

# 인증 기술

---

2024. 9

컴퓨터·소프트웨어공학과  
이 형 효  
(hlee@wku.ac.kr)

# 정보보호 목적(1)

- 비밀성(Confidentiality, Secrecy) 
  - 비인가 사용자에 의한 정보 조회 방지
- 무결성(Integrity) 
  - 비인가 사용자에 의한 정보 변경 방지
  - 적법한 사용자에 의한 부적절한 정보변경 방지
- 가용성(Availability) 
  - 적법한 사용자에 대한 서비스 거부 방지
  - The property that a product's services are accessible when needed and without undue delay

과도한, 심한, 지나친

# 정보보호 목적(2)

## ■ 책임성(Accountability)

- 자신이 수행한 작업에 대한 책임 부과
- 사용자가 실행한 작업의 내역(**로그, log**) 이용
- 일상생활에서 디지털 작업의 일상화로 그 중요성 증가

디지털포렌식(digital forensics)

전자적 증거물 등을 사법기관에 제출하기 위해  
데이터를 수집, 분석, 보고서를 작성하는 일련의 작업

## ■ 신뢰성(Reliability)

- 불의의 사고, 고장에 대한 대처 기능

# 컴퓨터 보안 정의

## ■ Computer Security



## ■ 그러나

- 보안에 대한 유일한 정의는 존재하지 않음
  - 책이나 문서마다 사용하는 정보보안에 대한 정의가 서로 다를 수 있음

# 컴퓨터 보안의 오버헤드

비용, 부담

- 컴퓨터 보안기능 실행 위한 컴퓨팅 자원 추가
  - CPU time, Memory, S/W 등
- 사용자에게 사용의 불편함 초래  
교환, 거래
  - Tradeoff between *security* and *ease-of-use* 사용 편리성
- 보안 관리를 위한 비용  
그래픽 사용자 인터페이스
  - 우수한 GUI(Graphical User Interface)를 가진 보안 제품 구입
- 보안은 이러한 오버헤드를 부담할 만한 가치가 있음

# 정보보호 조건

## ■ 보호대상의 자산(**Asset**)

- 보호할 가치를 가진 보호대상이 존재
- 컴퓨터, 통신망, 정보 등

## ■ 시스템의 취약성(**Vulnerability**)

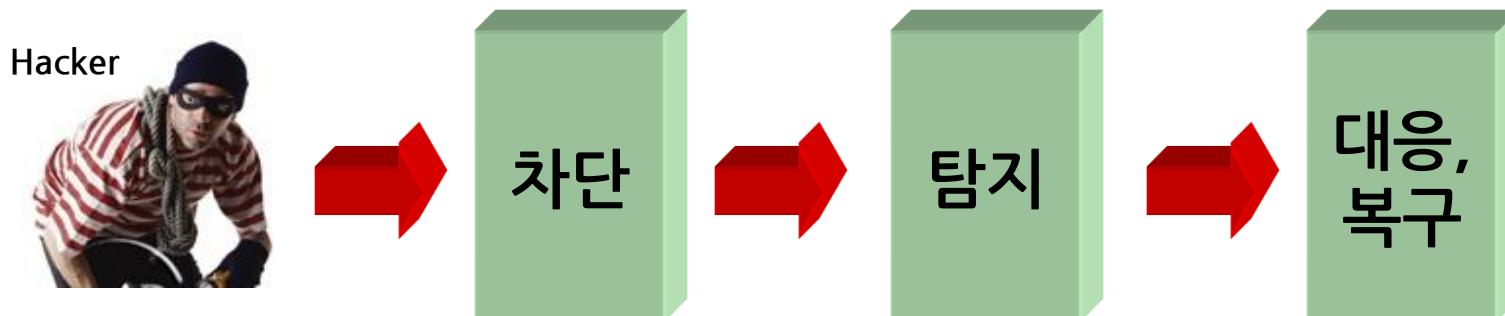
- 완전하지 않은 시스템의 보안
- 하드웨어 또는 소프트웨어의 결함, 오작동, 관리부실

## ■ 해커로부터의 위협(**Threats**)

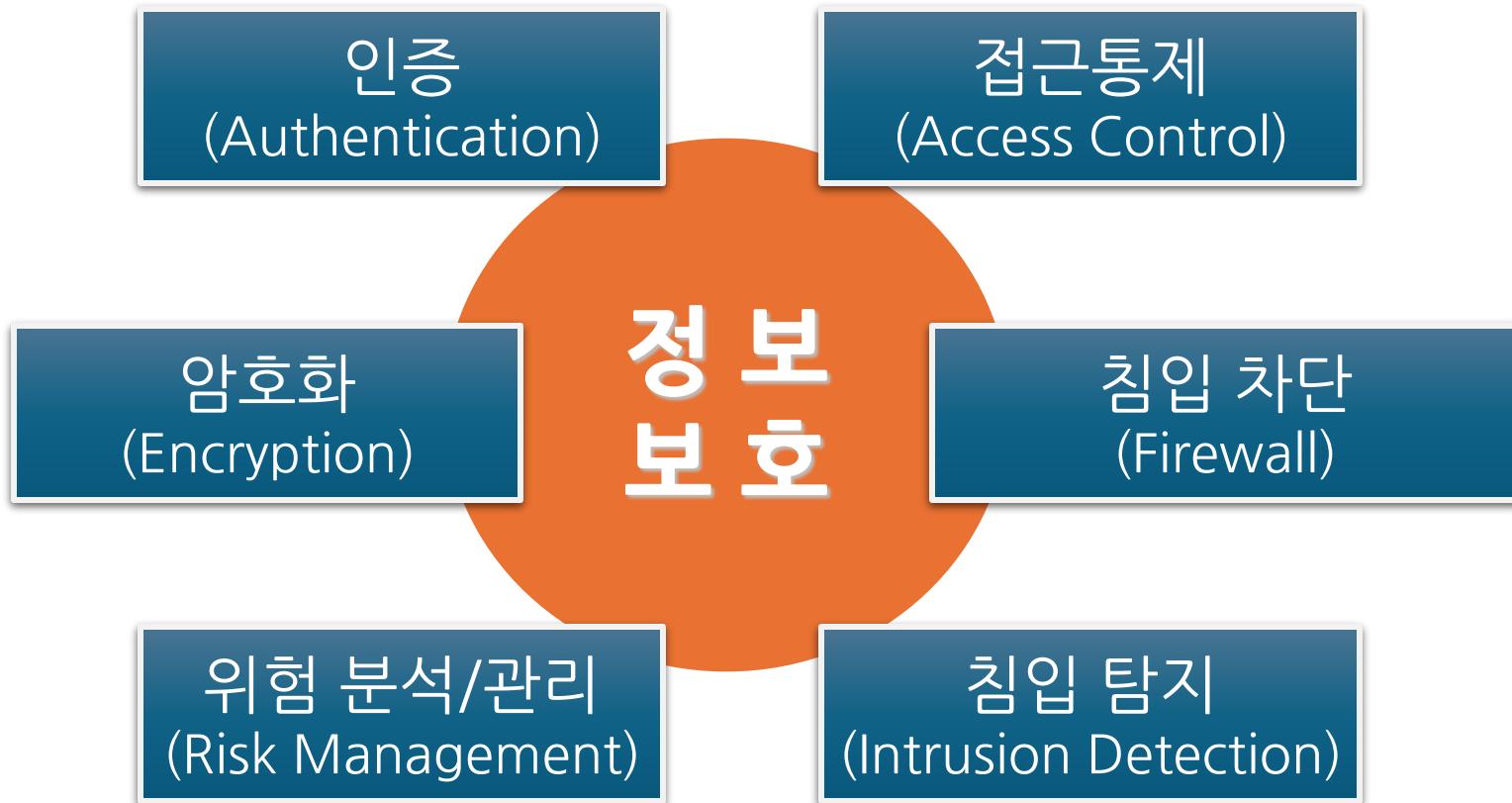
- 보호대상에 대한 내부 또는 외부로부터의 위협 존재
- 해커 또는 내부의 공격자

# 정보보호 방법, 절차

- 차단
  - 암호 기술, 방화벽(Firewall)
- 탐지(Detection)
  - 침입탐지시스템(IDS: Intrusion Detection System)
- 대응(Response), 복구(Recovery)
  - 침입 피해에 대한 대응 또는 복구



# 정보보안 기술



# 정보보호 용어 정의(1)

- **인증(Authentication)**
  - 사용자 신분 확인
- **접근통제(Access Control)**
  - 신분이 확인된 사용자의 정보에 대한 접근권한 확인
- **암호화(Encryption)**
  - 비인가 사용자가 정보를 조회할 수 없도록 자료를 다른 형태로 변형하는 작업
- **복호화(Decryption)**
  - 암호화의 역과정

# 정보보호 용어 정의(2)

## ■ 침입차단시스템(Firewall)

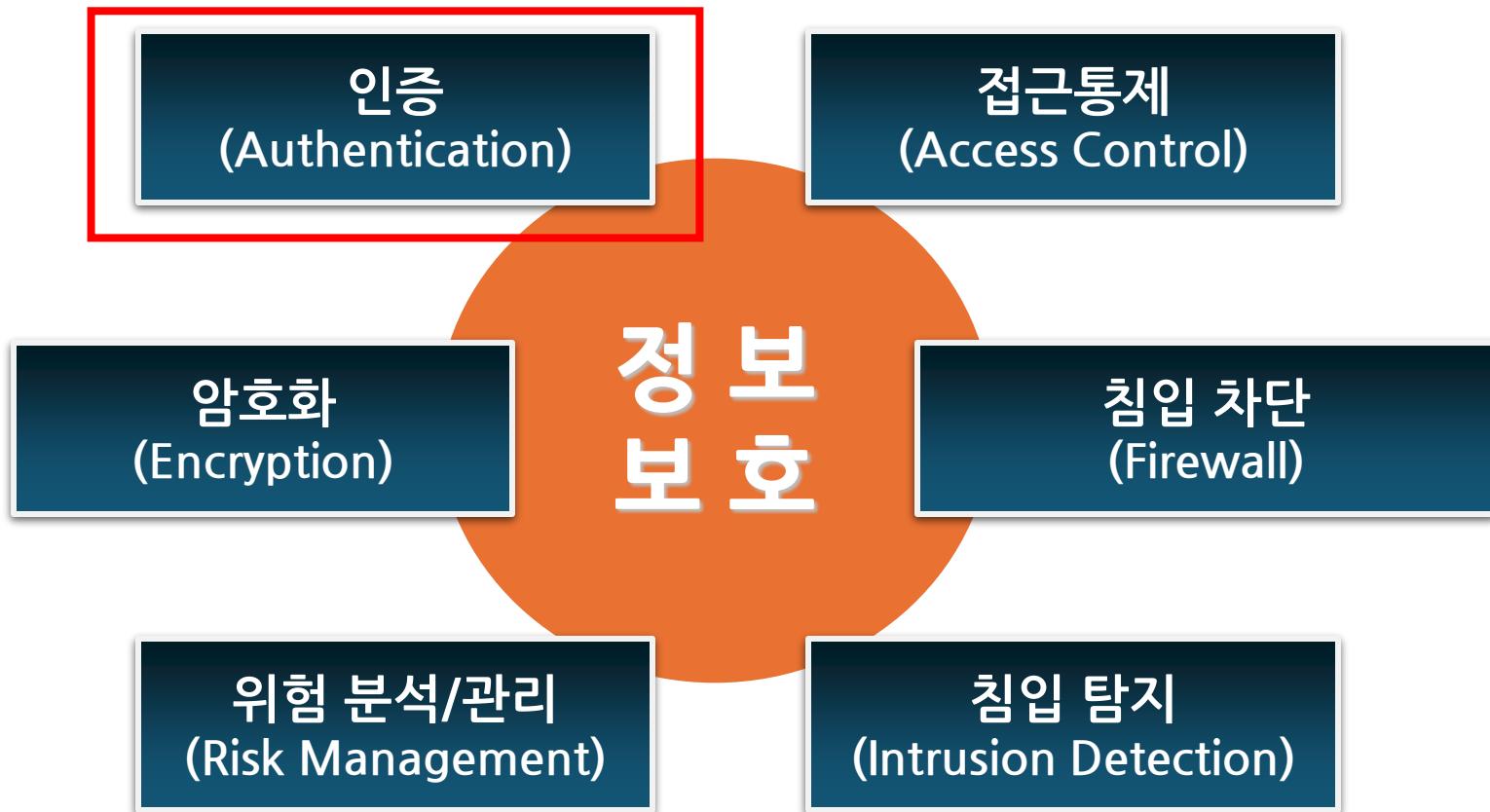
- 허용되지 않은 시스템으로부터의 접속 차단
- 하드웨어(H/W)와 소프트웨어(S/W)로 구성
  - ✓ 예: 라우터를 통한 패킷 필터링 방식

## ■ 침입탐지시스템(IDS)

- Intrusion Detection System
- 시스템에 침입한 불법 사용자를 탐지하여 경고 발생

# 인증 (Authentication)

# 정보보안 기술 - 인증



# 인증(Authentication)(1)

- 사용자의 신분을 확인하는 절차
- 필요성(인증결과의 활용)
  - 인증 이후 사용자의 권한을 확인하는 데 사용
  - 사용자의 사용기록(로그) 작성에 사용
- Identification vs. Authentication
  - Identification: announce who you are
  - Authentication: prove that you are who you claim to be

# 인증(Authentication)(2)

The image consists of two side-by-side screenshots. On the left is a screenshot of a web service login page titled '웹정보서비스'. It features two input fields: '웹정보서비스 아이디' and '비밀번호', each with a placeholder icon (envelope for ID and lock for password). Below these fields are two buttons: a blue '로그인' button and a black '회원가입' button. At the bottom of the page are links for '아이디 찾기' and '비밀번호 찾기'. Two red arrows point from the text 'Identification?' to the 'ID' and 'password' fields. On the right is a screenshot of a notice board for 'WONKWANG UNIVERSITY'. The board displays several bullet points in Korean:

- 원광대학교 교내 소프트웨어설치 안내
- 불법소프트웨어 사용 및 지적재산권에 대한 안내
- 원광대학교 알리미 설치 안내
- 웹메일(Office365) 사용안내

Below the text, there is a photograph of three people standing outdoors near a glass railing. In the foreground, there are some cartoonish illustrations of characters.

Identification?

Authentication?

# 인증 (Authentication)(3)

실생활에서의 **식별** 사례

실생활에서의 **인증** 사례

# 인증(Authentication)(4)

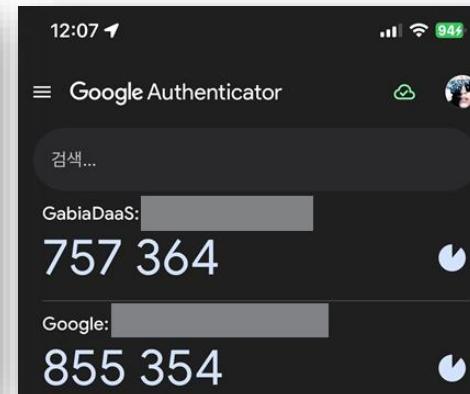
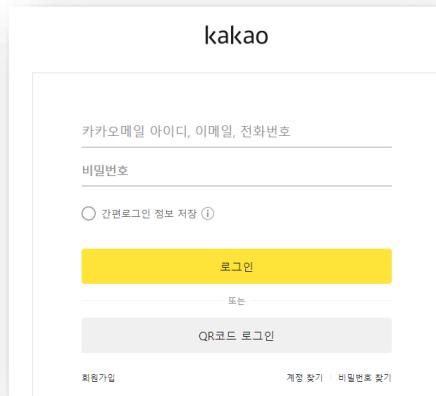
## ■ 인증 (Authentication)

- 컴퓨터 사용자의 신원을 확인하는 절차

✓ *something you know*

✓ *something you have*

✓ *something you are*



# 인증 방식(1)

- 특별한 정보를 알고 있는지
  - Something you know
  - ID/PASSWORD
- 특별한 물건을 가지고 있는지
  - Something you hold (have)
  - 스마트폰, OTP 단말기, 열쇠, 신분증
- 특별한 신체적 특성을 가지고 있는지
  - Something(Who) you are
  - 지문, 홍채, 정맥, DNA, … (생체인식 biometrics)

# 인증 방식(2)

- 특별한 행동특성을 보이고 있는지
  - **What you do**
  - 타이핑 속도와 시간 간격
- 특별한 장소에 있는지
  - **Where you are**
  - 사용자의 컴퓨터 접속 장소 (GPS 정보)

# 인증 방식(3)

## ■ 다중 요소 인증

- MFA(Multi-Factor Authentication)
- 2FA (2-Factor Authentication)
- 서로 다른 인증 방식 2개 이상을 적용한 인증 방식
  - ✓ (O) Something you **know** + Something you **have**
  - ✓ (O) Something you **know** + Something you **are**
  - ✓ (O) Something you **have** + Something you **are**
  - ✓ (O) Something you **know** + Something you **have** +  
Something you **are**
  - ✓ (X) Something you **know** + Something you **know**
  - ✓ (X) Something you **have** + Something you **have**

# 전통적 인증방식 취약점

- 물리적 신분증 조작/변조 사례



# 지문 인식 취약점(1)

- 지문 복제 사례



# 지문 인식 취약점(2)

- 가짜 손가락 이용 범죄 사례 (SBS, 2019)



# 홍채 인식 특징 및 취약점

## ■ 생체 인식 사례: 홍채(iris) 인식

- 홍채의 모양과 색, 모세혈관의 형태소를 분석하여 사람을 인식하는 기술
- 한 번 패턴이 완성되면 오랜 기간 동안 변하지 않은 특징
- 정확성이 우수한 생체기술로 평가



참고영상 (영화)

마이너리티 리포트  
(2002)

독일 해커그룹(CCC),  
사진으로 성 갤럭시 S8  
홍채인식 뚫었다  
(2017.5.24)

# Username/Password 인증(1)

## ■ 간단한 사용자 인증 방식

- **Something you know** 방식
- 정보보안의 첫 단계
- 사용자에게 최소의 불편함 제공
- 구현이 비교적 간단함

## ■ Username 제시

- **Identification** 과정

## ■ Password 제시

- **Authentication** 과정

# Username/Password 인증(2)

## ■ 인증 과정

- 제시된 사용자ID/비밀번호 정보를 등록된 사용자ID/비밀번호 정보와 비교
- 패스워드 파일>Password file)
- Linux 시스템의 경우 /etc 디렉토리에 존재

## ■ 초기의 UNIX 계열 시스템

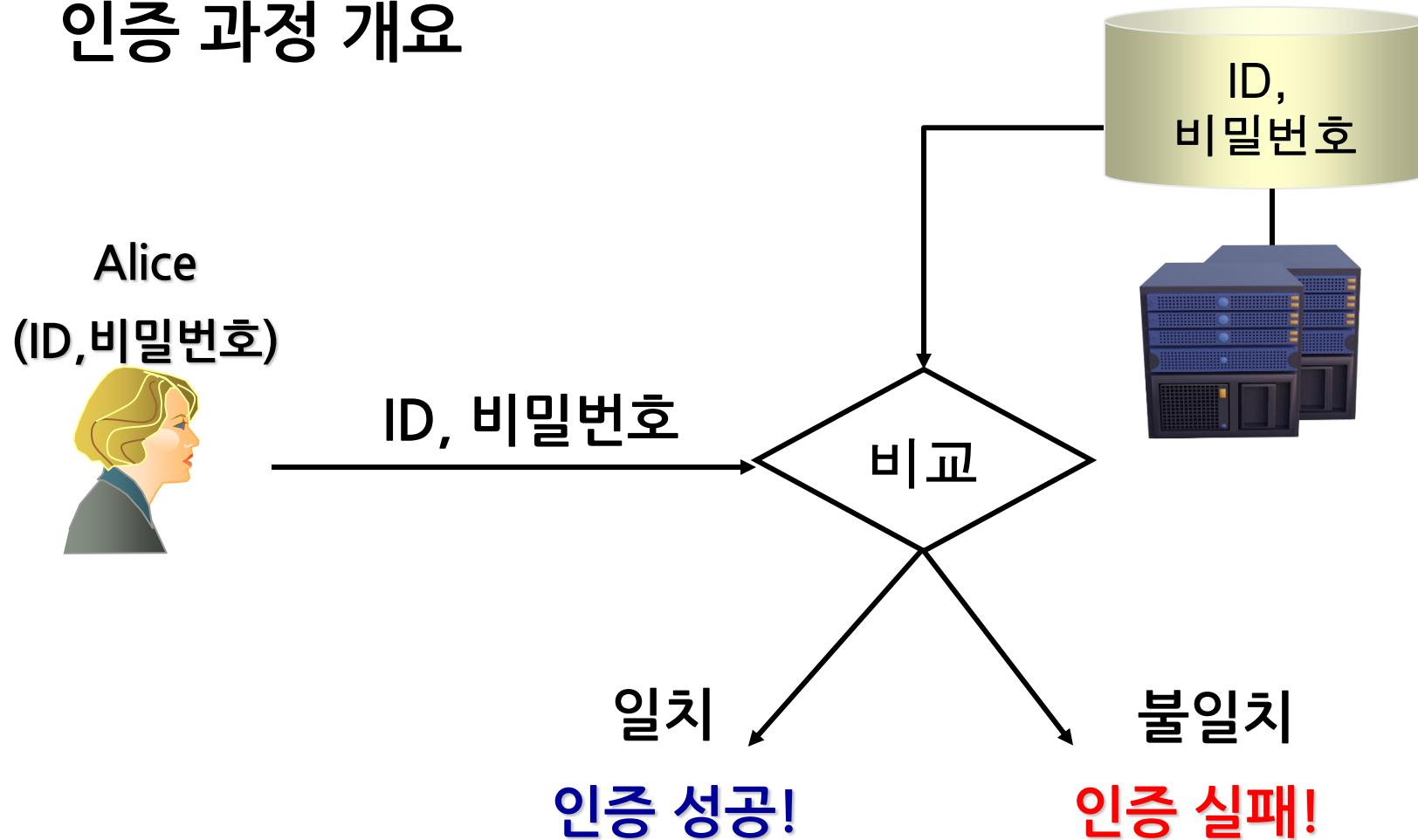
- 패스워드 파일 안에 암호화된 비밀번호 포함

## ■ 현재의 UNIX 계열, Linux 시스템

- 보안문제로 암호화된 비밀번호를 분리, 저장

# Username/Password 인증(3)

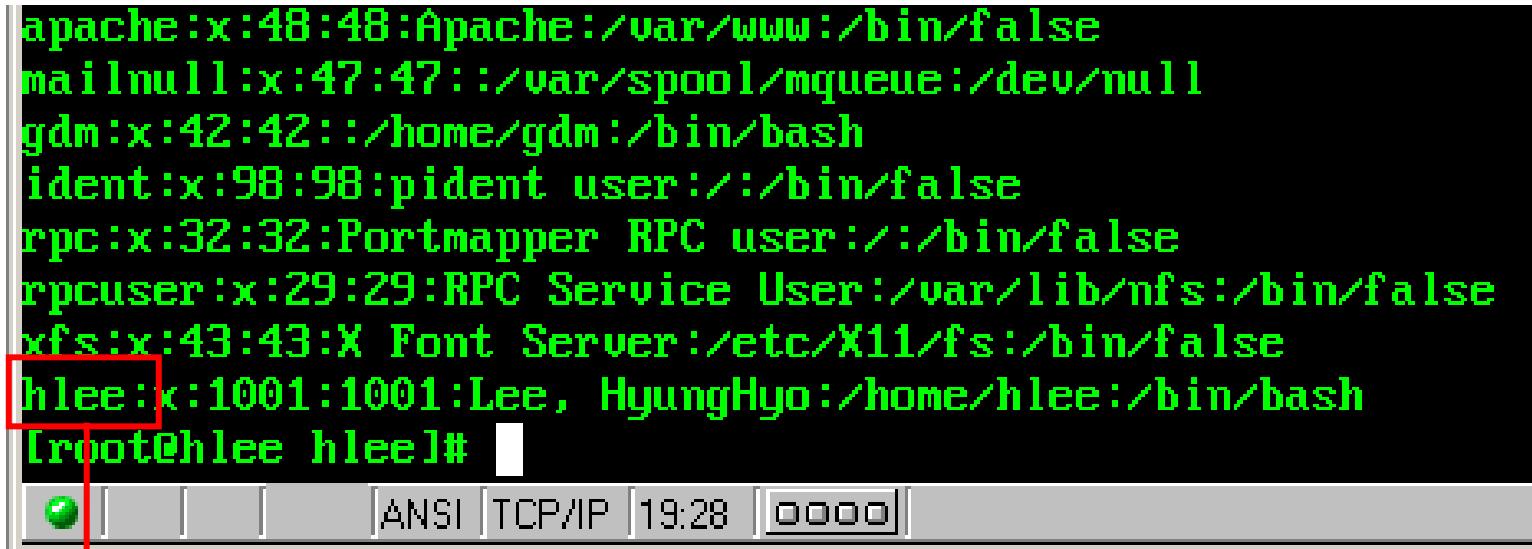
## ■ 인증 과정 개요



# Username/Password 인증(4)

- 패스워드 파일 내용 예(/etc/passwd)

```
apache:x:48:48:Apache:/var/www:/bin/false
mailnull:x:47:47::/var/spool/mqueue:/dev/null
gdm:x:42:42::/home/gdm:/bin/bash
ident:x:98:98:pident user:/:/bin/false
rpc:x:32:32:Portmapper RPC user:/:/bin/false
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/bin/false
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
hlee:x:1001:1001:Lee, HyungHyo:/home/hlee:/bin/bash
[root@hlee hlee]#
```



사용자ID

# Username/Password 인증(5)

- 암호화된 비밀번호가 저장된 파일 내용 예  
(/etc/shadow)

```
ftp:*:11570:0:99999:7:::  
nobody:*:11570:0:99999:7:::  
nscd:!!:11570:0:99999:7:::  
apache:!!:11570:0:99999:7:::  
mailnull:!!:11570:0:99999:7:::  
gdm:!!:11570:0:99999:7:::  
ident:!!:11570:0:99999:7:::  
rpc:!!:11570:0:99999:7:::  
rpcuser:!!:11570:0:99999:7:::  
xfs:!!:11570:0:99999:7:::  
hlee:kCK9MAI6T5VEQ:11571:0:99999:7:::  
[root@hlee hlee]#
```



사용자ID



암호화된 비밀번호

[ANSI TCP/IP 19:27 0000]

# Username/Password 인증(6)

- **일방 인증(Unilateral Authentication)**
  - 시스템 입장에서 사용자의 ID, 비밀번호만 점검
  - 사용자가 접속한 시스템에 대한 인증기능 없음
- **(참고) 상호 인증(Mutual Authentication)**
  - 시스템이 사용자를 인증하는 과정
  - 사용자가 시스템을 인증하는 과정
  - 속임 프로그램(spoofing program)에 대한 대응 방법

# Username/Password 인증(7)

## ■ 반복 인증(Repeated Authentication)

- 세션의 시작뿐만 아니라 사용 중 일정기간마다 인증기능 수행
- 예: 자동 화면잠금 후 비밀번호 입력 요구

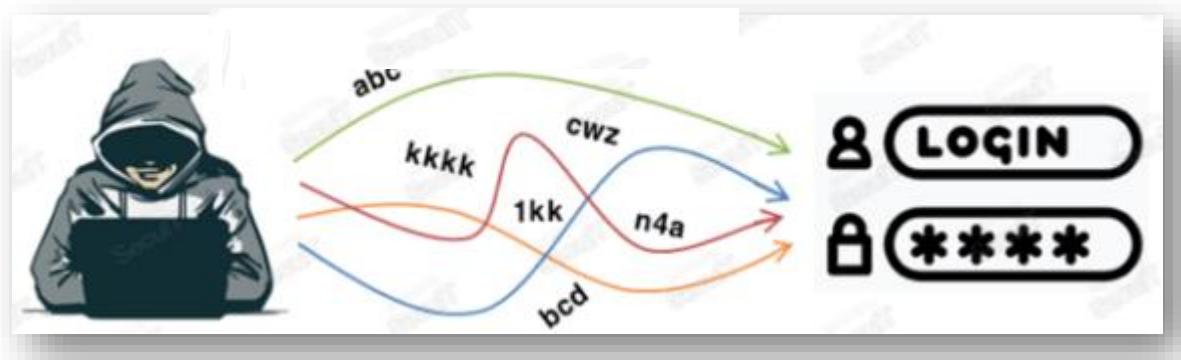
# Username/Password 인증(8)

## ■ 보안 위협

- 패스워드 추측(password guessing) 공격
- 로그인 프로그램 속임(login program spoofing) 공격

# Username/Password 인증(9)

- 패스워드 추측(password guessing) 공격
  - Brute force 공격 (무차별 대입 공격)
    - ✓ 가능한 모든 글자, 단어의 조합 이용



- 사전(Dictionary) 공격
  - ✓ 사전(dictionary)에 나타난 모든 단어를 이용



# Username/Password 인증(10)

## ■ 패스워드 추측(password guessing) 공격 대응

- 모든 계정에 패스워드 설정
- 긴 길이의 패스워드 사용
- 대문자, 소문자, 숫자, 특수문자를 혼합한 패스워드 사용
- 평범한 패스워드 사용 금지
  - ✓ 사전에 포함된 단어 사용 금지
- 패스워드 aging (패스워드 유효기간 설정)
- 로그인 시도 횟수 제한
- 최종 로그인 성공시간, 접속 IP를 사용자에게 제공

# Username/Password 인증(11)

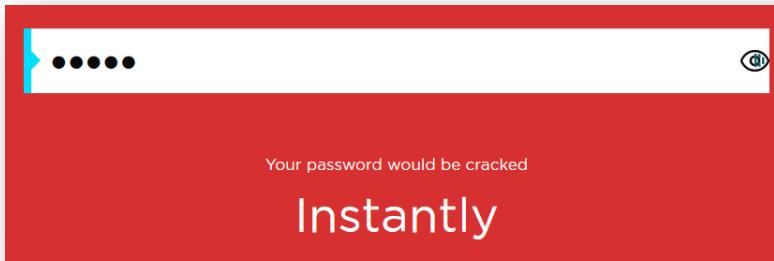
- 주어진 패스워드의 사전공격 취약점 점검 사이트
  - <https://www.security.org/how-secure-is-my-password/>



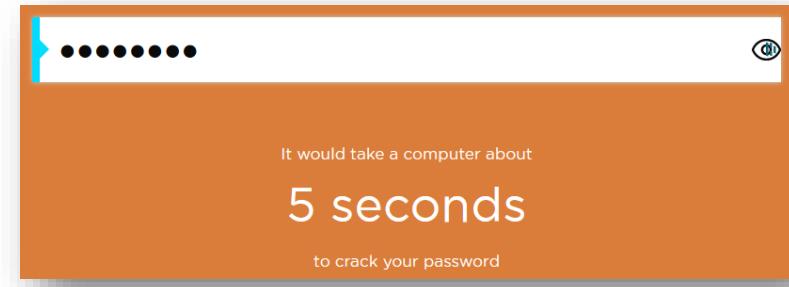
# Username/Password 인증(12)

## ■ 주어진 패스워드의 사전공격 취약점 점검 사이트

<apple>



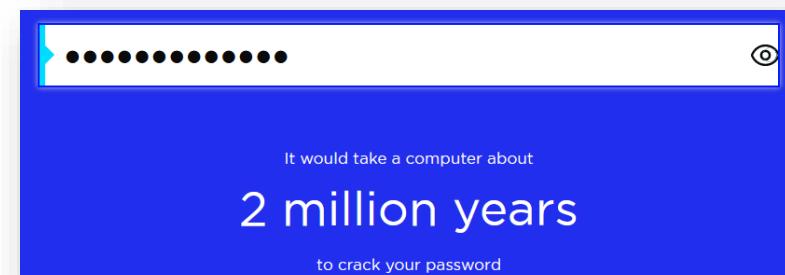
<wonkwang>



<wonkwang1234>



<HongGilDong@1>



# Username/Password 인증(13)

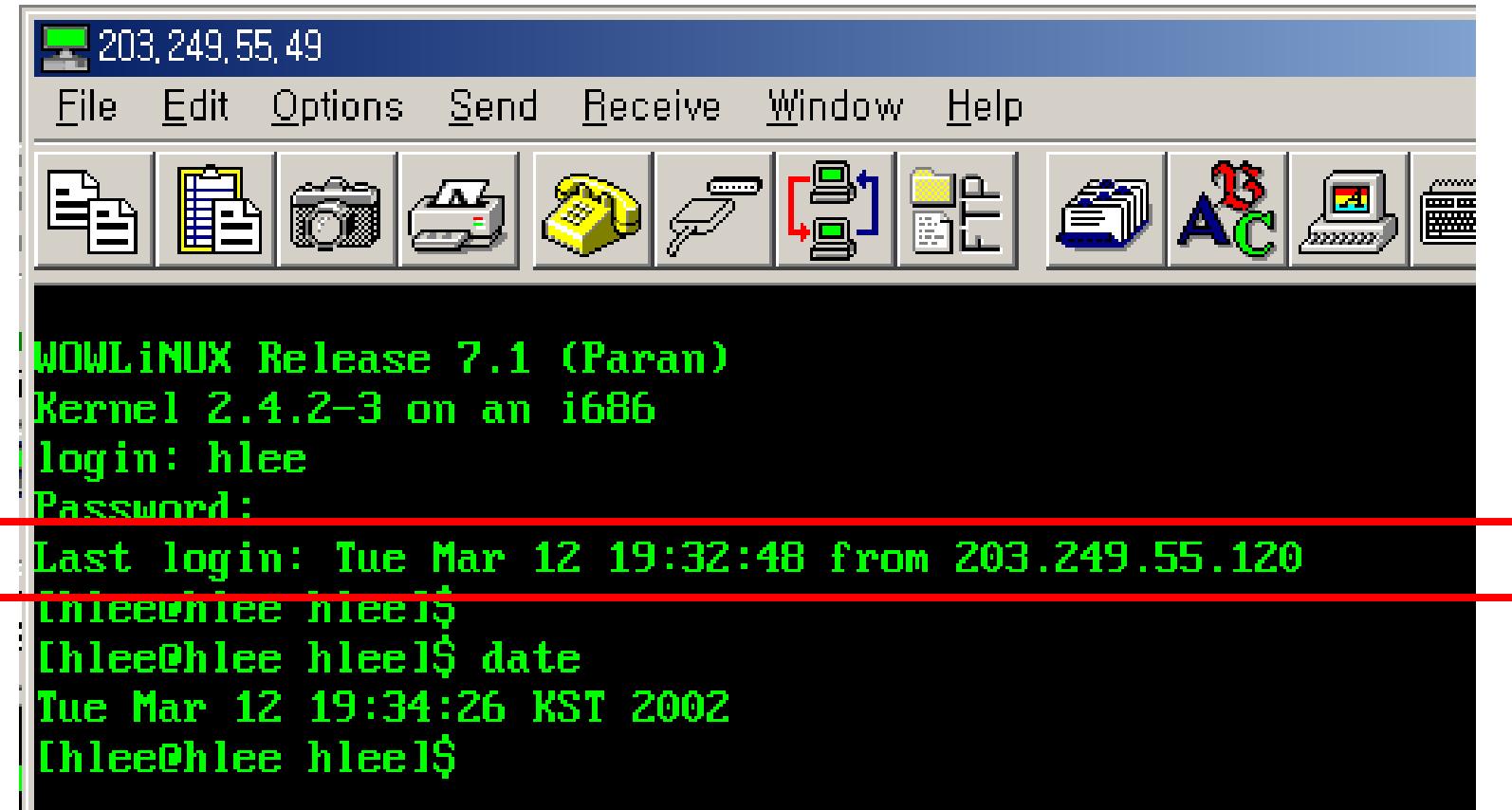
## ■ Top weakest passwords

Top 10 Worst Passwords - Historic Analysis

	2023	2015	2010	2005	2000
#1	123456	123456	123456	password	password
#2	123456789	password	password	123456	123456
#3	qwerty	12345	12345678	12345678	12345678
#4	password	12345678	qwerty	abc123	qwerty
#5	1234567	qwerty	abc123	qwerty	abc123
#6	12345678	1234567890	123456789	monkey	monkey
#7	12345	1234	111111	letmein	1234567
#8	iloveyou	baseball	1234567	dragon	letmein
#9	111111	dragon	iloveyou	111111	trustno1
#10	Covid	football	adobe123	baseball	dragon

# Username/Password 인증(14)

- 패스워드 추측(password guessing) 공격 대응



# Username/Password 인증(15)

- 로그인 프로그램 속임(login program spoofing) 공격
  - Username/password를 확인하는 시스템(SW) 위장
    - ✓ 가짜 로그인 프로그램 실행
  - 사용자가 입력한 username/password 불법 취득
    - ✓ 사용자 패스워드 입수 후 가짜 로그인 프로그램 종료

# Username/Password 인증(16)

## ■ 로그인 프로그램 속임(login program spoofing) 공격



# Username/Password 인증(17)

- 로그인 프로그램 속임(login program spoofing) 공격 대응
  - 로그인 성공 후 실패한 로그인 횟수 정보 제공
    - ✓ 로그인 실패 후 성공 한 세션에서 실패 로그인 횟수가 0이면 패스워드 속임 공격인지 의심, 점검 필요
  - 상호 인증(Mutual Authentication)
    - ✓ 사용자가 자신이 접속한 시스템을 인증하는 과정 추가

# 인증 종류

- **개체 인증(Entity Authentication)**
  - 사용자 또는 프로세스의 신분 확인
  - 예: Username/Password 인증
- **자료 출처 인증(Data Origin Authentication, Message Authentication)**
  - 자료의 전송자 또는 전송시스템 확인
  - 예: 전자서명(digital signature)

# 인증 기술 정리

- 인증의 정의
  - identification, authentication
- 인증을 수행하는 목적(필요성)
- 인증방식의 종류(분류) 및 특징
- ID/PW 방식의 인증 특징, 취약점 등
- 인증의 종류
  - 개체 인증, 메시지 출처 인증

# 간편 인증(1)

## ■ 간편 인증 도입 배경

- 복잡한 공인인증서 발급 및 설치, 사용 문제점 보완(2020.12)



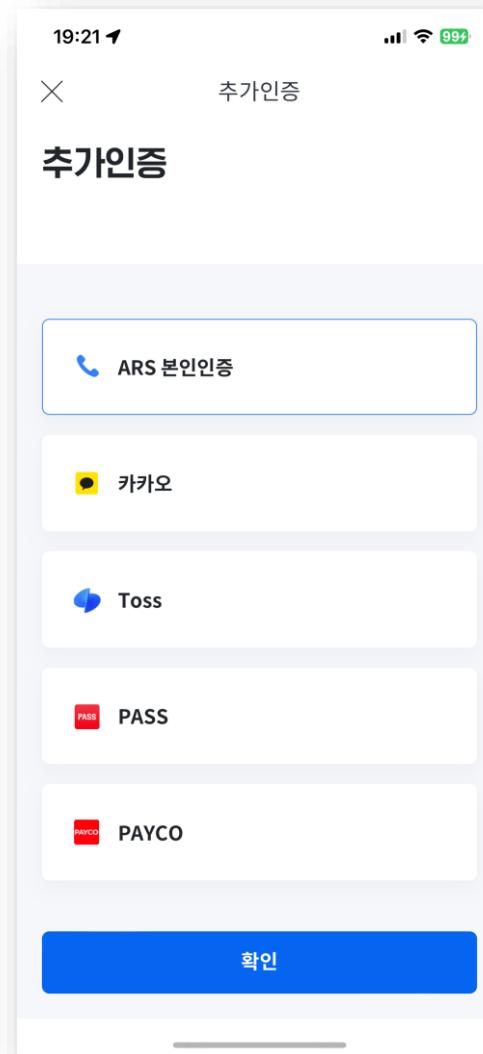
# 간편 인증(2)

## ■ 간편 인증 추진

- 2020.12: 전자서명법 개정 및 공인인증서 독점적 지위 폐지
- 2020.12: 공공분야 민간 전자서명 확대 도입, 5개 시범사업자
  - ✓ 카카오, 통신사PASS, 한국정보인증(삼성PASS), KB국민은행, NHN페이코 (5종)
- 2021.11: 네이버, 신한은행 추가 (5종 → 7종)
- 2022.2: 토스, 뱅크샐러드 추가 (7종 → 9종)
- 2022.11: 드림인증, 하나은행, NH농협은행 (9종 → 12종)
- 2023.12: 카카오뱅크, 우리은행 (12종 → 14종)

# 간편 인증(3)

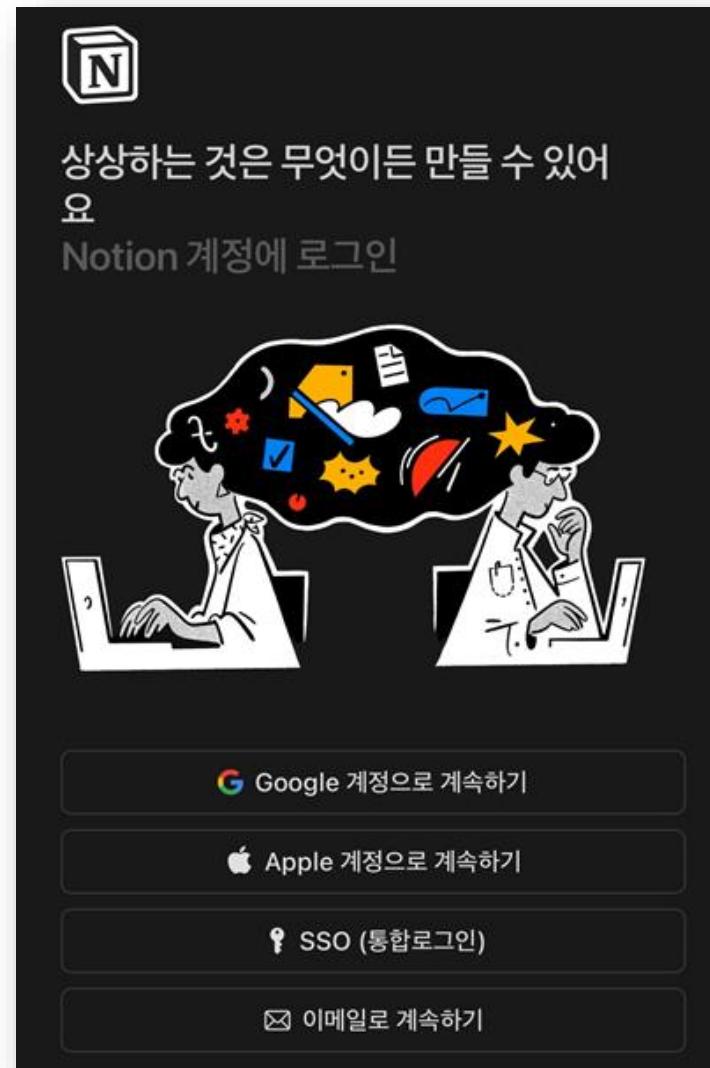
## ■ 인터넷뱅킹 시 추가인증 화면(예)



# 소셜 인증(Social login/sign-in/sign-on)

## ■ 특징

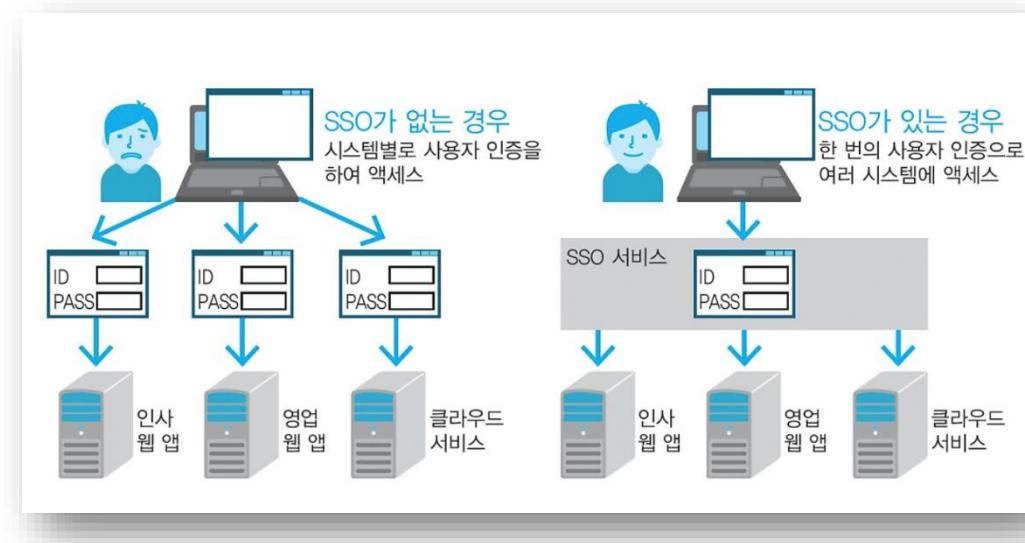
- 서비스 이용을 위해
- 해당 사이트가 제공하는 회원가입 절차 없이
- 주요 소셜 사이트에 인증 절차를 수행하고
  - ✓ 구글, 페이스북(메타), 애플, MS 등
- 서비스를 이용하는 방식



# SSO(Single Sign-On)

## ■ 특징

- 한 사이트에 로그인 후
- 다른 사이트를 접속할 때
- 추가적인 로그인 없이
- 로그인 상태를 유지하는 서비스



## ■ 조건

- SSO 서비스 이용 사이트와 SSO 서비스 제공 사이트 간  
사전 협약체결 및 SW 설치 필요

# Q & A

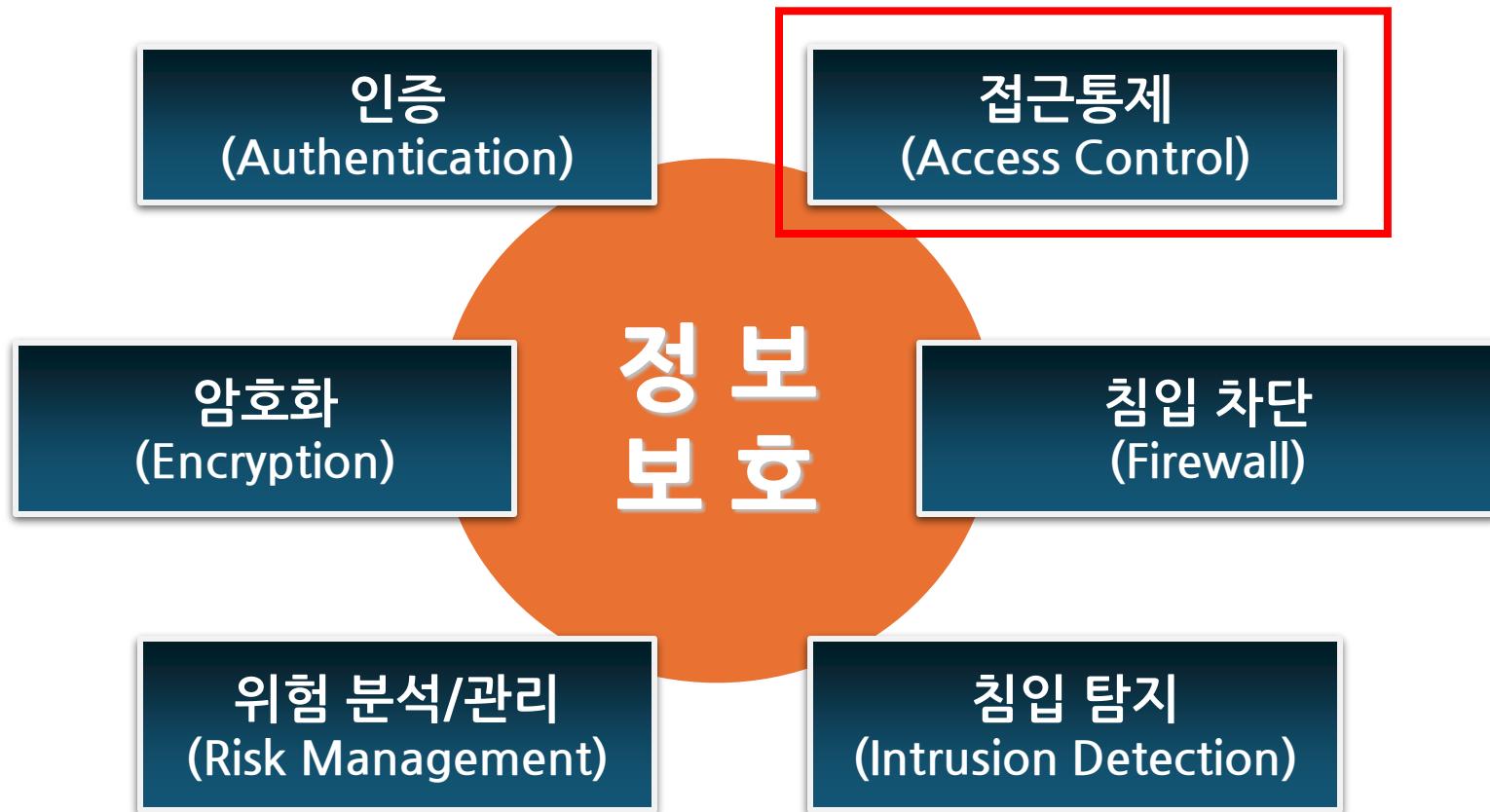
# 접근통제(Access Control) [ 인가(Authorization) ]

---

2024. 9

컴퓨터·소프트웨어공학과  
이 형 효  
(hlee@wku.ac.kr)

# 정보보안 기술 - 접근통제



# 정의

- 컴퓨팅 자원의 사용을 위해
- 인증과정을 성공적으로 마친 사용자를 대상으로
  - Identification, authentication
- 컴퓨터 시스템 자원에 대한
  - 메모리, 파일, I/O(프린터, 네트워크, ...)
- 접근허용 여부를
  - 읽기, 쓰기, 추가, 삭제, 생성, ...
- 결정하는 절차
  - 허가(grant), 거부(deny)

# 주요 용어(terminology)

## ■ 주체(subject)

- 능동적 개체(active entity)
- 예: 프로세스(=실행중인 프로그램)

## ■ 객체(object)

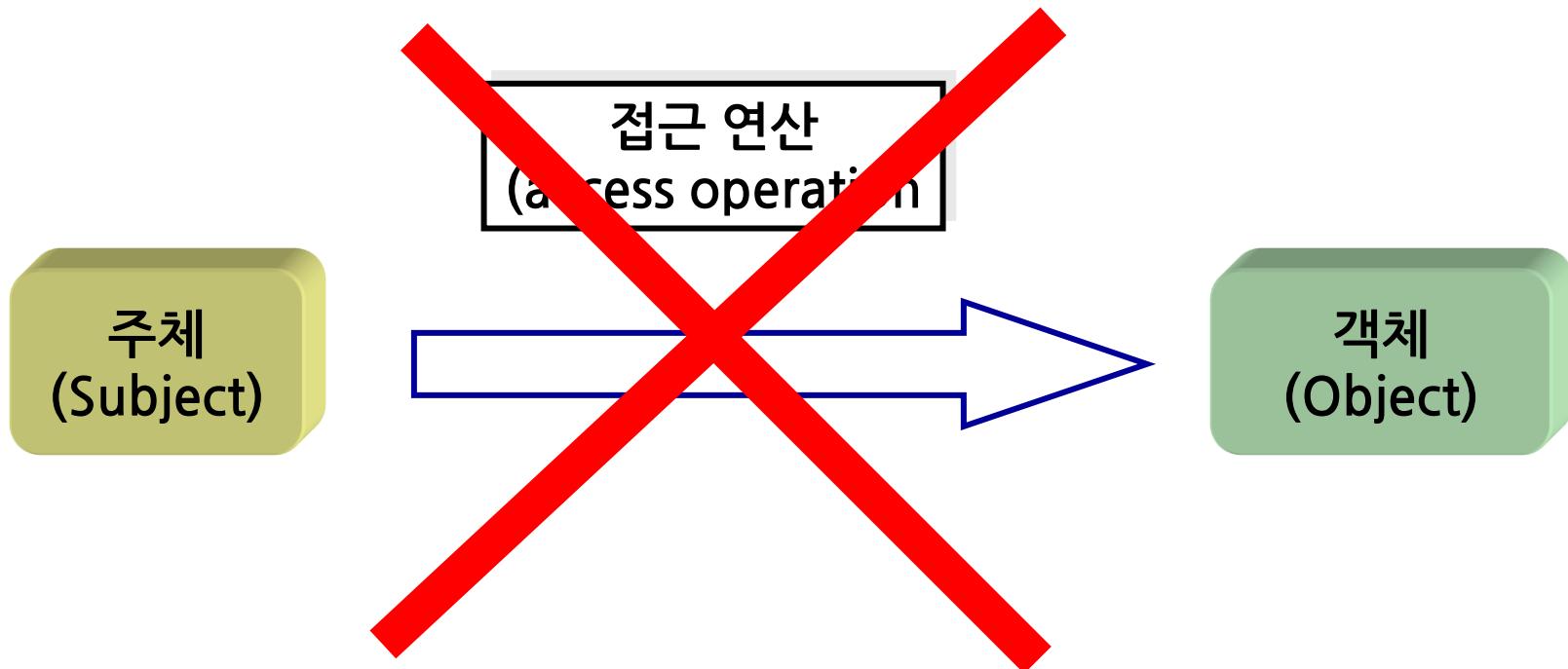
- 수동적 개체(passive entity)
- 예: 컴퓨터 자원(메모리, 파일, 프린터, 네트워크 장치, …)

## ■ 접근연산(access operation)

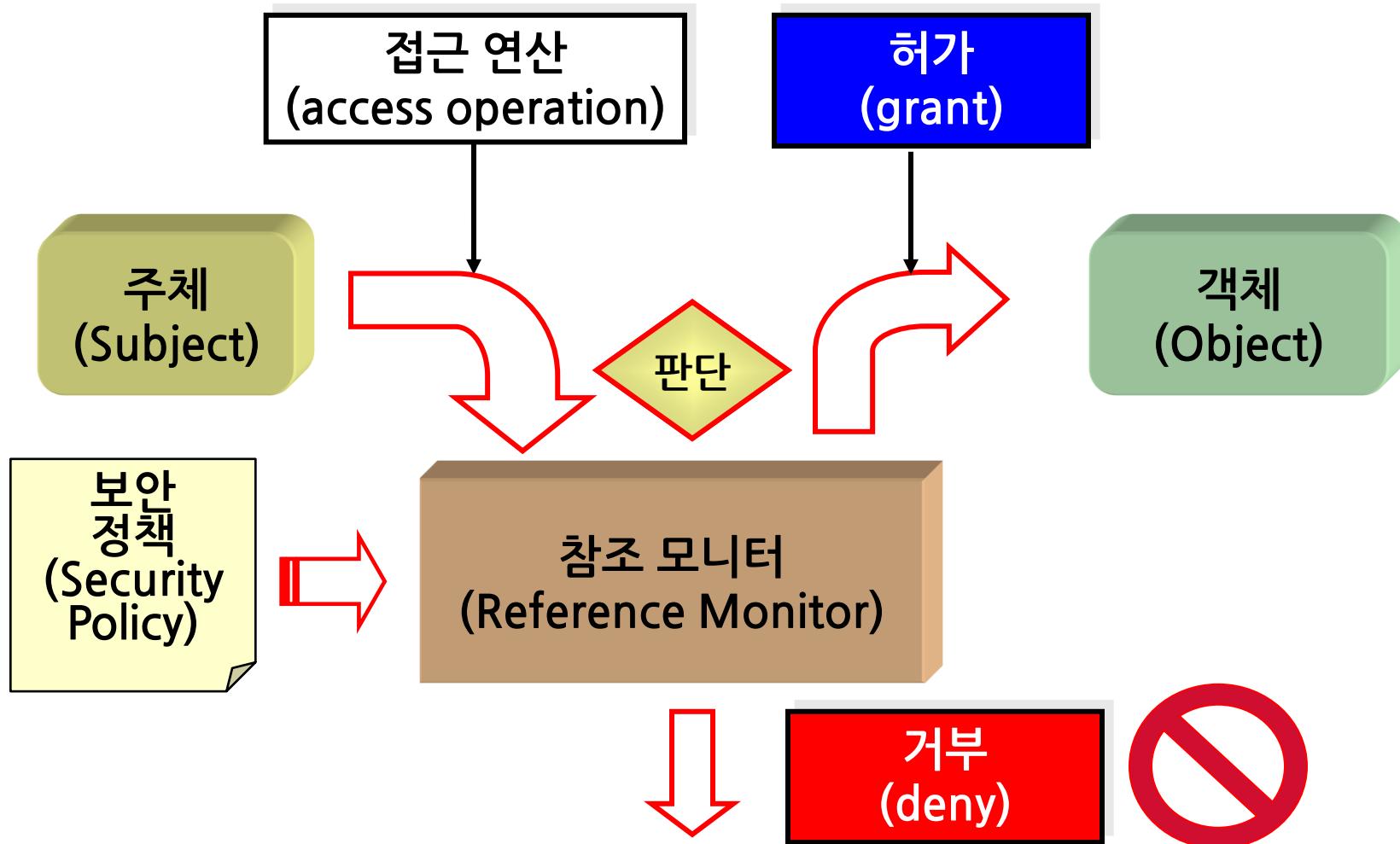
- 읽기, 쓰기, 추가, 생성, 삭제, …

## ■ 참조 모니터(reference monitor)

# 동작 구조(1)



# 동작 구조(2)



# 구성요소: 주체(Subject)

- 컴퓨터 시스템의 자원을 사용하는 개체

- 종류

- 사용자
- **프로세스**  
(= 실행 중인 프로그램)
- 컴퓨터 시스템에서는  
프로세스가 사용자를  
대신하여  
기능을 수행

이름	상태	27% CPU	54% 메모리	0% 디스크	0% 네트워크
앱 (9)					
> 4K Video Downloader	효...	0%	9.8MB	0MB/s	0Mbps
> KakaoTalk(32비트)	효...	0%	49.6MB	0MB/s	0Mbps
> Microsoft PowerPoint(32비트)(2)	효...	11.7%	530.1MB	0MB/s	0.1Mbps
> Outlook (new)(8)	효...	0%	335.1MB	0.1MB/s	0Mbps
> Waterfox(15)	효...	1.4%	803.9MB	0.1MB/s	0Mbps
> Windows 탐색기(2)	설정	0%	173.9MB	0MB/s	0Mbps
> 작업 관리자	작업 관리자	0%	0MB	0MB/s	0Mbps
> 터미널(4)	터미널	4.2%	79.0MB	0MB/s	0Mbps
백그라운드 프로세스 (151)					
> Acrobat Update Service(32비트)		0%	0.1MB	0MB/s	0Mbps
> adb.exe(32비트)		0%	0.8MB	0MB/s	0Mbps
> AhnLab Safe Transaction Application		0%	0.7MB	0MB/s	0Mbps

# 구성요소: 객체(Object)

- 컴퓨터 시스템의 자원
  - 특정한 기능(서비스)을 수행하는 개체
- 종류
  - 파일
  - 폴더
  - 프린터
  - 네트워크 자원 (폴더, 파일, 프린터, …)

# 구성요소: 접근연산(Access Operation)

- 주체가 요청한 객체에 대한 행위(**연산**)
- 예
  - 읽기(read)
  - 쓰기(write)
  - 추가 append)
  - 실행(execute)
- 접근연산에 대한 정의가 상황에 따라 다름

# 구성요소: 참조 모니터(1)

## ■ 기능

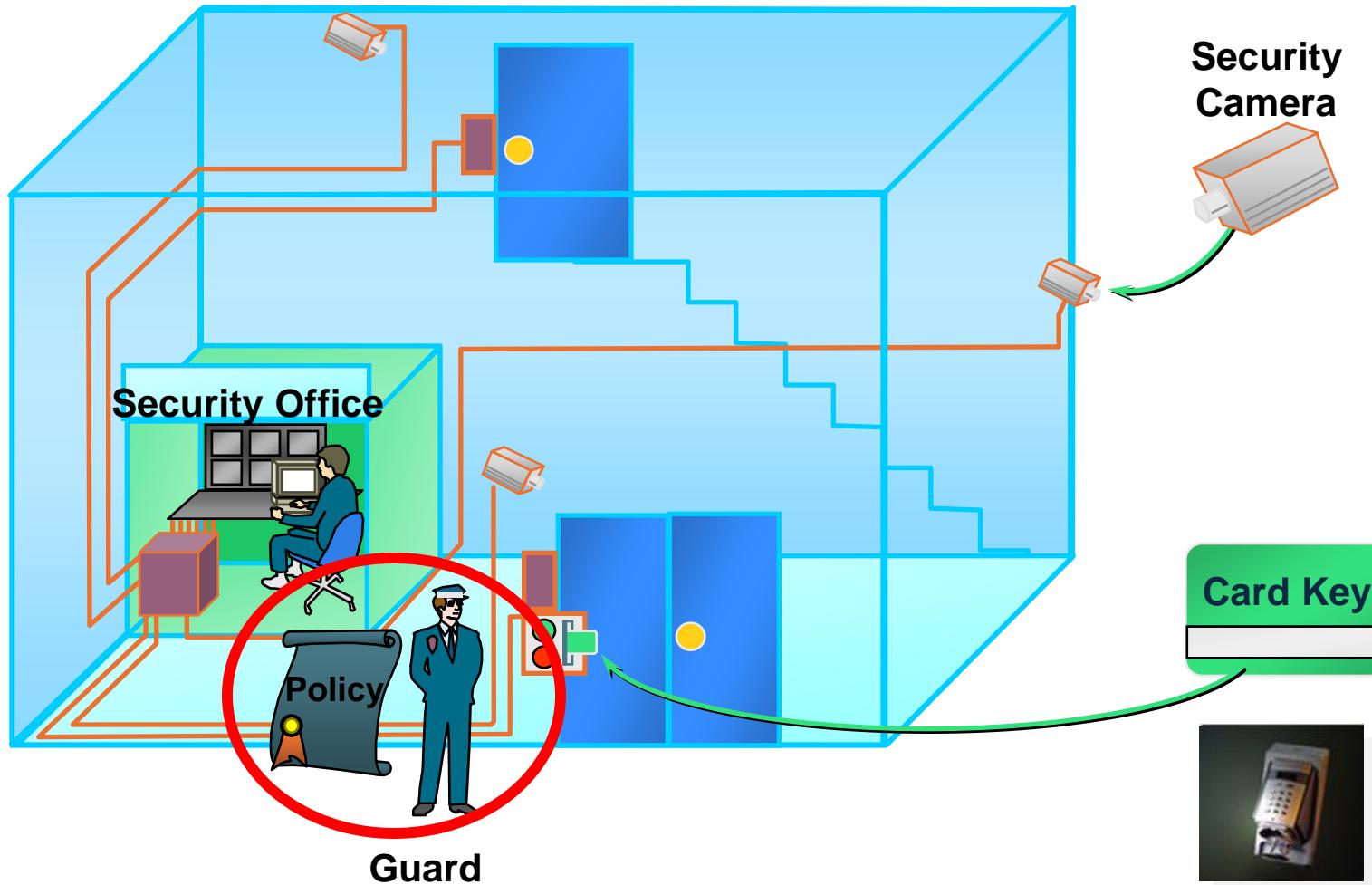
- 객체(Object)에 대한 주체(Subject)의 접근요청을 허가(grant)하거나 거부(deny)하는 판단
  - 주체가 객체에 대한 주어진 접근을 요청할 권한이 부여되어 있는지(**authorized**) 확인
  - 주체와 객체간의 모든 접근요청 처리

## ■ 조건

- Tamperproof      변조, 위조
  - Always invoked
  - Small enough to be suitable to analysis and testing  
                        적합한      분석



# 구성요소: 참조 모니터(2)



# 접근통제를 위한 자료구조(1)

## ■ 정의

- 주체(Subject)가 접근(access operation)할 수 있는 객체(Object)에 대한 정보를 표현하는 구조

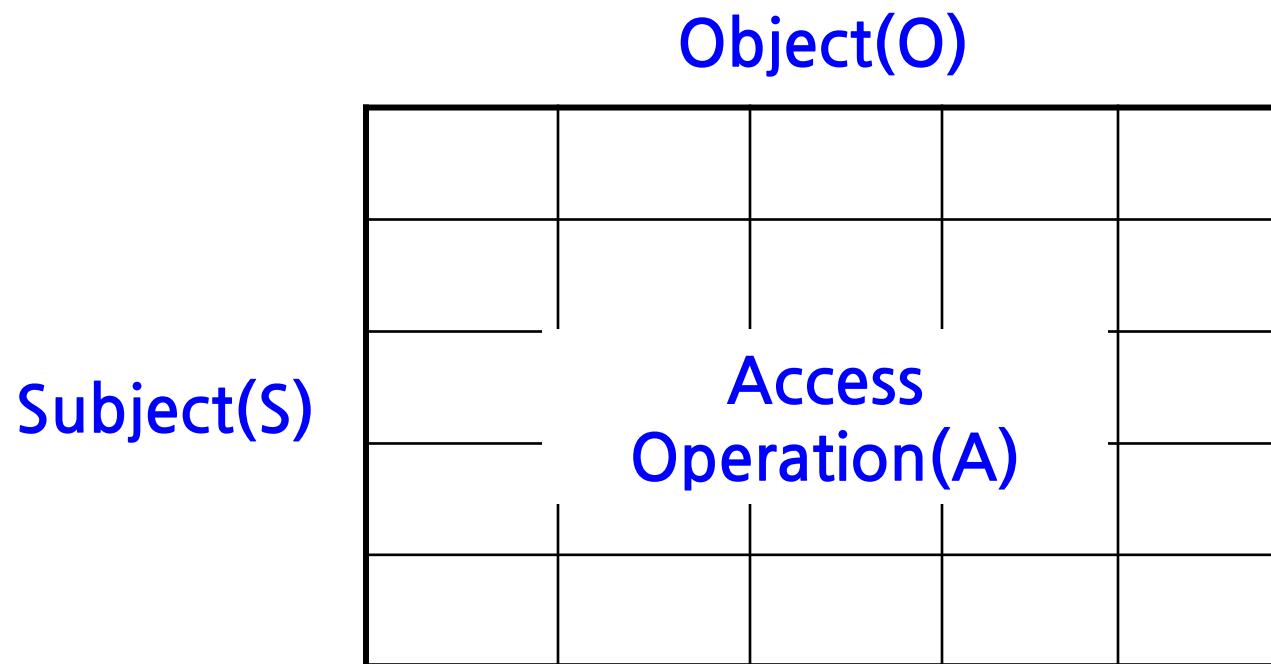
## ■ 표기법(Notation)

- **S**: 주체들의 집합(a set of subjects)
- **O**: 객체들의 집합(a set of objects)
- **A**: 접근연산의 집합(a set of access operations)

# 접근통제를 위한 자료구조(2)

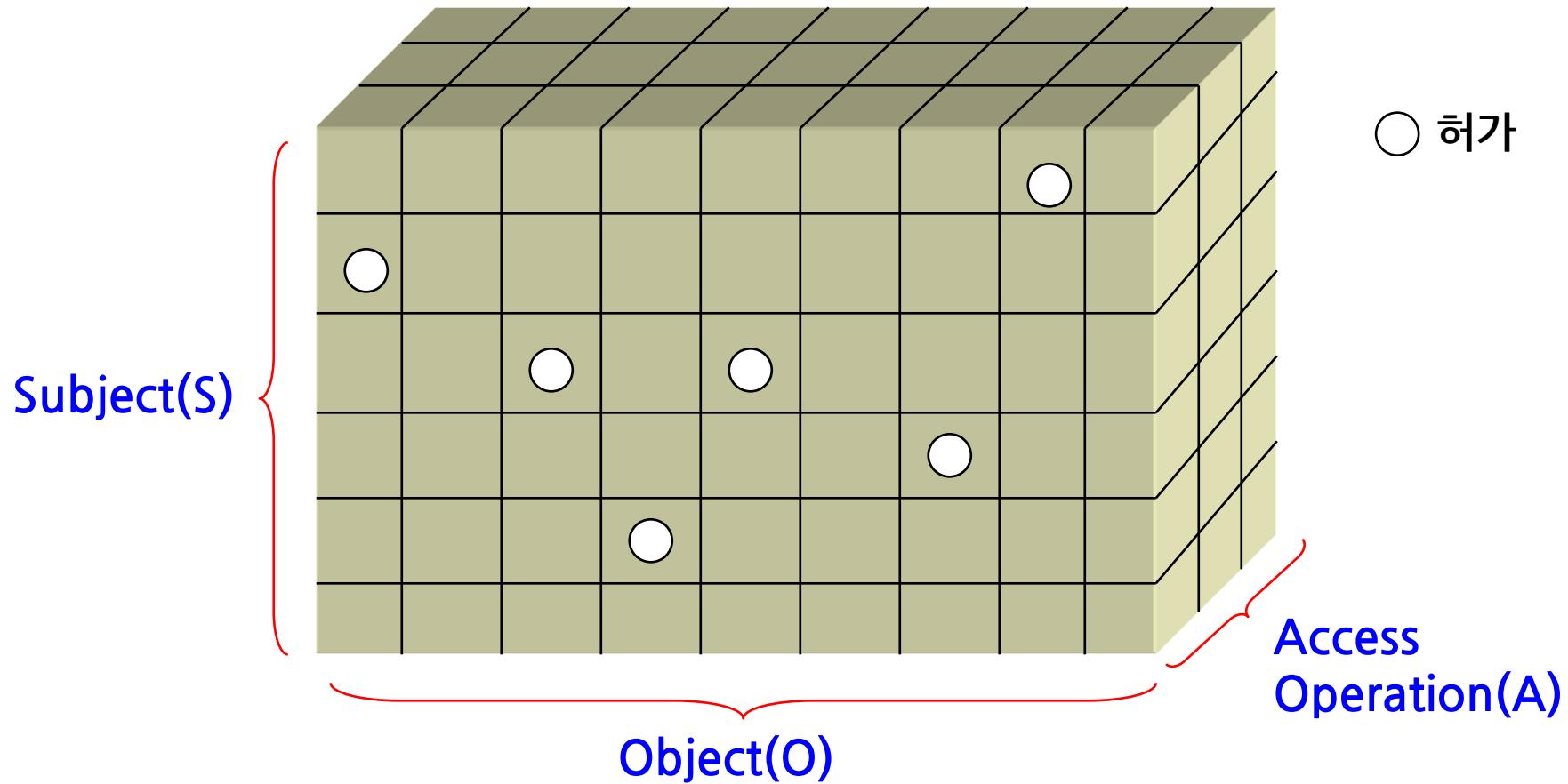
- **접근통제행렬**(Access Control Matrix)

- 개념적 자료 구조



# 접근통제를 위한 자료구조(3)

## ■ 접근통제행렬(Access Control Matrix)



# 접근통제를 위한 자료구조(4)

## ■ 접근통제행렬(Access Control Matrix)(예)

- Subjects: Alice, Bob
- Objects: bill.doc, edit.exe, fun.com, readme.txt
- Access Operations: execute(x), read(r), write(w)

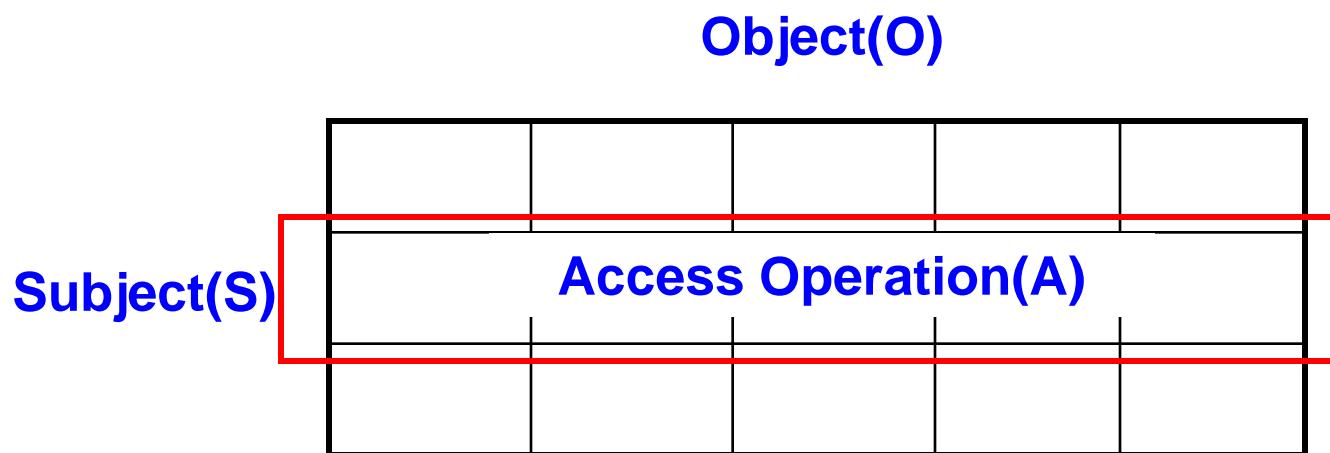
	bill.doc	edit.exe	fun.com	readme.txt
Alice	-	{r, x}	{r, x}	{r}
Bob	{r, w}	{r, x}	{r, w, x}	{r, w}

# 접근통제를 위한 자료구조(5)

- **접근통제행렬(Access Control Matrix) 문제점**
  - 주체와 객체의 수가 많은 경우 행렬의 크기가 커짐
    - ✓ 큰 크기의 기억장소 필요( $|S| \times |O| \times |A|$ )
    - ✓ 행렬의 일부분에만 접근연산 정보 포함
    - ✓ 기억장소의 낭비 초래
- **접근통제행렬은 주체와 객체, 주체에게 허용된 접근연산을 나타내는 개념적인 자료로 활용**

# 접근통제를 위한 자료구조(6)

- **Capability**
  - 주체 중심
  - 주체가 접근할 수 있는 객체에 대한 정보 저장
- 접근통제행렬과의 관계



# 접근통제를 위한 자료구조(7)

## ■ 예

- Alice's capability
  - ✓ edit.exe: read, execute; fun.com: read, execute; readme.txt: read
- Bob's capability
  - ✓ bill.doc: read, write; edit.exe: read, execute; fun.com: read, write, execute; readme.txt: read, write

	<b>bill.doc</b>	<b>edit.exe</b>	<b>fun.com</b>	<b>readme.txt</b>
<b>Alice</b>	-	{r, x}	{r, x}	{r}
<b>Bob</b>	{r, w}	{r, x}	{r, w, x}	{r, w}

# 접근통제를 위한 자료구조(8)

## ■ Capability 문제점

- 주어진 객체에 접근할 수 있는 주체들은?
- 주체별 capability의 안전한 보호 문제

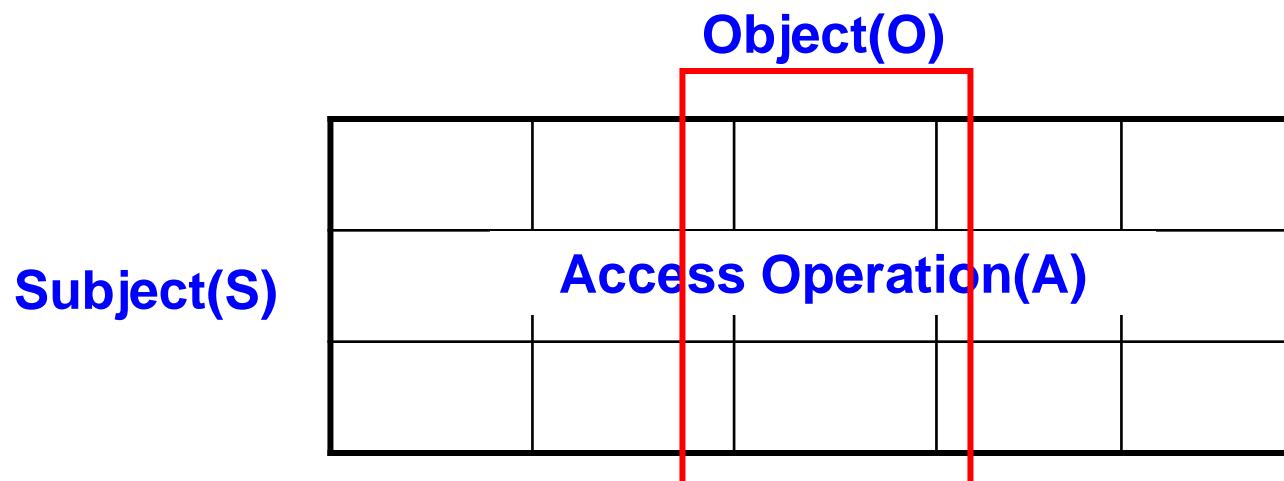
# 접근통제를 위한 자료구조(9)

- Access Control Lists

- 객체 중심

- 객체에 접근이 허용된 주체와 접근연산 정보 저장

- 접근통제행렬과의 관계



# 접근통제를 위한 자료구조(10)

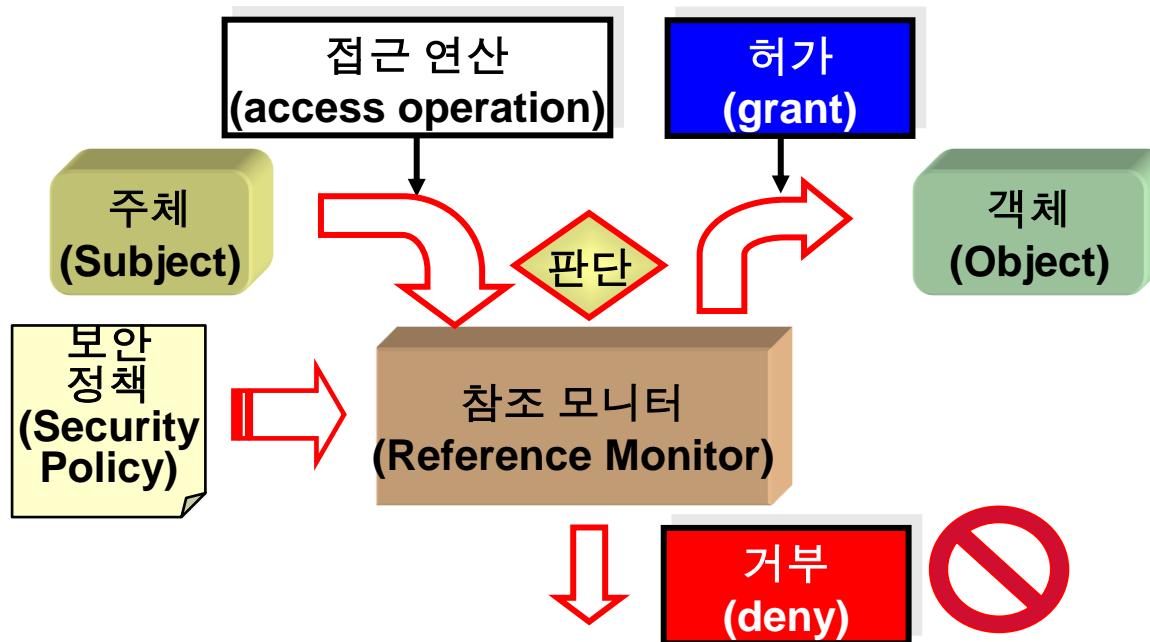
## ■ 예

- ACL for bill.doc
  - ✓ Bob: read, write
- ACL for edit.exe
  - ✓ Alice: read, execute; Bob: read, execute
- ACL for fun.com
  - ✓ Alice: read, execute; Bob: read, write, execute
- ACL for readme.txt
  - ✓ Alice: read; Bob: read, write

	bill.doc	edit.exe	fun.com	readme.txt
Alice	-	{r, x}	{r, x}	{r}
Bob	{r, w}	{r, x}	{r, w, x}	{r, w}

# 접근통제 정책(Access Control Policy)

- 주체(Subject)가 요청한 객체(Object)에 대한 접근 연산(access operation)의 허가 여부(grant/deny)를 결정하는 규칙들의 집합



# 접근통제 정책(1)

## ■ 대표적 정책

- **임의적 접근통제(DAC: Discretionary Access Control)**
  - ✓ 자율적 접근통제
- **강제적 접근통제(MAC: Mandatory Access Control)**
  - ✓ 다중 등급 보안(MLS: Multi-Level Security) 접근통제
- **역할기반 접근통제(RBAC: Role-Based Access Control)**

# 접근통제 정책(2)

## ■ DAC (임의적 접근통제)

- 객체의 소유자가 객체에 접근할 수 있는 주체와 접근연산 종류를 **임의적**으로 결정
- **Ownership(소유권)** 기반
- 권한배정과 회수가 자유로운 환경에 적용

## ■ MAC (강제적 접근통제)

- 모든 주체와 객체에 적용되는 공통의 **규칙** 존재
- 엄격한 정보흐름 통제가 필요한 환경에 적용

# 접근통제 정책(3)

## ■ RBAC (역할기반 접근통제)

- 권한의 배정이 사용자가 아닌 역할(role)에 배정
  - ✓ presidency: 대통령직
  - ✓ president: 대통령
- 사용자는 역할을 배정받음으로써 권한 수행 가능
- 역할을 기반으로 권한배정과 관리가 이루어지는 기업환경에 적용

# 임의적 접근통제(DAC) 정책

- **DAC: Discretionary Access Control**
  - Discretionary: 임의의, 자유 재량의, 임의로 결정할 수 있는
- 객체(Object)에 대한 접근 여부가 객체 소유자의 임의적 결정에 따른
  - Ownership-based
- 학교, 연구기관 및 기업환경에 적용
  - UNIX, Linux, Windows 운영체제

# 강제적 접근통제(MAC) 정책

- MAC: Mandatory Access Control
- 객체(Object)에 대한 접근 여부는 엄격히 정의된 규칙(Rule)에 의해 결정
  - 객체의 소유자가 객체에 대한 접근을 임의적으로 허용하거나 거부할 수 없음
- 국방, 정보기관 등과 같이 정보에 대한 접근이 엄격히 통제되는 환경에 적합

# BLP 모델(1)

- Bell-LaPadula Disclosure Model(1973)
- MLS(Multi-Level Security) 모델
  - 다중 등급 보안
- 특 징
  - 주체와 객체에 **보안레이블(Security Label)** 부여
  - 주체와 객체의 보안레이블을 비교하여 접근 허용 여부 판단
  - **정보의 일방향 흐름**(uni-directional information flow) 지원을 통한 **비밀성**(confidentiality) 지원

# BLP 모델(2)

## ■ 구성 요소

- 주체들의 집합: S
- 객체들의 집합: O
- 접근 연산 집합 A = {read, write, append, execute}
- 보안 레이블(Security Label)
  - ✓ 구성: 보안 등급(Security Level)과 범주(Category)의 집합
  - ✓ 보안 등급:
    - Top Secret(TS), Secret(S), Confidential(C), Unclassified(U)
  - ✓ TS > S > C > U

# BLP 모델(3)

- 주체와 객체에 보안 레이블 부여
  - 주체: 인가등급, 객체: 비밀등급
  - 보안관리자에 의해 결정
  - 보안 레이블의 임의적 변경은 불가능
- 레이블간 지배 관계(Dominance Relationship)
  - Label1 = (Level1, Category1)
  - Label2 = (Level2, Category2)
  - Label1 **dominates** Label2 iff
    - Level1  $\geq$  Level2  $\wedge$
    - Category1  $\supseteq$  Category2

# BLP 모델(4)

## ■ 예

- $L1 = (TS, \{\text{영업부}, \text{관리부}, \text{회계부}\})$
- $L2 = (S, \{\text{영업부}, \text{관리부}\})$
- $L3 = (U, \{\text{인사부}\})$
- *Question 1: L1 dominates L2? (true or false)*
- *Question 2: L1 dominates L3? (true or false)*
- *Question 3: L2 dominates L3? (true or false)*

# BLP 모델(5)

## ■ 두 가지 보안 특성

- Simple Security Property(*ss-property*)

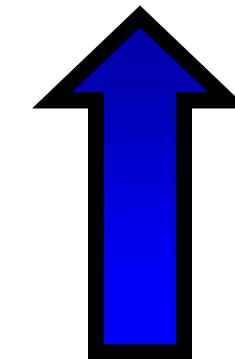
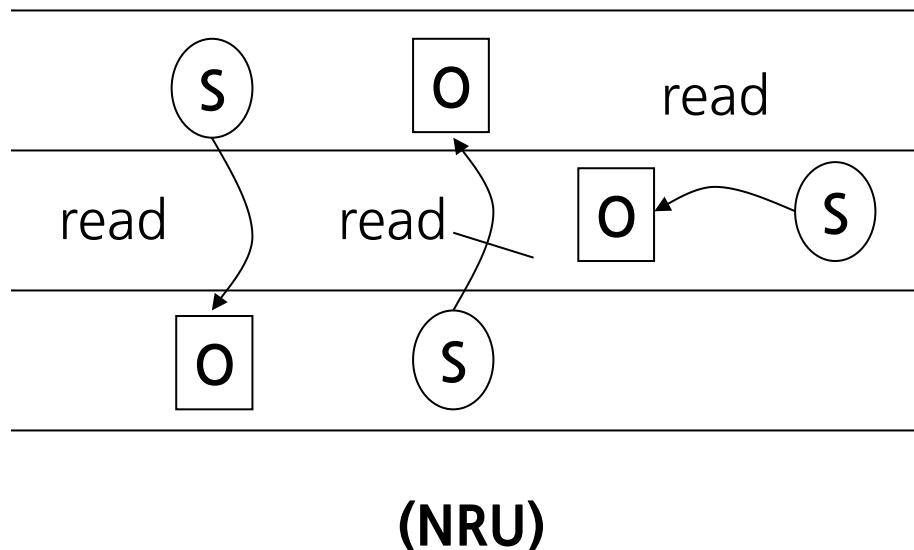
- ✓ *No Read-Up(NRU) security policy*
  - ✓ 주체의 보안레이블이 객체의 보안레이블을 지배하는 경우  
주체가 객체의 정보를 읽을 수 있음(read)

- Star Property(*\*-property*)

- ✓ *No Write-Down(NWD) security policy*
  - ✓ 객체의 보안레이블이 주체의 보안레이블을 지배하는 경우  
주체가 객체에 정보를 쓰거나 추가할 수 있음  
(write, append)

# BLP 모델(6)

- Simple Security Property (*ss-property*)
  - *No Read-Up(NRU) security policy*
  - 주체의 보안레이블이 객체의 보안레이블을 지배하는 경우  
주체가 객체의 정보를 읽을 수 있음(read)

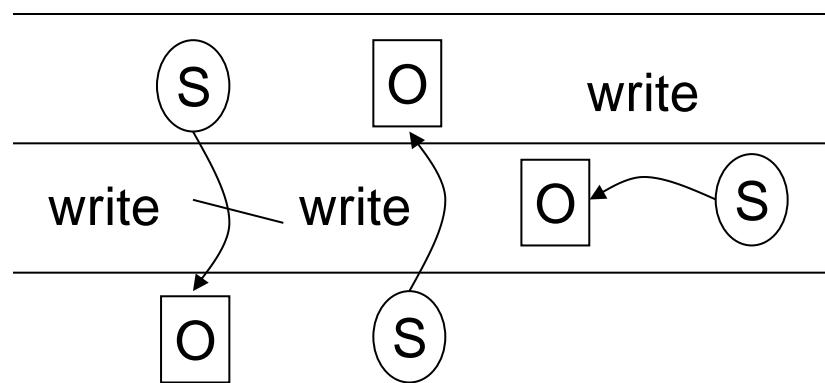


정보의 흐름 방향

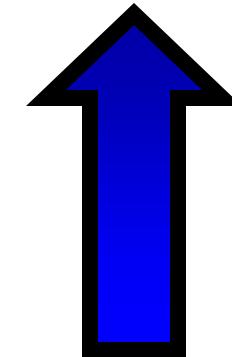
# BLP 모델(7)

- Star Property (*\*-property*)

- *No Write-Down(NWD) security policy*
- 객체의 보안레이블이 주체의 보안레이블을 지배하는 경우  
주체가 객체에 정보를 쓰거나 추가할 수 있음  
(write, append)



(NWD)

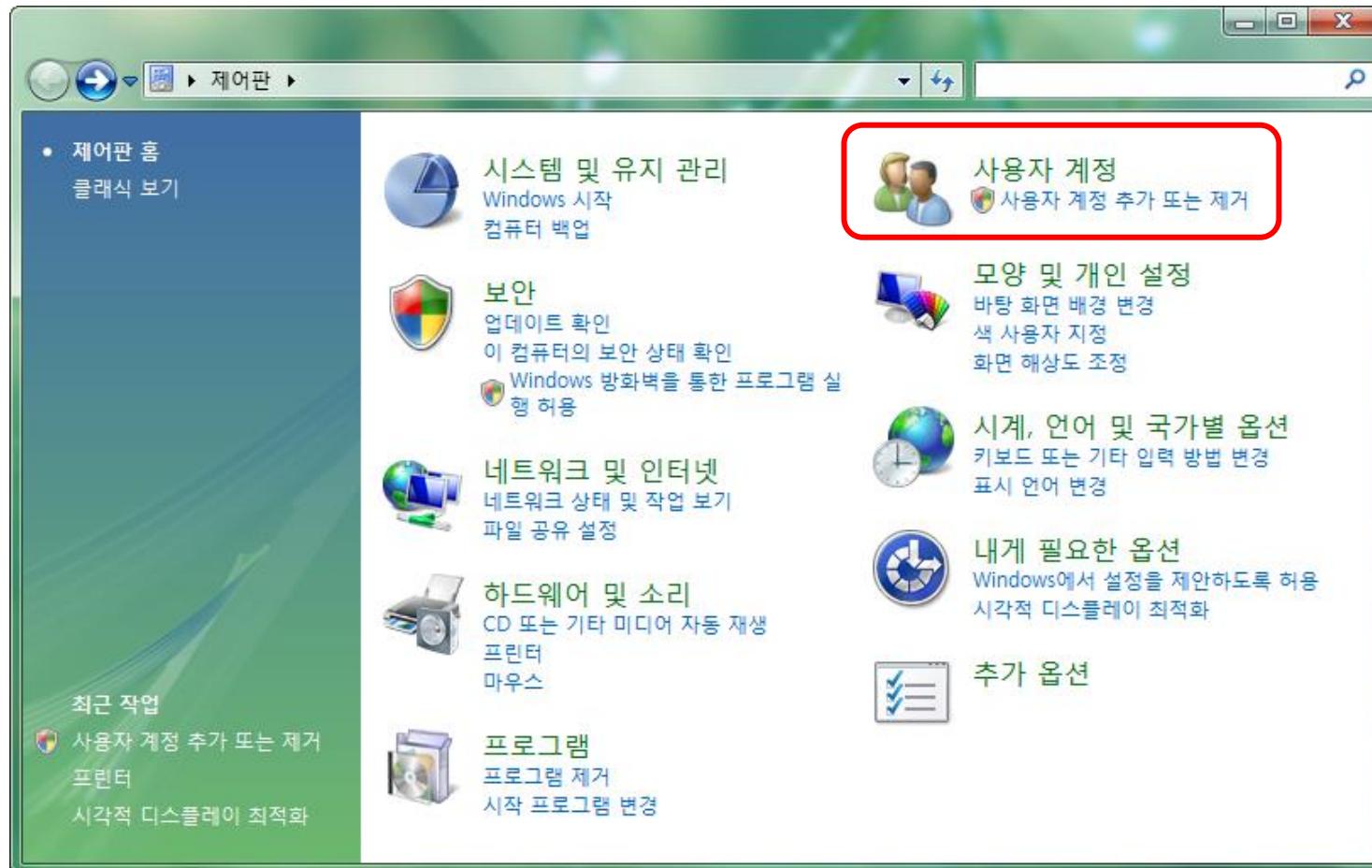


정보의 흐름 방향

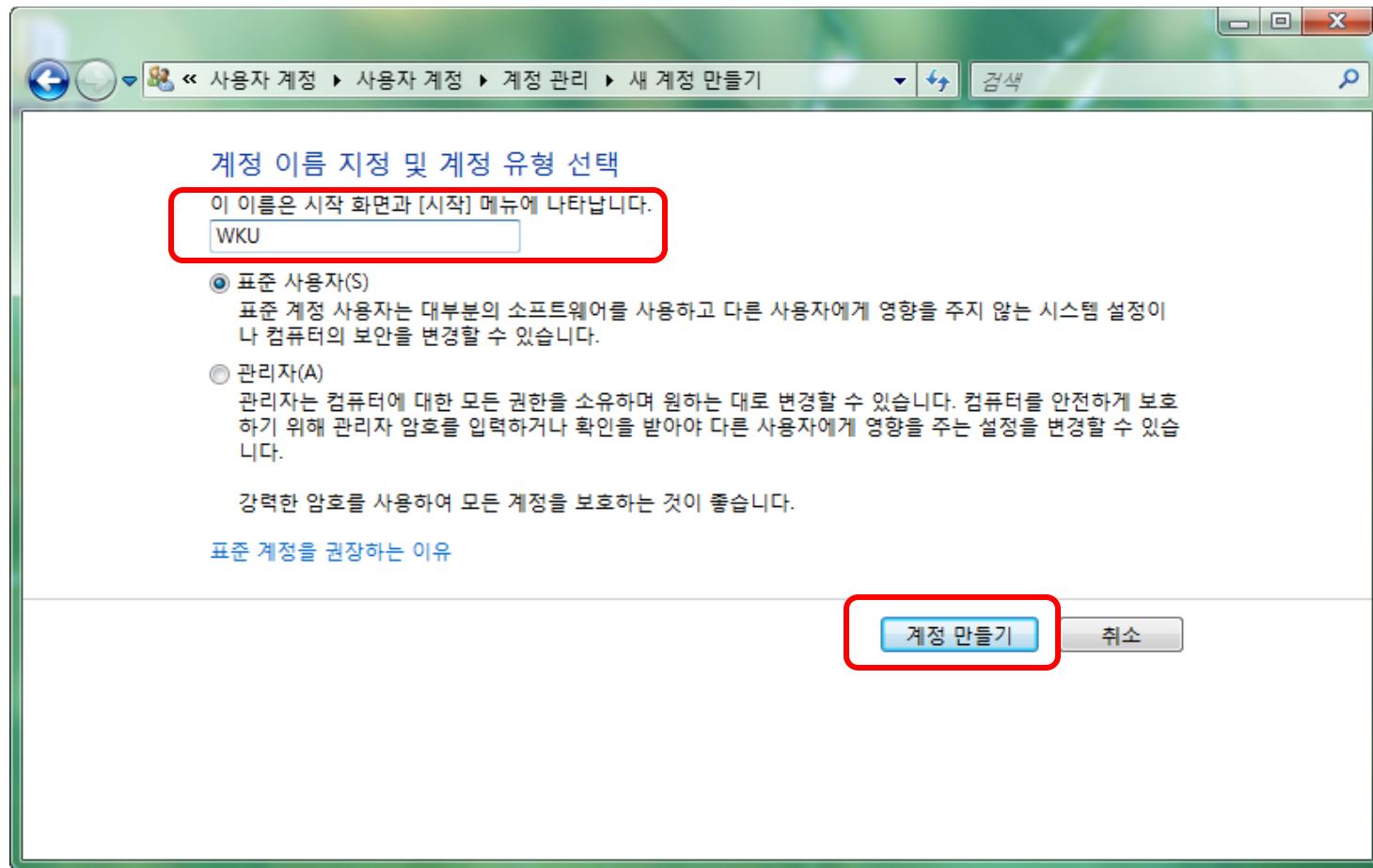
# BLP 모델(8)

- 강제적 접근통제(MAC) 정책 지원
- MLS(Multi-Level Security) 정책 지원
- 주체, 객체에 보안 레이블 부여
  - 보안 레이블 = (보안 등급, 범주의 집합)
- 정보의 일방향 흐름 통제를 통한 비밀성 유지
  - *ss-property*(NRU)
  - *\*-property*(NWD)
- 국방분야 등 특정 분야에서만 활용

# Windows 접근통제 사례(1)

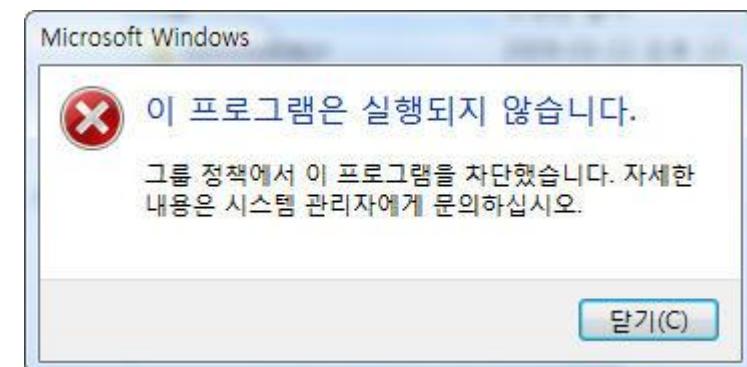
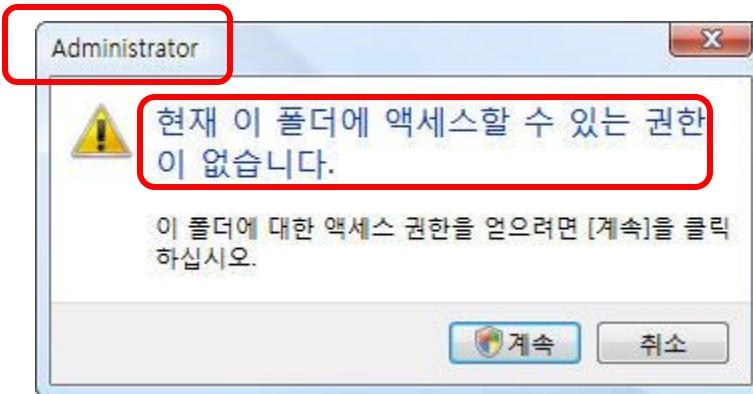


# Windows 접근통제 사례(2)



# Windows 접근통제 사례(3)

- 표준 사용자가 관리자 소유 폴더에 접근하는 경우
  - 관리자: Administrator



# Linux 접근통제 사례(1)

- 사용자: hlee
- 프로세스: more
- 객체: /etc/passwd, /etc/shadow
- 접근 연산: read
- % more /etc/shadow
- % more /etc/passwd

# Linux 접근통제 사례(2)

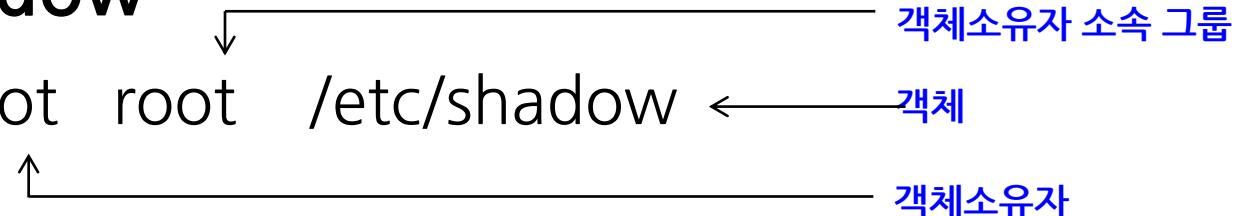
## ■ 객체: /etc/passwd

- rw-r--r-- root root /etc/passwd

rw- r-- r--			
객체소유자	객체소유자	기타 사용자	
권한	소속 그룹	권한	

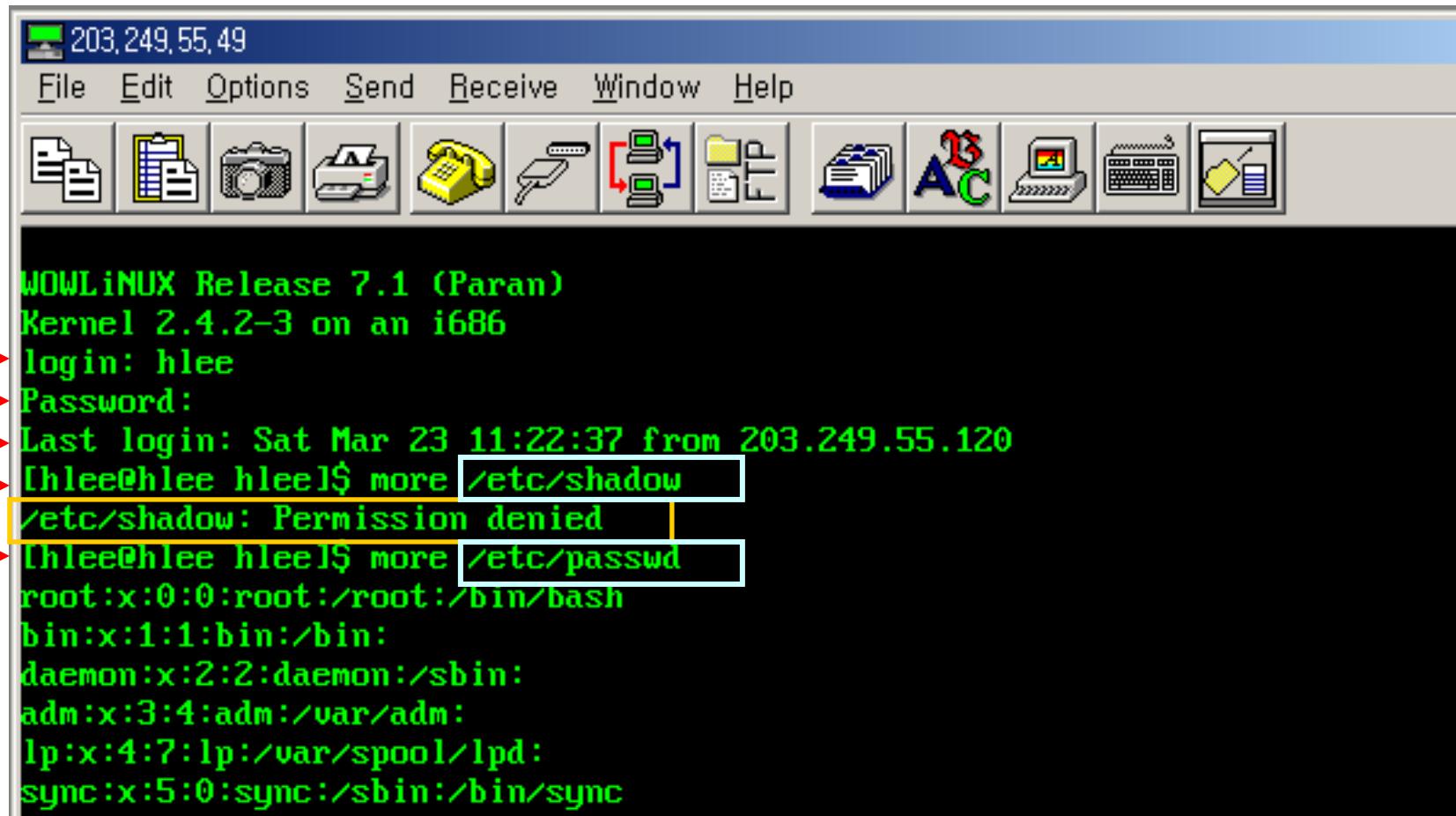
## ■ 객체: /etc/shadow

- rw----- root root /etc/shadow



```
[hlee@hlee hlee]$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1020 Mar 20 11:02 /etc/passwd
[hlee@hlee hlee]$ ls -l /etc/shadow
-rw----- 1 root root 774 Mar 20 11:02 /etc/shadow
[hlee@hlee hlee]$
```

# Linux 접근통제 사례(3)



The screenshot shows a terminal window titled "203,249,55,49" with a blue header bar containing the title and a menu bar with File, Edit, Options, Send, Receive, Window, Help. Below the menu is a toolbar with various icons. The main window displays a terminal session:

```
WOWLiNUX Release 7.1 (Paran)
Kernel 2.4.2-3 on an i686
login: hlee
Password:
Last login: Sat Mar 23 11:22:37 from 203.249.55.120
[hlee@hlee hlee]$ more /etc/shadow
/etc/shadow: Permission denied
[hlee@hlee hlee]$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
[hlee@hlee hlee]$
```

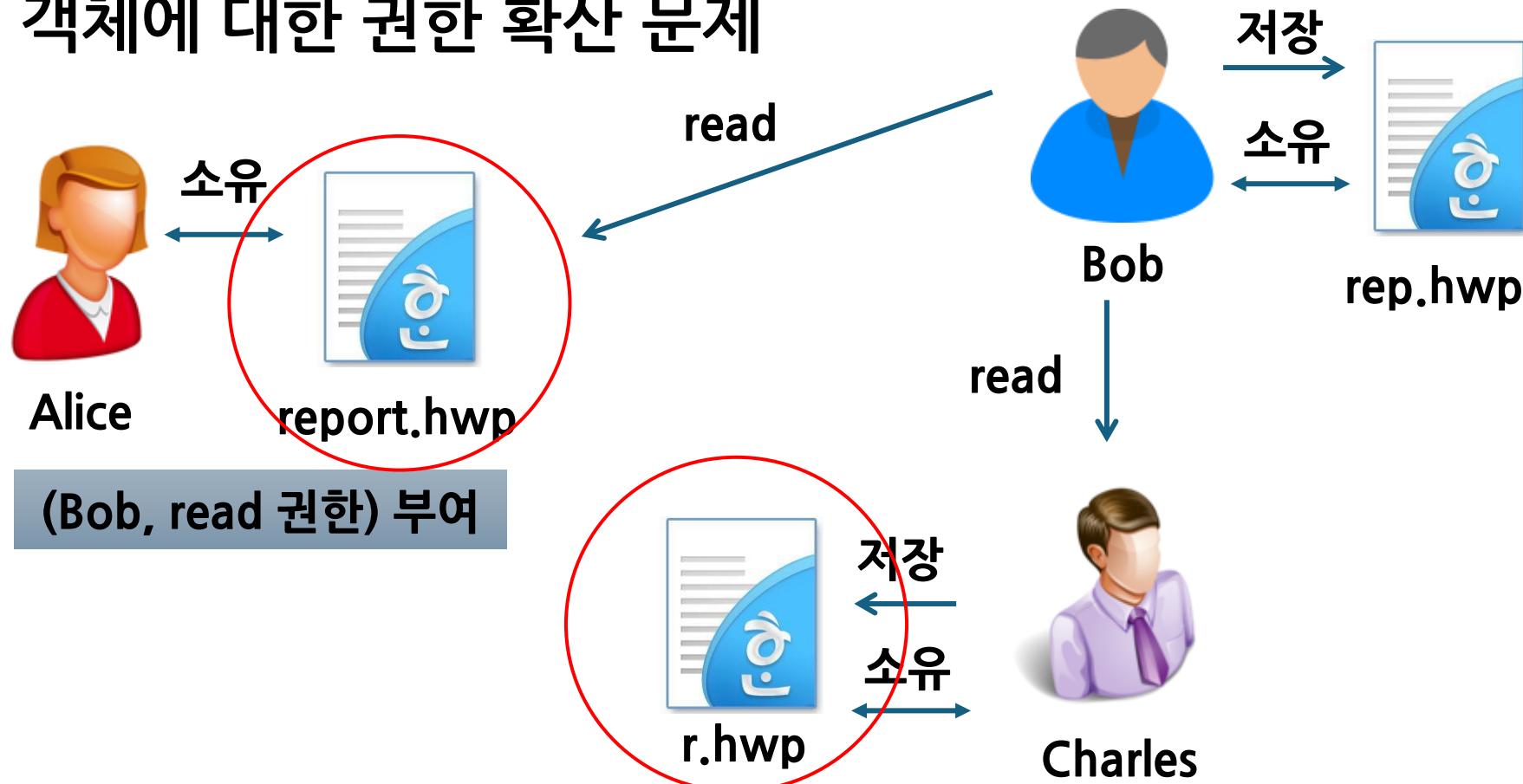
Red arrows point to several lines of the terminal output:

- An arrow points to the "login: hlee" line.
- An arrow points to the "Password:" line.
- An arrow points to the "Last login:" line.
- An arrow points to the command "more /etc/shadow". The output of this command, "etc/shadow: Permission denied", is highlighted with a yellow box.
- An arrow points to the command "more /etc/passwd". The output of this command, "root:x:0:0:root:/root:/bin/bash" and subsequent lines, is highlighted with a yellow box.

# RBAC(1) - DAC 문제점

- 객체 소유자가 의도하지 않는  
객체에 대한 권한 확산 문제

(Charles, read 권한) 부여



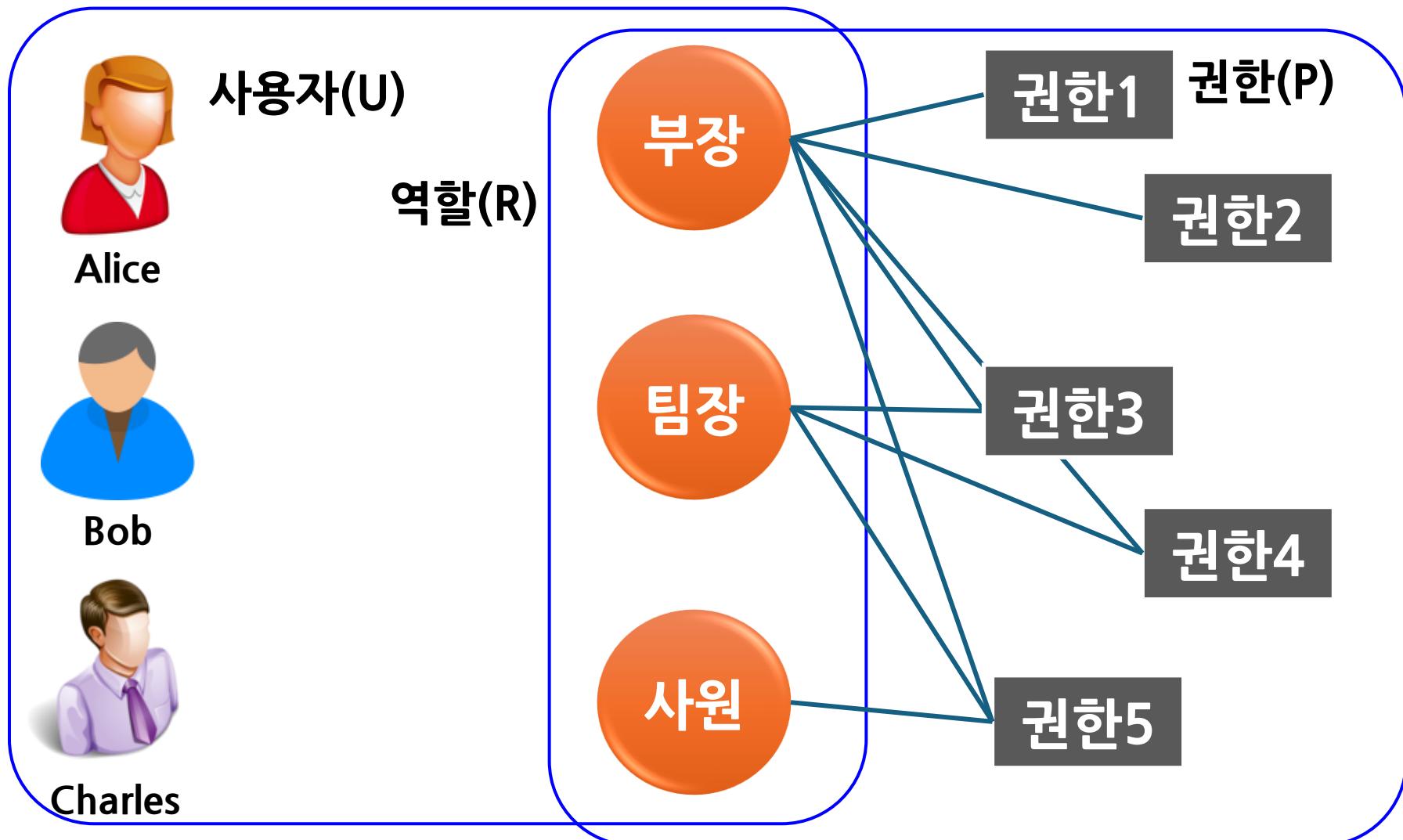
# RBAC(2)

- 역할(role)이 수행할 수 있는 권한은 미리 결정됨
  - 기업의 업무처리규정 등
  - 역할 - 권한 배정
- 사용자는 역할에 배정됨으로써 역할에 배정된 권한을 수행함
  - 기업의 인사발령 등
  - 사용자 - 역할 배정

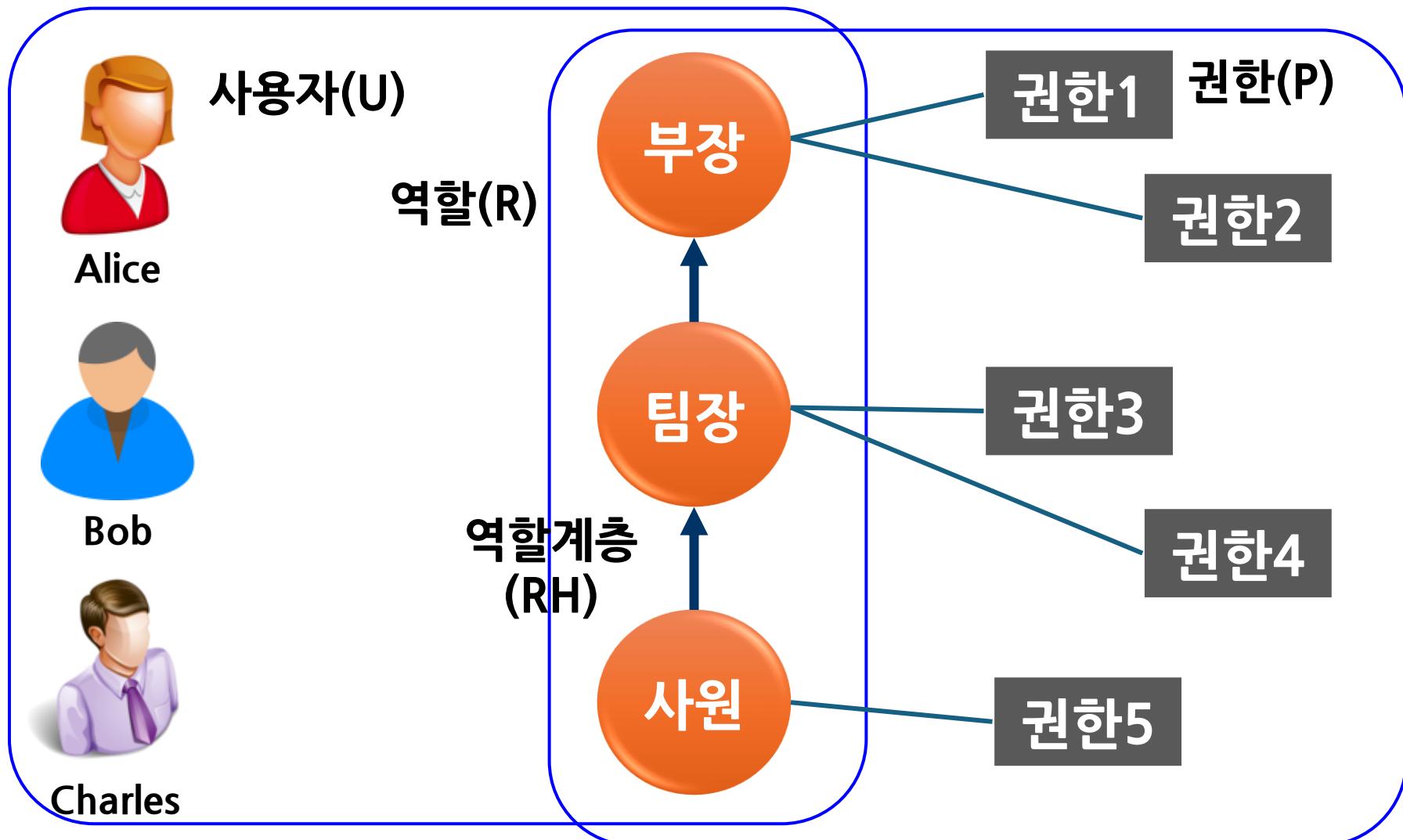
# RBAC(3)

- 역할 간 상속(inheritance)을 통해 권한관리 단순화
  - 하위 역할에 배정된 권한이 상위 역할로 상속
  - 역할 간 권한의 중복배정 번거로움 해결
  - 역할 계층 (role hierarchy)

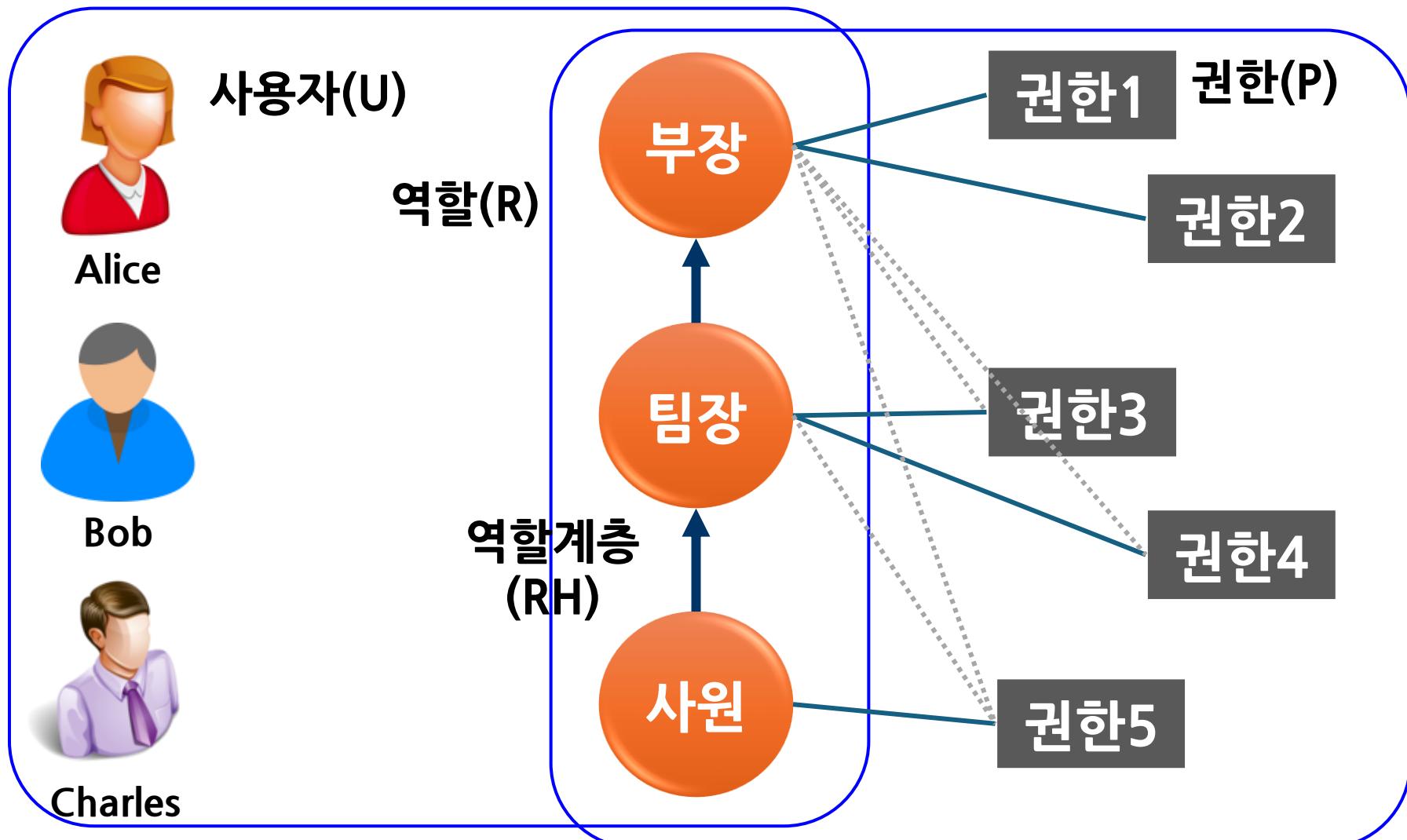
# RBAC(4) - 역할 ↔ 권한 배정



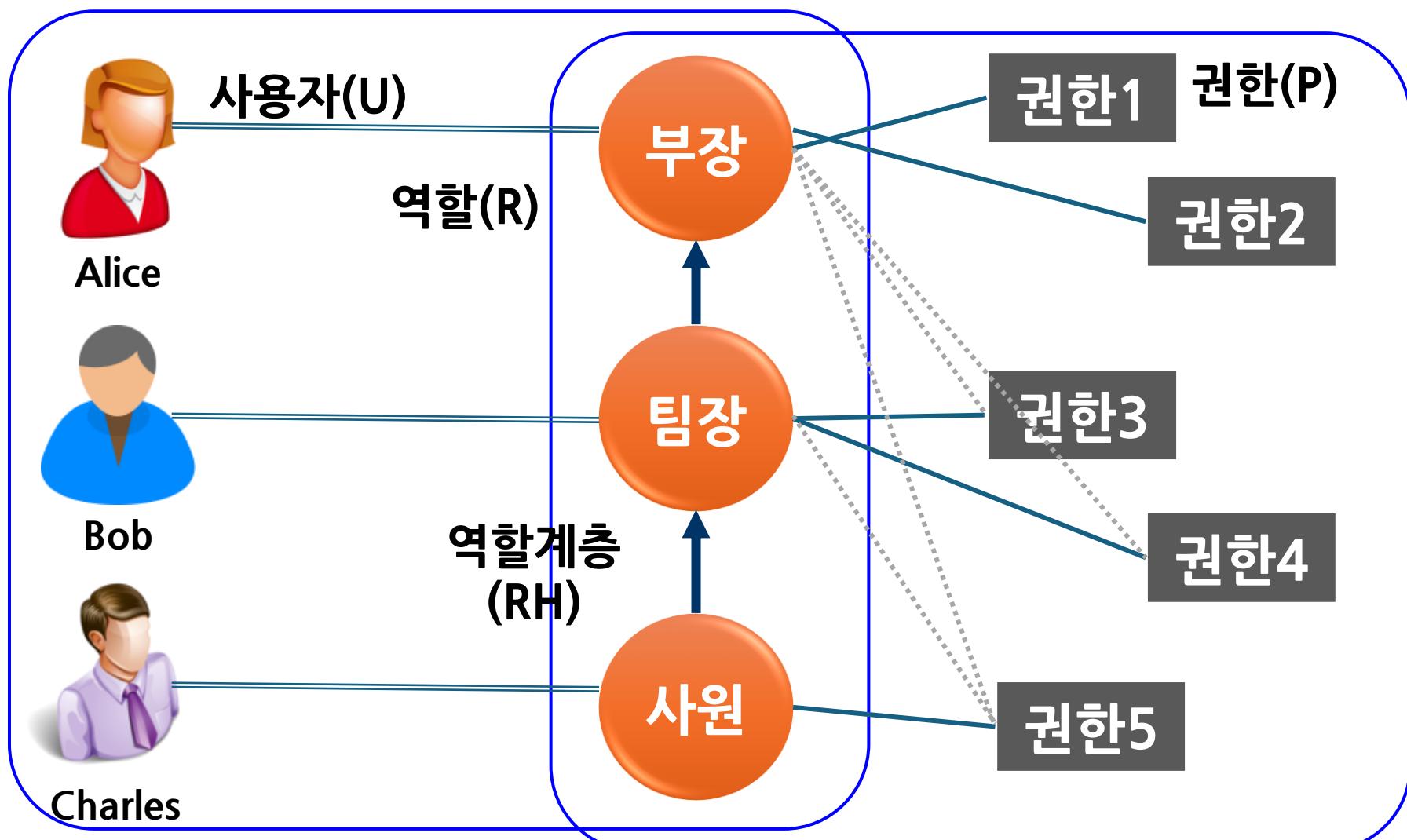
# RBAC(5) - 역할 계층 설정(1)



# RBAC(5) - 역할 계층 설정(2)



# RBAC(6) - 사용자 ↔ 역할 배정



# 접근통제 정리

- 접근통제 정의
  - 인증기능과의 관계
- 접근통제모델 구성요소
  - 구성요소 기능 및 예제
- 주요 접근통제모델
  - DAC, MAC, RBAC
  - 각 접근통제 모델별 특징, 적용 분야

# Q & A

---

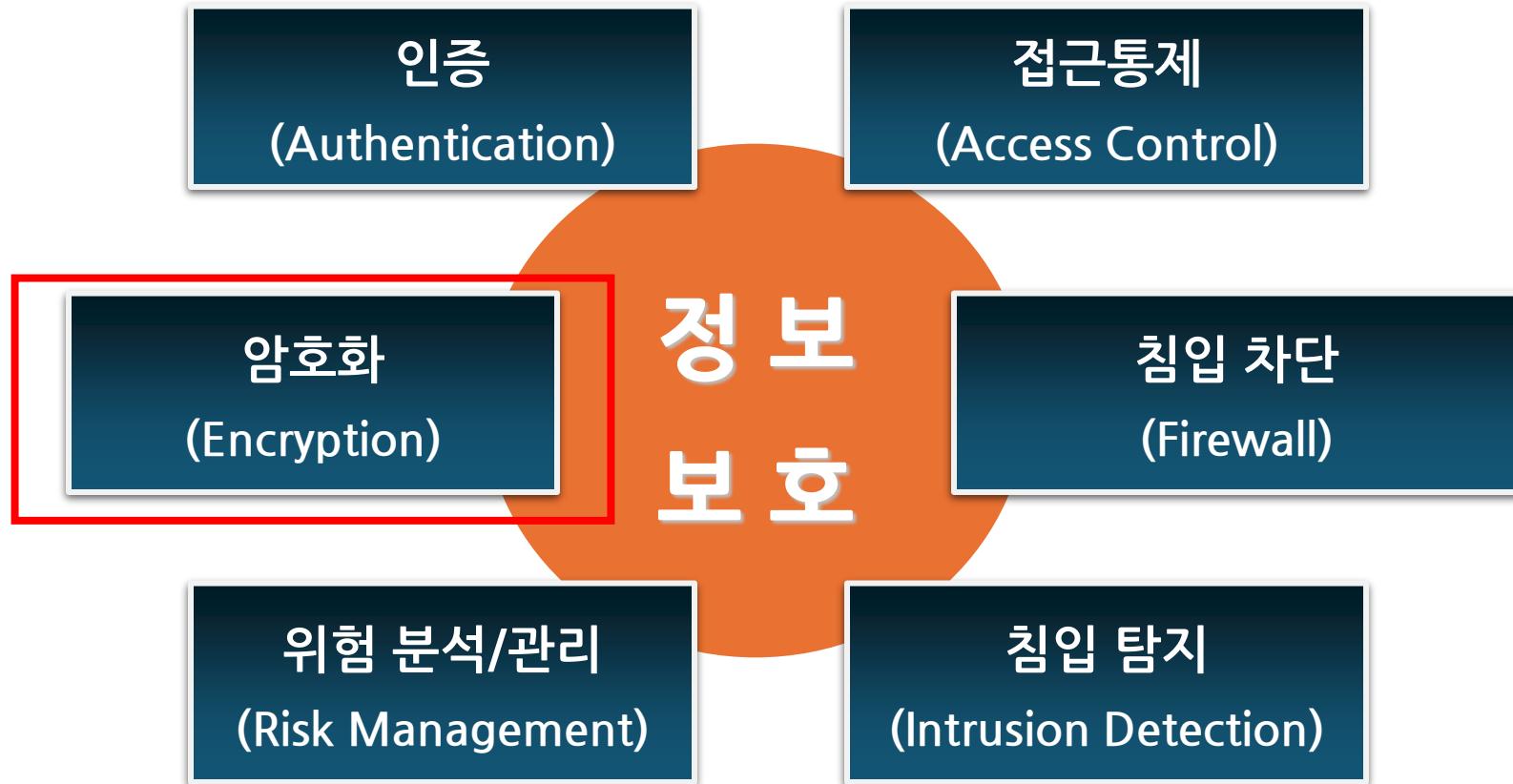
# 암호 기술 (Encryption)

2024. 9

컴퓨터·소프트웨어공학과  
이 형 효  
(hlee@wku.ac.kr)



# 정보보안 기술 - 암호기술



# 정의

- 암호화, 복호화를 위한 원리, 수단, 방법 등을 취급하는 기술이나 과학
  - 암호화: 평 문 → 암호문
  - 복호화: 암호문 → 평 문

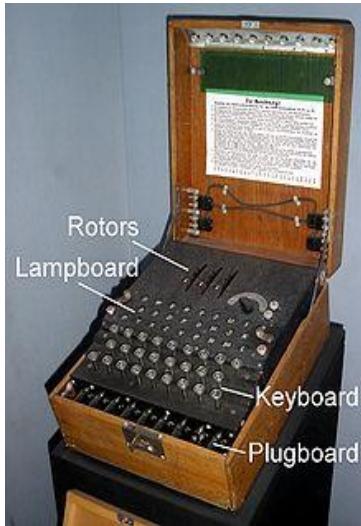
# 주요 용어(terminology)(1)

- 평문, 원문(**P**, plain text)
  - 암호화되지 않은 상태의 정보
- 암호문(**C**, cipher text)
  - 암호화 과정을 거쳐 암호화된 정보
- 암호화 과정(**E**, Encryption)
  - plain text → cipher text
  - 암호 알고리즘(=암호 방법)
- 복호화 과정(**D**, Decryption)
  - cipher text → plain text

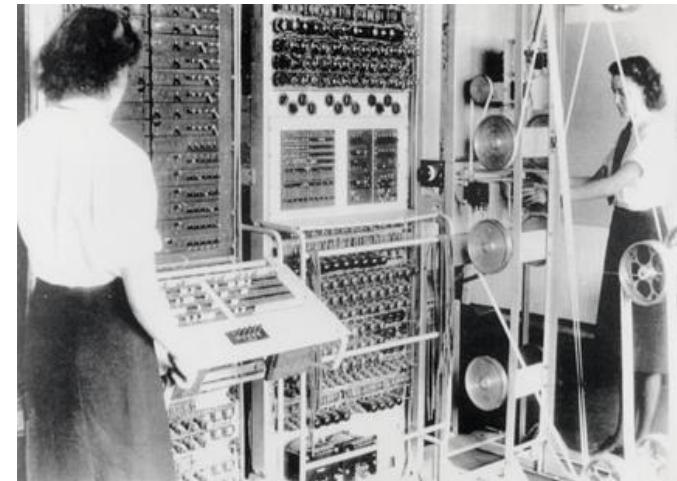
# 주요 용어(terminology)(2)

- **암호 키(Encryption Key)**
  - 암호화 과정에 사용되는 정보(키)
- **복호 키(Decryption Key)**
  - 복호화 과정에 사용되는 정보(키)
- **암호 시스템(Cryptosystem)**
  - 암호/복호 알고리즘과 암호/복호 키
- **암호 분석가(Cryptanalyst)**
  - 수학 또는 기타 지식을 이용하여 암호문을 분석하는 사람

# 암호시스템과 전쟁



독일군 암호 장비(enigma) 수수께끼, 암호



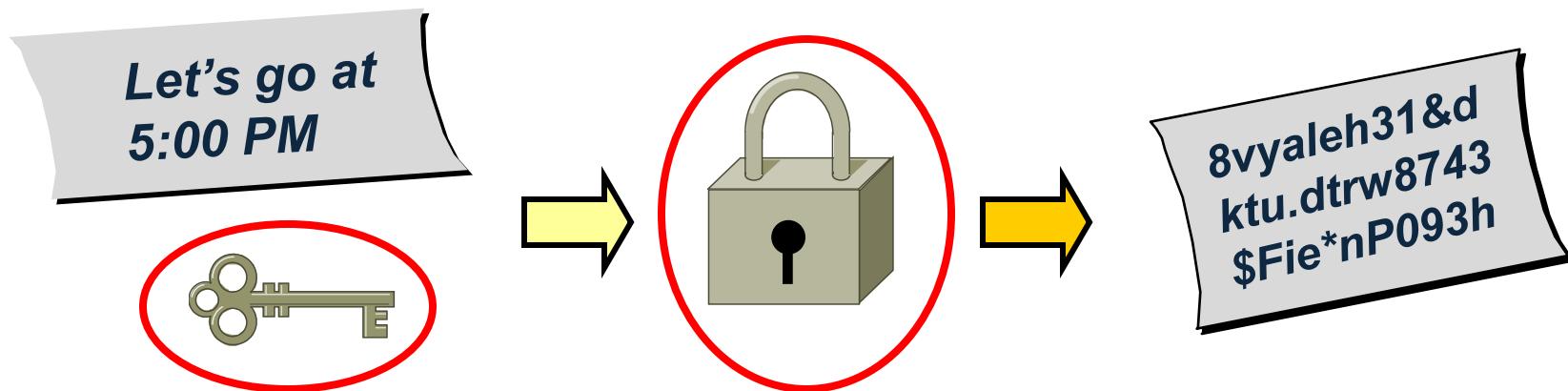
영국의 Colossus 컴퓨터  
(1943, 1944)



Alan Turing(1912~1954)

# 주요 용어(terminology)(3)

## ■ 암호화(Encryption)

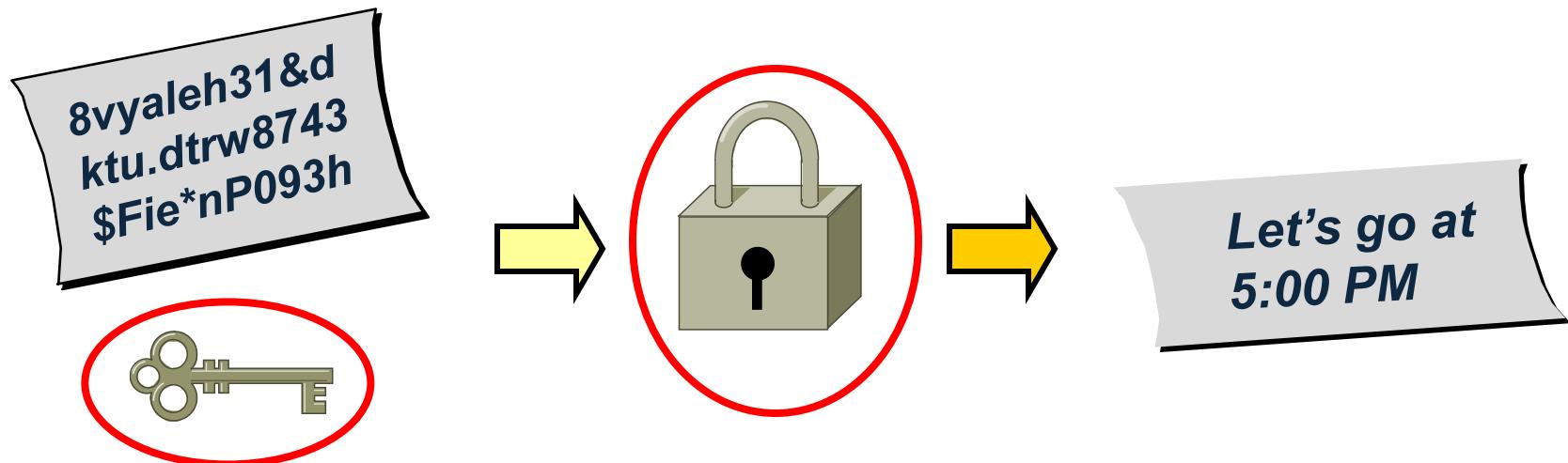


- 평문(plain text)
- 암호 키(encryption key)
- 암호 알고리즘(encryption algorithm)
- 암호문(cipher text)

암호시스템

# 주요 용어(terminology)(4)

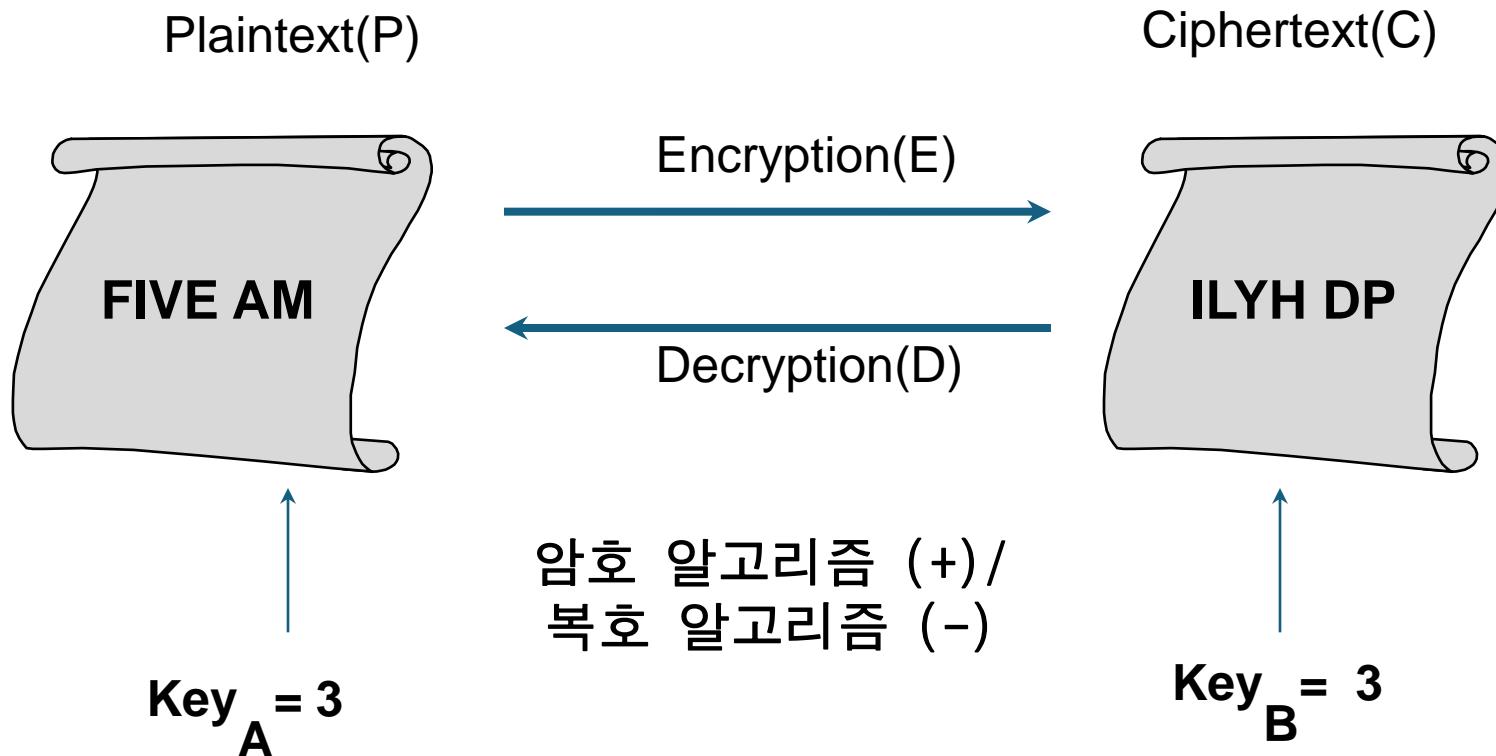
## ■ 복호화(Decryption)



- 암호문(cipher text)
- 복호 키(decryption key)
- 복호 알고리즘(encryption algorithm)
- 평문(plain text)

암호시스템

# Caesar Cipher(1)



# Caesar Cipher(2)

- 암호 알고리즘(방법)

- 

- 복호 알고리즘(방법)

- 

- 암호 키

- 

- 복호 키

-

# Caesar Cipher(3)

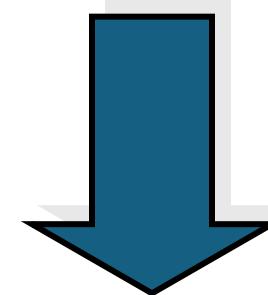
- Substitution 방식
  - 치환(置換), 대체(代替) 방식
- 취약점
  - 가능한 키의 수가 25개로 제한
    - ✓ 암호 키: 1, 2, 3, …, 25
    - ✓ 복호 키: 1, 2, 3, …, 25
  - 글자와 단어의 반복 횟수로 평문 추정 가능

# Caesar Cipher(4)

- 암호문: I L Y H D P

- (1) H K X G C O
- (2) G J W F B N
- (3) F I V E A M
- (4) E H U D Z L
- ...
- (24) K O A J F R
- (25) J M Z I E Q

D W W D F N



ATTACK

# Caesar Cipher(5)

## ■ 문제점

- 암호/복호 알고리즘이 널리 알려져 있음
  - ✓ 덧셈, 뺄셈
- 사용가능한 키의 수가 작고 한정적임
  - ✓ 25개의 키
- 평문의 내용이 인식가능함
  - ✓ 영어
- 따라서, brute-force 공격에 취약

# Caesar Cipher(6)

## ■ 키에 대한 Brute-Force 공격

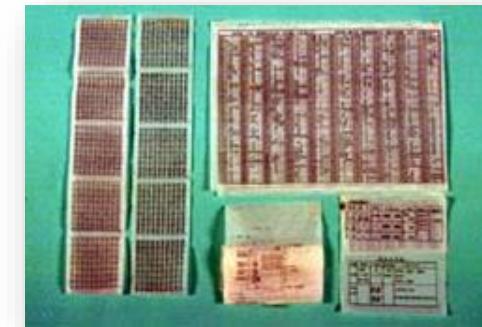
- 성능이 뛰어난 컴퓨터를 이용한 키 계산 시간

키 크기 (bit)	가능한 키의 수	1 마이크로초( $\mu s$ )당 1번의 암호화 실행	1 마이크로초 ( $\mu s$ ) 당 $10^6$ 번의 암호화 실행
32	$2^{32} = 4.3 \times 10^9$	$2^{31} = 35.8 \text{ min.}$	2.15 ms
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} = 1142(\text{Y})$	10.01(H)
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} = 5.4 \times 10^{24} (\text{Y})$	$5.4 \times 10^{18} (\text{Y})$
26문자	$26! = 4 \times 10^{26}$	$6.4 \times 10^{12}(\text{Y})$	$6.4 \times 10^6(\text{Y})$

# Substitution(대체) 암호

## ■ 예

- 암호 알고리즘: **대체표**
- 복호 알고리즘: **대체표**
- 암호/복호 알고리즘(대체표)을 공개하지 않는 방식



평문                   SELL 100 SHARES OF ABCD INDUSTRIES. JOHN SMITH.

**대체표**

A	B	...	E	...	H	...	S	...	9	0	þ	,	.
J	U	...	P	...	4	...	5	...	L	V	X	Z	1

**암호문**

5PEEXIVVX54JYP5X,2XJUTKXBCK95RYBP51XQ,4CX53BR41

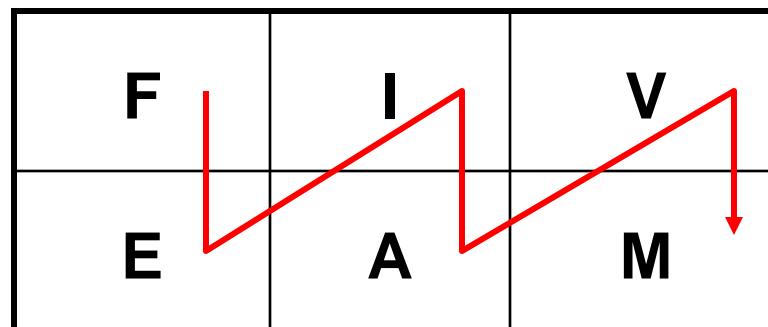
# Transposition(전치) 암호(1)

## ■ Transposition

- 전치(轉置, 위치를 바꿈)
- 단순한 글자의 대체 대신 글자의 순서를 변경

## ■ 예

F I V E A M



F E I A V M

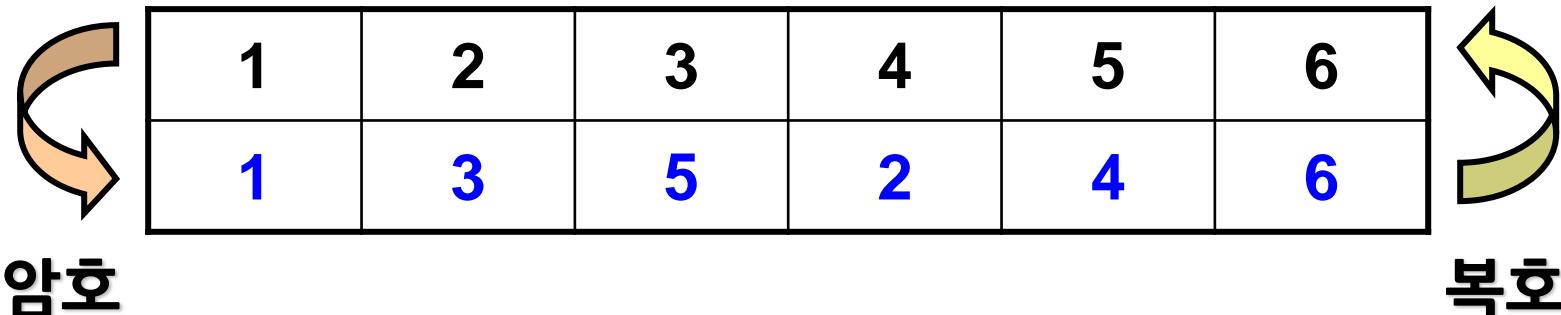
# Transposition(전치) 암호(2)

- 예(계속)

F I V E A M

F E I A V M

F	I	V
E	A	M



# Diffusion 기법

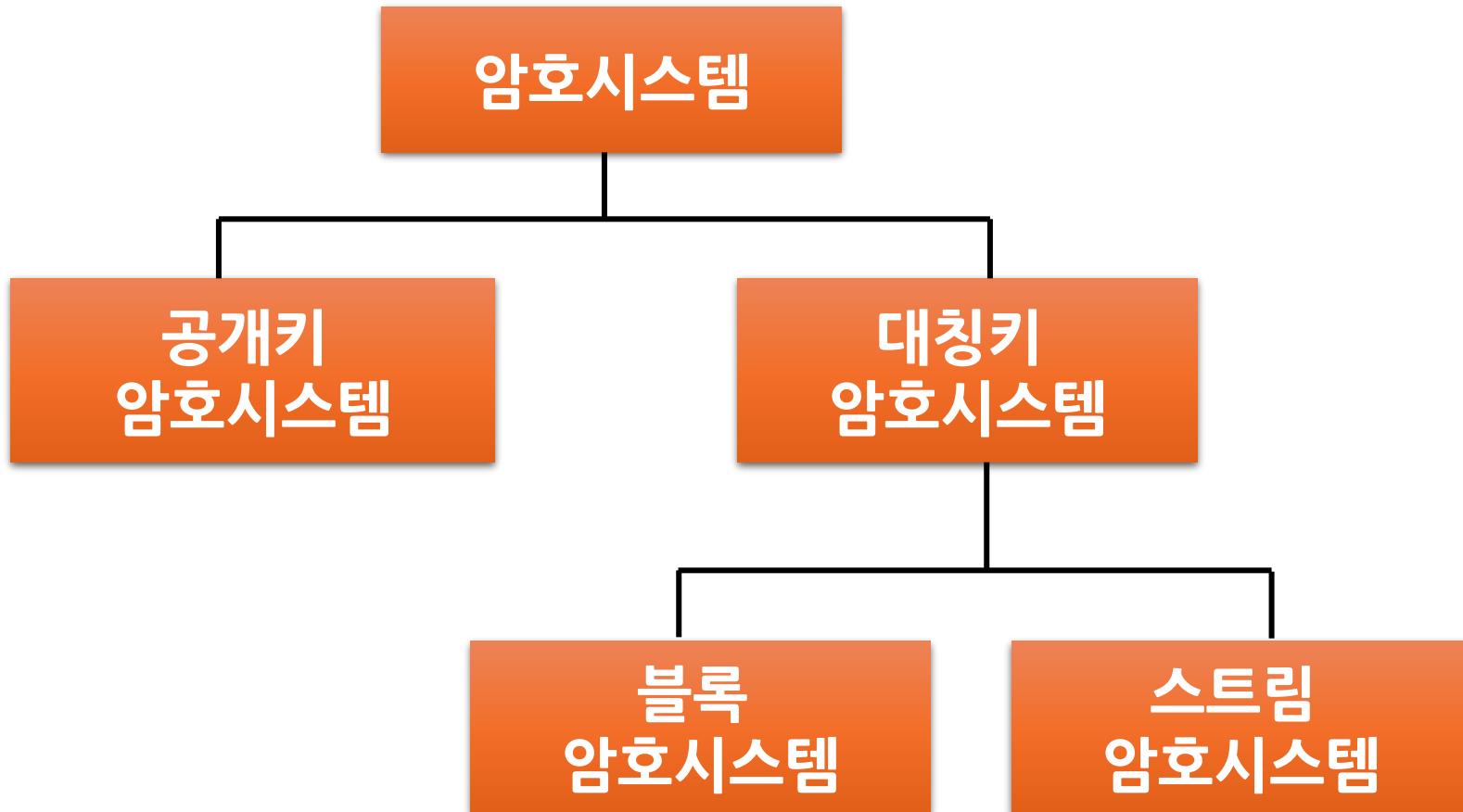
- Diffusion
  - 확산, 흐림
- 암호화 과정
  - Substitution 기법
  - Transposition 기법
- 목적
  - 평문과 암호문 간의 연관성을 쉽게 알지 못하도록
  - 암호문으로부터 평문을 추론할 수 없도록

# Confusion 기법

## ■ 목적

- 암호문과 암호키간의 연관성이 없도록
- 암호분석가가 수집된 여러 개의 암호문을 분석하여 암호키를 추론해 낼 수 없도록 하는 성질

# 암호시스템 분류



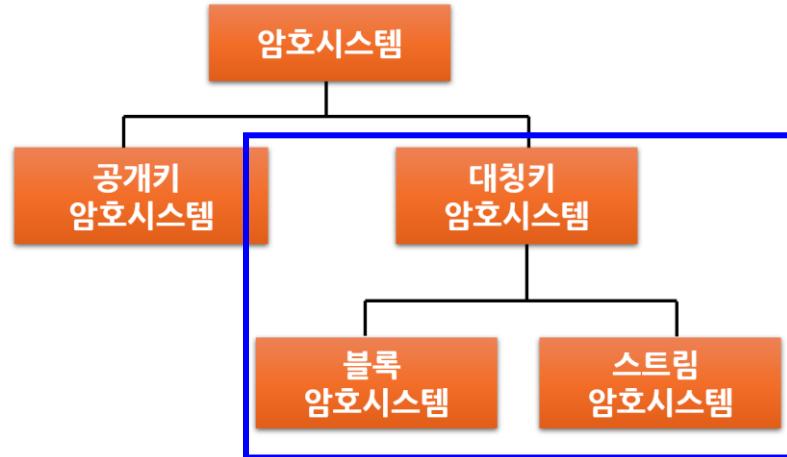
# 대칭키 암호 시스템(1)

## ■ Symmetric Cryptosystem

- 비밀키 암호 시스템
- 관용키 암호 시스템

## ■ 특징

- 오랜 역사
- 암호키와 복호키가 동일
- 알고리즘의 동작속도가 빠름
- 키의 길이가 공개키 암호 시스템에 비해 짧음



# 대칭키 암호 시스템(2)

## ■ 표기법

- 암호 알고리즘:  $E$ , 암호키:  $Ke$
- 복호 알고리즘:  $D$ , 복호키:  $Kd$

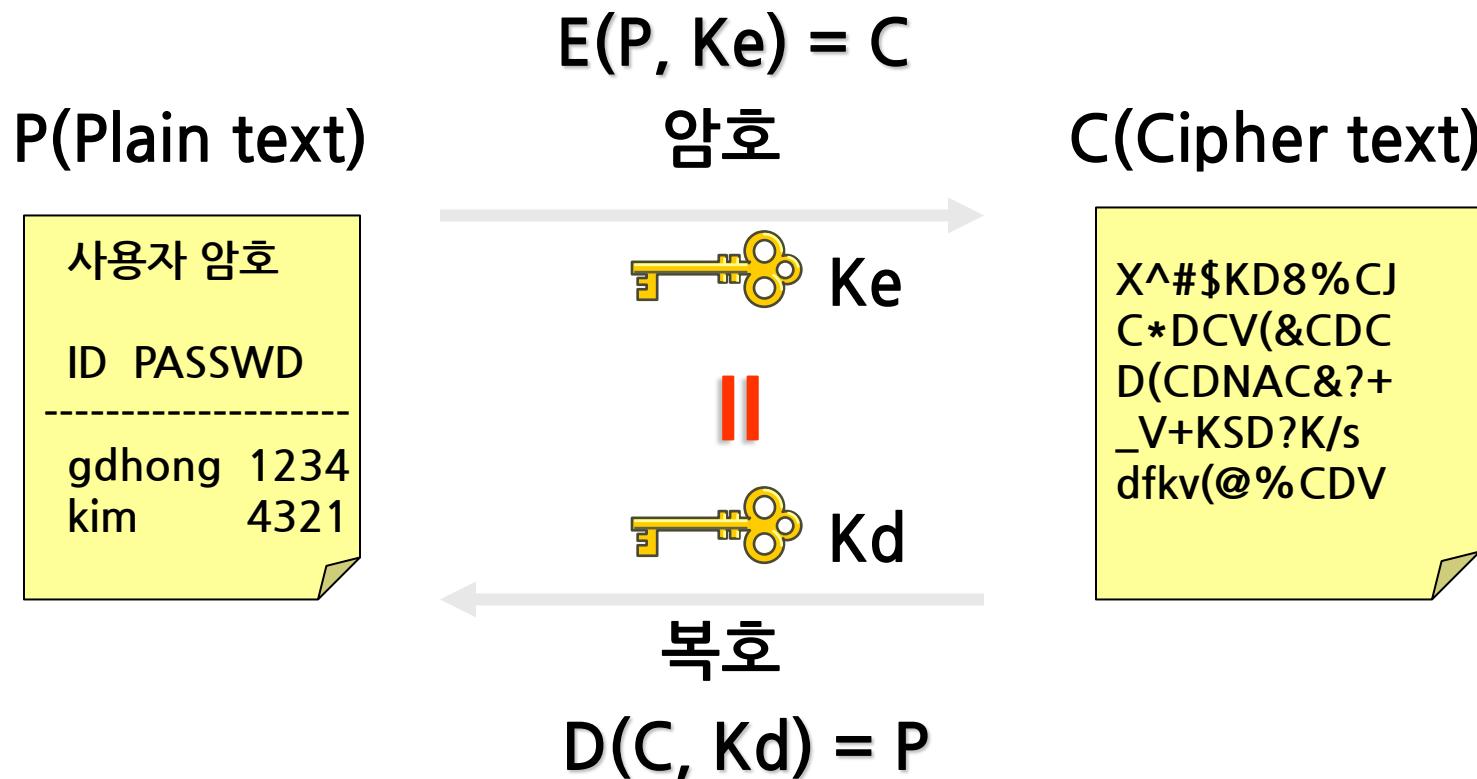
## ■ 암호화 과정

- $E(P, Ke) = C$

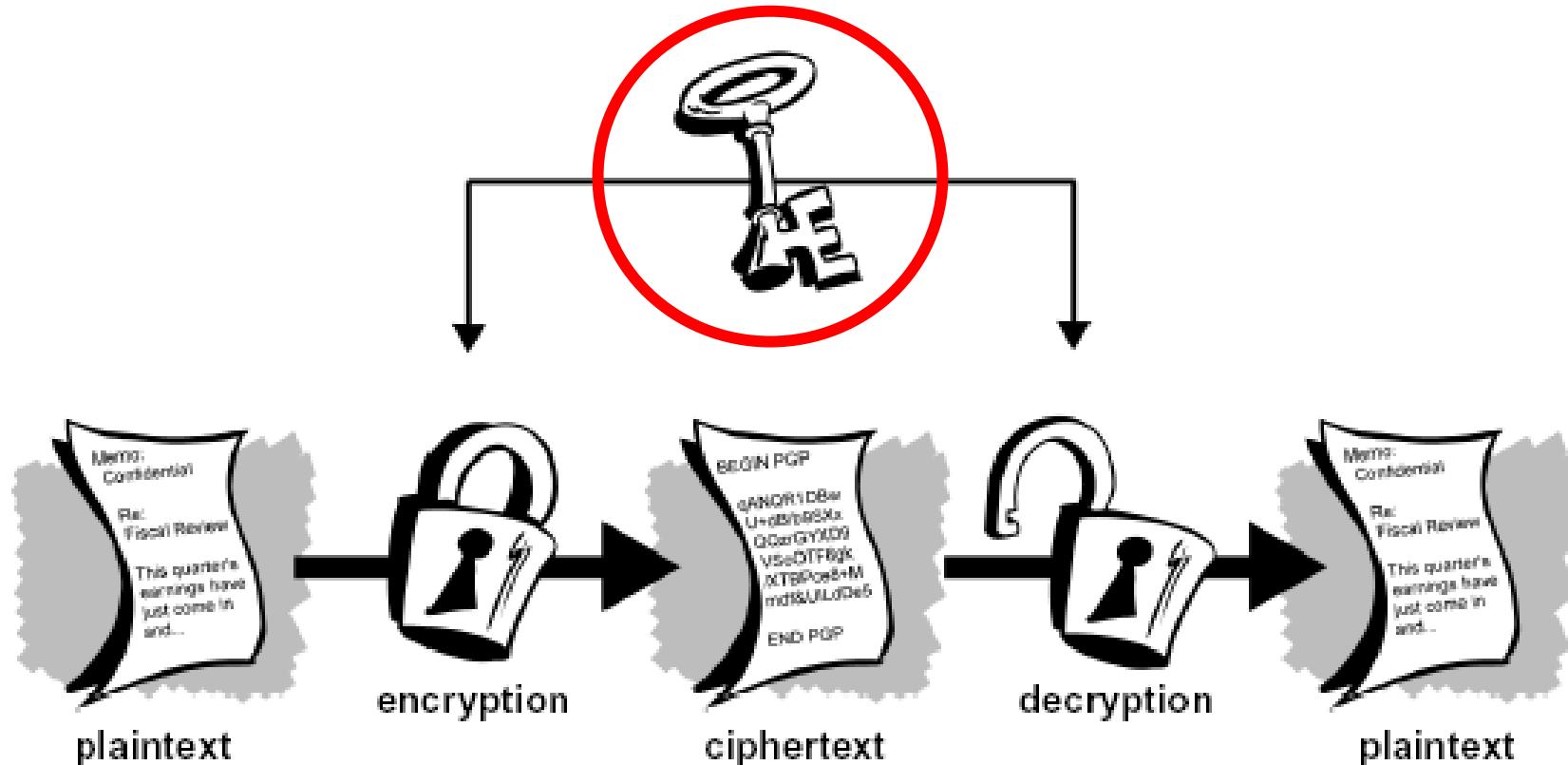
## ■ 복호화 과정

- $D(C, Kd) = P$

# 대칭키 암호 시스템(3)



# 대칭키 암호 시스템(4)

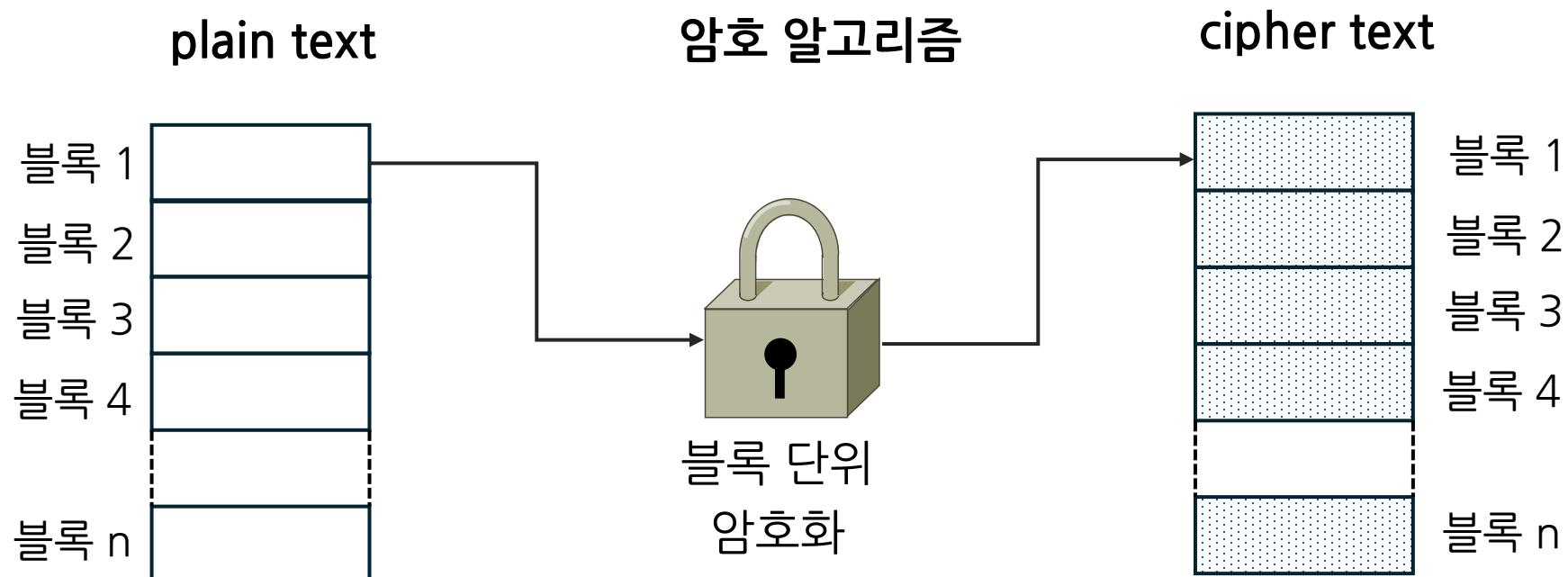


# 대칭키 암호 시스템(5)

- 암호키 = 복호키
- 암호키(복호키)는 메시지 송, 수신자만이 인지
  - 메시지 수신자에게 복호키 전송 필요
  - 전송방법, 전송과정의 안전성 문제
  - **필연적으로 키 분배(key distribution) 문제 발생**
    - ✓ **키 분배 문제:** 안전하게 암호키(복호키)를 수신자에게 전달하는 문제
- 동작 속도가 빠름

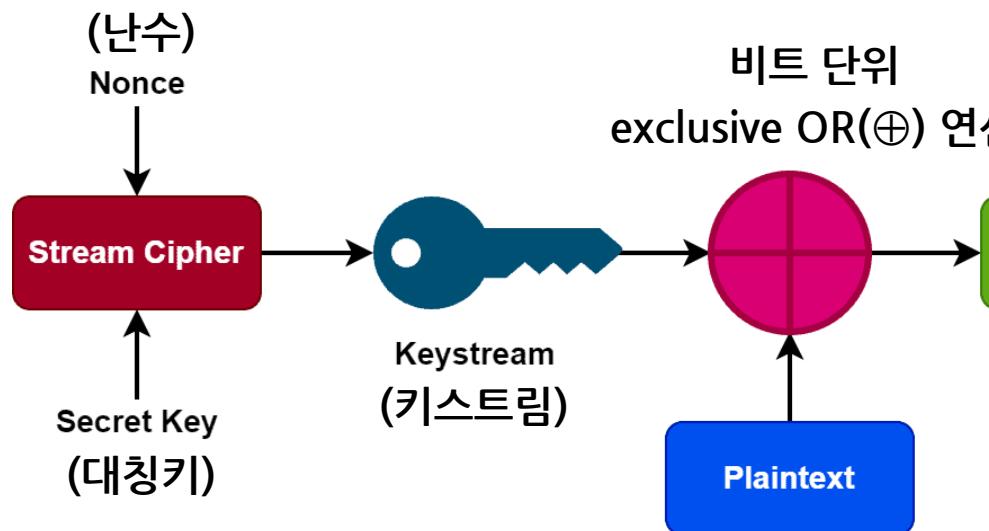
# 대칭키 암호 시스템(6) - 블록 암호

- 주어진 평문을 정해진 길이의 **블록**(64 혹은 128비트)으로 나누어 **블럭단위**로 암호화 수행
  - 대부분의 암호 알고리즘에서 채택



# 대칭키 암호 시스템(7) - 스트림 암호

- 평문과 같은 길이의 키 스트림(stream)을 생성
- 평문과 키 스트림을 비트단위로 합하여(Exclusive OR) 암호문을 얻는 알고리즘

exclusive OR( $\oplus$ ) 연산

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

# 공개키 암호 시스템(1)

## ■ Asymmetric Cryptosystem

- 비대칭키 암호 시스템

## ■ 특징

- 사용자당 2개의 키 소유

✓ 개인키: 사용자가 안전하게 보관

✓ 공개키: 일반에게 공개

- 개인키와 공개키는 서로 다름

✓ 공개키를 이용하여 개인키를 알아내기가 매우 어려움

- 키의 길이가 대칭키 암호 시스템에 비해 긴 특징

- 알고리즘의 동작속도가 느림



# 공개키 암호 시스템(2)

## ■ 표기법

- 암호 알고리즘: E, 암호키: Ke
- 복호 알고리즘: D, 복호키: Kd

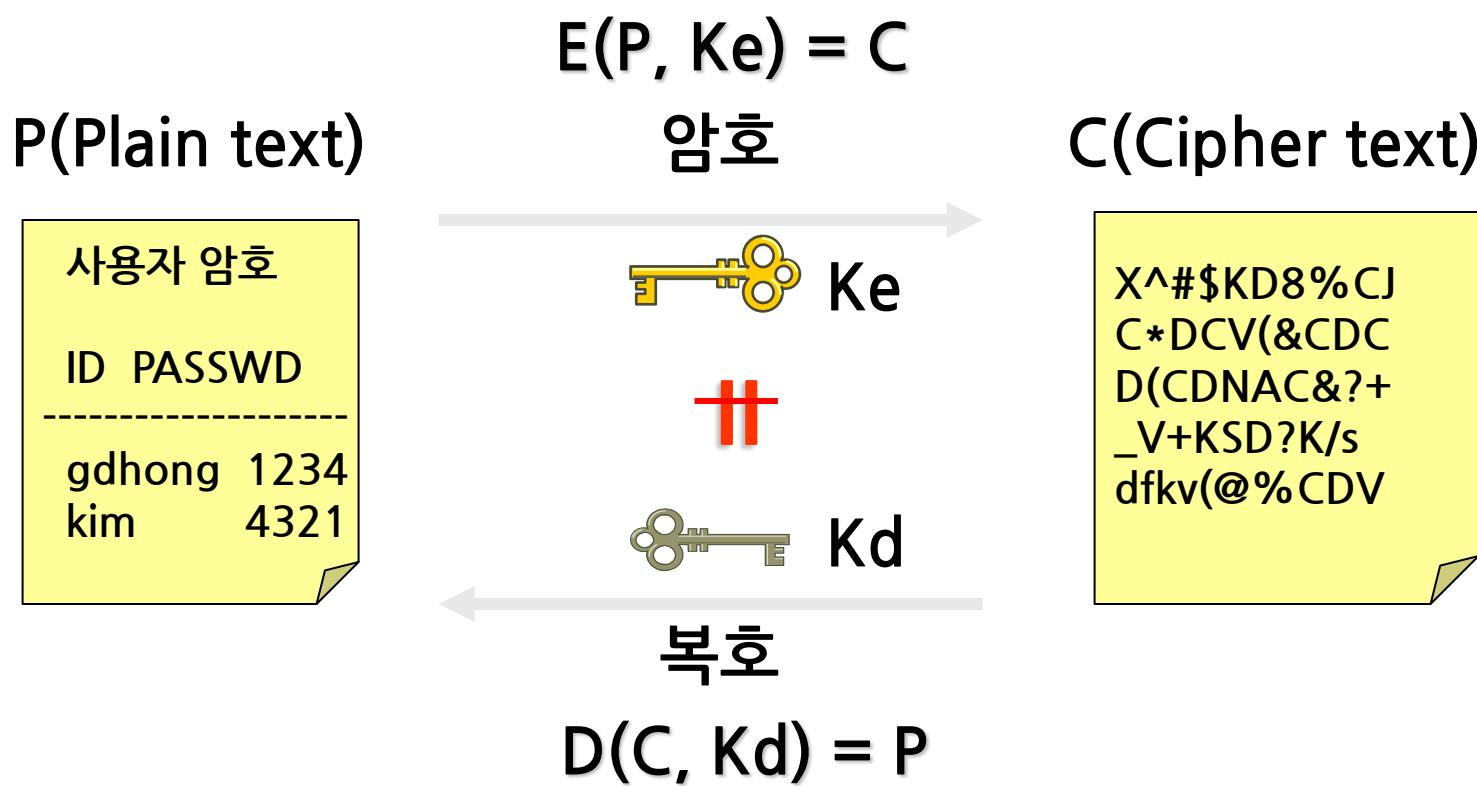
## ■ 암호화 과정

- $E(P, Ke) = C$

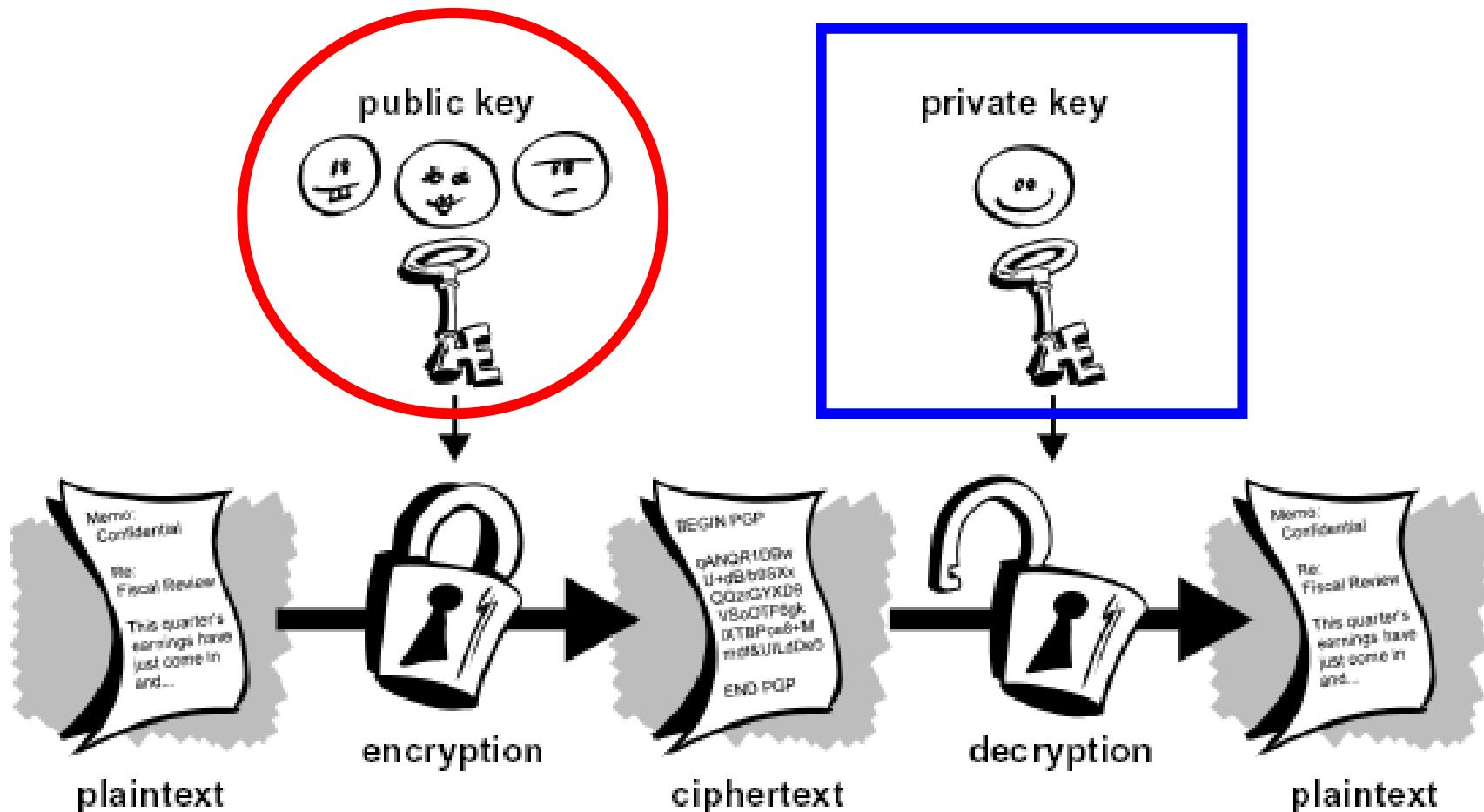
## ■ 복호화 과정

- $D(C, Kd) = P$

# 공개키 암호 시스템(3)



# 공개키 암호 시스템(4)



# 공개키 암호 시스템(5)

## ■ 특징

- 한 사용자당 2개의 키(**공개키**, **개인키**) 소유
- **개인키(private key), 비밀키(secret key)**
  - ✓ 사용자만이 알고 있는 키
  - ✓ 다른 사람에게 공개해서는 안되는 키
  - ✓ 개인키가 노출될 경우 심각한 보안 위협 발생
- **공개키(public key)**
  - ✓ 다른 사람들에게 공개되는 키
- **공개키에서 개인키를 추론해 내기는 거의 불가능**

# 공개키 암호 시스템(6)

- 동작 모드
  - 암호 모드(Encryption mode)
  - 인증 모드(Authentication mode)
- 암호 모드
  - 메시지의 내용을 메시지 수신자만 복원할 수 있도록
  - 메시지의 비밀성 보호 목적(confidentiality)
- 인증 모드
  - 메시지를 보낸 사람이 자신이 메시지를 보냈음을 확인
  - 메시지 출처 인증 목적(data origin authentication)

# 공개키 암호 시스템(7)

## ■ 암호 모드

- 메시지 수신자의 공개키로 암호화
- 메시지 수신자의 개인키로만 복호화 가능
- 메시지 수신자만이 메시지 내용 복원(복호화) 가능

## ■ 인증 모드

- 메시지 발신자의 개인키로 암호화
- 메시지 수신자는 메시지 발신자의 공개키로 복호화
- 메시지 발신자의 공개키로 내용이 복원되었다면  
메시지 발신자의 비밀키로 암호화되었음을 의미

# 공개키 암호 시스템(8)

- 암호 모드 - 지정된 수신자만이 복호화할 수 있도록

(상황) Alice가 Bob에게만  
중요 문서(M)를 전달

Alice

A<sub>개</sub>  
A<sub>공</sub>



Q1. 오직 Bob만이 복호화하도록  
하는 복호키는 ?

Q2. 해당 복호키와 짹이 되는  
암호키는?

Q3. Alice는 암호키를 어떻게  
알아낼까? (키 분배 문제)

Q4. Charles는 암호문을  
복호화할 수 있을까?

Bob

B<sub>개</sub>B<sub>공</sub>C<sub>개</sub>C<sub>공</sub>

Charles

# 공개키 암호 시스템(9)

- 인증 모드 - 보낸 사용자가 누구인지 확인할 수 있도록

(상황) 중요 문서(M)를 Alice가 보낸 사실 증명 필요

Alice

A<sub>개</sub>  
A<sub>공</sub>



Q1. 오직 Alice만이 알 수 있는 암호키?

Q2. 해당 암호키와 짹이 되는 복호키는?

Q3. Bob은 복호키를 어떻게 알아낼까? (키 분배 문제)

Q4. Charles는 암호문을 복호화할 수 있을까?

Bob

B<sub>개</sub>B<sub>공</sub>C<sub>개</sub>C<sub>공</sub>

Charles

# DES(Data Encryption Standard)(1)

## ■ 역사

- 1972: 미국 NIST에 의해 표준 암호기술 개발 시작
- 1977: DES를 표준 암호기술로 확정

## ■ 특징

- **56-bit** 길이의 암호 키 이용
  - ✓ 실제 암호 키의 길이 64-bit중 8-bit는 패리티로 사용
- Feistel 방식의 **블록 암호 방식**
  - ✓ Substitution, transposition 기법 이용
- IBM에서 제안한 Lucifer 시스템을 개량
  - ✓ 128-bit 길이의 암호 키

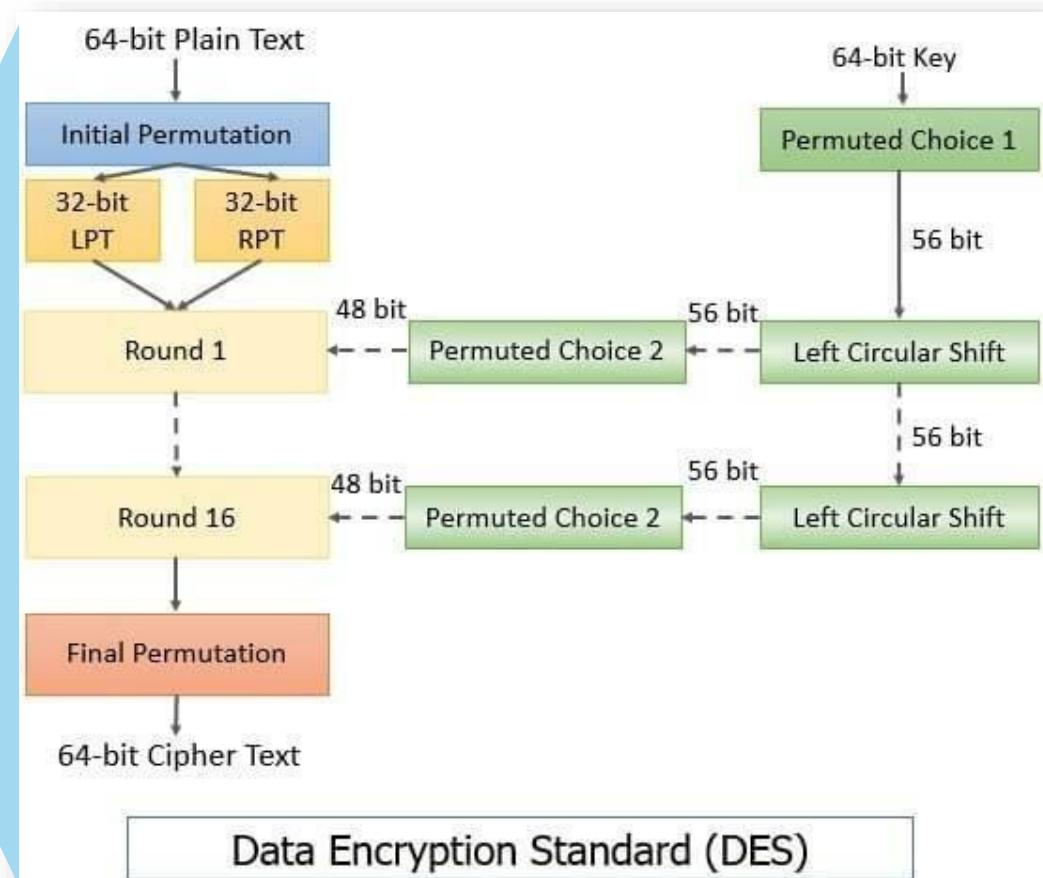
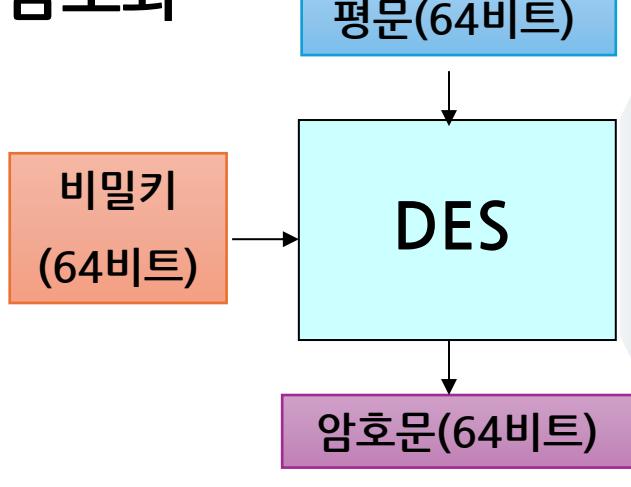
# DES(Data Encryption Standard)(2)

## ■ 문제점

- 56-bit 길이의 암호 키 채택과 trapdoor 존재에 대한 계속적인 논쟁
- 1977년 이후 매 5년마다 DES 알고리즘에 대한 안전성 검토 절차 마련
  - ✓ 1982년, 1987년, 1992년
- 1997년 DES의 후속 알고리즘 제안
  - ✓ AES(Advanced Encryption Standard)
- 2000년 10월 Rijndael 암호 알고리즘을 DES 후속 암호 알고리즘으로 채택 (개발국: 벨기에)

# DES(Data Encryption Standard)(3)

**암호화**



# RSA(1)

- 1977년, MIT의 Rivest, Shamir, Adleman
- 소인수 분해의 어려움에 근거
- 시스템 구성
  - 공개키 :  $n (=p \times q)$ ,  $e$
  - 개인키 :  $n (=p \times q)$ ,  $d$
- 암호화 :  $C = E(M) = M^e \pmod{n}$
- 복호화 :  $M = D(E(M)) = (M^e)^d \pmod{n} = M$

# RSA(2)

- **Encryption Speed**
  - 대칭키 암호시스템보다 상당히 느린 동작 속도 특징
- **키 크기**
  - 안전한 키 최소 길이 - 1024-bit
  - 키의 길이는 256-bit 배수로 증가
  - 현재 안전한 키 길이로 2048-bit 권장

# RSA(3)

## ■ RSA 알고리즘 정리

- 키 생성

- ✓  $p, q$  선택 ( $p, q$ 는 소수),  $n = p \times q$  계산
- ✓  $\phi(n) = (p-1) \times (q-1)$  계산
- ✓ 정수  $e$  선택 ( $\gcd(\phi(n), d) = 1, 1 < d < \phi(n)$ ) ( $e$ 와  $\phi(n)$  서로 소)
- ✓  $d$  계산 ( $e \times d \bmod \phi(n) = 1$  관계 성립)
- ✓ 공개키 ( $KU = \{e, n\}$ )
- ✓ 개인키 ( $KR = \{d, n\}$ )

- 암호화

- ✓  $C = M^e \pmod{n}$

- 복호화

- ✓  $M = C^d \pmod{n}$

# RSA(4)

## ■ RSA 알고리즘 사용 예

- 공개키와 개인키 생성

1. 두 소수  $p = 7$ ,  $q = 17$  을 선택

2.  $n = pq = 7 \times 17 = 119$  계산

3.  $\phi(n) = (p-1) \times (q-1) = 96$  계산

4.  $\phi(n) = 96$ 과 서로 소이고  $\phi(n)$ 보다 작은  $e$  선택 ( $e = 5$ )

5.  $d \times e \text{ mod } 96 = 1$  이고  $d < 96$  인  $d$ 를 결정 ( $d = 77$ )

⇒ 공개키 KU = {5, 119}, 개인키 KR = {77, 119}

# RSA(5)

## ■ RSA 알고리즘 사용 예(계속)

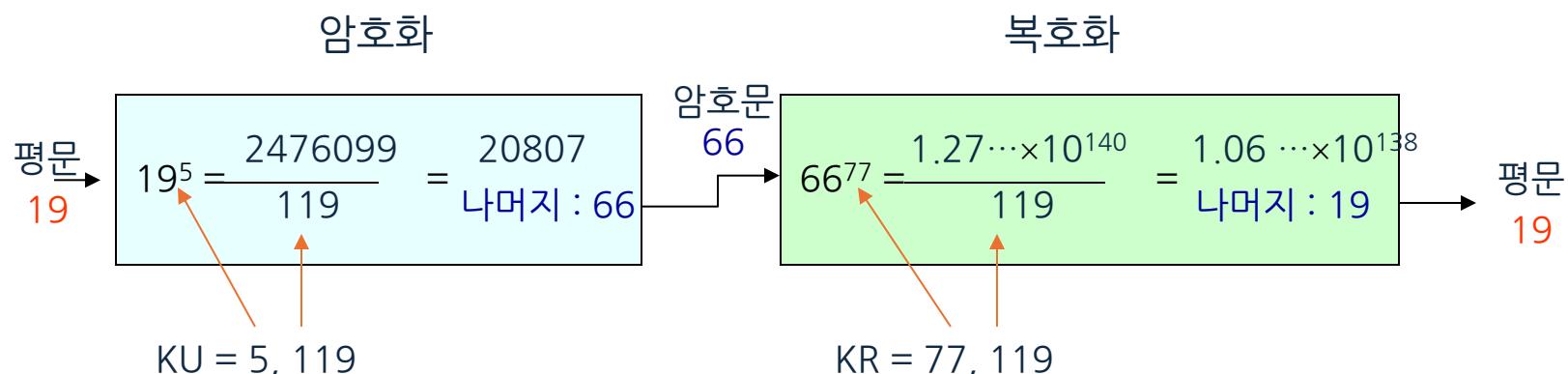
- 암호화와 복호화

: 평문 메시지  $M = 19$  일 경우

✓ 암호문 :  $19^5 \bmod 119 \Rightarrow 66$

✓ 복호문 :  $66^{77} \bmod 119 \Rightarrow 19$

- 공개키  $KU = \{5, 119\}$
- 개인키  $KR = \{77, 119\}$



# 대칭키, 공개키 암호 시스템 비교(1)

	대칭키 암호시스템	공개키 암호시스템
키의 상호관계	암호키 = 복호키	암호키 ≠ 복호키
암호키	대칭키	공개키 또는 개인키
복호키	대칭키	공개키 또는 개인키
대표적 예	DES/RC4/ <b>SEED/ARIA</b>	RSA/DH/DSS/ECC
암호키 전송	필요	불필요
키 개수	$n(n-1)/2$	$2n$
안전한 인증	곤란	용이
암호화 속도	고속	저속
경제성	높음	낮음
전자서명 활용	복잡	간단

# 대칭키, 공개키 암호 시스템 비교(2)

## ■ SEED

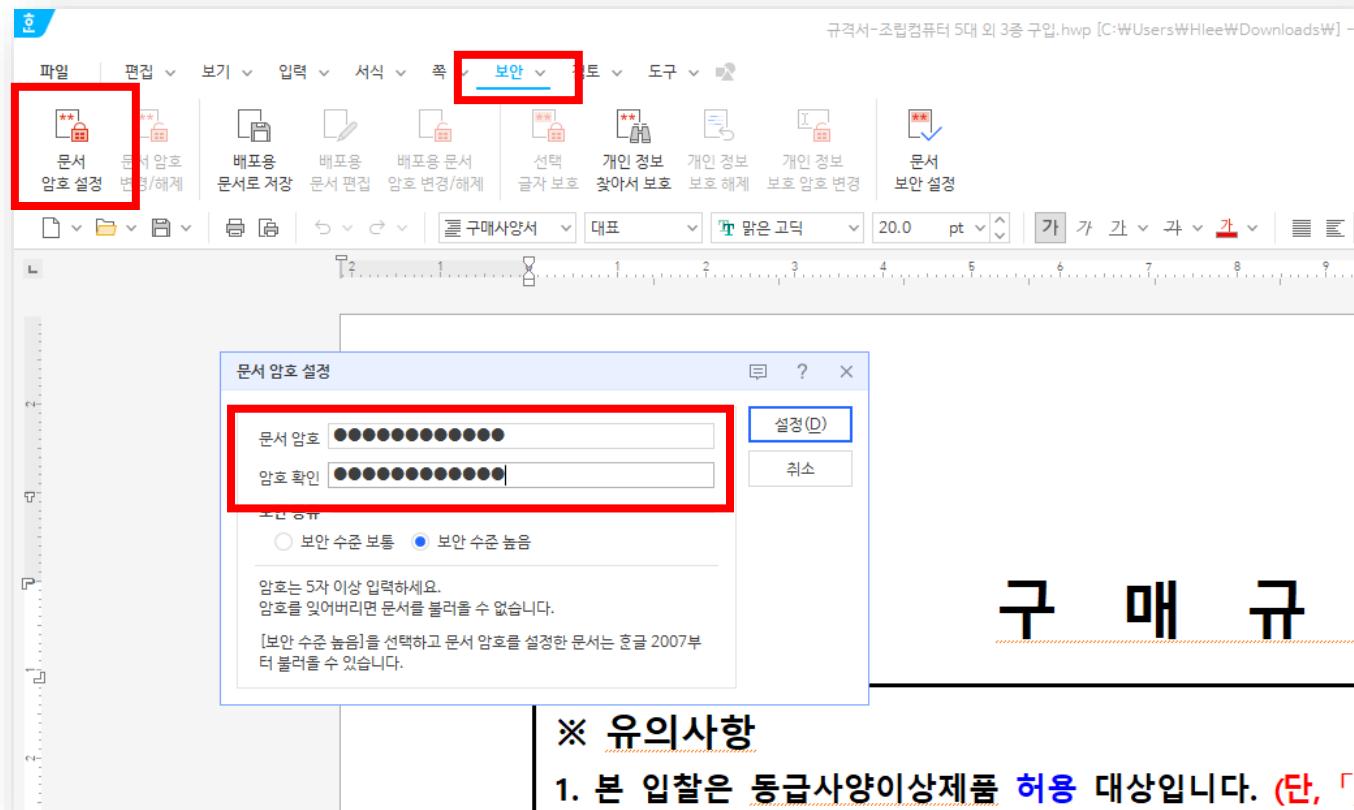
- 1999년 한국인터넷진흥원(KISA) 개발 대칭키 암호 알고리즘
- DES 암호화 방식 채택
- 국제 표준 등재 (암호키 길이 128-bit 버전)

## ■ ARIA(Academy, Research Institute, Agency)

- 2003년 학계, 연구소, 정부기관 공동 개발
- 경량 환경 및 하드웨어 구현 목적
- 키 길이: 128/192/256비트
- 국내 개발 보안 소프트웨어에 의무 사용

# 암호 사용 사례(1)

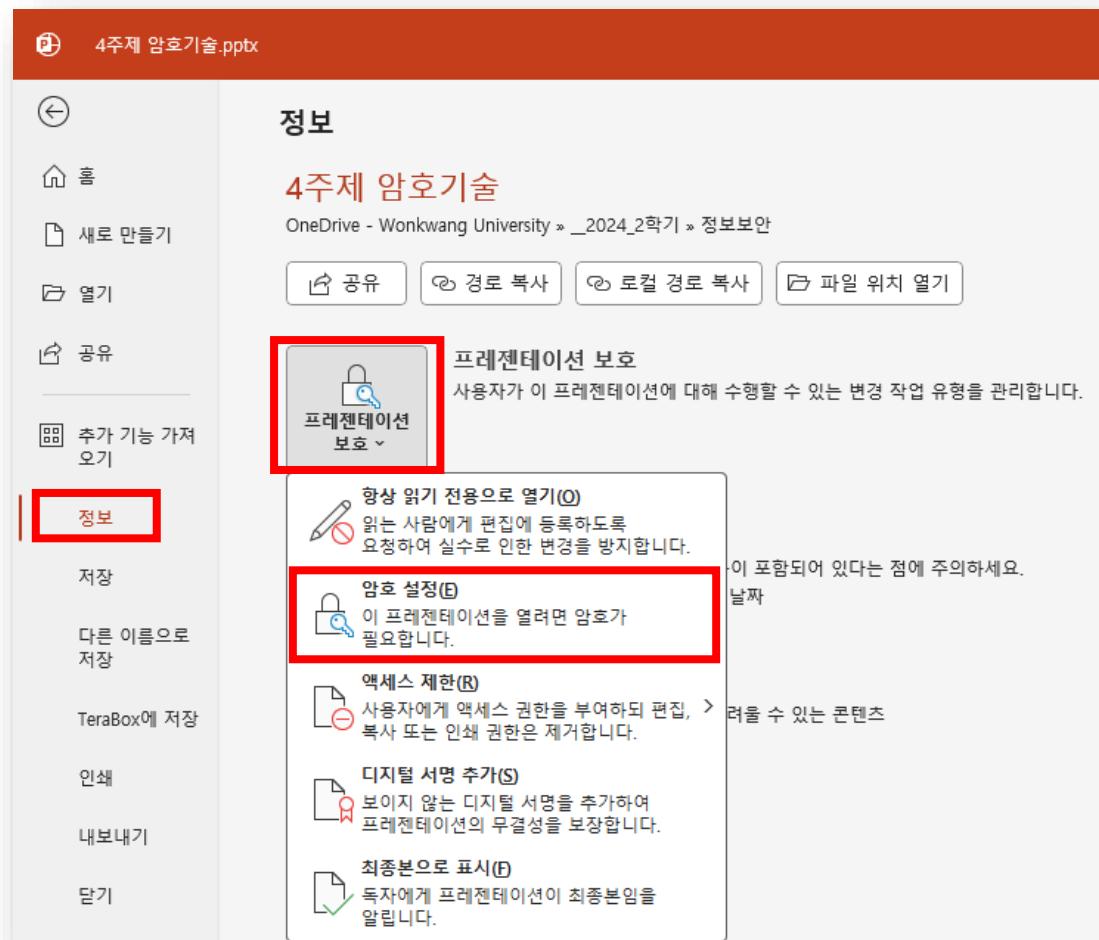
- 한글 워드프로세서 (ARIA 암호 알고리즘 사용)
  - [보안] - [문서 암호 설정]



# 암호 사용 사례(2)

## ■ MS Powerpoint

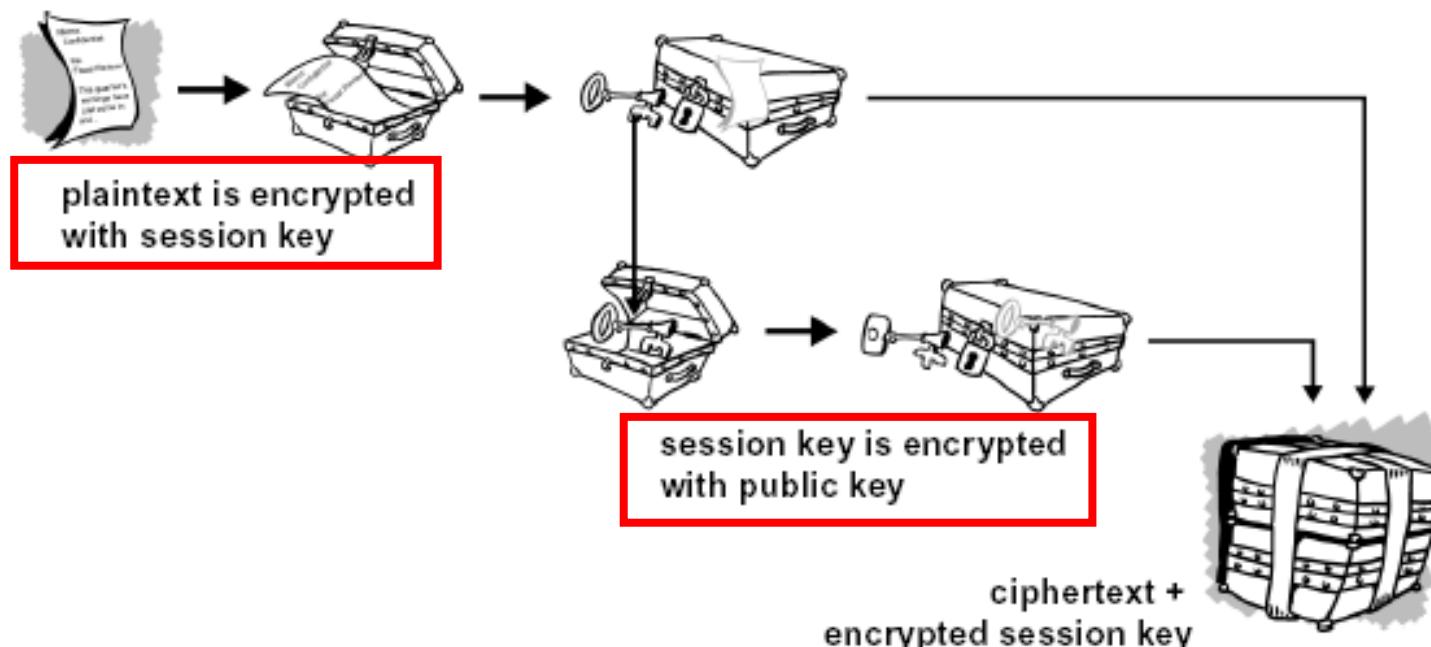
- [파일] - [정보] -  
[프레젠테이션 보호]  
- [암호 설정]



# 대칭키, 공개키 암호 시스템 활용 사례(1)

## ■ 송신측(암호화 과정)

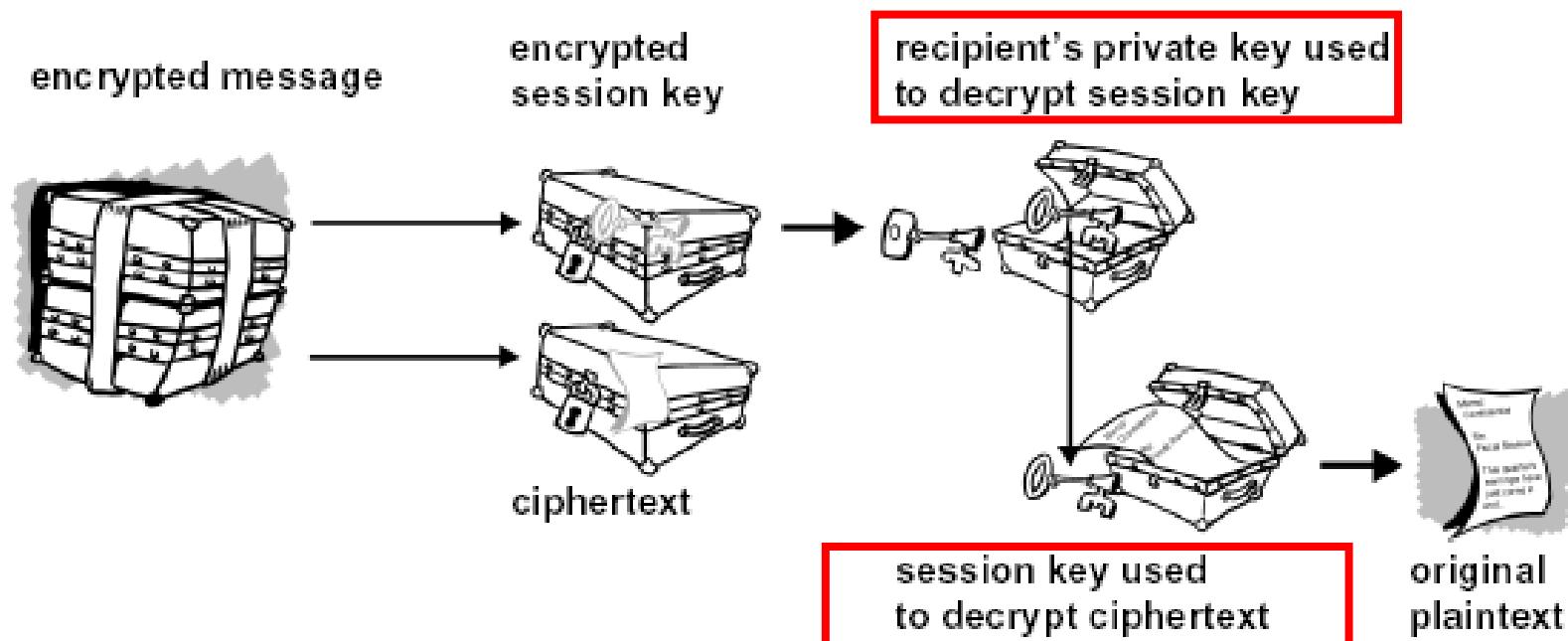
- 대칭키 암호시스템 - 평문 암호용 (빠른 동작 속도)
- 공개키 암호시스템 - 세션키 암호용 (키 교환 문제 해결)



# 대칭키, 공개키 암호 시스템 활용 사례(2)

## ■ 수신측(복호화 과정)

- 대칭키 암호시스템 - 암호문 복호용 (빠른 동작 속도)
- 공개키 암호시스템 - 세션키 복호용 (키 교환 문제 해결)



# 기타 공개키 암호 시스템(1)

- DH
  - Diffie-Hellman
- ECC
  - Elliptic Curve Encryption
- DSS
  - Digital Signature Standard

# 기타 공개키 암호 시스템(2)

## ■ DH(Diffie-Hellman)

- 1976년 Diffie, Hellman에 의해 개발
- 대칭키의 안전한 교환 절차 정의
- 메시지 암호화 기능 없음
- 전자서명 기능 없음

# 기타 공개키 암호 시스템(3)

## ■ ECC(Elliptic Curve Cryptosystem)

- 1985년 Neil Koblitz, Victor Miller에 의해 개발
- RSA나 DSA에 비해 보안성 우수
  - ✓  $10^{12}$  MIPS year의 경우
  - ✓ RSA, DSA: 1024 비트 요구
  - ✓ ECC: 160 비트 요구
- 메시지 암호 기능 제공
- 전자서명 기능 제공

# 기타 공개키 암호 시스템(4)

## ■ DSS(Digital Signature Standard)

- 1987년 미국의 NIST에 의해 제안
- 전자서명 목적의 공개키 암호 알고리즘
  - ✓ 전자문서 작성자의 신원 확인(authentication)
  - ✓ 전자문서의 변경여부 확인(integrity)
- 메시지의 암호화 기능 없음
- 키 교환 기능 없음

# 해쉬 함수(Hash Function)와 전자 서명(Digital Signature)

2024. 10.

컴퓨터·소프트웨어공학과  
이 형 효  
(hlee@wku.ac.kr)



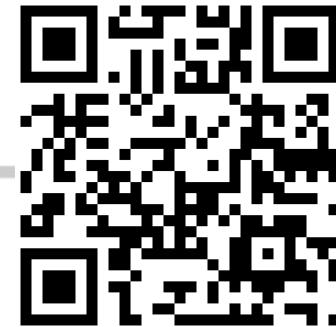
# 해쉬 함수

# (Hash Function)

# 해쉬 함수 일반(1)

- 다양한 길이의 긴 문서를 입력 받아 일정 크기의 출력(메시지 디제스트)를 생성
  - 계산 속도가 빠름
  - 역계산(inverse computation)이 불가능
    - ✓ One-Way Function
  - 데이터 변조 검증에 사용
    - ✓ Integrity 유지여부 점검에 이용
  - 전자서명 과정에 활용
- 암호 알고리즘 활용 가능(cryptographic hash)

# 해쉬 함수 일반(2)



## ■ 동작 구조

입력  
(길이에 무관)



해쉬 함수  
(MD4, MD5,  
SHA-1, SHA-256, SHA-512,  
RIPEMD-160, …)



9bcfacf49c4636ee69e6ca22aae984c6  
출력(메시지 디아제스트, 해쉬값, …)  
(비교적 짧은 길이)

# 해쉬 함수 조건 [ $h(M) = md$ ]

- ① 계산 속도가 빨라야
  - 계산 절차(알고리즘)이 간단해야
- ② 역계산(inverse computation)이 불가능 해야
  - 일방향 함수(one-way function)
  - $md$ 값을 분석하여  $M$ 을 알아내기는 사실상 불가능
- ③ 충돌(collision)이 없어야
  - 서로 다른 두 입력에 대한 해쉬 결과는 서로 달라야
  - $M_1 \neq M_2 \Rightarrow h(M_1) \neq h(M_2)$

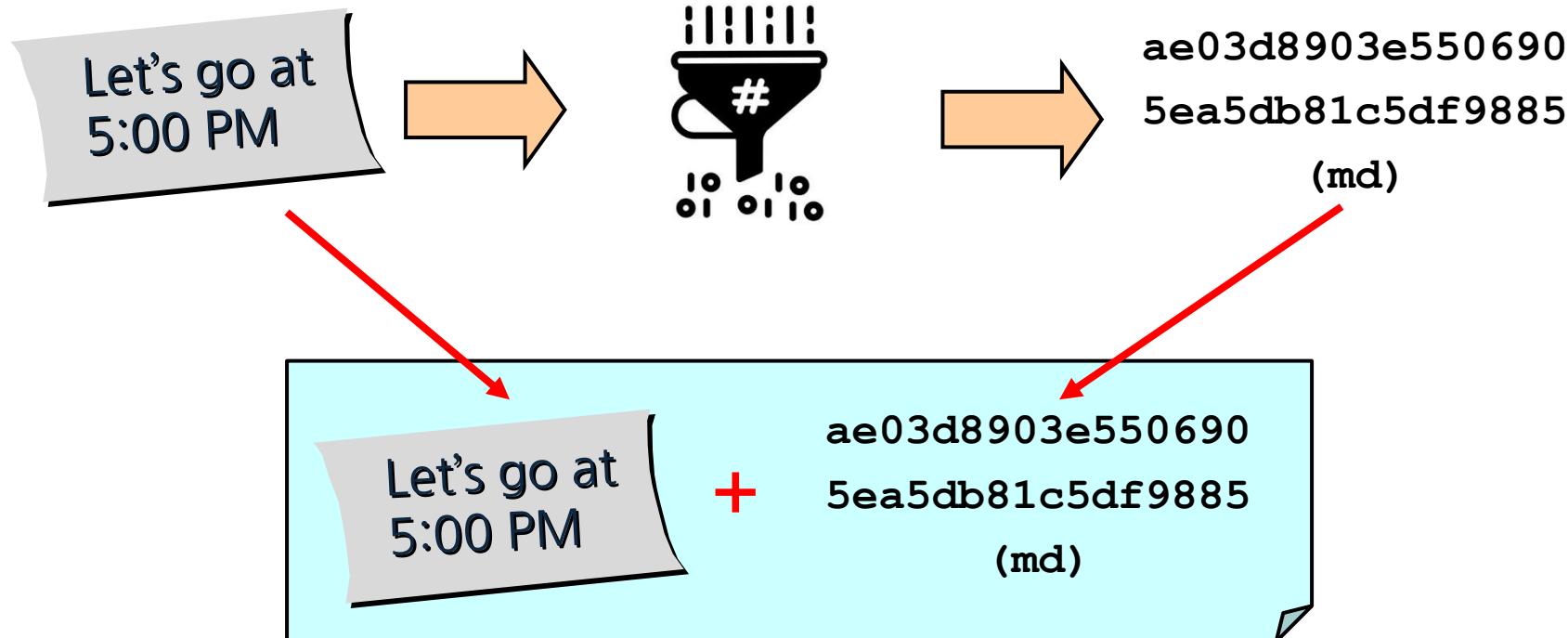
(주) 해쉬 함수 동작 절차(알고리즘)은 공개(public)

# 해쉬 함수 목적

- **정보 무결성(integrity) 확인**
  - 정보의 변경 여부 확인용
  - Message digest, Integrity Checking Value(ICV)
- **비밀키와 함께 사용하는 경우 메시지 출처 인증  
(data origin authentication) 기능 제공**
  - 메시지 다이제스트를 생성할 때 입력으로 메시지와 함께 송신자와 수신자가 공유하는 비밀키를 입력으로 이용

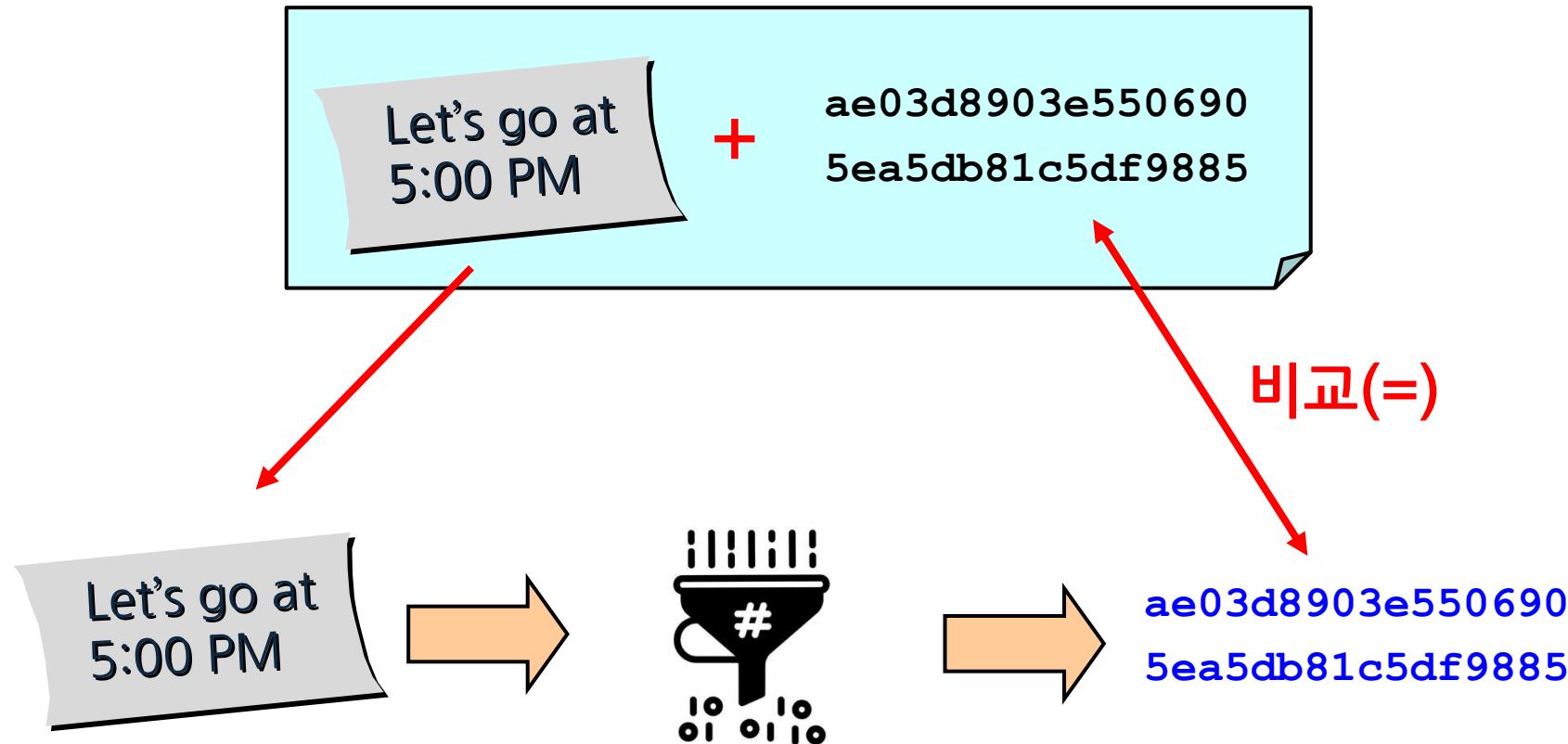
# 해쉬 함수 동작(1)

## ■ 송신자 측



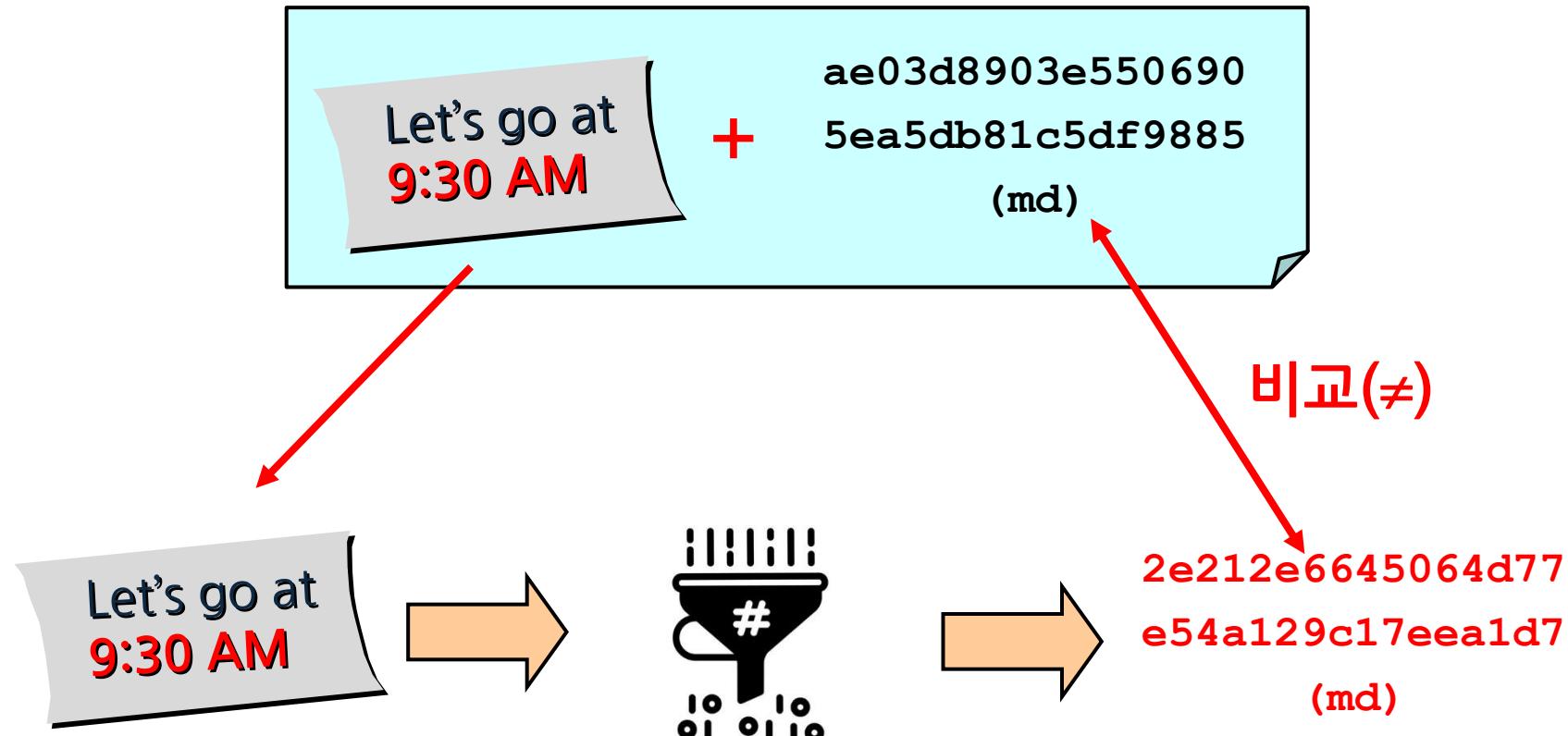
# 해쉬 함수 동작(2)

## ■ 수신자 측



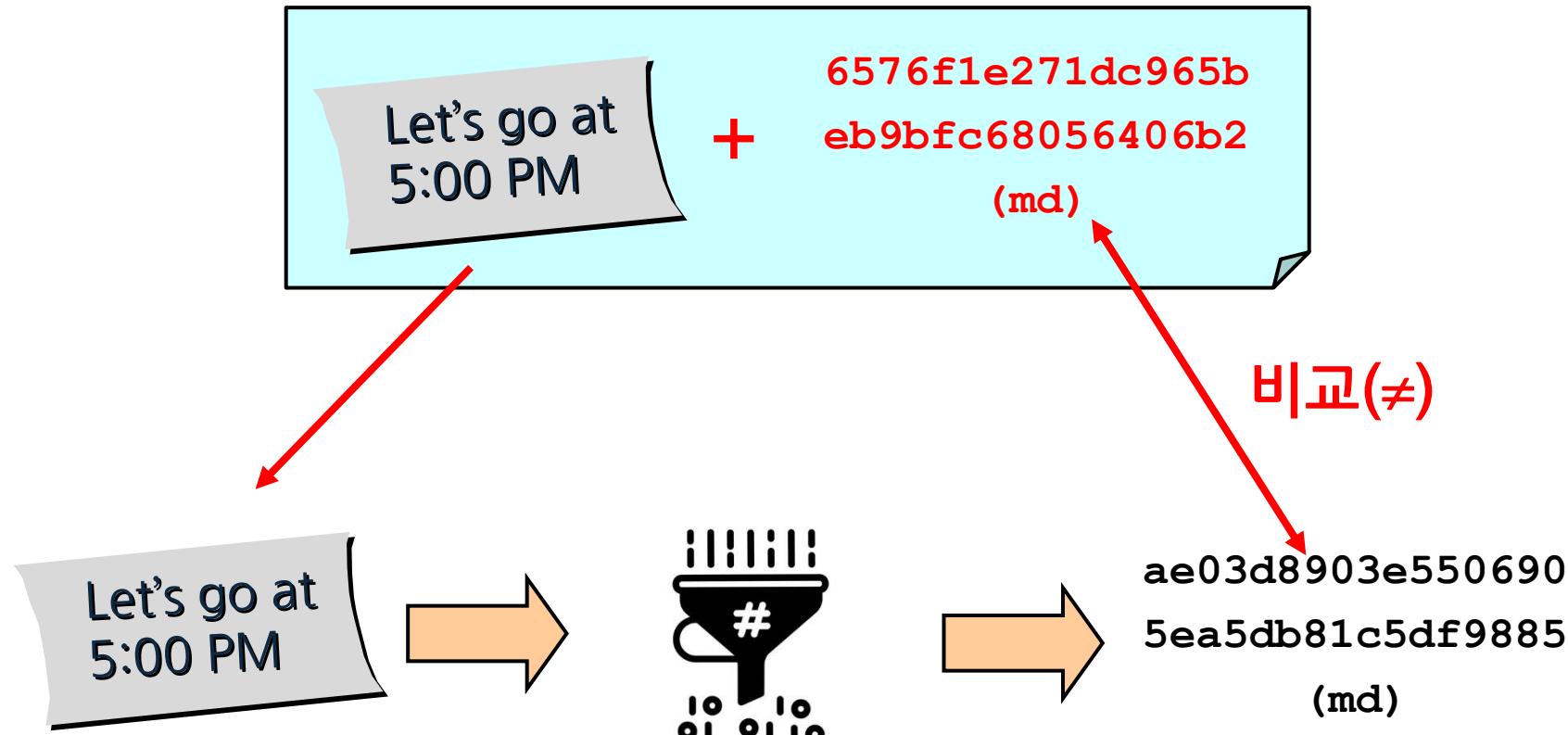
# 해쉬 함수 동작(3)

## ■ 수신자 측 (전송 중 메시지 변경)



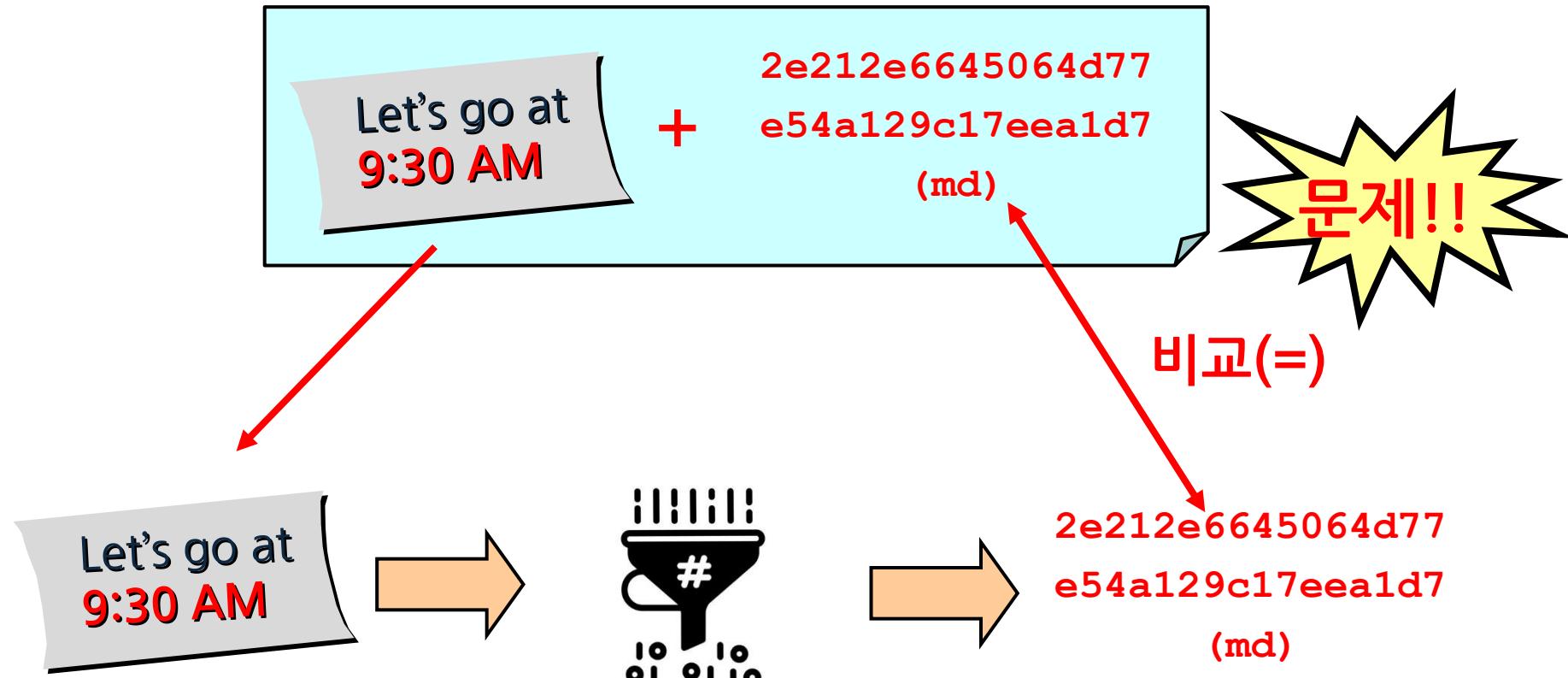
# 해쉬 함수 동작(4)

- 수신자 측 (전송 중 메시지 다이제스트 변경)



# 해쉬 함수 동작(5)

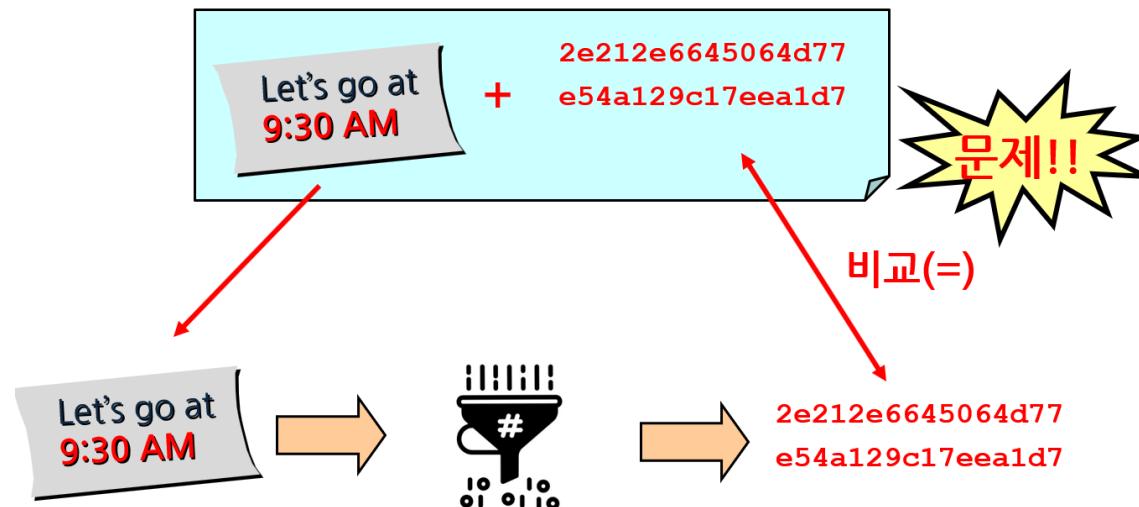
- 수신자 측 (전송 중 메시지, 메시지 다이제스트 변경)



# 해쉬 함수 동작(6)

## ■ 전송 중인 메시지 변경 후 메시지 다이제스트 생성

- 해쉬 함수의 동작절차(알고리즘)는 공개되어 있어 누구나 사용 가능
  - ✓ 메시지의 변경 여부 확인 불가능 (메시지 다이제스트 검증 성공 문제)
  - ✓ 메시지의 송신자 확인 불가능 (제3자에 의한 메시지 위조 가능성)

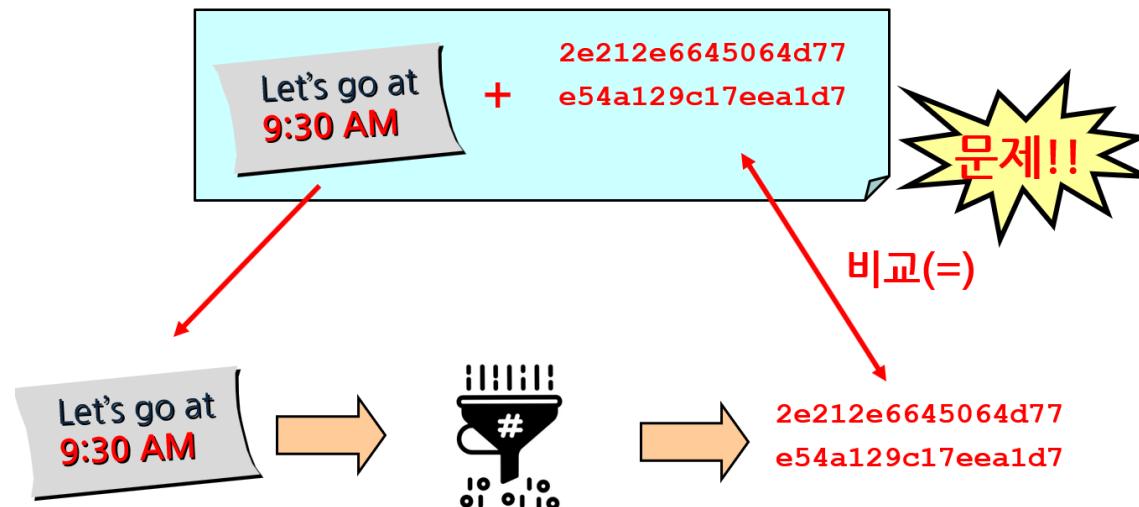


# 해쉬 함수 동작(7)

## ■ 전송 중인 메시지 변경 후 메시지 다이제스트 생성

- 해결 방안

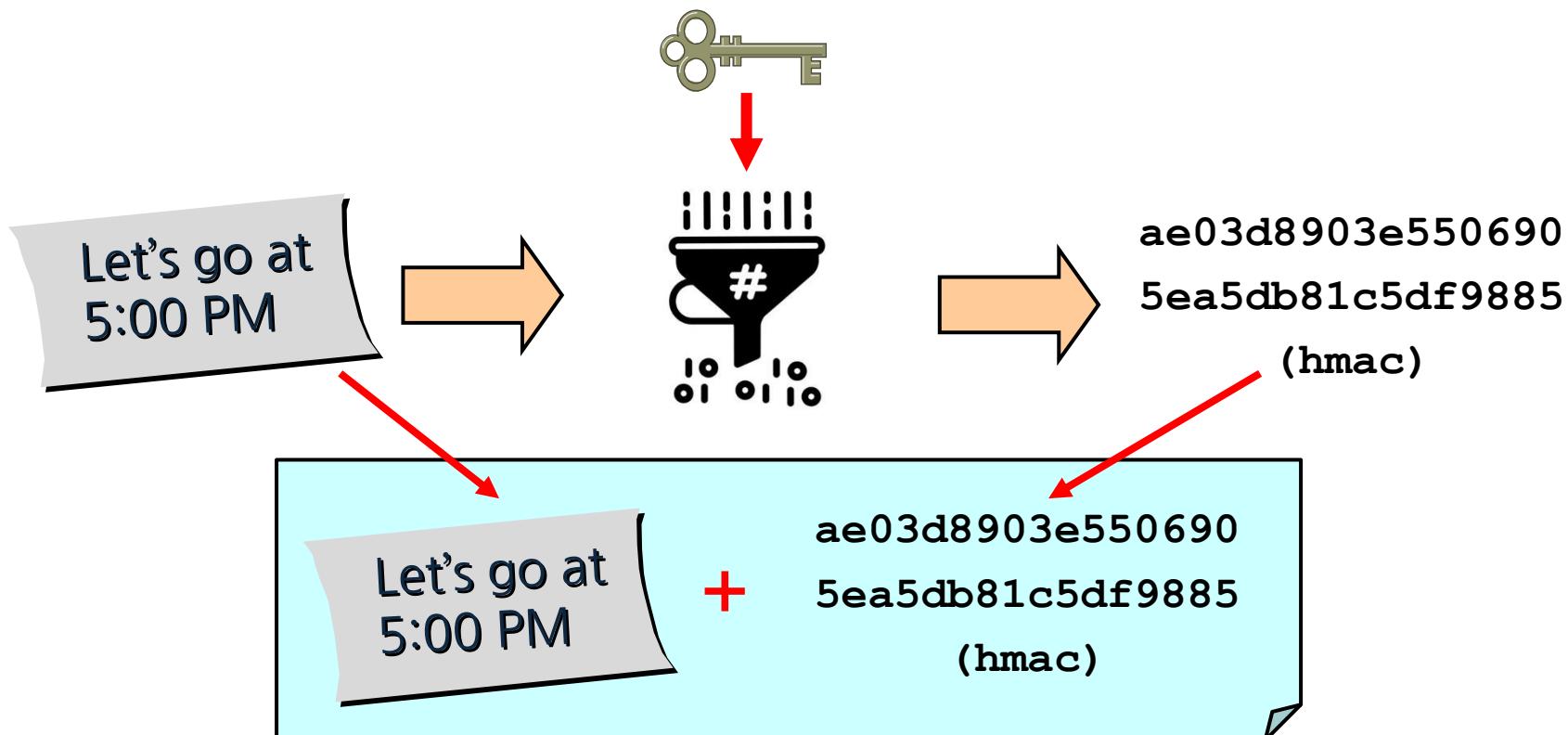
- ✓ 송신자와 수신자만이 알고 있는 비밀 정보 활용
- ✓ 제3자에 의한 메시지 위조, 변조 방지 가능



# HMAC(해쉬 기반 메시지 인증 코드) (1)

## ■ 송신자 측

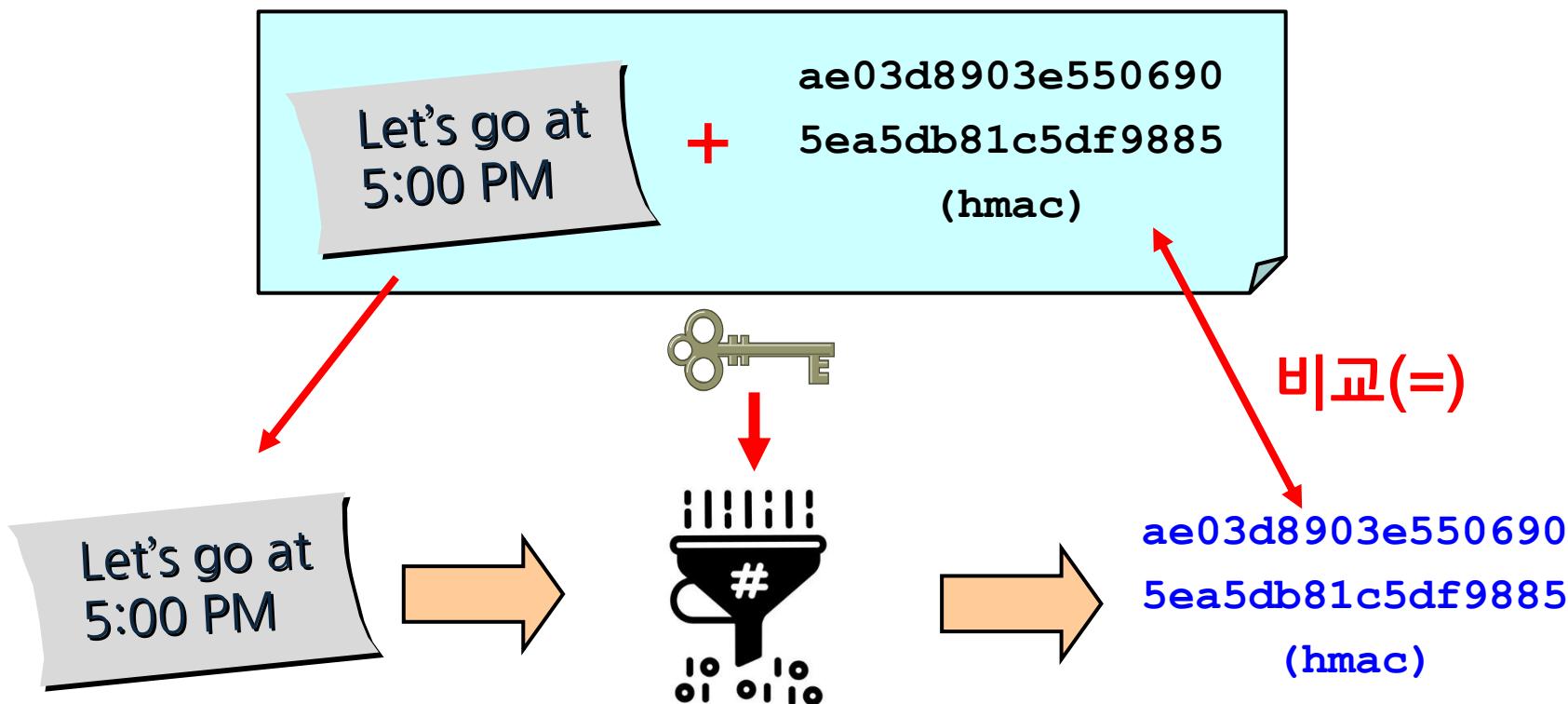
- 해쉬 함수에 송신자와 수신자만 공유하는 비밀키 입력



# HMAC(해쉬 기반 메시지 인증 코드) (2)

## ■ 수신자 측

- 검증 절차 시 송신자와 수신자만 공유하는 비밀키 입력



# HMAC(해쉬 기반 메시지 인증 코드) (3)

- 송신자, 수신자만 공유하는 대칭키 -  $k$ 
  - 사전에 안전한 방법으로 대칭키를 공유한다고 가정
- 송신자
  - $h(M, k) = \text{HMAC}$  (Hash-based Message Authentication Code)
- 수신자
  - $(M, \text{HMAC})$  수신
  - $h(M, k) = \text{HMAC}'$  계산
  - 수신된 **HMAC** 과 계산된 **HMAC'** 비교

# HMAC(해쉬 기반 메시지 인증 코드) (4)

## ■ HMAC 검증 성공 시 알 수 있는 사실

- 송신 과정 중 메시지 변경 X
  - ✓ 근거: 메시지 무결성 검증 (data integrity) 성공
- 송신 과정 중 HMAC 변경 X
  - ✓ 근거: 메시지 무결성 검증 (data integrity) 성공
- 메시지를 보낸 사람은 대칭키  $k$ 를 사전에 공유한 사용자
  - ✓ 메시지 출처 인증 (data origin authentication) 가능
  - ✓ 근거: 송신자와 수신자만 공유하는 비밀키 이용
  - ✓ 비밀키가 제3자에 유출된 경우 메시지 무결성 및 출처 인증 불가능

# 해쉬 함수 종류

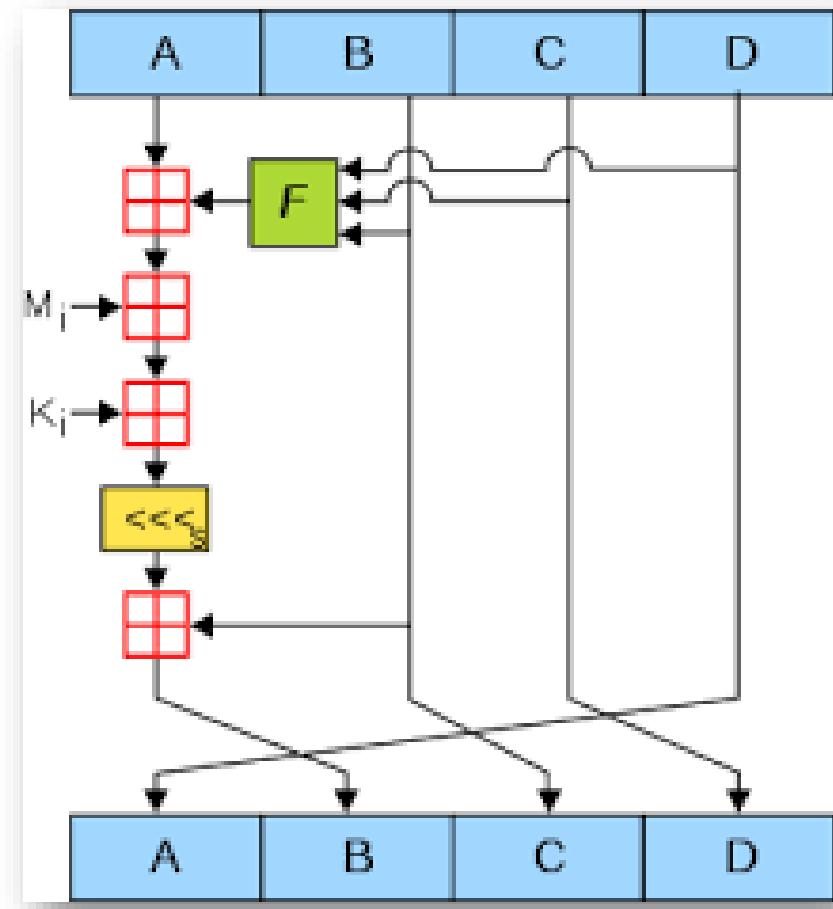
## ■ 종 류

- MD4(128-bit), **MD5**(128-bit)
  - ✓ RSA Data Security, Inc
- **SHA-1**(160-bit)
  - ✓ NIST
- **SHA-256**(256-bit), **SHA-512**(512-bit)
- RIPEMD-160(160-bit), RIPEMD-320(320-bit)
  - ✓ Europe Project RIPE

# MD5 해쉬 함수 (1)

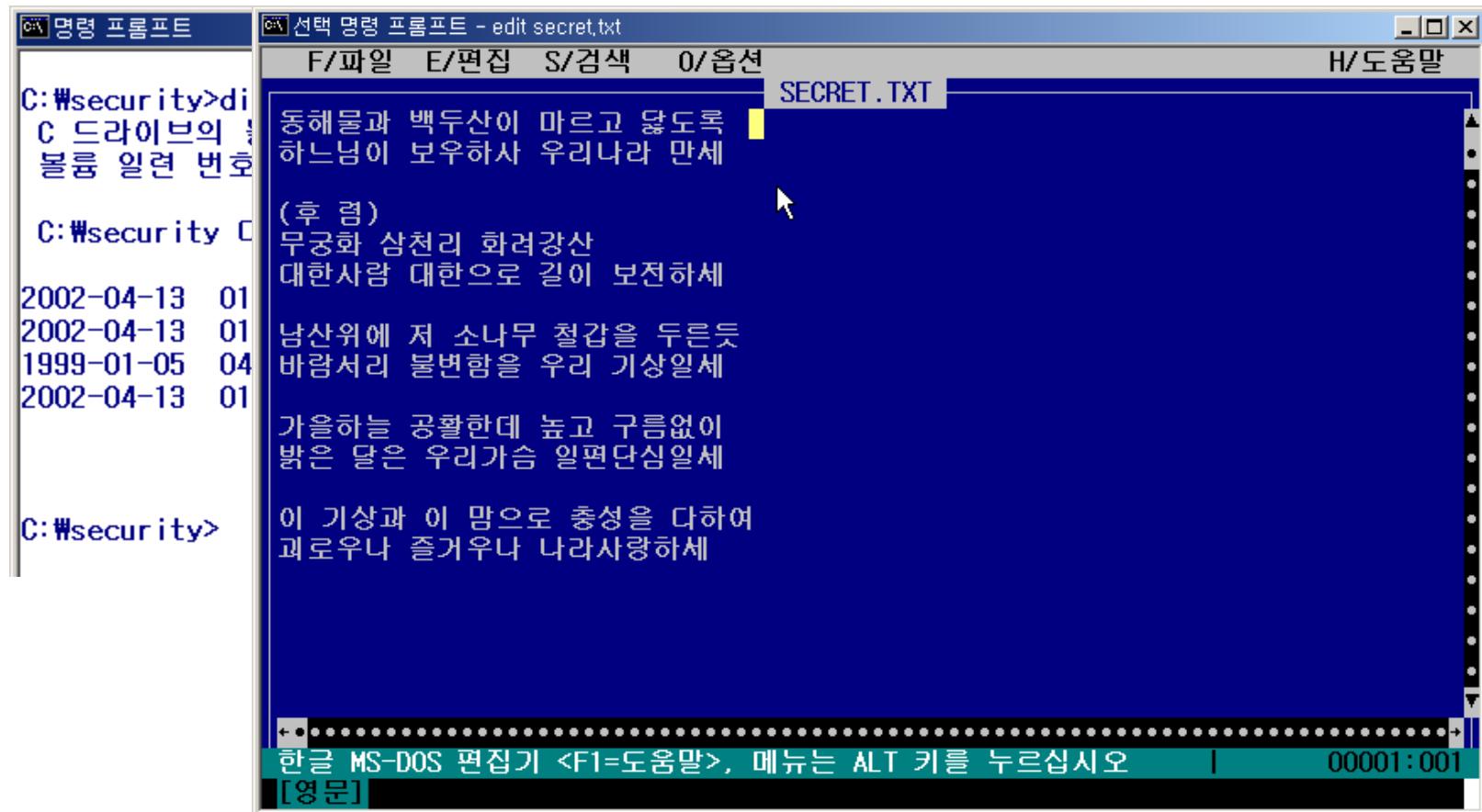
## ■ 기능

- 파일 또는 데이터에 대하여  
128-bit(16 바이트) md 생성  
**(정보의 무결성)**
- 보안 취약점으로 현재  
사용되지 않음



# MD5 해쉬 함수 (2)

## ■ 실행 예



# MD5 해쉬 함수 (3)

## ■ 실행 예 (만세 → 만새로 변경)

The screenshot shows a Windows command prompt window titled "명령 프롬프트 - edit secret.txt". The file "SECRET.TXT" contains the Korean phrase "동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세". The MD5 hash of this text is displayed in red as "B40E52A885B0D37AD6A317C1D42A78B0". Below this, another instance of the text is shown with a different MD5 hash in red, "E58A11ED6DD641C4B86E125B82B11A04". At the bottom, a command prompt shows the command "md5 secret.txt" followed by the same red hash value.

```
C:\#security>md5 secret.txt  
E58A11ED6DD641C4B86E125B82B11A04
```

# CertUtil - 해쉬값 계산 (Windows 환경)

- CertUtil -hashfile <입력 파일명> <해쉬 알고리즘>
  - 해쉬 알고리즘: md5, sha1, sha256, sha512

```
명령 프롬프트
C:\Users\Hlee>certutil -hashfile test.db md5
MD5의 test.db 해시:
a57797257a9fe1110c149cd7840bb58f
CertUtil: -hashfile 명령이 성공적으로 완료되었습니다.

C:\Users\Hlee>certutil -hashfile test.db sha1
SHA1의 test.db 해시:
4c34fa222709cb88c86b972f55542969a767c30a
CertUtil: -hashfile 명령이 성공적으로 완료되었습니다.

C:\Users\Hlee>certutil -hashfile test.db sha256
SHA256의 test.db 해시:
d11ddd9c8ef44e37508c895f67e8cff4876c58c44c9c5ec26a1ea5fb011873c6
CertUtil: -hashfile 명령이 성공적으로 완료되었습니다.

C:\Users\Hlee>certutil -hashfile test.db sha512
SHA512의 test.db 해시:
17809b94a3edeb0c594a4c651b64f1615833830dd129cedef0e0a840b0941f7111285f6ba3e5981572545193268b118ca148ed5ef562a0d98b3bf403
41f9c092
CertUtil: -hashfile 명령이 성공적으로 완료되었습니다.
```

# 해쉬함수와 비밀번호 저장 (1)

## ■ 비밀번호 저장 방식

### 평문 형식

```
alice : 123456  
bob : qwerty
```

### 암호문 형식

```
alice:@d$aDa#4hjko  
bob:>d+=fdQxF&DV
```

### 해쉬값 형식

```
alice : e10adc3949ba59abbe56e057f20f883e  
bob : d8578edf8458ce06fb5bb76a58c5ca4
```

# 해쉬함수와 비밀번호 저장 (2)

## ■ 비밀번호 저장 방식 - (1) 평문 형식 저장

- 비밀성 보장 X
- 현실성 X

# 해쉬함수와 비밀번호 저장 (3)

## ■ 비밀번호 저장 방식 - (2) 암호문 형식

- 대칭키 암호시스템을 활용하여 암호화된 비밀번호 저장
- Plain text 저장 방식보다는 안전
- 모든 사용자의 비밀번호를 하나의 대칭키로 암호화
- (위험) 대칭키 유출 시 모든 사용자의 비밀번호 노출 위험
- (위험) 안전하지 않은 대칭키 사용 시 **brute force 공격** 및 **사전(dictionary) 공격**에 취약

# 해쉬함수와 비밀번호 저장 (4)

## ■ 비밀번호 저장 방식 - (3) 해쉬값 형식

- 해쉬함수를 적용한 결과값(해쉬값) 저장
- 해쉬함수의 one-way 특성으로 비밀번호 추론 가능성이 낮음
- (위험) 안전하지 않은 비밀번호 사용 시 **brute force 공격** 및 **사전 공격**에 취약
- (위험) **rainbow table 공격**에 취약

# 해쉬함수와 비밀번호 저장 (5)

## ■ 비밀번호 저장 방식 - (3) 해쉬값 형식

### • rainbow 테이블 공격

- ✓ 해쉬값 계산에 소요되는 시간 절약을 위해
- ✓ 사용 가능성이 높은 (**비밀번호**, ①해쉬값) 쌍을 미리 계산하여 저장
- ✓ 획득한 **사용자의 비밀번호** ②해쉬값과 rainbow 테이블 해쉬값 비교
- ✓ 만일 일치하는 항목이 있을 경우(① = ②) 해당 사용자의 **비밀번호** 추론 가능
- ✓ (문제점) rainbow 테이블 저장에 필요한 대규모 기억 장치 소요

# 해쉬함수와 비밀번호 저장 (6)

- 비밀번호 저장 방식 - (3') 해쉬값, Salt 저장
  - (문제점) 동일한 비밀번호에 대한 동일한 해쉬값 생성

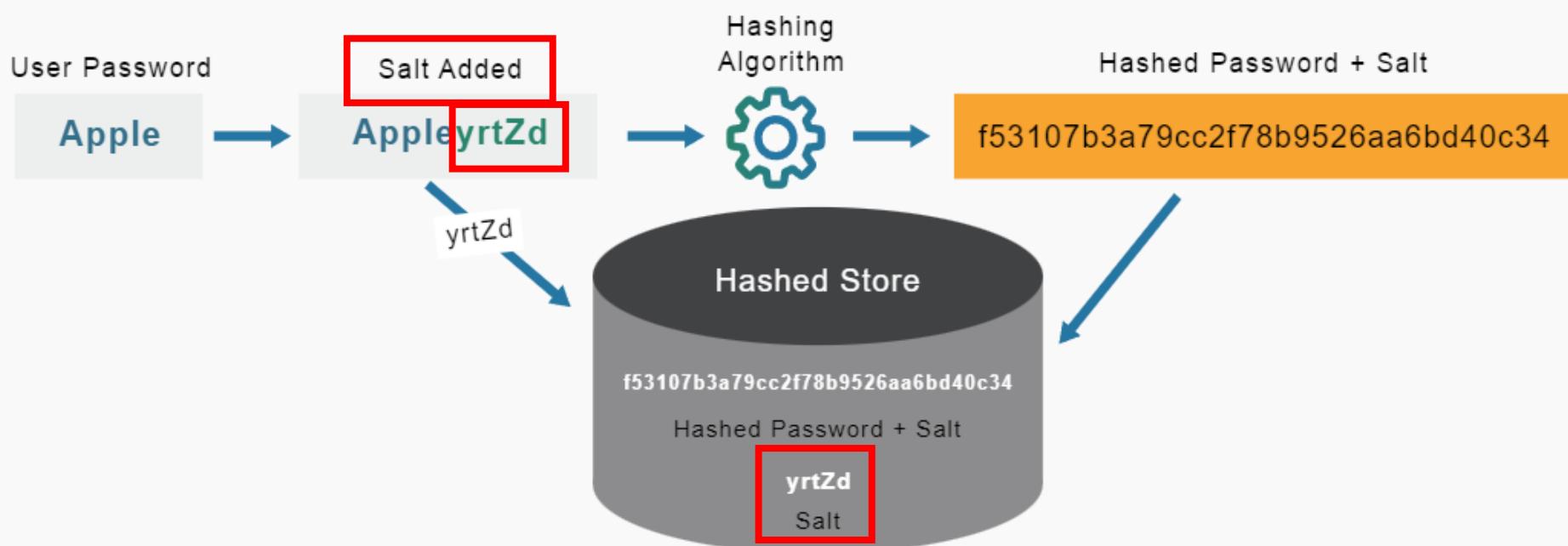
- ✓ 서로 다른 사용자가 동일한 비밀번호를 사용하는 경우 해쉬값도 동일
  - ✓ 해쉬값이 같으면 다른 사용자의 비밀번호 추론 가능

```
alice : e10adc3949ba59abbe56e057f20f883e
bob   : d8578edf8458ce06fbc5bb76a58c5ca4
tom   : 1bef85b3f0aab14873ae81f2a1bc6a33
park  : e10adc3949ba59abbe56e057f20f883e
```

- (대안) 해쉬값 계산 시 사용자마다 다른 정보 추가 - Salt

# 해쉬함수와 비밀번호 저장 (7)

- 비밀번호 저장 방식 - (3') 해쉬값, Salt 저장
  - Salt를 이용하는 해쉬 함수 동작



# 해쉬함수와 비밀번호 저장 (8)

- 비밀번호 저장 방식 - (3') 해쉬값, Salt 저장
  - /etc/shadow 파일 내 저장 Salt 정보

```
hlee@hlee-PRIME-Win11Pro:/mnt/c/Users/Hlee$ tail /etc/passwd
systemd-timesync:x:996:996:systemd Time Synchronization:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
syslog:x:102:102::/nonexistent:/usr/sbin/nologin
systemd-resolve:x:991:991:systemd Resolver:/usr/sbin/nologin
uuidd:x:103:103::/run/uuidd:/usr/sbin/nologin
Landscape:x:104:105::/var/lib/landscape:/usr/sbin/nologin
polkitd:x:990:990:User for polkitd:/usr/sbin/nologin
ubuntu:x:1000:1001:Ubuntu:/home/ubuntu:/bin/bash
hlee:x:1002:1002,,,,:/home/hlee:/bin/bash
hlee@hlee-PRIME-Win11Pro:/mnt/c/Users/Hlee$ tail /etc/shadow
tail: cannot open '/etc/shadow' for reading: Permission denied
hlee@hlee-PRIME-Win11Pro:/mnt/c/Users/Hlee$ sudo tail /etc/shadow
systemd-timesync:!*:19836:::::
dhcpcd:!*:19836:::::
messagebus:!:19836:::::
```

해  
시  
알고  
리즘

Salt

(비밀번호 + Salt) 해쉬값

\$y\$j9T\$Rm69V.Z7qJEjIdbp9Nw3q1\$HxjoQ4GlIXKooshjPizsafxz2Dmm2yX0B/Nt1Fp2kH0

```
polkitd:!*:19836:::::
ubuntu:!:19975:0:99999:7...
hlee:$y$j9T$Rm69V.Z7qJEjIdbp9Nw3q1$HxjoQ4GlIXKooshjPizsafxz2Dmm2yX0B/Nt1Fp2kH0 19975:0:99999:7:::
hlee@hlee-PRIME-Win11Pro:/mnt/c/Users/Hlee$ |
```

# 전자 서명

# (Digital Signature)

# 전자 서명(Digital Signature)((1))

## ■ 정의

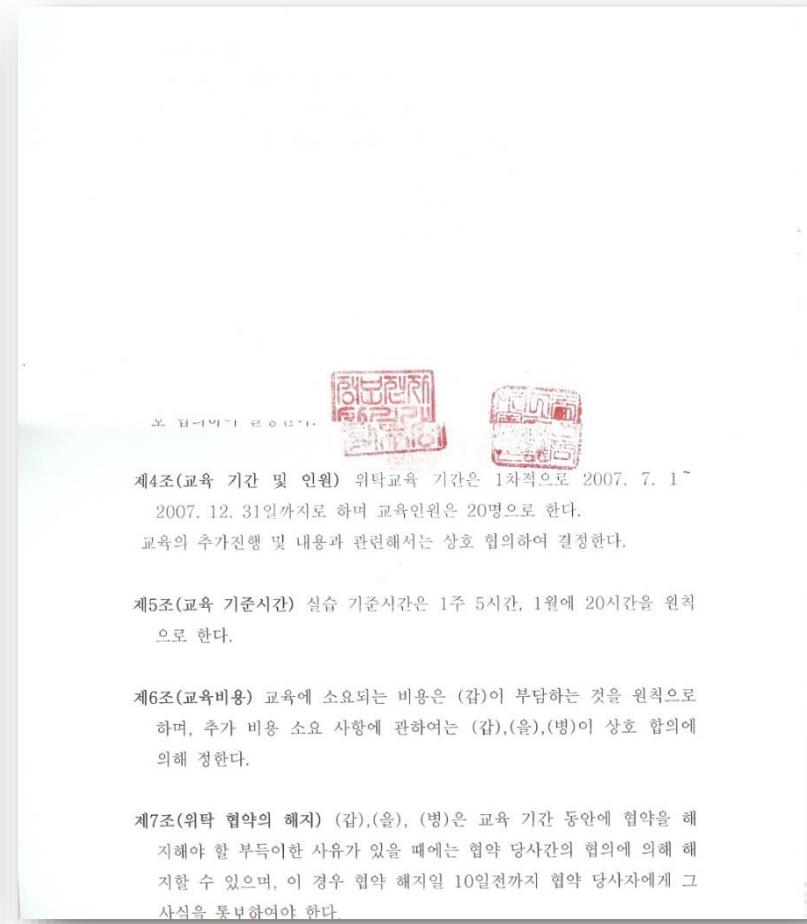
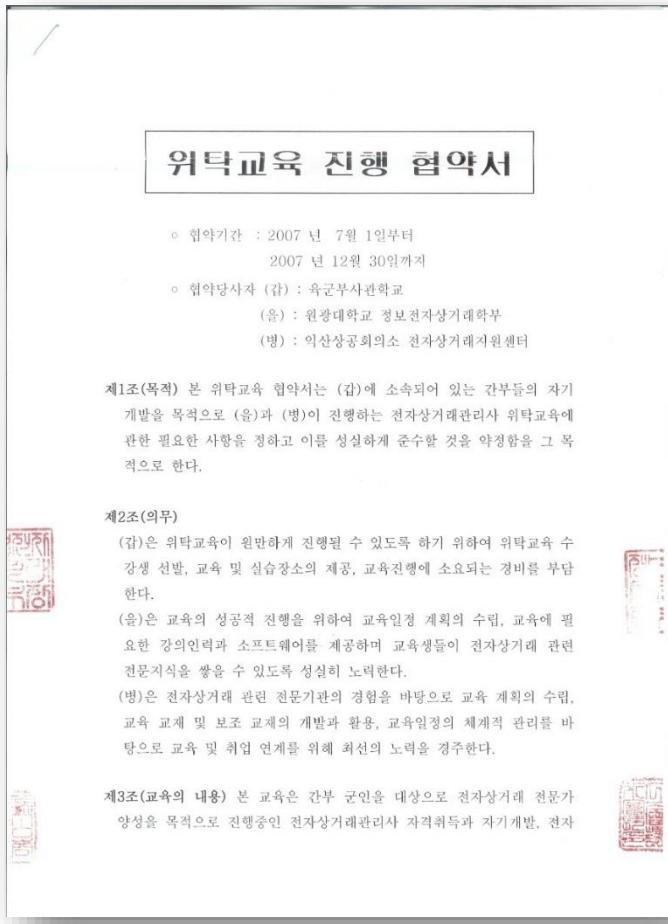
- 전자문서를 **작성자의 신원**과
- 전자문서의 **변경여부를 확인**할 수 있도록
- **전자서명 생성키(작성자의 개인키)**로 생성한
- 전자문서에 대한 **작성자의 고유한 정보**

(비유) 온라인 상의 자필 서명



# 전자 서명(Digital Signature)(2)

## ■ 현실 세계의 서명



# 전자 서명(Digital Signature)(3)

## ■ 목 적

- **서명자 신원 확인**(Authentication)

- ✓ 서명자 확인

- **위조 불가**(Unforgeable)

- ✓ 서명자만이 서명 가능

- **변조 불가**(Unalterable)

- ✓ 내용 수정 불가능

- **재사용 금지**(Unreusable)

- ✓ 다른 문서에 재사용 불가

- **부인 봉쇄**(Non-repudiation)

- ✓ 서명사실 부인 불가능

# 전자 서명(Digital Signature)(4)

## ■ 원리

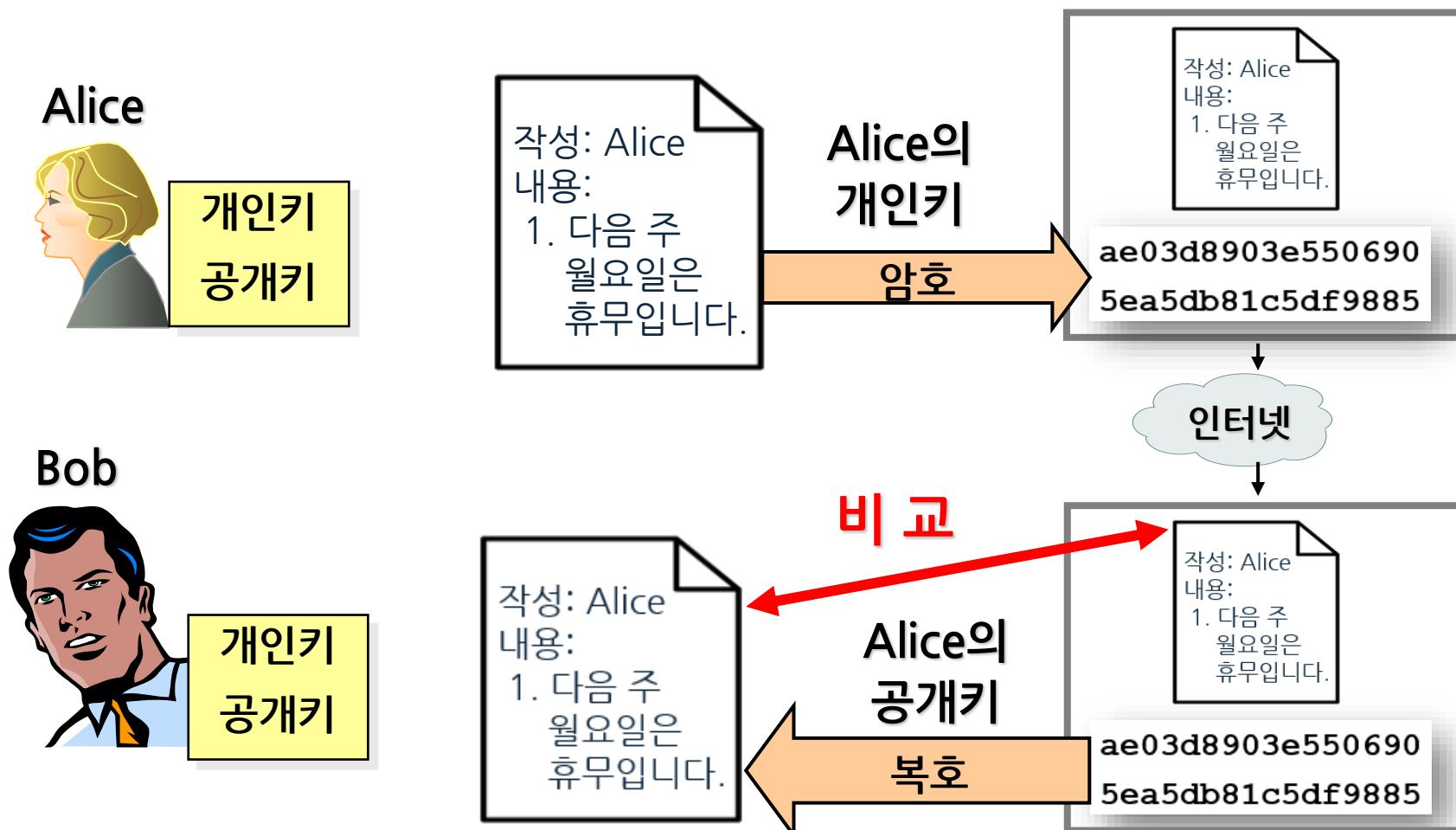
- 전자 서명 생성키와 전자 서명 검증키를 이용해 데이터를 암호화, 복호화하는 기술
- 데이터를 전자서명 생성키(작성자의 개인키)로 암호화하면
- 전자서명 생성키와 짹이 되는 전자서명 검증키(작성자의 공개키)에 의해서만 복호화되어 원래의 데이터를 얻을 수 있음

# 전자 서명(Digital Signature)(5)

- 전자 서명 **생성키**
  - 전자 서명자(송신자)의 개인키
- 전자 서명 **검증키**
  - 전자 서명자(송신자)의 공개키
- **공개키 암호시스템**의 **인증모드** 절차와 동일
  - 메시지 무결성 보장 목적

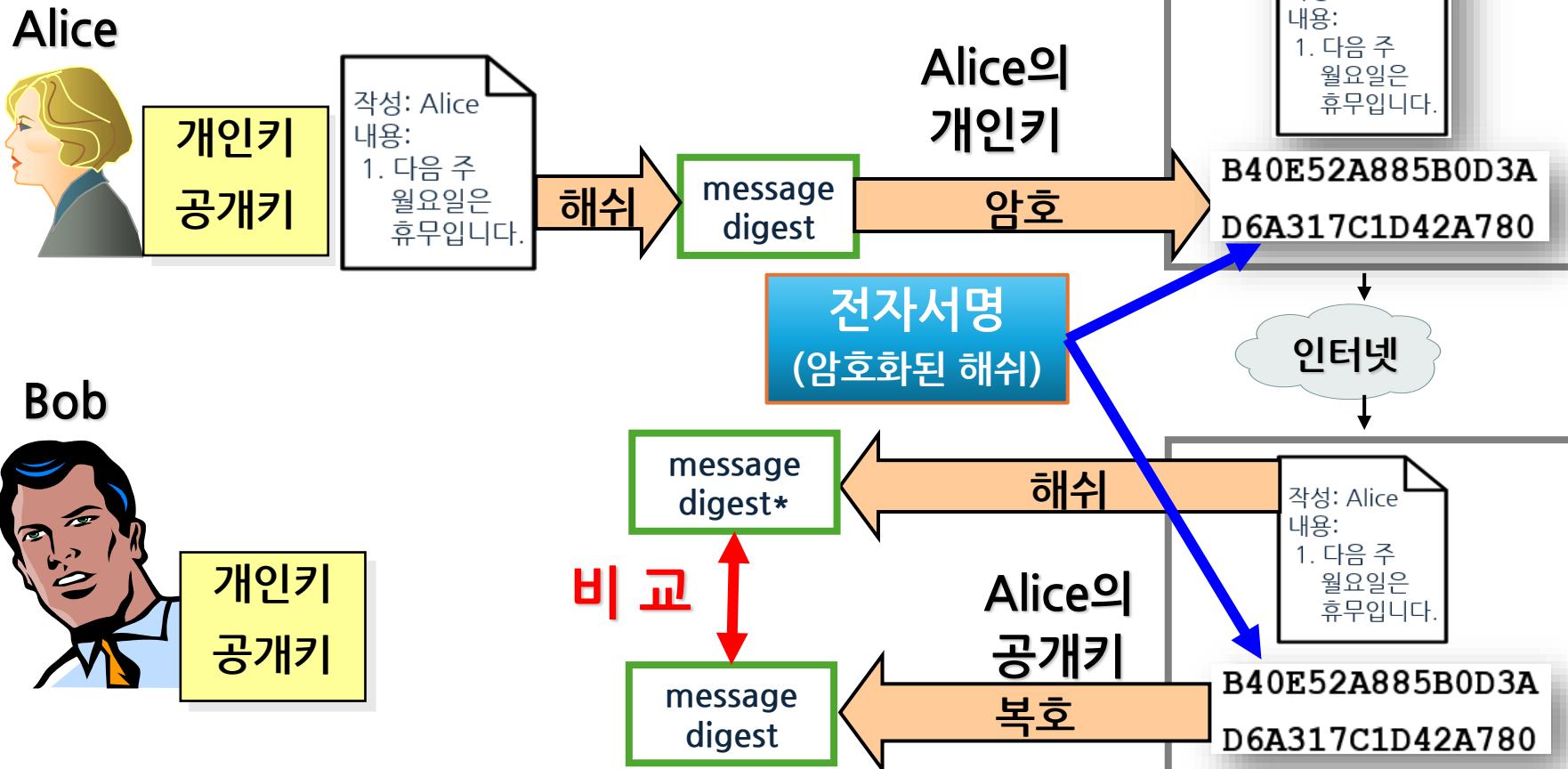
# 전자 서명(Digital Signature)(6)

## ■ 전자 서명 생성 및 검증 절차



# 전자 서명(Digital Signature)(7)

- 전자 서명 생성 및 검증 절차 (해쉬 함수 적용)



# 전자 서명(Digital Signature)(8)

## ■ 전자서명 검증 성공 시 확인할 수 있는 사실

전자서명 목적	질문	근거
서명자 신원 확인	Q1. Alice가 수신된 메시지를 작성한 것이 맞는가?	
위조 불가 (Unforgeable)	Q2. 제3자가 메시지를 작성하여 Alice가 작성한 것처럼 속일 수 있을까?	
변조 불가 (Unalterable)	Q3. Alice가 작성한 메시지를 제3자가 수정하여 보내고 검증 절차를 마칠 수 있을까?	
재사용 금지 (Unreusable)	Q4. 제3자가 현재 메시지의 전자서명을 다른 메시지와 함께 보내 검증 절차를 마칠 수 있을까?	
부인 봉쇄 (Non-repudiation)	Q5. Alice가 나중에 자신이 이 메시지를 보낸 사실이 없다고 부인할 수 있을까?	

# Q & A