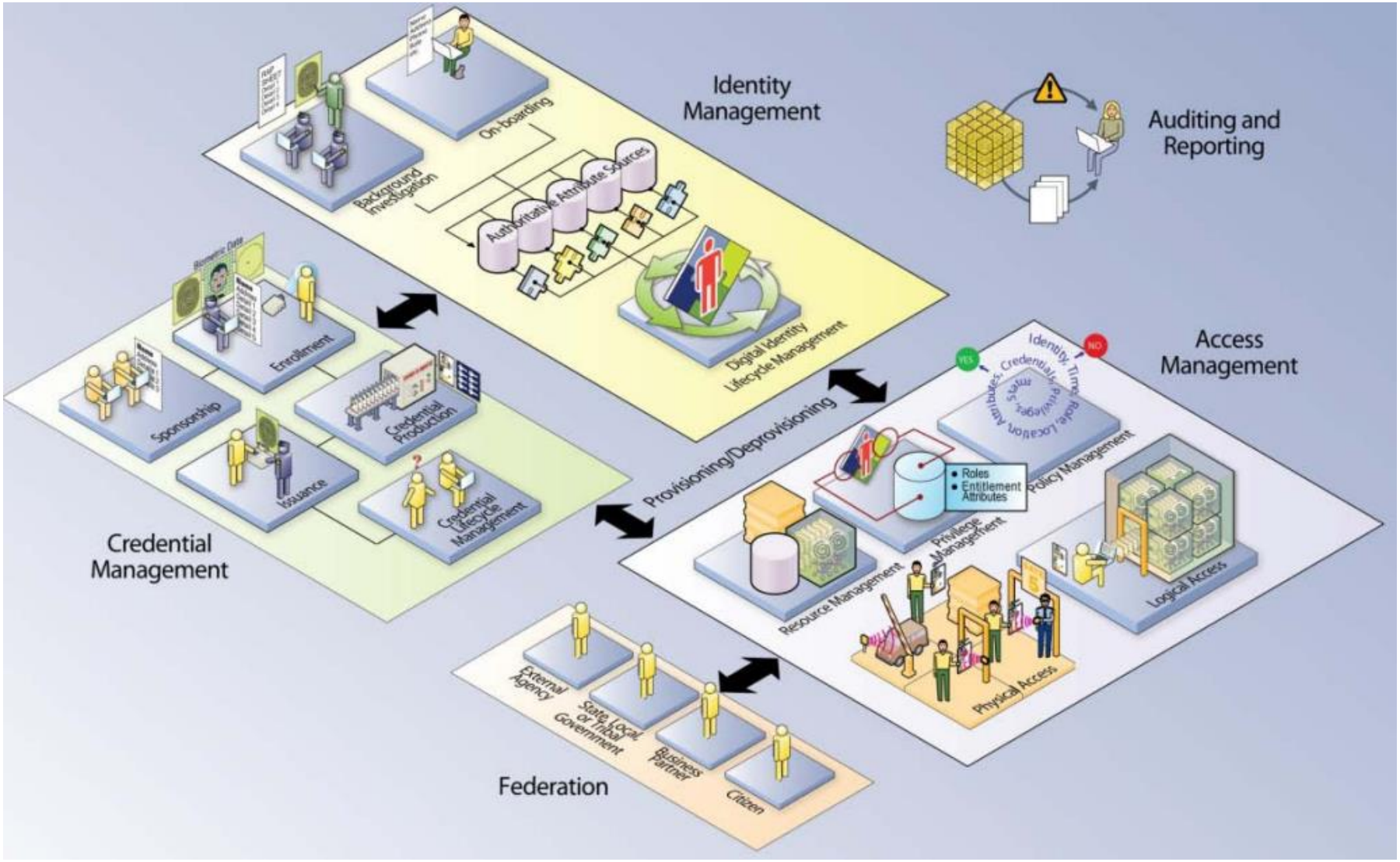


# 3-7. 신원, 신용장, 접근 관리: ICAM

# 7. 신원, 신용장, 접근 관리: ICAM

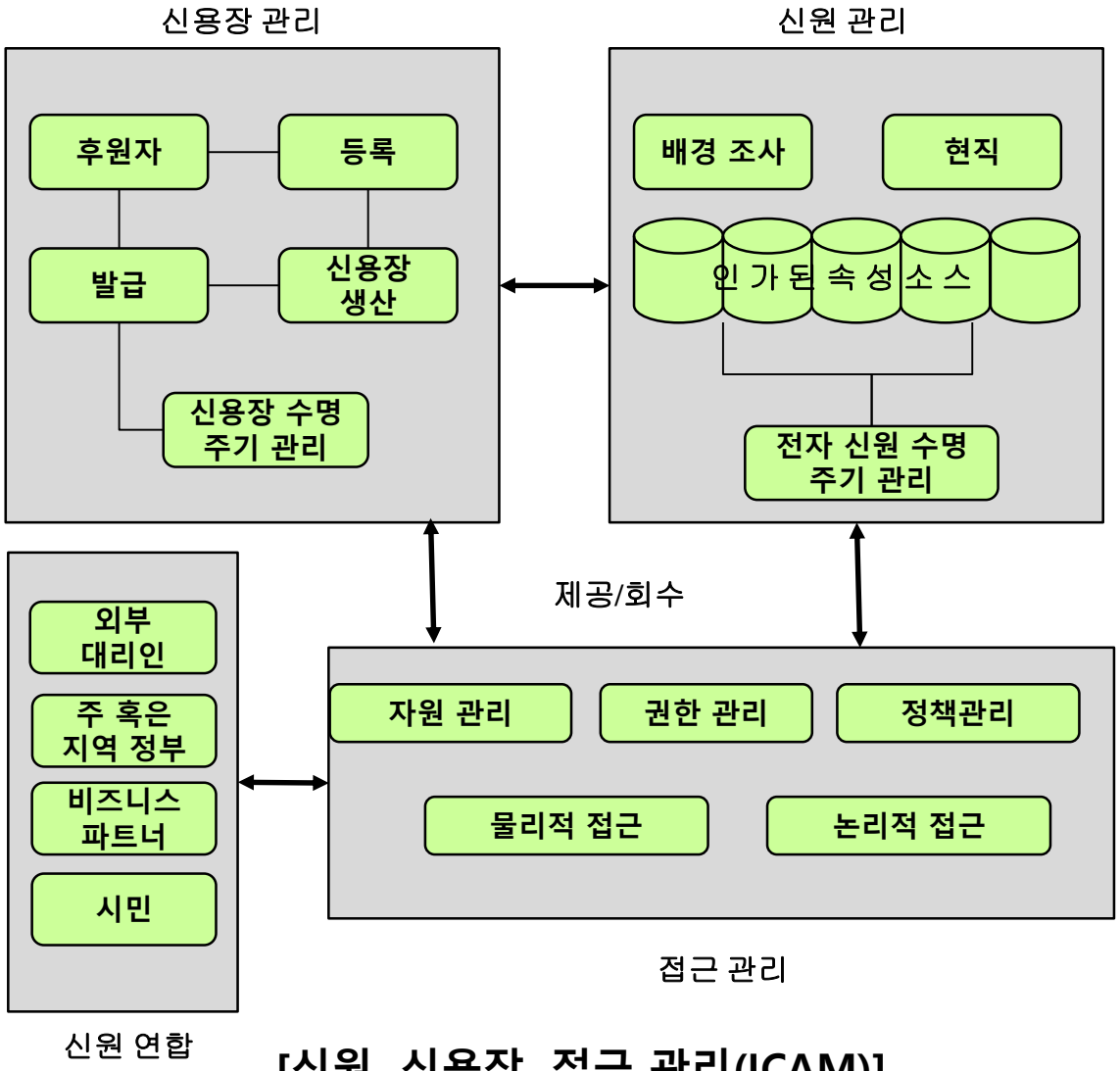
- ICAM(Identity, Credential, and Access Management)
  - 미국정부에 의해 개발된 전자 신원, 신용장, 접근 제어를 관리하고 구현하는 포괄적인 기법
  - 접근 제어에 대한 단일화된 접근

- ICAM 의 주요 서비스
  - Digital Identity
  - Credentialing
  - Privilege Management
  - Authentication
  - Authorization & Access
  - Cryptography
  - Auditing & Reporting


Identity Management
On-boarding
Background Investigation
Authoritative Attribute Sources
Digital Identity Lifecycle Management
Auditing and Reporting
Access Management
Identity
Credential
Time
Role, Location, Attributes
Privilege
Status
Roles
Entitlement Attributes
Policy Management
Privilege Management
Resource Management
Logical Access
Physical Access
Provisioning/Deprovisioning
Credential Management
Enrollment
Sponsorship
Credential Production
Issuance
Credential Lifecycle Management
Federation
External Agency
State, Local, or Tribal Government
Business Partner
Citizen

# 신원관리(Identity Management)

- 속성을 전자 신원에 부여하고 전자 신원을 개인 혹은 NPE와 연결
  - NPE (비인간 객체): processes, applications, and automated devices seeking access to a resource 등
- 목적 : 특정 상황에 독립적인 신뢰할 수 있는 전자 신원의 수립
- 응용 프로그램과 프로그램의 접근 제어
  - 응용 프로그램이나 프로그램의 특정 사용에 대한 신원의 전자적 표현을 생성
- 신원 관리와 보호 : 응용프로그램에 연관된 미션에 부차적인 것으로 다루어짐
- 신원 관리의 마지막 요소 - 수명 주기 관리
  - 메커니즘, 정책, 개인 신원 정보 보호
  - 신원 데이터 접근 제어
  - 인가된 신원 데이터를 필요한 응용프로그램들이 공유하는 기술
  - 기업 신원 폐기




[신원, 신용장, 접근 관리(ICAM)]

# 신용장 관리(Credential Management)

- 신용장 - 신원을 가입자에 의해 소유되고 제어되는 토큰과 공식적으로 묶은 객체 혹은 자료 구조
  예) 스마트카드, 개인/공개 암호 키, 공인인증서

- 신용장의 수명 주기 관리

- 신용장 관리의 논리적 요소
  1. 인가된 개인은 신용장의 필요성을 수립하기 위해 개인 혹은 존재를 후원
  2. 후원된 개인은 신용장에 등록
     - 이 과정은 일반적으로 신원 증명과 생물학적과 생체 인식 데이터 캡처로 구성
     - 신원 관리 컴포넌트가 관리하는 인가된 속성 데이터의 결합을 포함할 수 있음
  3. 신용장 생성
     - 신용장 유형에 따라 생성은 암호화, 전자 서명 사용 등을 포함
  4. 신용장 개인 혹은 NPE로 발급됨
  5. 신용장은 폐기, 재발급/대체, 기간만료 등 포함하는 수명 주기 동안 유지되어야 함

# 접근 관리(Access Management)

- 접근 관리 컴포넌트 - 자원 접근 허가되는 방법의 관리와 제어
  - 논리적/물리적 접근, 시스템 내부 / 외부 요소를 다룸

- 접근 관리 목적 - 개인이 보안에 민감한 컴퓨터 시스템, 데이터, 건물에 접근 시도할 때 적절한 신원 검증 이루어지도록 함

- 접근 제어 함수 - 접근을 요청하는 것들이 제시하는 신용장과 요청자의 전자 신원을 사용

- 기업 접근 제어 시설의 3가지 지원 요소
  - 자원 관리 : 접근 제어를 요구하는 자원에 대한 규칙 정의
    - 사용자 속성, 자원 속성,환경 조건 등 포함
  - 권한 관리 : 개인의 접근 프로필을 구성하는 권한과 권한 속성의 수립과 유지와 관련됨
    - 물리적/논리적 자원에 대한 접근 결정을 하는 기초로 사용되는 개별 특징 나타남
    - 권한은 전자 신원에 연결될 수 있는 속성으로 고려됨
  - 정책 관리 : 접근 업무에서 무엇이 허가되는지에 대한 통제

# 신원 연합(Identity Federation)

- 한 기관이 다른 기관이 발급한 전자 신원, 신원 속성, 신용장을 신뢰할 수 있게 해주는 기술, 표준, 정책, 프로세스 등

- 두 가지 질문

  1. 당신의 시스템에 접근하려는 외부 기관의 개인의 신원을 어떻게 신용할 것인가?

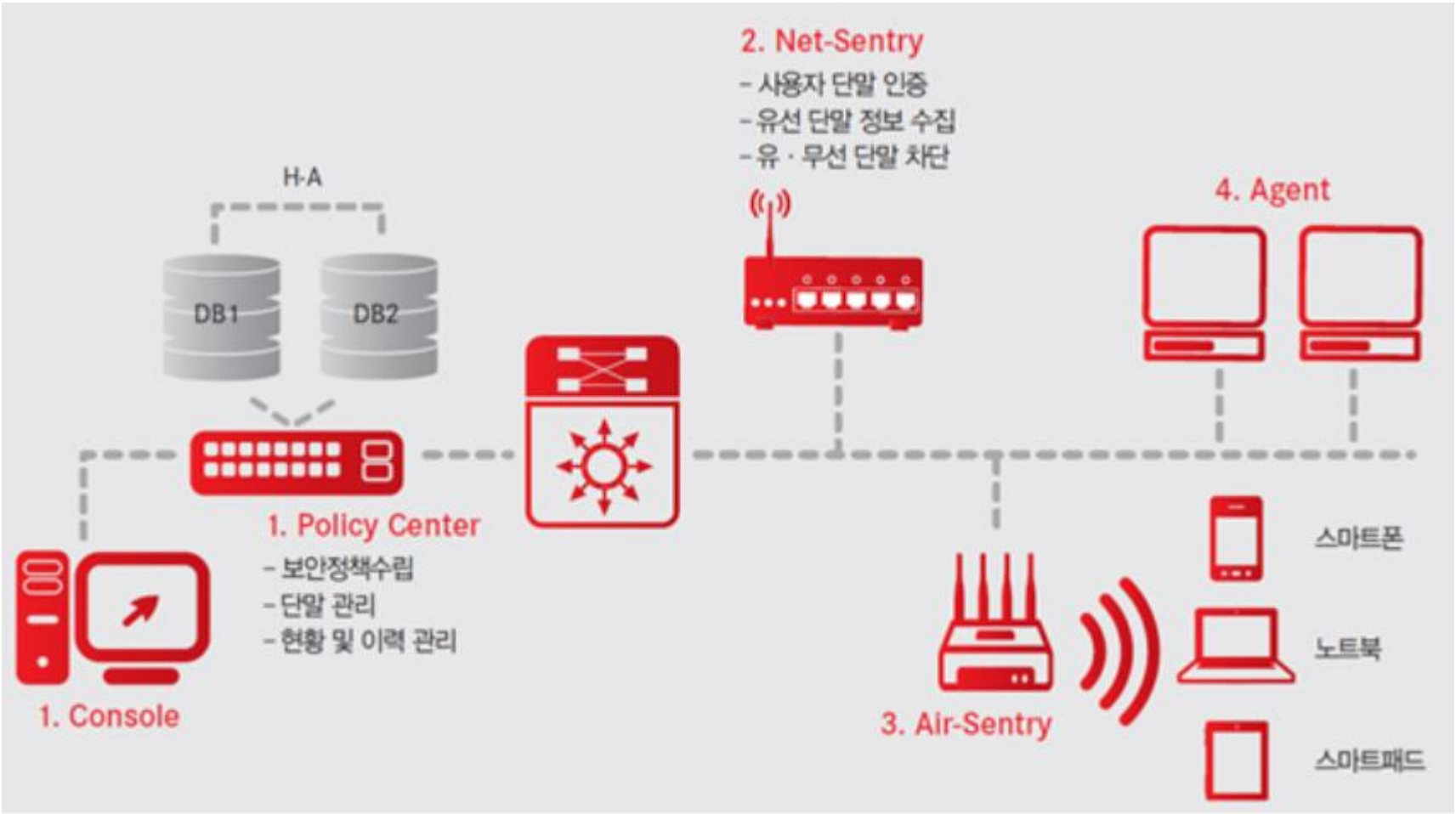
  2. 당신의 기관의 개인들이 외부 기관과 협동하기를 원할 때 어떻게 개인들의 신원을 보증할 것인가?

# 3-8. 실무 접근 제어 솔루션

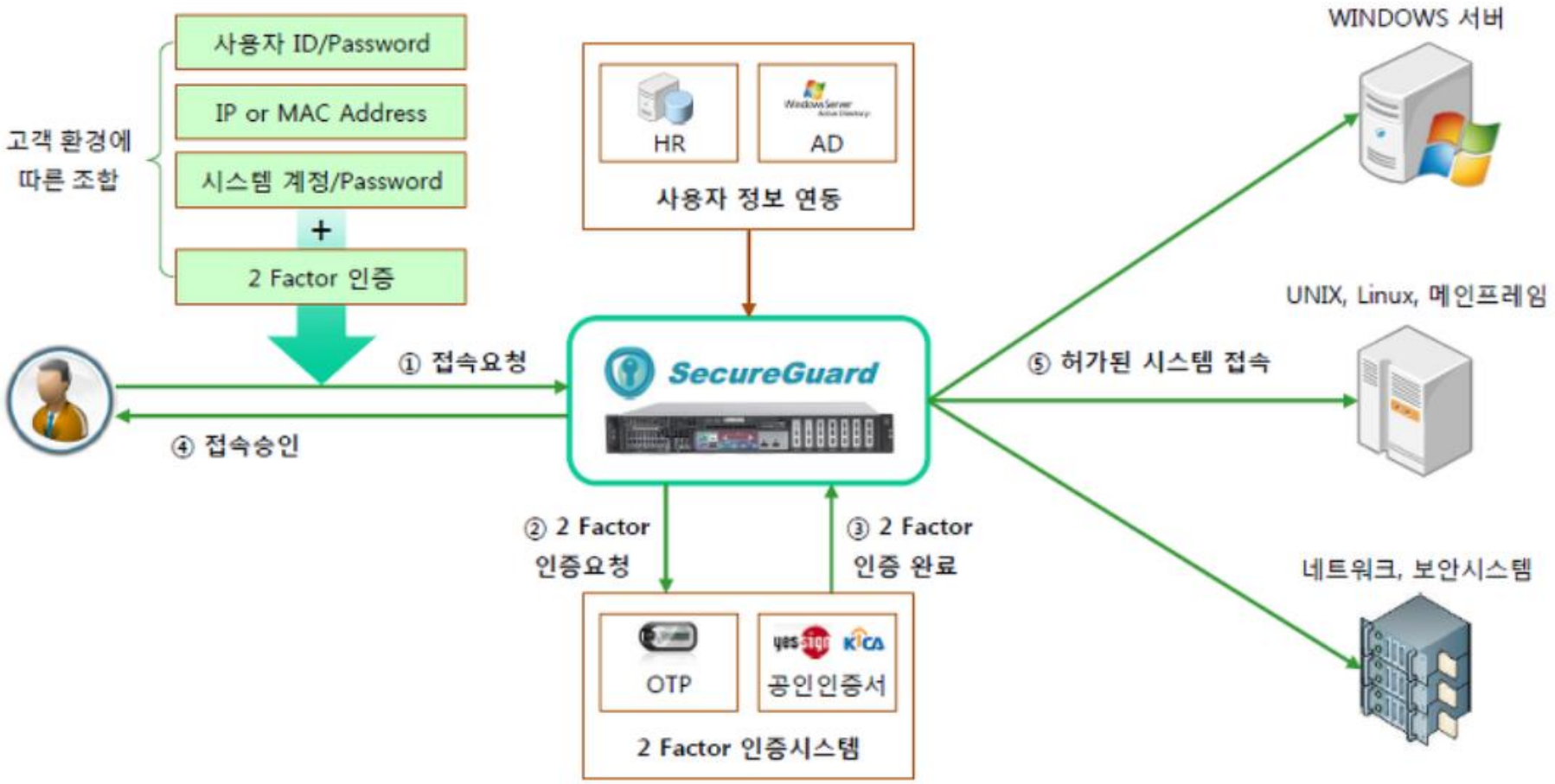
# 8. 실무 접근제어 솔루션

- 시스템 접근제어

- 네트워크 접근제어 시스템



- 시스템 접근제어 및 감사


사용자 ID/Password
IP or MAC Address
시스템 계정/Password
+
2 Factor 인증
고객 환경에 따른 조합
HR
AD
사용자 정보 연동
WINDOWS 서버
UNIX, Linux, 메인프레임
네트워크, 보안시스템
① 접속요청
④ 접속승인
SecureGuard
⑤ 허가된 시스템 접속
② 2 Factor 인증요청
③ 2 Factor 인증 완료
OTP
공인인증서
2 Factor 인증시스템

# 참고문헌


- Stallings, 컴퓨터보안 (Computer Security(GE), 복두출판사, 2016
- Information Security, IT COOKBOOK, 한빛미디어
- 김진보, 김미선, 서재현, "사물인터넷 서비스 접근제어를 위한 리소스 서비스 관리 모델 구현", 스마트미디어저널, 2016
- 이범기 외2인, “IoT에서 Capability 토큰 기반접근제어 시스템 설계 및 구현”, 한국정보보호학회논문지, 2015
- Luis Sanchez, Luis Munoz, Jose Antonio Galache, Pablo Sotres, Juan R. Santana, Veronica Gutierrez and Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, Evangelos Theodoridis and Dennis Pfisterer. "SmartSantander: IoT experimentation over a smart city testbed". Computer Networks, 61, 217~238, 2014.
- Capability-Based Access Control, Lecture Notes (Syracuse University), http://www.cis.syr.edu/~wedu/Teaching/CompSec/LectureNotes_New/Capability.pdf
- What is Identity, Credential, and Access Management (ICAM)?, https://www.idmanagement.gov/IDM/servlet/fileField?entityId=ka0t0000000TNJYAA4&field=File__Body__s
- http://www.kis.co.kr/sub02/02.php
- http://www.linkic.co.kr/main.php?mm=s2131
- http://www.mplsoft.co.kr/378

# Q & A


61

# 부록


62

# 임의 접근 제어(DAC)

## 접근 제어 시스템 명령

| 역할 | 명령 (by S0) | 인증 | 조작 |
|---|---|---|---|
| R1 | 전송 { a *} to S, X<br>전송 { a } to S, X | 'a*' in A[$S_0$, X] | 저장 { a *} in A[S, X]<br>저장 { a } in A[S, X] |
| R2 | 허가 { a *} to S, X<br>허가 { a } to S, X | '소유' in A[$S_0$, X] | 저장 { a *} in A[S, X]<br>저장 { a } in A[S, X] |
| R3 | 삭제 a from S, X | '소유' in A[$S_0$, S]<br>or<br>'제어' in A[$S_0$, X] | 삭제 a from A[S, X] |
| R4 | w <- 읽기 S, X | '소유' in A[$S_0$, S]<br>or<br>'제어' in A[$S_0$, X] | 복사 A[S, X] into w |
| R5 | 객체 생성 X | 없음 | A에 X를 위한 행 추가; store 'owner' in A[$S_0$, X] |
| R6 | 객체 삭제 X | '소유' in A[$S_0$, X] | A에 X를 위한 행 삭제 |
| R7 | 주체 생성 S | 없음 | A에 S를 위한 열 추가; 객체 S 생성 수행; store 'control' in A[S, S] |
| R8 | 주체 삭제 S | '소유' in A[$S_0$, S] | A에 X를 위한 열 삭제; 객체 S 삭제 수행 |

# 접근 제어 시스템에 정의 될 수 있는 규칙 집합의 예

- 규칙 R1 = α* 가 $A[S_0,X]$ 가정하에 $S_0$ 가 객체 X 에 대한 접근 권한 α 를 가지고 있다는 의미, 카피 플래그가 존재하기 때문에 이 권한(카피 플래그와 함께 또는 카피 플래그 없이)을 다른 주체에게 양도, 주체는 새로운 주체가 악의를 가지고 권한을 가지면 안 되는 또 다른 주체에게 그 권한을 양도할 우려가 있다면 카피 플래그 없이 권한 양도
  EX) $S_1$은 접근 행렬 $F_1$ 행 모든 엔트리에 '읽기' 또는 '읽기*' 표시 가능
- 규칙 R2 = $S_0$ 가 객체 X에 대한 소유자로 지정, $S_0$ 가 X에 대한 '소유' 권한을 가고 있다면 $S_0$ 는 모든 S에 대해서 A[S, X]에 대한 접근 권한을 추가
- 규칙 R3 = $S_0$ 는 $S_0$ 가 컨트롤하고 있는 주체에 대한 접근 행렬의 행의 모든 엔트리로부터 접근 권한을 삭제할 수 있고 $S_0$ 가 소유하고 있는 객체에 대한 접근 행렬의 행의 모든 엔트리로부터 접근 권한을 삭제 허용
- 규칙 R4 = 주체가 접근 행렬을 소유하거나 컨트롤 하는 경우에 행렬의 일부분을 읽을 수 있도록 허용
- 규칙 R5~R8 = 주체와 객체의 생성과 삭제를 통제
- 규칙 R5 = 주체는 객체를 소유하고 있는 경우 새로운 객체 생성, 그 객체에 권한을 승인과 삭제 가능
- 규칙 R6 = 객체의 소유자는 객체 삭제, 결과 접근 행렬에서 그에 해당하는 행을 삭제
- 규칙 R7 = 주체가 새로운 주체를 만들 수 있도록 한다. 생성자가 새로운 주체를 소유하고 새로운 주체는 자신에 대한 접근 제어 소유
- 규칙 R8 = 주체의 소유자가 그 주체에게 임명된 접근 행렬의 열과 행(만약, 주체 행이 있다면)을 삭제 허용

# 추가적인 규칙 또는 대안적인 규칙이 포함될 수 있는 예

- 타깃이 되는 주체에 양도된 권한이 더해지고 양도해주는 주체에서는 그 권한이 삭제되는 'transfer-only' 권한이 정의 , 카피 플래그 '소유자' 권한에 동반하는 것을 허용 X, 객체나 주체에 대한 소유자의 숫자는 한개로 제한
- 하나의 주체가 다른 주체를 생성하고 그 주체에 대해서 '소유자' 권한을 갖는 능력은 주체 계층 구조 정의하는 데 사용
  ex) $S_1$이 $S_2$ 와 $S_3$ 를 소유하고 있고 $S_2$와 $S_3$ 는 $S_1$ 에 종속 (그림 4.3)
  $S_1$ 이 가지고 있는 권한을 $S_2$ 접근권한에 승인,삭제 (표 4.2)
- 주체는 가지고 있는 접근 권한의 부분 집합과 함께 다른 주체를 생성
  ex) 주체가 충분히 신뢰할 수 없는 응용프로그램을 호출하고 그 응용프로그램이 다른 주체에 접근 권한을 양도하는 것을 원하지 않을 경우에 매우 유용

# 예 : UNIX 파일 접근 제어

- UNIX 파일은 inode(index node) 를 사용하여 관리
  - inode 는 특정 파일에 필요한 핵심 정보가 있는 제어 구조
  - 여러 개의 파일 이름은 단일 inode 와 연관
  - 활성화된 inode는 정확히 하나의 파일과 연관
  - 디스크에는 파일 시스템의 모든 파일 inode를 포함하는 inode 테이블 또는 inode 목록이 있음
  - 파일이 열리면 inode는 메인 메모리로 옮겨지고, 메모리에 상주하는 inode 테이블에 저장
- 디렉터리 - 계층적 트리 구조
  - 파일, 다른 디렉터리가 포함될 수 있음
  - 연관된 inode에 대한 포인트와 파일 이름을 포함

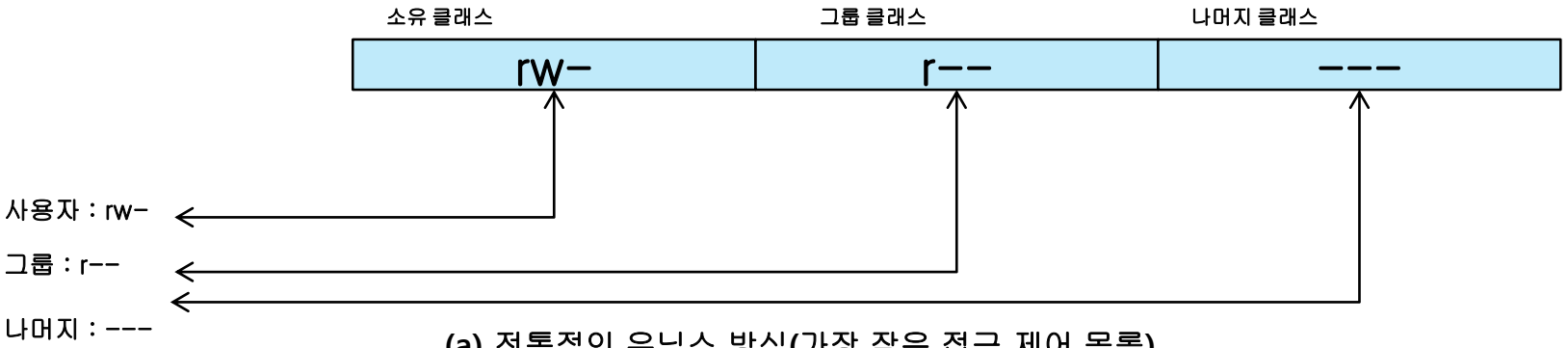
# 전통적인 UNIX 파일 접근 제어

- 유일한 사용자 식별 번호(사용자 ID)를 할당 받음
- 그룹 ID로 식별되는 기본 그룹의 구성원
- 특정 그룹에 속함
- 소유자 ID, 그룹 ID 및 보호 비트는 파일의 inode의 일부
- 총 12개의 보호비트의 집합과 연관
  - 9개의 보호 비트는 파일 소유자, 그룹 구성원 및 다른 모든 사용자들에 대해 읽기,쓰기,실행 권한을 지정

    .읽기와 쓰기 비트 : 디렉터리 안의 파일 목록 나열과 이름을 바꾸거나 삭제 할 수 있는 권한을 부여함

    .실행 비트 : 디렉터리로 내려 가거나 파일 이름을 검색하는 권한 부여

    .나머지 3개 비트: 파일이나 디렉터리의 특수한 부가적 행동을 정의




(a) 전통적인 유닉스 방식(가장 작은 접근 제어 목록)

**유닉스 파일 접근 제어**

- 사용자 ID 설정(SetUID)
  - 사용자의 권한이 있어야만 실행을 할 수 있는 파일
    → 그 권한을 일시적으로 파일을 실행하는 일반 사용자들에게 부여하기 위해 사용

- 그룹 ID 설정(SetGID) :
  - 그룹의 권한이 있어야만 실행을 할 수 있는 파일인 경우,
    → 그 권한을 일시적으로 파일을 실행하는 일반 사용자들에게 부여하기 위해 사용

- Sticky 비트
  - 파일에 설정할 때: 시스템이 실행 후 파일 내용을 메모리에 유지해야 함
  - 디렉터리 적용할 때 : 디렉터리의 파일 소유자만이 파일의 이름을 바꾸거나 이동하거나 삭제 할 수 있음

- Superuser
  - 일반적인 접근제어 제한에서 제외하고 시스템 전체에 대한 접근 할 수 있음

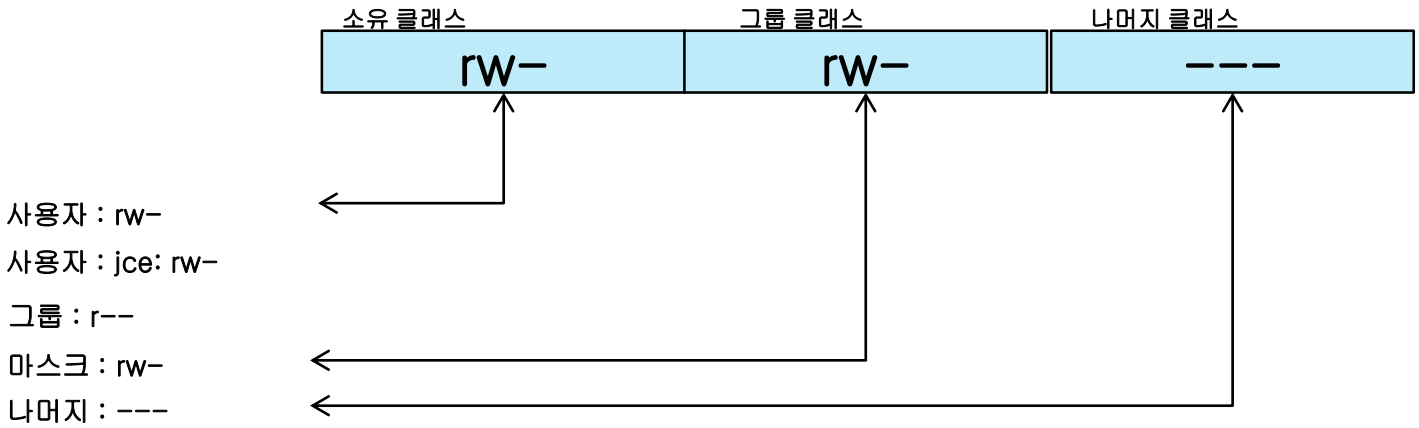
# UNIX의 접근 제어 목록(ACLs)

- 현대의 UNIX와 UNIX 기반 운영체제(FreeBSD, OpenBSD, Linux, Solaris) 목록 지원

- FreeBSD
  - 관리자가 setfacl 명령어를 사용함으로써 UNIX 사용자 ID와 그룹의 목록을 지정
  - 파일에 할당 보호 비트는 읽기, 쓰기, 실행
  - 파일에 ACL 이 있을 필요 없음.
  - 파일은 확장된 ACL이 있는지 여부를 나타내는 부가적인 보호 비트가 포함

– FreeBSD와 확장 ACL사용을 지원하는 UNIX 구현

:9개 비트 허가필드에서 소유자 클래스와 나머지 클래스 항목에는 최소 ACL의 경우와 같은 의미함



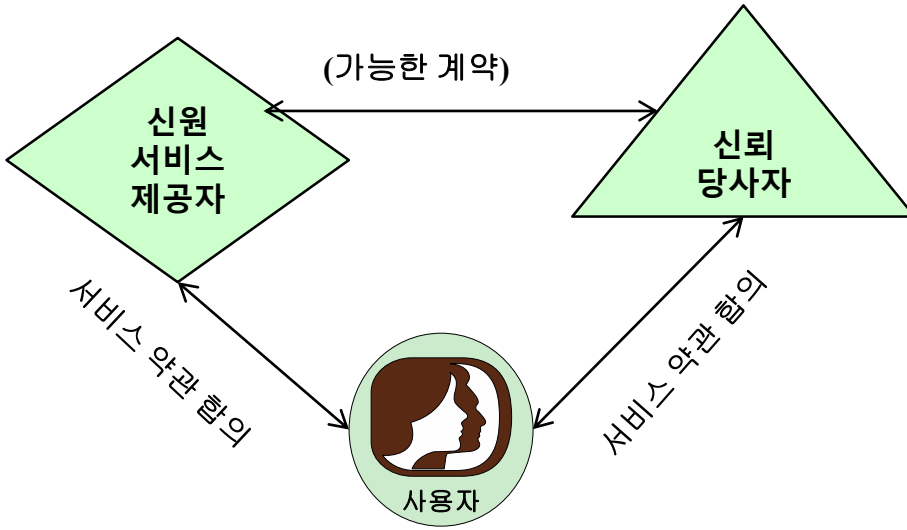

**유닉스 파일 접근 제어(확장된 접근 제어 목록)**

- 프로세스가 파일 시스템 객체에 대한 접근 요청하면 두 단계가 형성
  - 1단계 : 요청 하는 프로세스에 가장 일치하는 ACL 항목 선택
    ACL 항목은 소유자,지명된 사용자,그룹 등 단일 항목만 접근을 결정
  - 2단계 : 일치하는 항목이 있는지 확인
    - 선택된 항목이 충분한 승인 정보를 포함하고 있는지 검사
    - 프로세스는 둘 이상의 그룹에 속할 수 있음
    - 프로세스는 한 개 이상의 그룹 멤버일 수 있기 때문에 한 개 이상의 그룹 항목이 일치될 수 있음

# 신뢰 프레임워크

- 전통적 신원 교환 접근
  - 온라인/네트워크 업무, 조직과 온라인 고객과 같은 개인 사용자 사이에서는 신원 정보의 공유가 일반적으로 요구됨
    - 간단한 이름/ 숫자 식별자에 추가적으로 관련 속성의 호스트도 포함할 수 있음
  - 정보를 제공하거나 수신하는 쪽은 정보에 관련된 보안과 프라이버시 이슈에 대한 신뢰 레벨이 필요함

- 신뢰 당사자
  - 사용자가 어느 정도 인증되었고, 신원 서비스 제공자가 입력한 사용자 속성은 정확하고,
  - 신원 서비스 제공자는 이러한 속성들에 대해 인가되었음을 요구함
- 신원 서비스 제공자
  - 사용자에 대해 정확한 정보를 가지고 있고, 만약 정보를 공유한다면, 신뢰 당사자는 계약 조건과 법에 맞게 사용할 것에 대한 보장을 요구
- 사용자
  - 신원 서비스 제공자와 신뢰 당사자가 민감한 정보에 대해 신뢰받고 사용자의 취향을 유지
  - 사용자 프라이버시를 존중할 것에 대한 보장을 요구




(a) 전통적인 신원 정보 교환 관계자의 삼각형

**신원 정보 교환 접근 방법**

# 공개 신원 신뢰 프레임워크(Open Identity Trust Framework)

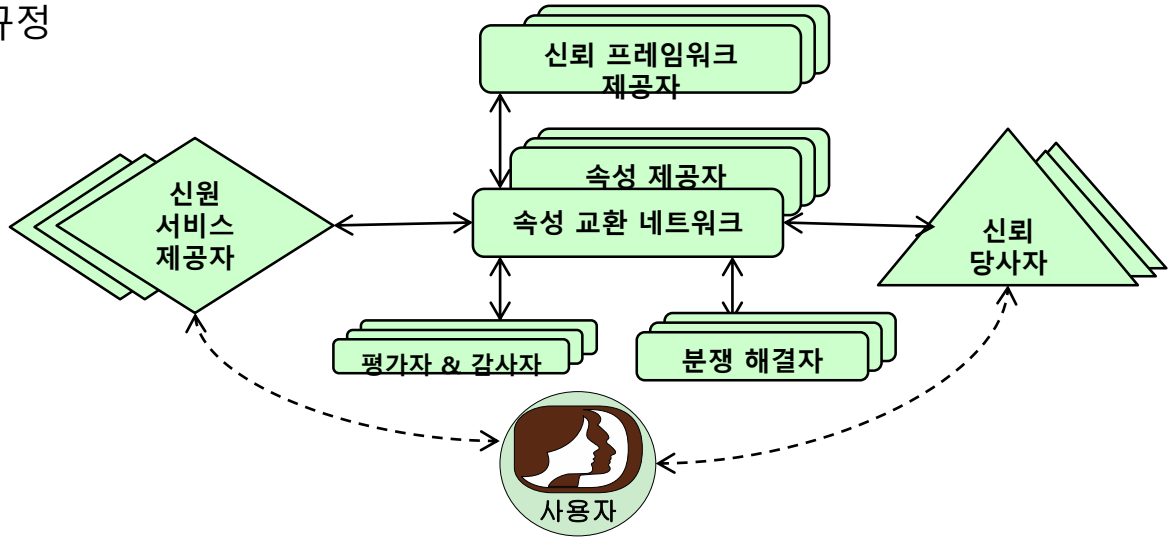
- **약어**
  - **공개ID(OpenID)** : 사용자가 타사 서비스를 사용하여 웹마스터가 자신의 임시 시스템을 제공할 필요가 없고 사용자가 자기 전자 신원을 강화할 수 있게 하여 특정 사이트에서 인증되도록 하는 공개표준
  - **OIDF(OpenID Foundation)** : 공개ID  재단은 공개 ID 기술의 증진, 보호를 위한 개인과 기업들의 국제적 비영리 기관
  - **ICF(Information card Foundation)** : 정보 카드 에코시스템의 발전을 위해 일하는 기업과 개인의 비영리 커뮤니티.
  - **OITF(Open Identity Trust Framework)** : OIDF와 ICF가 개발한 신원과 속성 교환을 위한 신뢰 프레임워크의 표준화된 공개 스펙
  - **AXN(Attribute Exchange Network)** : 신원 서비스 제공자와 신뢰 당사자가 대용량 사용자가 주장, 허가, 검증된 온라인 신원 속성을 저렴하고 효율적으로 접근하도록 하는 온라인 인터넷 규모의 게이트웨이

- 신뢰 프레임 워크(Trust Framework)
  - 인증 프로그램으로 기능
  - OIX(Open Identity Exchange) - 신뢰 프레임 워크를 트랜잭션의 여러 관계자들로부터의 검증 가능한 약속의 집합
    1. 약속의 전달을 보장하는 제어(규정과 계약상 의무)
    2. 오류에 대한 처리 방법을 포함

  - 신뢰 프레임 워크는 멤버들이 유사한 목표와 관점을 갖는 하나의 커뮤니티에 의해 개발

- 공개 신원 신뢰 프레임워크(Open Identity Trust Framework)
  - 신뢰 당사자(RPs) : 서비스 제공자
    - 서비스를 특정 사용자에게 전달
      신원과 의도한 사용자의 속성에 대한 신용이 있어야 하고, 속성과 신원을 증명하기 위해 제시된 다양한 신용장에 의존
  - 주체 : 고객, 고용인 등 RP 서비스의 사용자
  - 속성 제공자(Aps) : 특정 정보의 권한의 소스, 도출된 속성의 중개자
  - 신원 제공자(IDPs) : AXN 혹은 사용자 속성을 사용되는 전자 신원을 생성하는 다른 호환되는 신원과 접근관리(IDAM) 시스템을 통해 사용자 신용장을 인증, 주체의 이름을 보증할 수 있는 존재
  - AXN의 부분으로 중요한 지원 요소
    - 평가자 : 신원 서비스 제공자와 RPs 를 평가하고 OITF 제공자의 청사진을 따를 수 있음을 보증
    - 감사자 : 관계자의 사례가 OITF 의 합의 사항에 부합되는지 확인
    - 분쟁 해결자 : OIX(Open Identity Exchange) 가이드라인에 따라 분재의 해결과 조정
    - 신뢰 프레임워크 제공자 : 정책 입안자의 요구 사항을 신뢰 프레임워크의 청사진에 번역해 놓고 OITF 스펙의 최소 요구 사항에 부합되게 진행하도록 하는 기관

- 동작
  - 실선 화살표 - 신뢰 프레임워크 제공자가 구현의 기술적, 동적, 법적 요구 사항에 합의
  - 점선 - 요구 사항에 의해 잠재적으로 영향을 받는 다른 합의
  - 참여 기관의 책임자 - 신원 정보의 인가에 필요한 기술적 운영적 그리고 법적인 요소들을 결정하고, 필요 요소들을 구현하기 위하여 OITF 제공자를 선택
  - OITF 제공자들 - 필요한 요소들을 OITF 제공자의 부가적인 조건들을 포함하는 청사진으로 번역
  - OITF 제공자들은 신원 정보를 교환할 때 신뢰 프레임워크 조건들을 준수하기 위해 신원 서비스 제공자들과 RP 그리고 계약사항을 면밀히 검사
  - 계약 - 계약의 해석과 준수를 위한 논쟁 해결자와
  - 감사관에 관한 조항들을 규정




**(b)** 신원 속성 교환 요소

# 사례연구 : 은행을 위한 RBAC 시스템


- 사례연구 : 은행을 위한 RBAC 시스템


  - 1990년대에는 간단한 DAC 시스템이 각각 서버와 대형 컴퓨터에 사용되었다. 관리자는 각각의 호스트에 지역 접근 제어 파일을 관리하고 각각의 호스트의 각 응용프로그램에 대한 각 직원들의 접근 권한을 정의했고 이러한 시스템은 부담이 크고 시간이 많이 걸리고 또한 에러가 자주 발생했음

  - 이 시스템을 개선하기 위해서, 은행은 전 조직에 걸치는 RBAC 구조를 도입했고 이 구조에서는 더 큰 보안성을 위해 접근 권한의 결정이 세 가지의 다른 관리 단위로 분류되었음

  - 기관 내 역할은 공식 지위와 직무의 조합으로 정의
    - 모든 경우에, 은행의 역할 구조화는 공식적인 지위에 기반한 상속 계층을 발전시키는 자연스러운 수단을 이르게 함.
    - 은행 안에서는, 책임과 권력의 계층 구조를 나타내는 각각의 조직 내 공식적인 지위의 엄격한 순서가 존재함.
      Ex) 부장,그룹매니저, 그리고 직원은 내림차순의 관계

| 역할 | 기능 | 공식 직위 |
|---|---|---|
| A | 재정 분석가 | 사무원 |
| B | 재정 분석가 | 그룹 매니저 |
| C | 재정 분석가 | 부서장 |
| D | 재정 분석가 | 주니어 |
| E | 재정 분석가 | 시니어 |
| F | 재정 분석가 | 전문가 |
| G | 재정 분석가 | 보조 |
| … | … | … |
| X | 주식 기술자 | 사무원 |
| Y | E-commerce 지원 | 주니어 |
| Z | 은행 사무 | 부서장 |

**(a)** 기능과 공식 직위

은행 기능과 역할 예제

- 공식 지위가 직무와 혼합이 되면,표에 나타나는 것처럼 접근 권한 순서의 결과 나옴
  - 재무 분석가/그룹 매니저 역할(역할 B)은 재무 분석가/직원 역할(역할 A)는 3개의 응용프로그램에 대한 접근 권한을 가진 역할 A보다 많은 4개의 응용프로그램에 대한 접근 권한을 가지고 있는 역할 B가 더 많은 접근 권한을 가지고 있는 것을 나타냄
  - 반면에, 은행 사무/그룹 매니저와 재무 분석가/직원은 업무 분야가 다르기 때문에 그들 사이의 관계에는 계층 구조가 없음

- 우리는 역할의 지위가 다른 역할의 상위에 있고 그들의 직무가 같은 때에만, 역할이 다른 역할보다 상위의 역할인 역할 구조를 정의할 수 있음

- 역할 계층은 표에 제안된 것처럼 접근 권한 정의를 절약할 수 있음

(b) 승인 할당

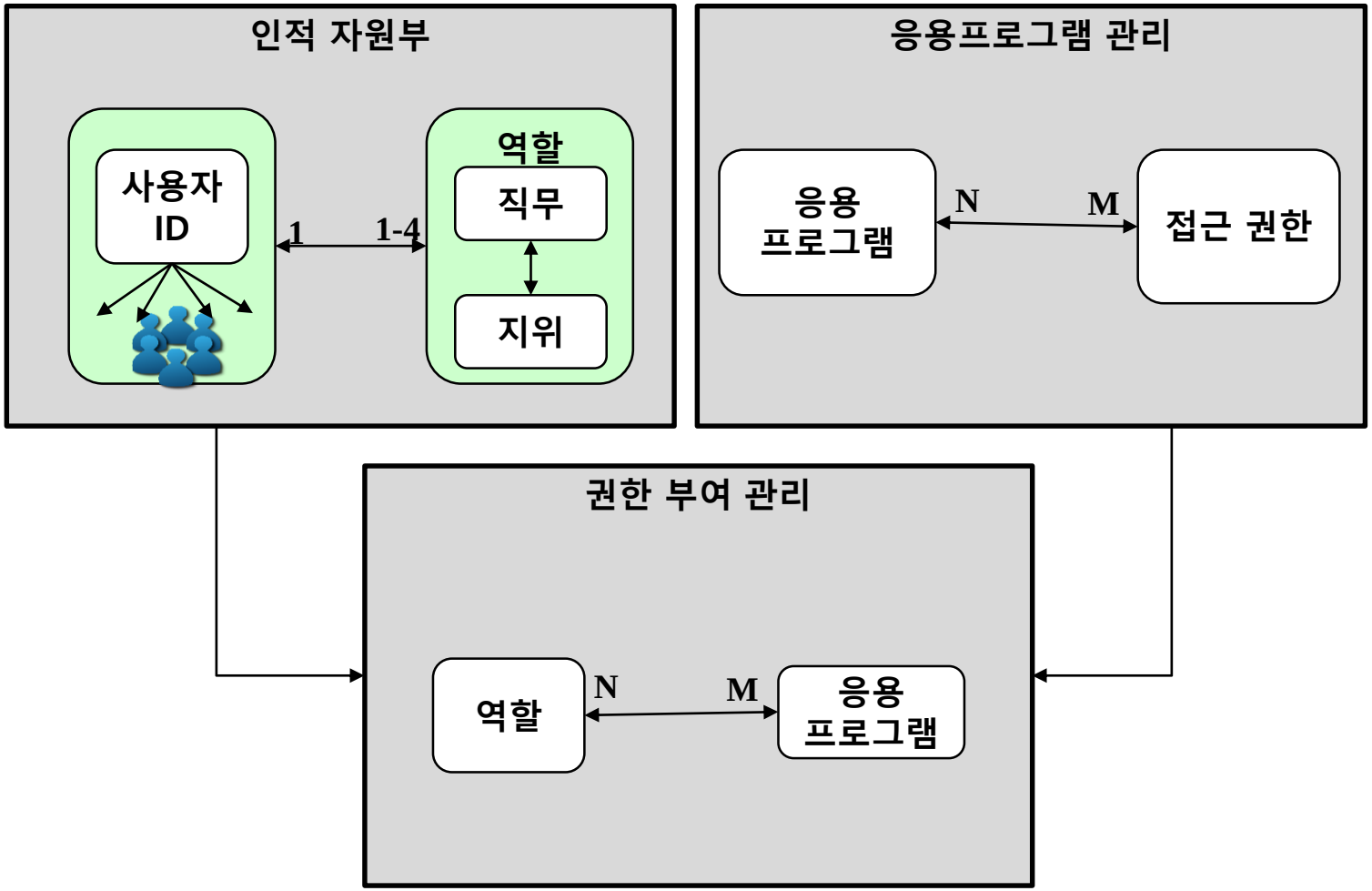
| 역할 | 응용 프로그램 | 접근 권한 |
|---|---|---|
| A | 금융 시장 도구 | 1,2,3,4 |
| | 파생상품 거래 | 1,2,3,7,10,12 |
| | 이자 도구 | 1,4,8,12,14,16 |
| B | 금융 시장 도구 | 1,2,3,4,7 |
| | 파생상품 거래 | 1,2,3,7,10,12,14 |
| | 이자 도구 | 1,4,8,12,14,16 |
| | 개인 소비자 도구 | 1,2,4,7 |
| … | … | … |

(c) 승인 할당과 상속

| 역할 | 응용 프로그램 | 접근 권한 |
|---|---|---|
| A | 금융 시장 도구 | 1,2,3,4 |
| | 파생상품 거래 | 1,2,3,7,10,12 |
| | 이자 도구 | 1,4,8,12,14,16 |
| B | 금융 시장 도구 | 7 |
| | 파생상품 거래 | 14 |
| | 개인 소비자 도구 | 1,2,4,7 |
| … | … | … |

## 은행 기능과 역할 예제

- 원래의 구조에서는,각 사용자에게 접근 권한을 직접 할당하는 것은 응용프로그램 수준에서 발생했고 각각의 응용프로그램과 연관 있었다. 새로운 구조에서는, 응용 프로그램 관리자가 각각의 응용프로그램과 연관된 접근 권한 집합을 결정
  그러나 특정한 작업 수행하는 특정 사용자는 그 응용프로그램과 관련된 모든 접근 권한 허가 받지 못할 수도 있음
- 접근 제어 관리의 예
  - 인적자원부는 시스템을 사용할 각각의 직원에게 유일한 사용자 ID를 부여하고, 사용자의 지위와 직무에 기반해서, 그 부서는 한 개 이상의 역할을 사용자에게 할당함.
  - 사용자/역할 정보는 인증 관리 모듈에 제공되고 사용자 ID와 역할을 접근 권한에 연관시키는 각 사용자에 대한 보안 프로필을 생성
    사용자가 응용프로그램을 호출했을 때, 사용자의 역할에 맞게 어떤 응용프로그램의 접근 권한의 부분 집합이 수행되어야 하는지를 결정하기 위해서 응용프로그램은 사용자에 대한 보안 프로필을 참고
  - 하나의 역할은 몇몇의 응용프로그램에 접근하기 위해서 사용될 수 있음.
    표 4.4b 에서 보면 역할 A는 매우 많은 접근 권한을 가지고 있지만, 그 권한 중에서 일부분만이 역할 A가 호출할 수 있는 각각 세 개의 응용프로그램에 해당


인적 자원부
사용자
ID
1
1-4
역할
직무
지위
응용프로그램 관리
응용
프로그램
N
M
접근 권한
권한 부여 관리
역할
N
M
응용
프로그램

접근 제어 관리의 예