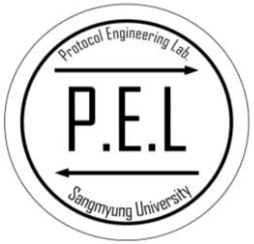


네트워크 보안 에센셜

4장 키 분배와 사용자 인증

Ji-Yeon Moon (jiyeon@pel.smuc.ac.kr)
Protocol **E**ngineering **L**ab., **Sangmyung** University

Content

대칭 암호를 이용한 대칭키 분배

- **대칭키 암호**
  - 메시지를 주고받는 양쪽이 반드시 동일한 키를 공유해야함
  - 키에는 반드시 다른 사람이 접근할 수 없도록 해야함

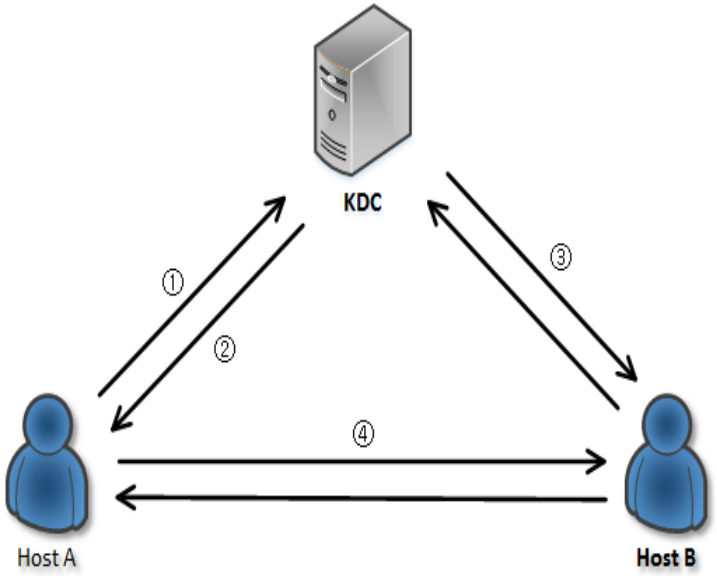
- **키 분배 기술**
  - 다른 사람이 키를 보지 못하게 함
  - 데이터를 교환하고자 하는 쌍방에게 전달하는 수단

대칭 암호를 이용한 대칭키 분배

- **키 분배 센터를 이용한 대칭키 분배 방법**
  - **키 분배 센터(KDC: Key Distribution Center)**
    - 어떤 시스템이 각각 통신을 할 권한이 있는지를 결정
    - 연결할 필요가 있는 두 시스템이 적법하다고 판단하면 세션키를 양쪽 개체에게 제공
  - **세션키(Session Key)**
    - 두 개의 종단 시스템이 통신하기 위한 논리적 연결이 구성되어야 함
    - 세션이 유지되는 동안 모든 사용자 데이터는 세션키로 암호화
    - 세션이 종료되면 세션키는 폐기
  - **영구 키(Permanent Key)**
    - 개체에게 세션키를 분배하기 위해 필요한 키

대칭 암호를 이용한 대칭키 분배

- **키 분배 센터를 이용한 대칭키 분배 방법**



① Host A가 KDC에게 연결 요청

- Host A와 KDC가 공유하고 있는 비밀키를 이용해서 암호화

② KDC가 연결 요청 허락 및 세션키 전달

- Host A와 KDC가 공유하고 있는 비밀키를 이용해서 암호화

③ KDC가 Host B에게 연결을 요청하며 이에 동의하면 세션키를 전달

- Host B와 KDC가 공유하고 있는 비밀키를 이용해서 암호화

④ Host A와 Host B가 통신

- 세션키를 이용하여 메시지를 암호화

대칭 암호를 이용한 대칭키 분배

- **키 분배 센터를 이용한 대칭키 분배 방법**
  - **문제점: 보관해야 할 키의 개수**
    - 클라이언트가 증가함에 따라 KDC에서 보관해야 할 키의 개수 또한 증가
    - 클라이언트가 n명일 때, 키의 개수는 n(n+1)/2
    - 따라서 KDC가 공격 당한다면 모든 사용자의 키가 노출

KERBEROS

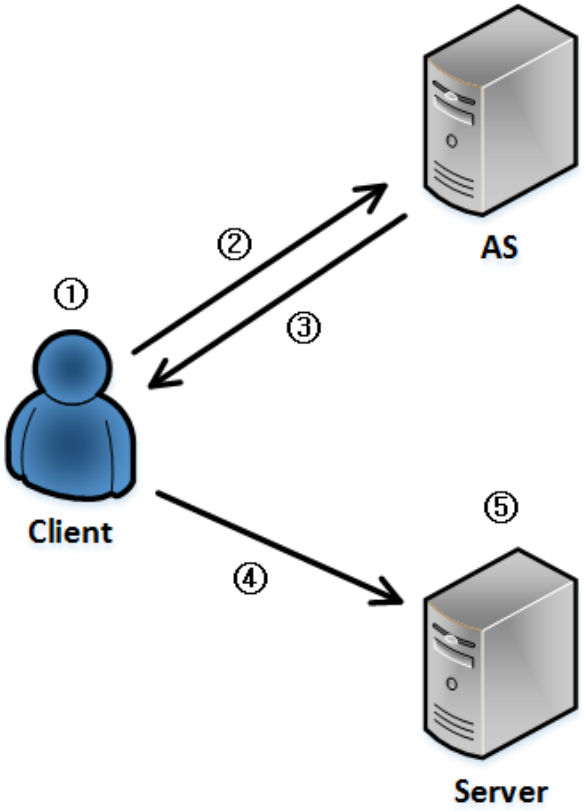
- **개요**
  - MIT에서 개발한 대칭키 방식의 키 분배 및 사용자 인증 서비스
  - 사용자가 어떤 서비스 또는 서버에 접속하기 위해서 사용자를 인증하고 네트워크 상의 서버에게 사용자의 신분을 보증
  - version 4와 version 5가 존재하며, version 4의 보안상 허점을 보완하여 version 5가 인터넷 표준으로 제안

KERBEROS

- **단순 인증 프로토콜**
  - **용어 설명**
    - C = 클라이언트
    - AS = 인증 서버 (AS: Authentication Server)
      - 모든 사용자의 패스워드를 데이터베이스에 저장
      - 각 서버와 비밀키를 공유
    - V = 서버
    - ID_C = 클라이언트의 식별자
    - ID_V = 서버의 식별자
    - PW_C = 클라이언트의 패스워드
    - AD_C = 클라이언트의 IP 주소
    - **K_{AS-V} = 인증 서버와 서버가 공유하는 비밀키**
    - || = 이어 붙이기

KERBEROS

- 단순 인증 프로토콜
  - 동작 과정



① 사용자 로그인

② $ID_C||ID_V||PW_C$ 전송

③ ID_C와 PW_C 그리고 Server 접근 여부 확인 후,

$Ticket = E(K_{AS-V}, [ID_C||AD_C||ID_V])$ 전송

④ $ID_C||Ticket$ 전송

⑤ $Ticket$을 복호화하여 ID_C를 비교한 후 서비스 승인

KERBEROS

- **단순 인증 프로토콜**
  - **문제점**
    1. 사용자가 입력해야할 패스워드의 횟수
       - 새로운 서비스를 이용할 때마다 새로운 티켓을 발행받기 위해서 패스워드를 입력해야함

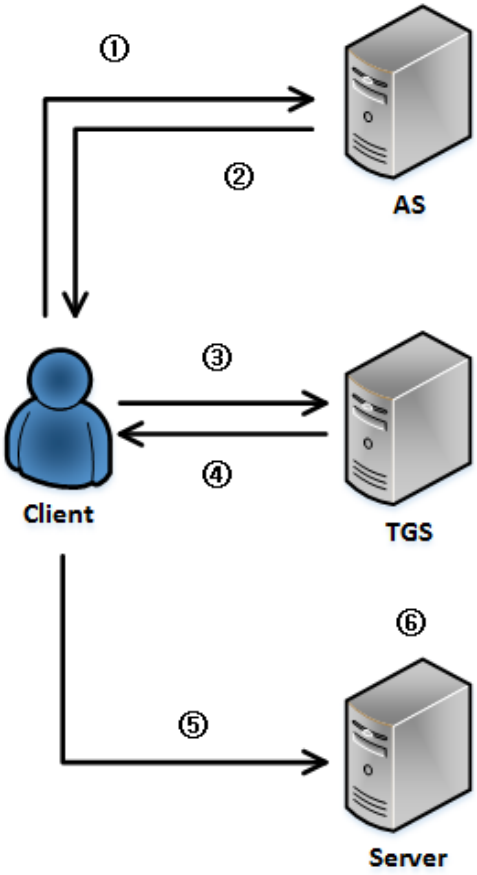
    2. 평문으로 전달되는 패스워드
       - 공격자는 패스워드를 가로채서 해당 사용자에게 허용된 모든 서비스를 이용

KERBEROS

- **TGS를 도입한 인증 프로토콜**
  - **용어 설명**
    - C = 클라이언트
    - AS = 인증 서버 (AS: Authentication Server)
    - TGS = 티켓 발행 서버 (TGS: Ticket-granting Server)
    - V = 서버
    - ID_C = 클라이언트의 식별자
    - ID_V = 서버의 식별자
    - ID_{TGS} = 티켓 발행 서버의 식별자
    - PW_C = 클라이언트의 패스워드
    - AD_C = 클라이언트의 IP 주소
    - **K_C = 클라이언트의 패스워드로부터 얻은 키**
    - **K_{AS-TGS} = 인증 서버와 티켓 발행 서버가 공유하는 비밀키**
    - **K_{TGS-V} = 티켓 발행 서버와 서버가 공유하는 비밀키**
    - TS = 타임스탬프
    - $Lifetime$ = 티켓 유효기간
    - || = 이어 붙이기

KERBEROS

- **TGS를 도입한 인증 프로토콜**
  - **동작 과정**



인증 서비스 교환: 티켓-발행 티켓을 취득하기

① $ID_C \parallel ID_{TGS}$ 전송

② $E(K_C,\ Ticket_{TGS})$ 전송

- $Ticket_{TGS} = E(K_{AS-TGS},\ [ID_C \parallel AD_c \parallel ID_{TGS} \parallel TS_1 \parallel Lifetime_1])$

티켓-발행 서비스 교환: 서비스-발행 티켓 취득하기

③ 사용자로부터 패스워드를 입력받은 후, K_C를 생성하여 메시지를 복호화하여 $ID_C \parallel ID_V \parallel Ticket_{TGS}$ 전송

④ $Ticket_{TGS}$을 복호화하여 ID_{TGS} 존재여부, 유효기간 만료여부, ID_C와 PW_C, Server 접근 여부를 확인한 후, $Ticket_V$ 전송

- $Ticket_V = E(K_{TGS-V},\ [ID_C \parallel AD_c \parallel ID_V \parallel TS_2 \parallel Lifetime_2])$

클라이언트/서버 인증 교환: 서비스 취득하기

⑤ $ID_V \parallel Ticket_V$ 전송

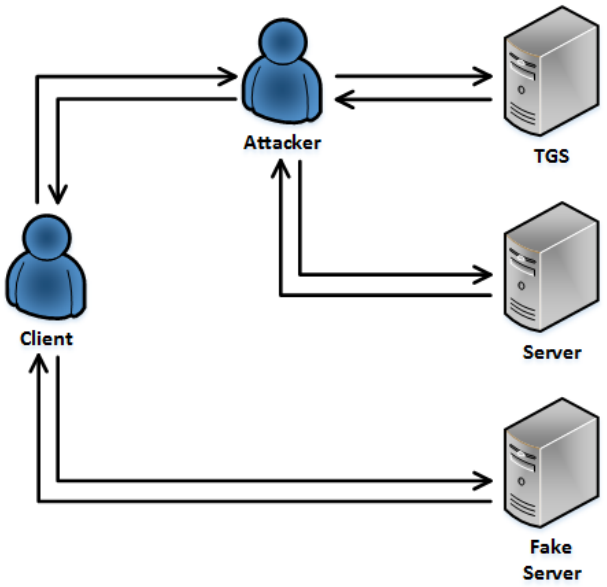
⑥ $Ticket_V$을 복호화하여 ID_V를 비교한 후 서비스 승인

KERBEROS

- **TGS를 도입한 인증 프로토콜**
  - **문제점**
    1. $Ticket$의 유효기간
       - 티켓의 유효기간이 짧으면, 사용자는 세션이 끝나지 않았음에도 불구하고 패스워드를 입력해야 하는 경우가 생김
       - 티켓의 유효기간이 길면, 사용자가 서비스 사용을 종료한 후에도 티켓은 사용이 가능하므로 공격자가 유효기간이 남아있는 티켓을 이용

KERBEROS

- **TGS를 도입한 인증 프로토콜**
  - **문제점**
    2. 서버 인증
       - 공격자가 서버의 식별자를 위조하여, 가짜 서버를 이용해 공격자는 서비스를 사용할 수 있으며 사용자는 서비스를 이용하고 있다고 착각 / 사용자가 서비스를 제공받지 못하도록 함

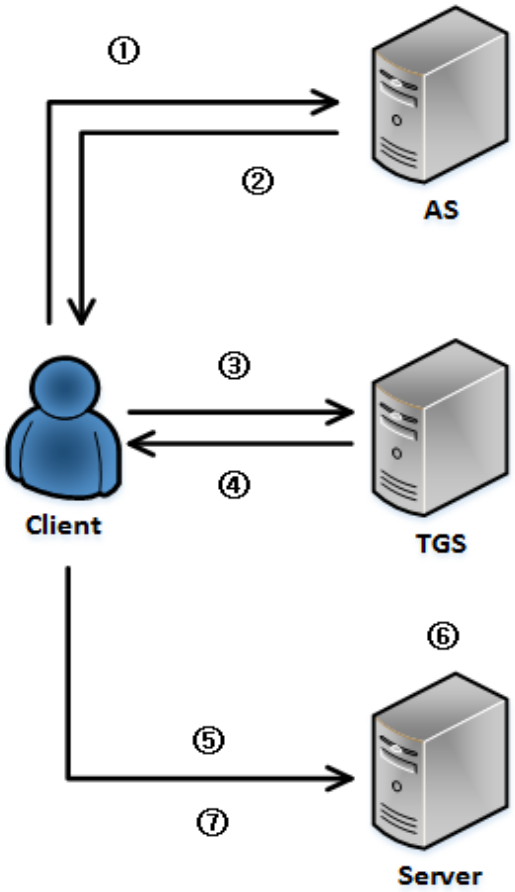


KERBEROS

- **Kerberos version 4 인증 프로토콜**
  - **용어 설명**
    - C = 클라이언트
    - AS = 인증 서버 (AS: Authentication Server)
    - TGS = 티켓 발행 서버 (TGS: Ticket-granting Server)
    - V = 서버
    - ID_C = 클라이언트의 식별자
    - ID_V = 서버의 식별자
    - ID_{TGS} = 티켓 발행 서버의 식별자
    - PW_C = 클라이언트의 패스워드
    - AD_C = 클라이언트의 IP 주소
    - **K_C = 클라이언트의 패스워드로부터 얻은 키**
    - **K_{C-TGS} = 클라이언트와 티켓 발행 서버 사이의 세션키**
    - **K_{C-V} = 클라이언트와 서버 사이의 세션키**
    - **K_{AS-TGS} = 인증 서버와 티켓 발행 서버가 공유하는 비밀키**
    - **K_{TGS-V} = 티켓 발행 서버와 서버가 공유하는 비밀키**
    - TS = 타임스탬프
    - $Lifetime$ = 티켓 유효기간
    - || = 이어 붙이기

KERBEROS

- **Kerberos version 4 인증 프로토콜**
  - **동작 과정**



인증 서비스 교환: 티켓-발행 티켓을 취득하기

① $ID_C \parallel ID_{TGS} \parallel TS_1$ 전송

② $E(K_C, [K_{C-TGS} \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{TGS}])$ 전송

- $Ticket_{TGS} = E(K_{AS-TGS}, [K_{C-TGS} \parallel ID_C \parallel AD_c \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

티켓-발행 서비스 교환: 서비스-발행 티켓 취득하기

③ 사용자로부터 패스워드를 입력받은 후, K_C를 생성하여 메시지를 복호화하여 $ID_V \parallel Ticket_{TGS} \parallel Authenticator_{C-TGS}$ 전송

- $Authenticator_{c-TGS} = E(K_{C-TGS}, [ID_C \parallel AD_c \parallel TS_3 \parallel Lifetime_3])$

④ $Ticket_{TGS}$과 $Authenticator_{C-TGS}$ 를 복호화한 뒤, ID_C와 AD_c를 일치하면 $E(K_{C-TGS}, [K_{C-V} \parallel ID_V \parallel TS_4 \parallel Ticket_V])$ 전송

- $Ticket_V = E(K_{TGS-V}, [K_{C-V} \parallel ID_C \parallel AD_c \parallel ID_V \parallel TS_4 \parallel Lifetime_4])$

클라이언트/서버 인증 교환: 서비스 취득하기

⑤ 메시지를 복호화하여 $Ticket_V \parallel Authenticator_{C-V}$ 전송

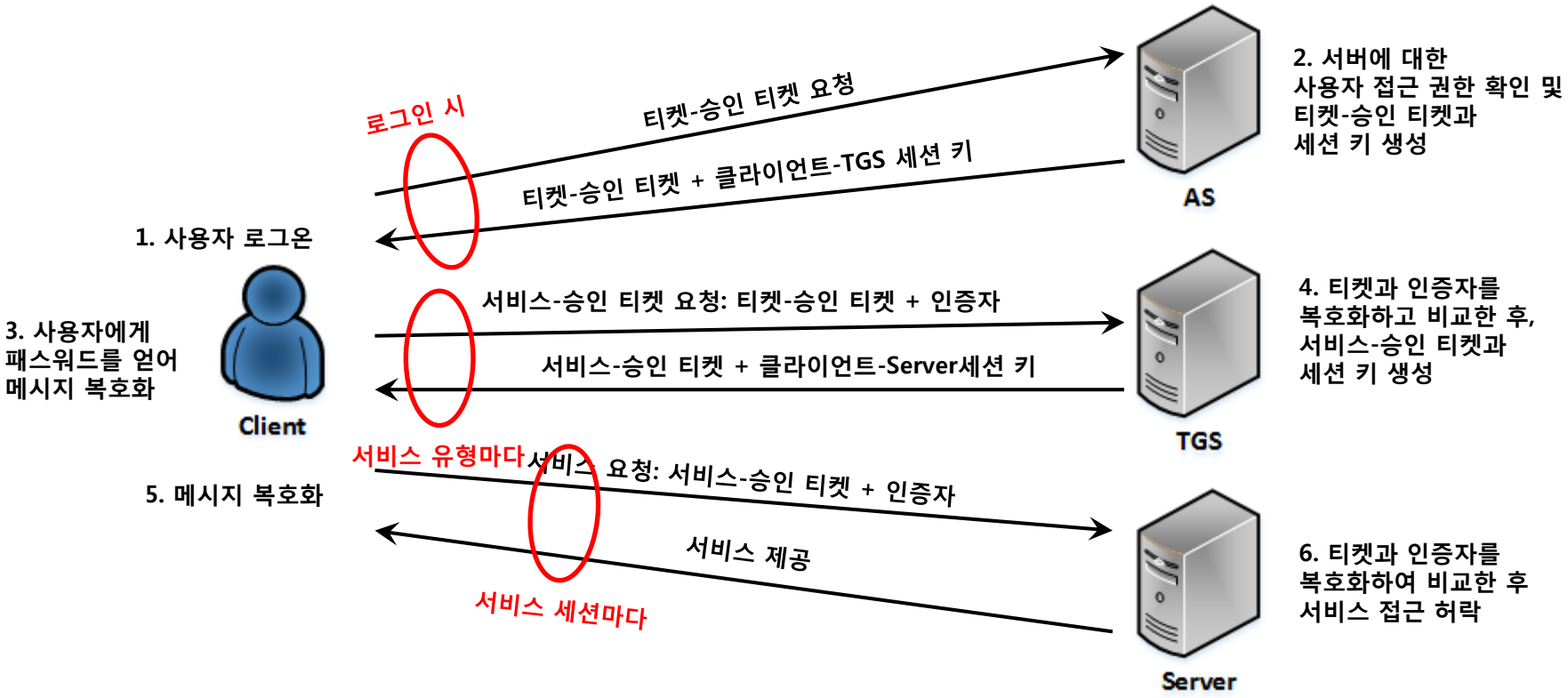
- $Authenticator_{C-V} = E(K_{C-V}, [ID_C \parallel AD_c \parallel TS_5])$

⑥ $Ticket_V$과 $Authenticator_{C-V}$ 를 복호화한 후 서비스 승인

⑦ 만일 상호간의 인증이 필요하다면 $E(K_{C-V}, [TS_5 + 1])$ 전송

KERBEROS

- **Kerberos version 4 인증 프로토콜**
  - **동작 과정 (개요)**



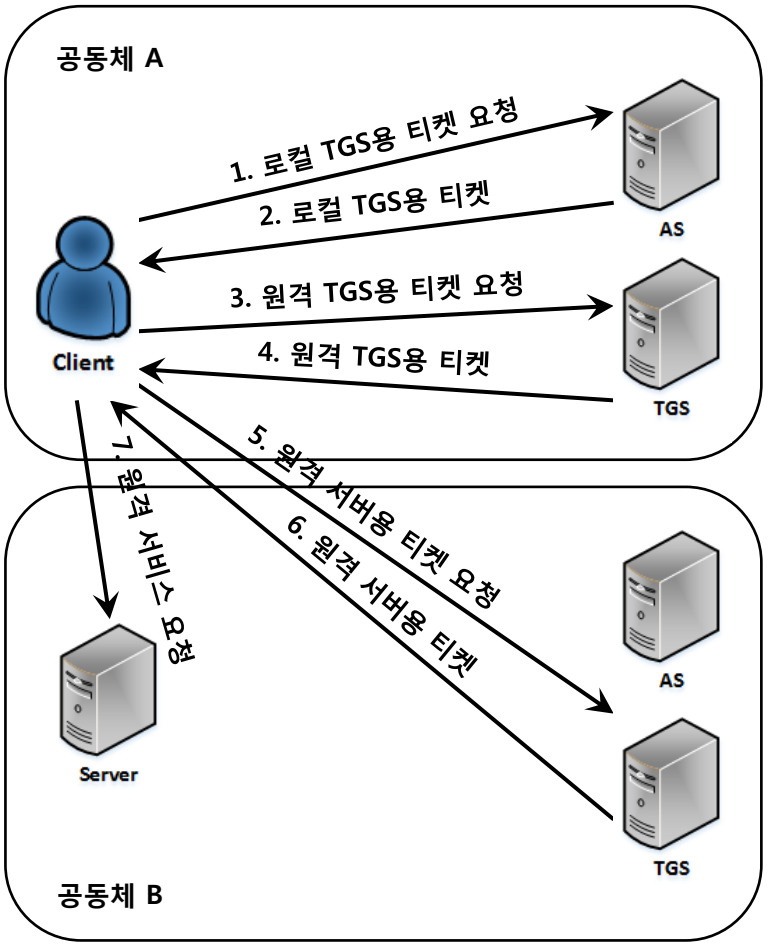
KERBEROS

- **Kerberos version 4 인증 프로토콜**
  - **완전한 Kerberos 환경**
    - Kerberos 서버, 다수의 클라이언트, 다수의 응용 서버로 구성된 환경

1. Kerberos 서버는 모든 사용자의 ID와 해시된 패스워드를 데이터베이스에 저장하고 있어야 함
2. Kerberos 서버는 각 서버와 비밀키를 공유해야 함
   → **Kerberos 공동체 (Kerberos realm)**
3. Kerberos 서버는 상호 교류하는 서로 다른 공동체 서버와 비밀키를 공유해야 함

KERBEROS

- **Kerberos version 4 인증 프로토콜**
  - **Kerberos 공동체 사이의 인증**



1. $ID_C \parallel ID_{TGS} \parallel TS_1$
2. $E(K_C,\ [K_{C-TGS} \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{TGS}])$
3. $ID_{REMTGS} \parallel Ticket_{TGS} \parallel Authenticator_C$
4. $E(K_{C-TGS},\ [K_{C-REMTGS} \parallel ID_{REMTGS} \parallel TS_4 \parallel Ticket_{REMTGS}])$
5. $ID_{REMV} \parallel Ticket_{REMTGS} \parallel Authenticator_C$
6. $E(K_{C-REMTGS},\ [K_{C-REMV} \parallel ID_{REMV} \parallel TS_6 \parallel Ticket_{REMV}])$
7. $Ticket_{REMV} \parallel Authenticator_C$

KERBEROS

- **Kerberos version 5 인증 프로토콜**
  - **Kerberos version 4의 환경적 결함 보완**
    1. DES 암호화 알고리즘을 사용해야 함
       → 모든 종류의 암호기술 사용 가능
    2. IP 주소를 사용해야 함
       → 어떤 유형의 네트워크 주소도 사용 가능
    3. 메시지 바이트 순서를 메시지에 따로 표기
       → 표준 언어인 ASN.1(Abstract Syntax Notation One)과 ASN.1의 기본 인코딩 규칙인 BER(Basic Encoding Rules)을 이용하여 바이트 순서를 정의
    4. 티켓의 유효기간은 최대 1280분으로 장시간 실행되는 응용 프로그램 같은 경우에는 부적합
       → 티켓에 시작시간과 만료시간을 명시
    5. 클라이언트가 사용하는 인증자는 한 가지 서버에만 사용 가능
       → 서로 다른 서버에서도 인증자 사용 가능
    6. N개의 공동체가 다른 모든 Kerberos 공동체와 상호 교류를 하기 위해서는 $N(N-1)/2$번의 키교환이 필요
       → 이보다 적은 숫자의 관계가 요구

KERBEROS

- **Kerberos version 5 인증 프로토콜**
  - **Kerberos version 4의 기술적 결함 보완**
    1. 티켓 이중 암호화로 인한 낭비
    2. PCPB(Propagating Cipher Block Chaining) 비표준 모드의 DES 사용으로 인한 암호문 블록을 교환하는 기법의 공격에 취약
       → CBC(Cipher Block Chaining) 모드 사용으로 무결성 제공
    3. 서비스 사용에 동일한 티켓이 반복적으로 사용될 수 있어 공격자가 이전 세션에서 가져온 메시지를 통해 재전송 공격이 가능
       → 클라이언트와 서버가 협상할 수 있도록 도입한 서브세션키를 사용하여 클라이언트가 접근을 시도할 때마다 서브세션키를 생성

KERBEROS

- **Kerberos version 5 인증 프로토콜**
  - **용어 설명**
    - C = 클라이언트
    - AS = 인증 서버 (AS: Authentication Server)
    - TGS = 티켓 발행 서버 (TGS: Ticket-granting Server)
    - V = 서버
    - ID_C = 클라이언트의 식별자
    - ID_V = 서버의 식별자
    - ID_{TGS} = 티켓 발행 서버의 식별자
    - PW_C = 클라이언트의 패스워드
    - AD_C = 클라이언트의 IP 주소
    - **K_C = 클라이언트의 패스워드로부터 얻은 키**
    - **K_{C-TGS} = 클라이언트와 티켓 발행 서버 사이의 세션키**
    - **K_{C-V} = 클라이언트와 서버 사이의 세션키**
    - **K_{AS-TGS} = 인증 서버와 티켓 발행 서버가 공유하는 비밀키**
    - **K_{TGS-V} = 티켓 발행 서버와 서버가 공유하는 비밀키**
    - || = 이어 붙이기
    - $Realm$ = 공동체
      - 사용자의 공동체를 나타냄
    - $Times$ = 시간
      - from: 요청된 티켓을 언제부터 사용하길 원하는지 표시
      - till: 요청된 티켓의 사용 만료시간을 표시
      - rtime: 요구되는 시작부터 만료시간까지의 시간을 표시
    - $Nonce$ = 난수
      - 응답이 재전송된 것이 아님을 확신시켜줌
    - **$Subkey$ = 서브키**
      - 응용 세션을 보호하기 위한 키로 클라이언트가 선택
    - $Seq\#$ = 순서번호
      - 서버가 세션동안 클라이언트에게 보내는 메시지에 재전송을 감지하기 위해 사용

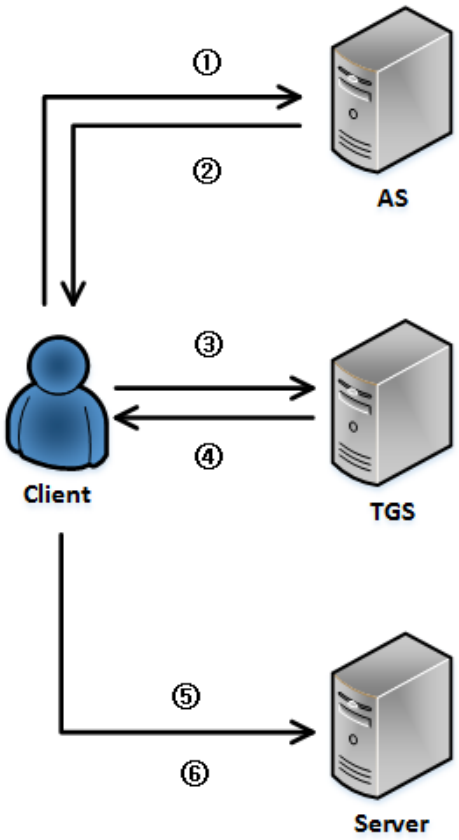
KERBEROS

- **Kerberos version 5 인증 프로토콜**
  - **용어 설명**
    - *Options* = 선택사항
      - 티켓 안의 플래그 설정을 요청

Flags	설명
INITIAL	AS를 이용하여 발행
PRE-AUTHENT	Client는 티켓이 발행되기 전에 KDC에 의해 인증
HW-AUTHENT	하드웨어의 이용이 필요한 초기 인증을 요구
RENEWABLE	교체 티켓을 얻기 위해 재사용될 수 있음
MAY-POSTDATE	날짜가 지난 티켓이 발행
POSTDATED	날짜가 지났음
INVALID	사용 전에 KDC에 의해 유효성 검증이 필요
PROXIABLE	티켓이 대리인임을 나타냄
PROXYFORWARDABLE	다른 네트워크 주소를 이용하여 발행 가능
FORWARDED	인증을 기반으로 발행

KERBEROS

- **Kerberos version 5 인증 프로토콜**
  - **동작 과정**



인증 서비스 교환: 티켓-발행 티켓을 취득하기

① $Options \parallel ID_C \parallel Realm_c \parallel ID_{TGS} \parallel Times \parallel Nonce_1$ 전송

② $Realm_c \parallel ID_C \parallel Ticket_{TGS} \parallel E(K_C, [K_{C-TGS} \parallel Times \parallel Nonce_1 \parallel Realm_c \parallel ID_{TGS}])$ 전송

- $Ticket_{TGS} = E(K_{AS-TGS}, [Flags \parallel K_{C-TGS} \parallel Realm_c \parallel ID_C \parallel AD_c \parallel Times])$

티켓-발행 서비스 교환: 서비스-발행 티켓 취득하기

③ $Options \parallel ID_V \parallel Times \parallel Nonce_2 \parallel Ticket_{TGS} \parallel Authenticator_{C-TGS}$ 전송

- $Authenticator_{C-TGS} = E(K_{C-TGS}, [ID_C \parallel Realm_c \parallel TS_1])$

④ $Realm_c \parallel ID_C \parallel Ticket_V \parallel E(K_{C-TGS}, [K_{C-V} \parallel Times \parallel Nonce_2 \parallel Realm_V \parallel ID_V])$ 전송

- $Ticket_V = E(K_{TGS-V}, [Flags \parallel K_{C-V} \parallel Realm_c \parallel ID_C \parallel AD_c \parallel Times])$

클라이언트/서버 인증 교환: 서비스 취득하기

⑤ $Options \parallel Ticket_V \parallel Authenticator_{C-V}$ 전송

- $Authenticator_{C-V} = E(K_{C-V}, [ID_C \parallel Realm_c \parallel TS_2 \parallel Subkey \parallel Seq\#])$

⑥ $E(K_{C-V}, [TS_3 \parallel Subkey \parallel Seq\#])$ 전송

KERBEROS

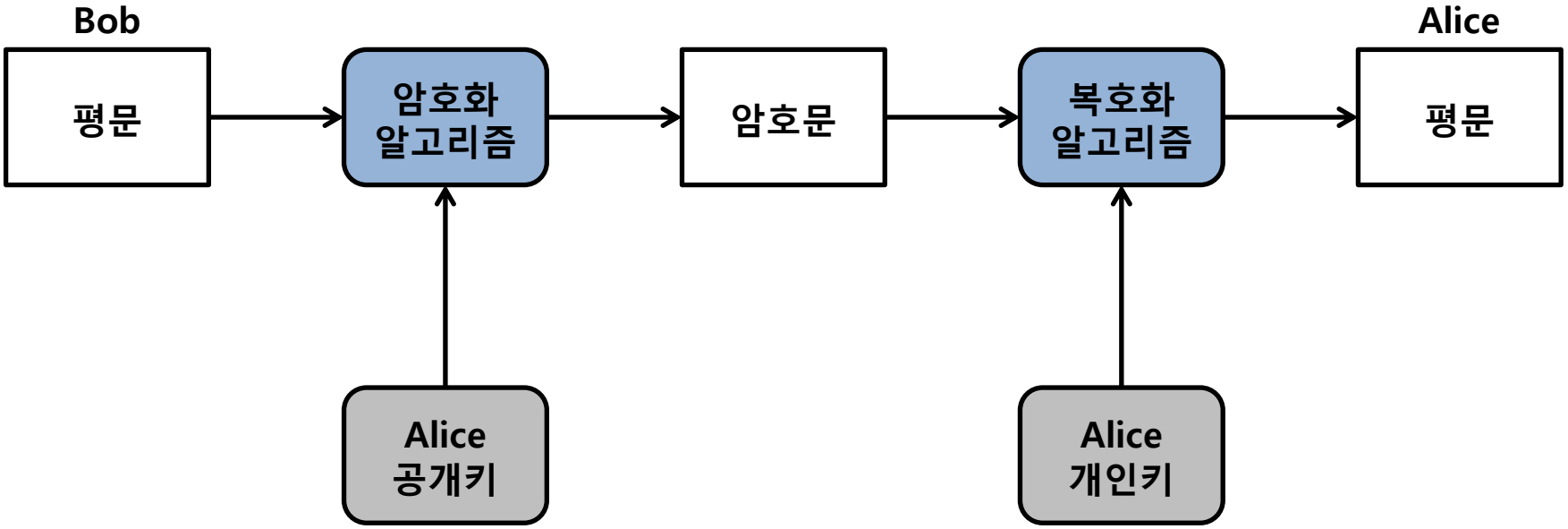
- **장점**
  - 사용자와 서비스 간의 통신 내용을 암호화 키 및 암호화 프로세스를 이용하여 보호하기 때문에 데이터의 기밀성과 무결성을 보장

- **단점**
  - 모든 사용자와 서비스의 암호화 키를 키 분배 센터가 가지고 있어 키 분배 센터에 오류가 발생하면 전체 서비스 사용이 불가
  - 사용자의 패스워드가 유출되어도 서버는 이를 인식하지 못하며, 사용자가 패스워드를 바꾸면 비밀키도 변경해야 함

Content

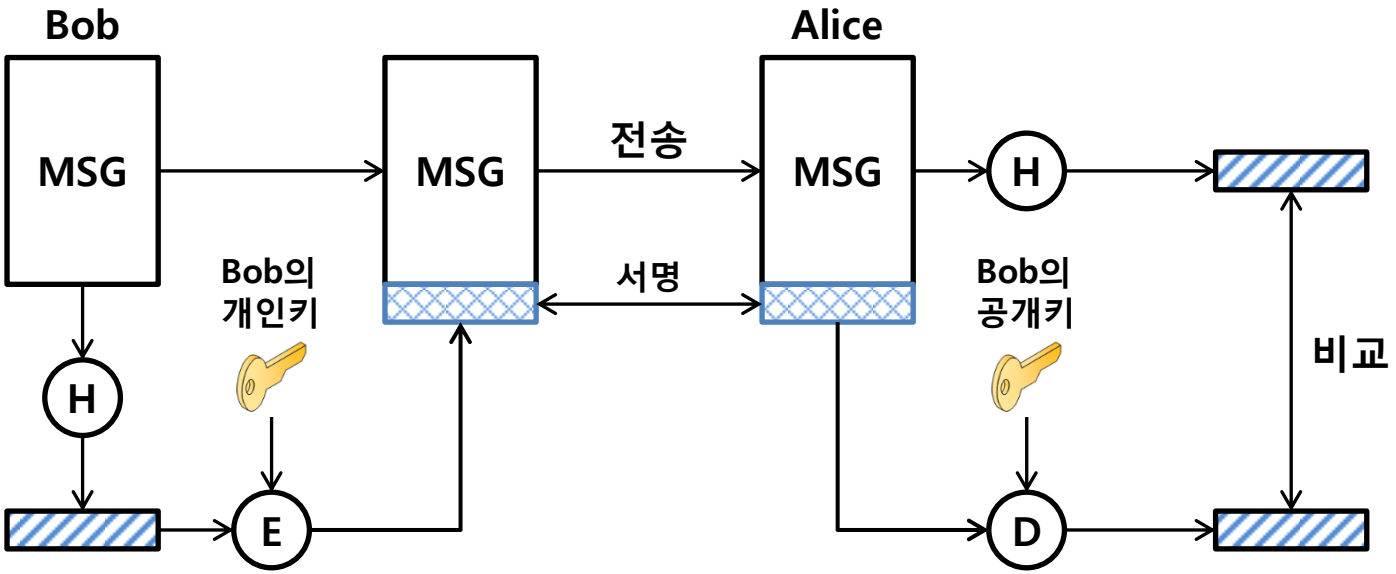
비대칭 암호를 이용한 키 분배

- **공개키 암호**
  - 어떤 공개키 알고리즘이 널리 사용되고 있다면 그 알고리즘을 사용하는 사람들에게 자신의 공개키를 공개하는 것



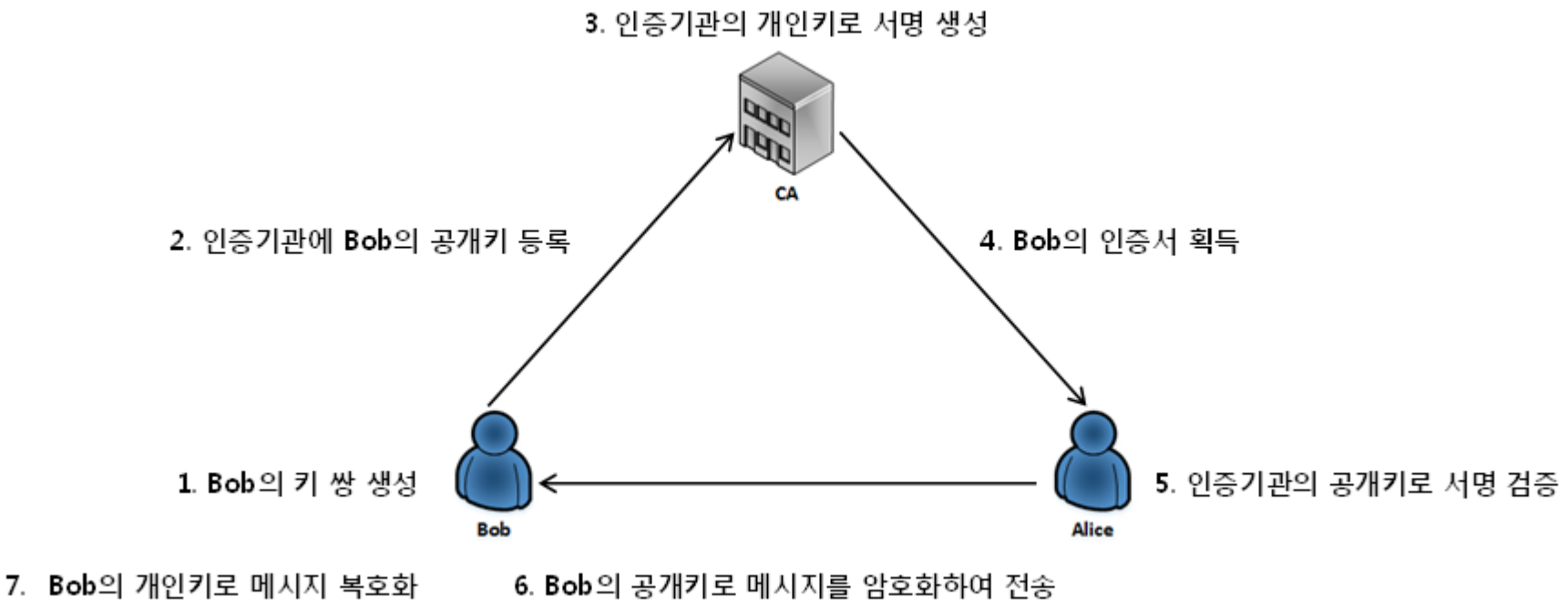
비대칭 암호를 이용한 키 분배

- **디지털 서명**
  - 데이터 통신에 있어서 기밀성은 보장할 수 없지만 메시지의 출처(본인 인증)와 데이터 무결성을 인증하는 것



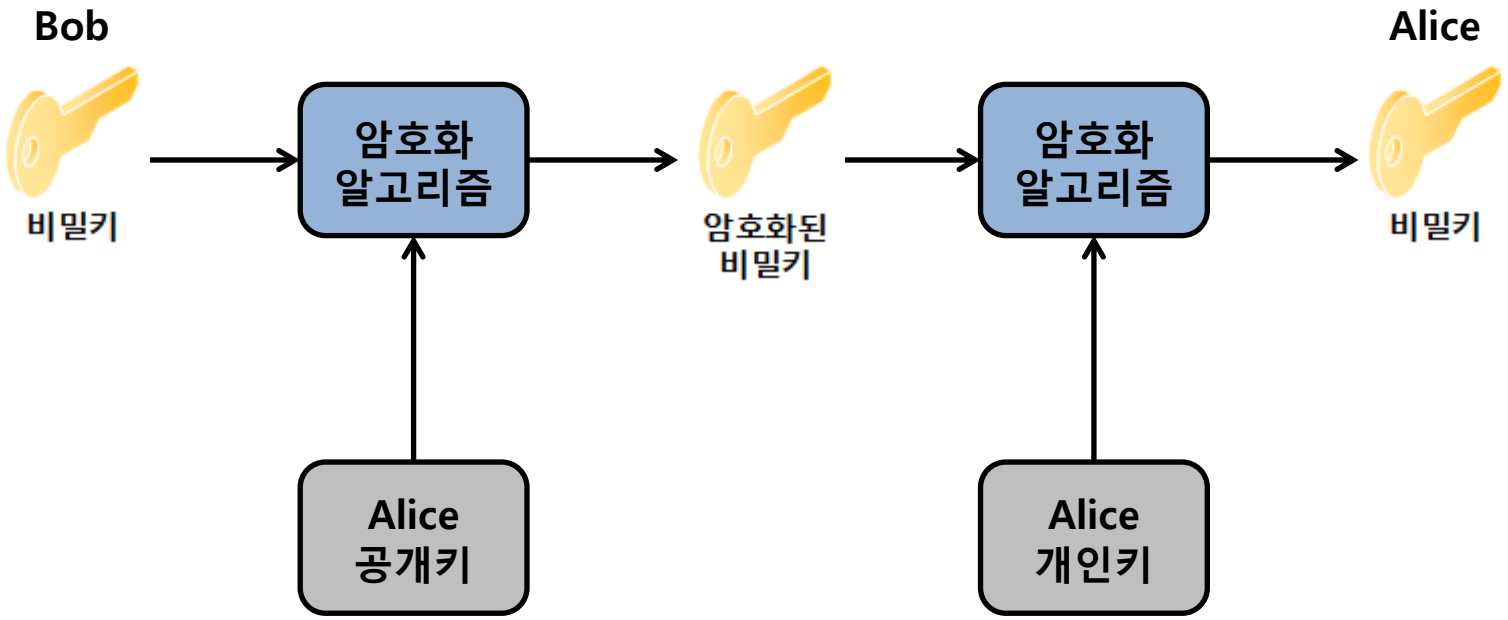
비대칭 암호를 이용한 키 분배

- **공개키 인증서**
  - 공개키 암호 방식에는 공격자가 특정 사용자로 가장하여 공개키를 공개할 수 있기 때문에 특정 사용자에게 전송되는 암호화된 메시지를 얻을 수 있고 가짜 키를 인증을 위해 사용할 수 있는 문제점이 존재
  - 공개키와 사용자 ID로 구성되어 있는 것을 신뢰할 만한 제 3자가 서명한 것
  - 제 3자는 정부기관, 금융기관 같은 사용자 모두가 신뢰하는 인증기관(CA: Certificate Authority)



비대칭 암호를 이용한 키 분배

- **공개키를 이용한 비밀키 분배**
  - 양측이 통신하는데 안전하게 비밀키를 공유하기 위해서 사용

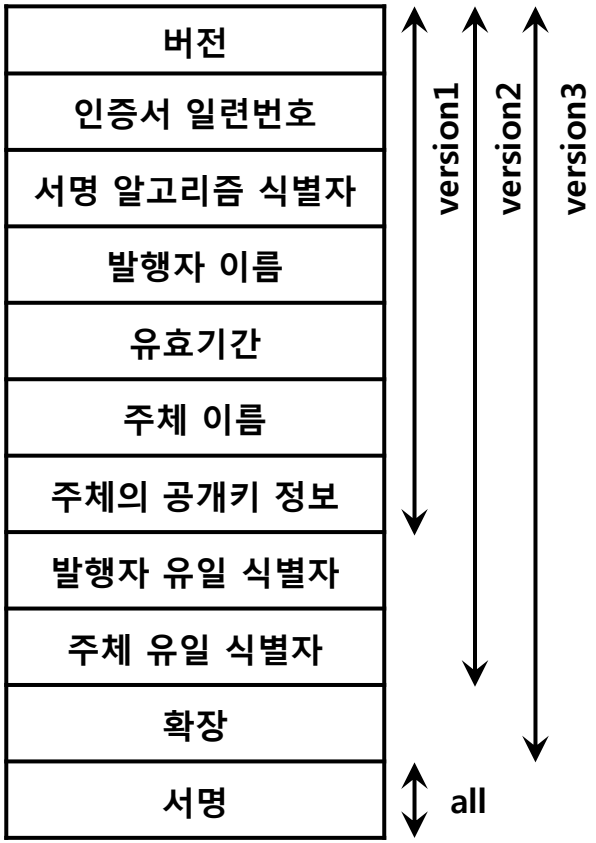


X.509 인증서

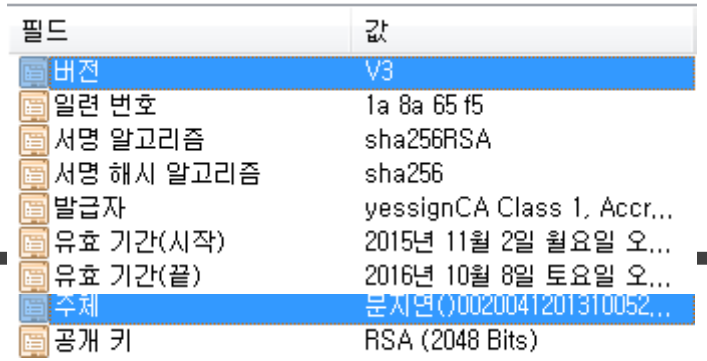
- **X.509 인증서**
  - 공개키 인증서를 이용한 인증 프로토콜로 공개키 암호와 디지털 서명을 이용
  - S/MIME, IP Security, SSL/TLS 등에 사용
  - 현재 version1부터 version3까지 존재하며, version3는 표준

X.509 인증서

- **X.509 인증서 형식: version1**

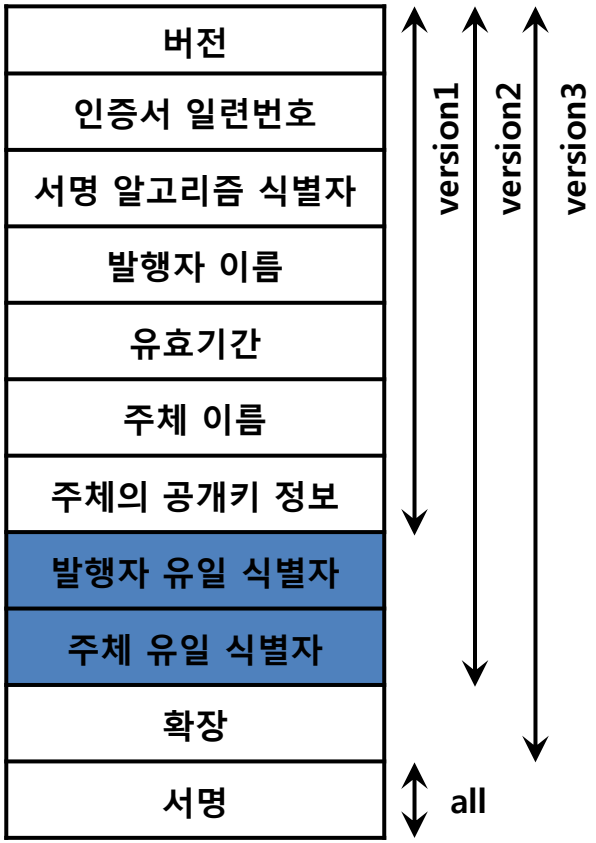


- **버전(Version):** 인증서 형식을 구별하기 위한 번호
- **일련번호(Serial number):** 인증기관에서 발행한 인증서의 일련번호
- **서명 알고리즘 식별자 (Sianature algorithm identifier):** 인증서 서명에 사용한 알고리즘 (ex RSA)
- **발행자 이름(Issuer name):** 인증서를 만들고 서명한 인증기관의 이름
- **유효기간(Period of validity):** 인증서가 유효한 시작일과 만료일
- **주체 이름(Subject name):** 인증서가 인증하는 사용자의 이름
- **주체의 공개키 정보(Subject's public-key information):** 주체의 공개키와 이 키를 사용하는 알고리즘 식별자
- **서명(Signature):** 인증기관의 개인키로 암호화한 해시값



X.509 인증서

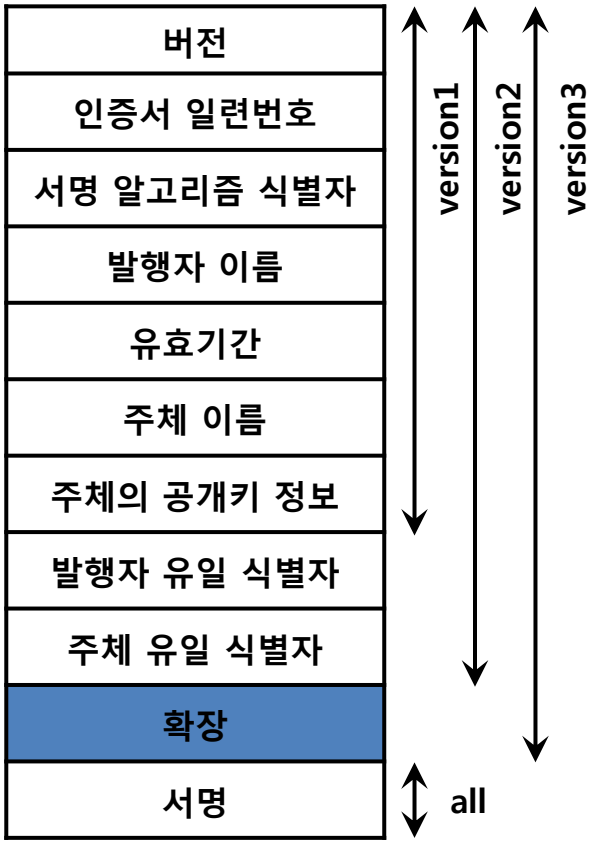
- **X.509 인증서 형식: version2**



- **발행자 유일 식별자(Issuer unique identifier):** 인증기관을 식별하기 위한 고유번호
- **주체 유일 식별자(Subject unique identifier):** 인증서 주체를 식별하기 위한 고유 번호

X.509 인증서

- **X.509 인증서 형식: version3**

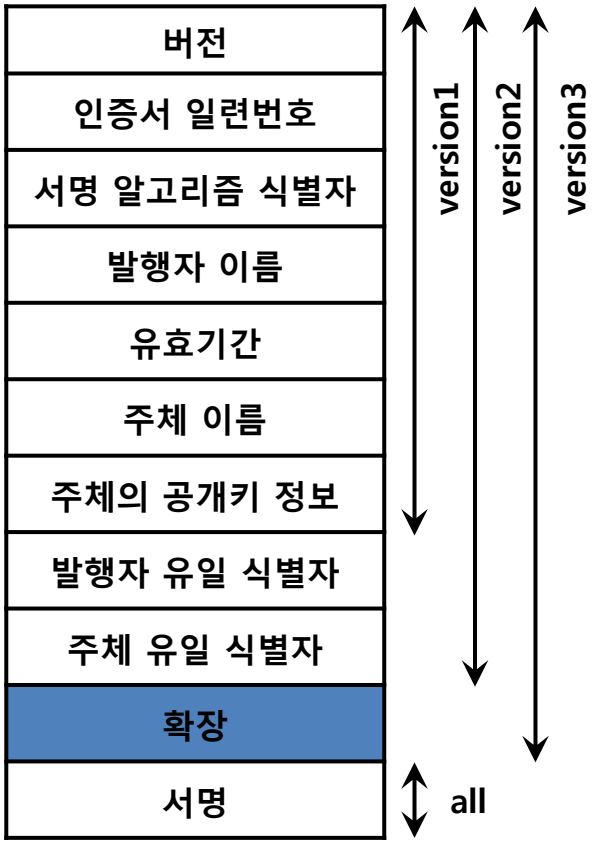


키와 정책정보

- **기관 키 식별자(Authority key identifier):** 인증서나 CRL에 있는 서명을 확인하는 데 쓰이는 공개키를 식별
- **주체 키 식별자(Subject key identifier):** 인증될 공개키를 식별
- **키 용도(Key usage):** 어떤 목적인지 어떤 정책 하에 사용될 것인지에 따라 키에 대한 제한조건
- **개인키 유효기간(Private-key usage period):** 공개키와 대응되는 개인키의 사용기간
- **인증서 정책(Certificate policies):** 해당 인증서가 지원되는 것으로 인식하는 정책 항목
- **정책 매핑(Policy mapping):** 다른 CA에 의해 발행한 인증서에만 사용되며 인증서를 발행한 CA의 정책이 발행받은 CA에 적용되는 다른 정책과 동일하게 간주될 수 있다는 것을 나타냄

X.509 인증서

- **X.509 인증서 형식: version3**

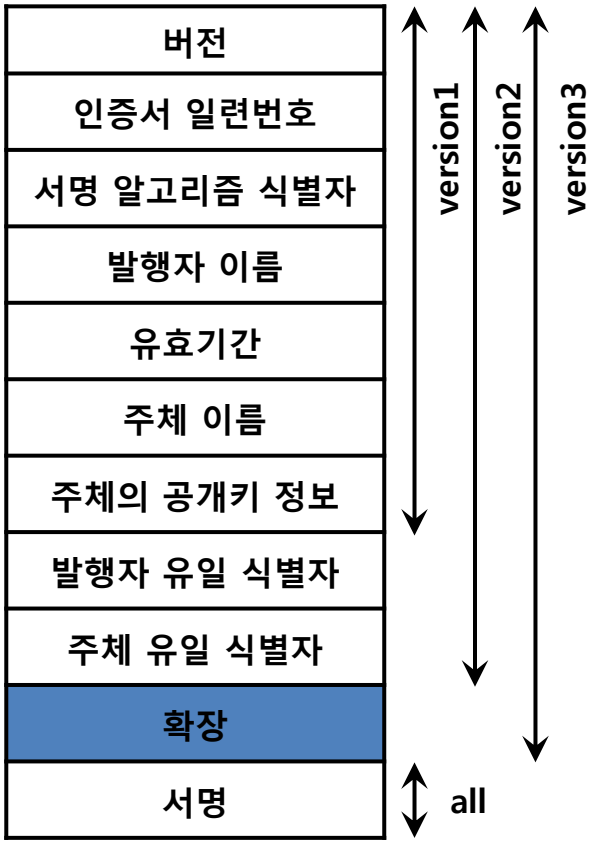


인증서 주체와 발행자 속성

- **주체 대체 이름(Subject alternative name):** 자체적으로 이름 형식을 가지고 있는 특정 응용프로그램(ex 전자메일, IPSec)을 지원하기 위해 대체 이름을 지원
- **발행자 대체 이름(Issuer alternative name)**
- **주체 디렉터리 속성(Subject directory attributes):** 인증서의 주체를 위한 어떤 것이든 X.500 디렉터리 속성 값

X.509 인증서

- **X.509 인증서 형식: version3**

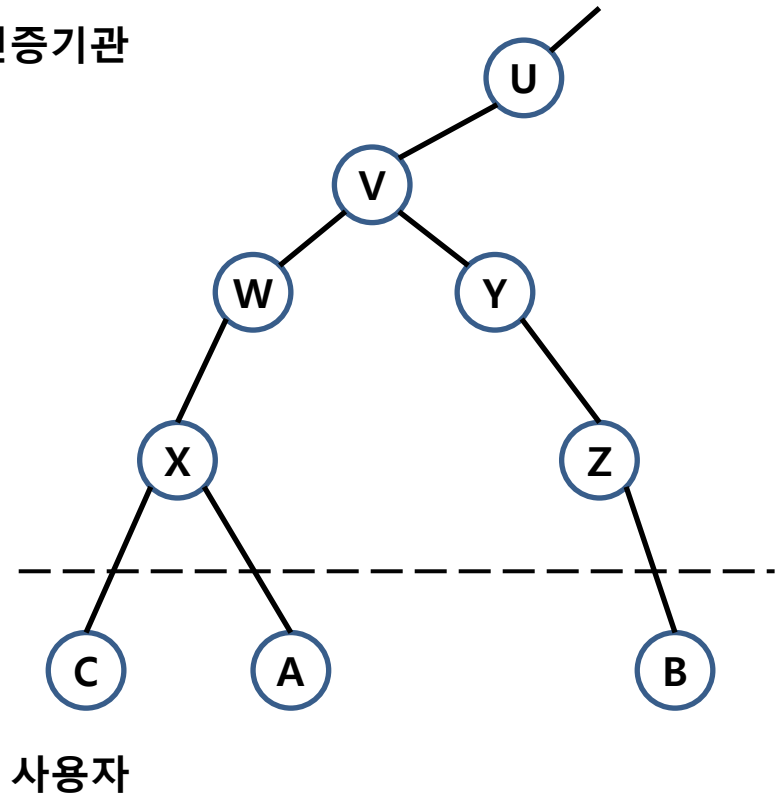


인증경로 제약조건

- **기본 제약(Basic constraints):** 주체가 CA 역할을 하는지 아닌지 그리고 인증 경로에 대한 정보
- **이름 제약(Name constraints):** 인증서에 나타나는 모든 주체의 이름이 위치해야 하는 이름 공간
- **정책 제약(Policy constraints):** 구체적인 인증서 정책 식별 요구 및 인증서 정책 매핑 금지 요구

X.509 인증서

- **X.509 인증서 체인 (Chain of Certificate)**
  - 서로 다른 기관으로부터 안전하게 공개키를 획득하기 위해서 인증 기관들을 계층구조(트리구조)로 연결한 것



사용자 A에서 사용자 B의 인증서를 얻기 위한 인증 경로

A<<X>>X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

X.509 인증서

- **X.509 인증서 취소**
  - **인증서 취소 목록 (CRL: Certificate Revocation List)**

서명 알고리즘 식별자
발행자 이름
최근 업데이트 일시
마지막 업데이트 일시
취소된 인증서
:
취소된 인증서
서명

  - **인증서 유효기간 만료 이외의 취소 사유**
    1. 사용자 개인키가 노출되거나 훼손된 경우
    2. CA가 사용자를 더 이상 인증해줄 수 없는 경우
    3. CA의 인증서가 노출되었거나 훼손된 경우

Content

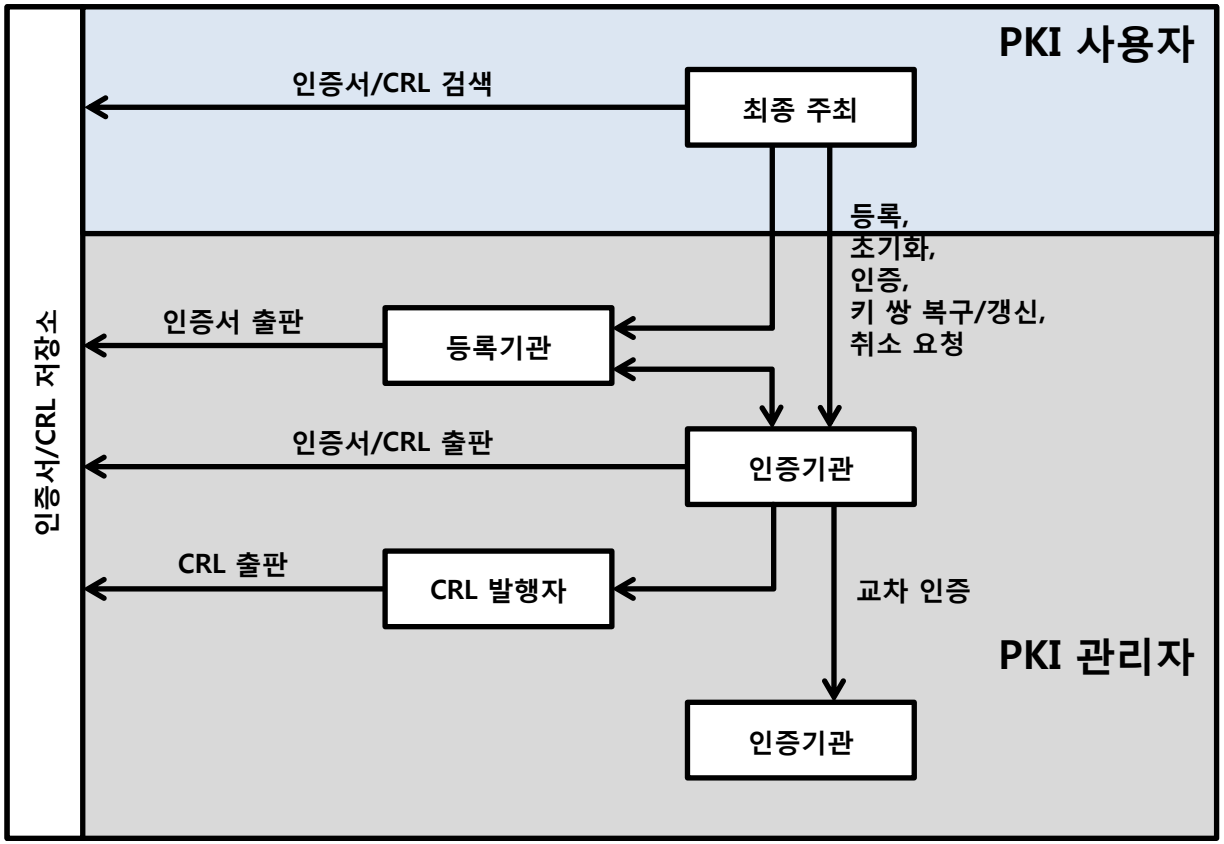
공개키 기반구조

- **공개키 기반구조 X.509 (PKIX: Public-Key Infrastructure X.509)**
  - X.509에 기초하여 디지털 인증서를 생성/관리/저장/분배/취소하는 데 필요한 하드웨어, 소프트웨어, 사용자, 정책 및 절차

  - **구성 요소**
    - 사용자(User): 종단 사용자, 종단 장치 등으로 PKI와 관련된 서비스를 사용
    - 인증기관(CA: Certificate authority): 인증서와 인증서 취소 목록(CRL)을 발행
    - 등록기관(RA: Registration authority): 선택적 요소로 사용자의 등록절차와 관련된 업무 수행
    - CRL 발행자(CRL issuer): 선택적 요소로 인증서 취소목록 발행
    - 저장소(Repository): 인증서와 인증서 취소목록을 저장하며 사용자가 검색할 수 있는 장소

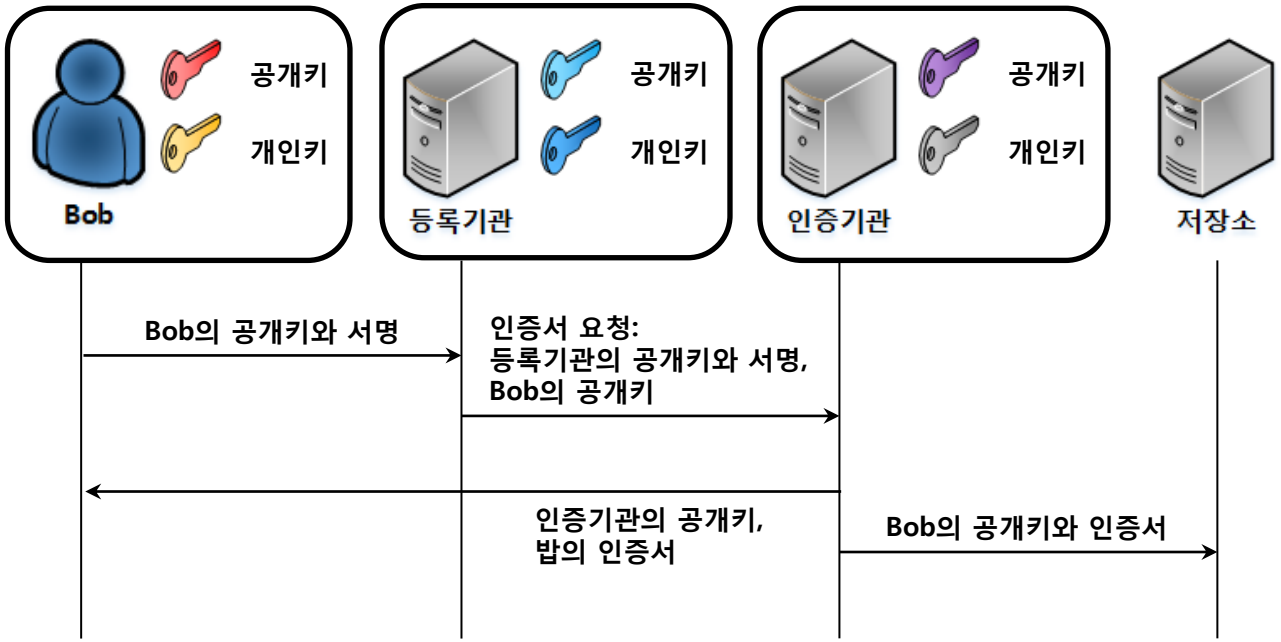
공개키 기반구조

- **PKIX 구조 모델**



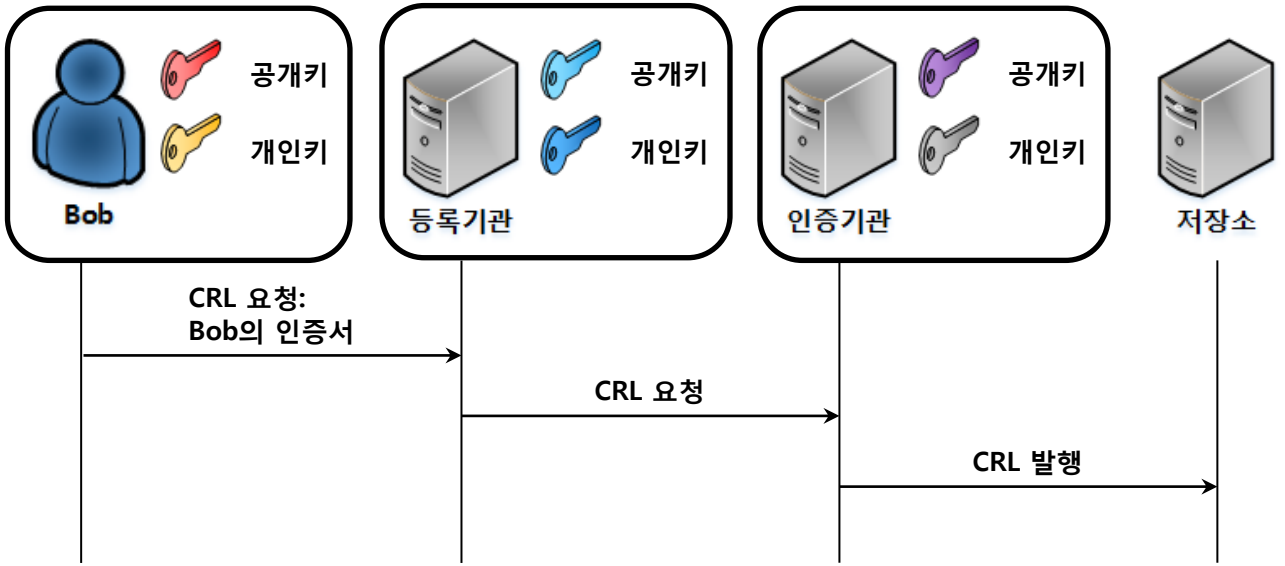
공개키 기반구조

- **PKIX 관리 기능**
  - **등록(Registration):** 사용자가 최초로 CA에게 자신을 알리는 절차
  - **초기화(Initialization):** 사용자의 시스템이 안전하게 작동되기 전에 키와 관련된 재료들을 설치 (ex CA의 공개키)
  - **인증(Certification):** 사용자에게 인증기관의 공개키를 발급하고 인증서를 사용자의 시스템에게 인증서를 돌려주며 인증서를 저장소에 저장



공개키 기반구조

- **PKIX 관리 기능**
  - **취소 요청(Revocation request):** 정상적인 키 사용이 불가능한 경우 인증서를 폐지하도록 요청 (ex 개인키 분실/도난, 이름 변경)



공개키 기반구조

- **PKIX 관리 기능**
  - **키 쌍 복구(Key pair recovery):** 정상적인 키 사용이 불가능한 경우를 대비하여 개인키를 획득하는 절차

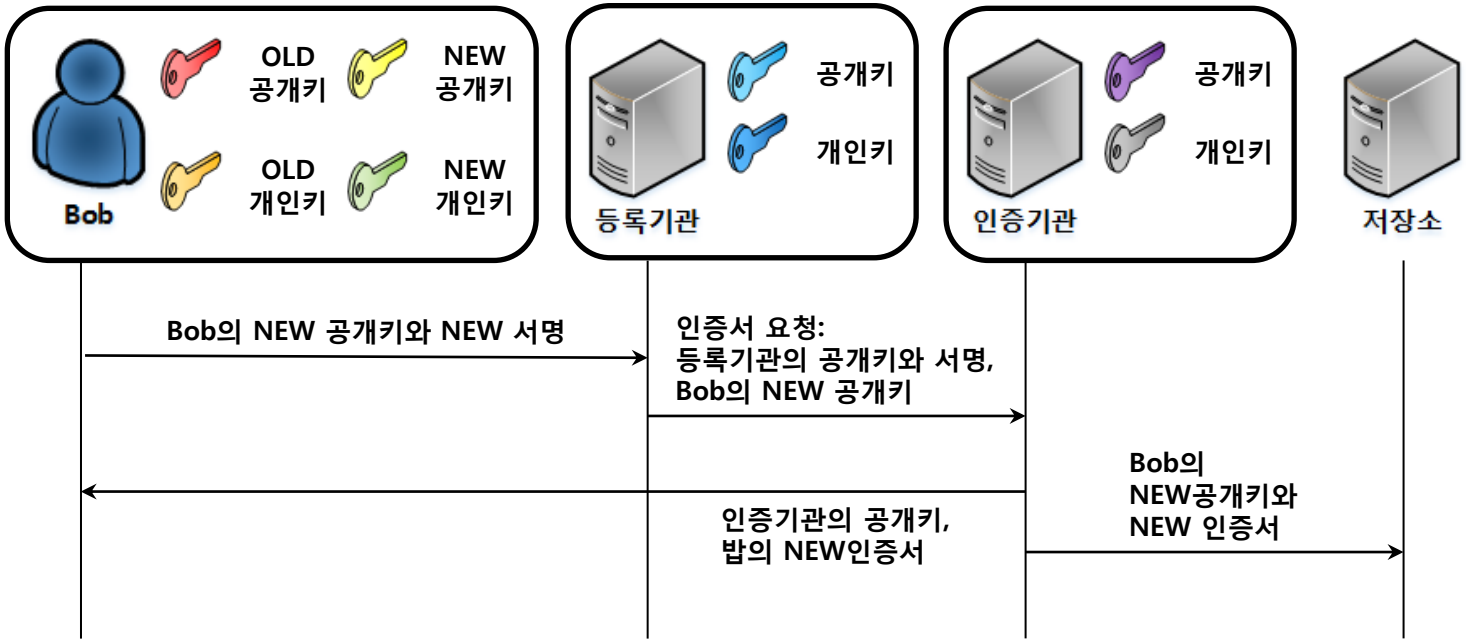
    ✓ 위탁 방식: 사용자의 비밀키 전부 또는 일부를 신뢰받는 제 3자에게 위탁하는 방식으로 신뢰도가 낮은 위탁기관을 사용하는 경우 문제가 발생

    ✓ 캡슐화 방식: 각각의 메시지 전송 마다 키 복구에 필요한 정보를 담은 영역을 생성해서 해당 메시지를 복구할 수 있는 정보를 데이터에 부가하는 방식으로 복구 영역을 사용자가 생성하기 때문에 수정과 조작이 용이하며 프로토콜 자체가 복구 영역에 대한 확장 영역을 지원해야 함

    ✓ TTP 방식: 신뢰할 수 있는 제 3자인 TTP를 가정하여 복구될 사용자의 비밀키를 그 사용자의 신뢰기관으로 지정된 기관에서 생성하고 사용자에게 분배하는 방식으로 많은 개인의 정보가 전적으로 TTP에게 의존되며 너무 많은 TTP가 요구됨

공개키 기반구조

- **PKIX 관리 기능**
  - **키 쌍 갱신(Key pair update):** 모든 키 쌍을 정기적으로 갱신하기 위해서 새로운 인증서를 발급 (ex 인증서의 유효기간 종료된 경우, 인증서가 취소된 경우)



공개키 기반구조

- **PKIX 관리 기능**
  - **교차 인증(Cross certification):** 한 CA가 다른 CA에게 인증서 발급하며 두 개의 CA가 정보를 교환하여 서로 다른 CA의 인증서를 가진 사용자를 인증

공개키 기반구조

- **PKIX 메시지 포맷**
  - **메시지 포맷**

PKIMessage ::= SEQUENCE {

header PKIHeader,

body PKIBody,

protection [0] PKIProtection OPTIONAL,

extraCerts [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL

공개키 기반구조

- **PKIX 메시지 포맷**
  - **메시지 헤더 포맷**

PKIHeader ::= SEQUENCE {

pvno INTEGER { ietf-version2 (1) }, // 프로토콜의 버전을 나타냄

sender GeneralName, // 송신자 이름

recipient GeneralName, // 수신자 이름

messageTime [0] GeneralizedTime OPTIONAL, // 송신자가 메시지를 생성하는 시간을 포함하는 필드

protectionAlg [1] AlgorithmsIdentifier OPTIONAL, // 메시지를 보호하기 위해 사용되는 알고리즘 명시

senderKID [2] KeyIdentifier OPTIONAL, // 메시지 보호를 위해 사용된 키를 식별하기 위한 필드

recipKID [3] KeyIdentifier OPTIONAL, // 메시지 보호를 위해 사용된 키를 식별하기 위한 필드

transactionID [4] OCTET STRING OPTIONAL, // 요청메시지와 이에 대한 응답메시지를 식별하기 위한

senderNonce [5] OCTET STRING OPTIONAL, // 메시지의 재사용 공격을 막기 위해 사용되는 필드

recipNonce [6] OCTET STRING OPTIONAL, // senderNonce에 따른 값이 들어가는 필드(동일한 값 포함)

freeText [7] PKIFreeText OPTIONAL,

generalInfo [8] SEQUENCE SIZE (1..MAX) OF InfoTypeAndValue OPTIONAL}

PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String

공개키 기반구조

- **PKIX 메시지 포맷**
  - **메시지 바디 포맷**

PKIBody ::= CHOICE {

ir CertReqMessage, // 초기화 요청

cr CertReqMessage, // 등록 및 인증서 요청

p10cr CertificateionRequest, // p10cr인증요청

popdecr POPODecKeyRespContent, // pop 응답

kup CertRepMessage, // 키 갱신 응답 메시지

krp KeyRecRepContent, // 키 복구 응답

rp CertRepMessage, // 인증서 폐지 응답메시지

ccp CertReqMessage, // 교차인증 응답

cann CertAnnContent, // 인증 Ann

crlann CRLAnnContent, // CRL 발행

nested NestedMessageContent, // 내부 메시지

genp GenRepContent, // 일반적인 응답메시지

ip CertRepMessage, // 초기화 응답

cp CertReqMessage, // 등록 및 인증서 응답

popdecc POPODecKeyChallContent, // pop challenge

kur CertReqMessage, // 키 갱신 요청 메시지

krr CertReqMessage, // 키 복구 요청

rr CertReqMessage, // 인증서 폐지 요청 메시지

ccr CertReqMessage, // 교차인증 요청

ckuann CAKeyUpdAnnContent, // 키 갱신 Ann

rann RevAnnContent, // 폐지 Ann

conf PKIConfirmContent, // 확인 메시지

genm GenMsgContent, // PKI 개체 간 일반적인 요청메시지

error ErrorMsgContent } // 에러 메시지

통합신원관리

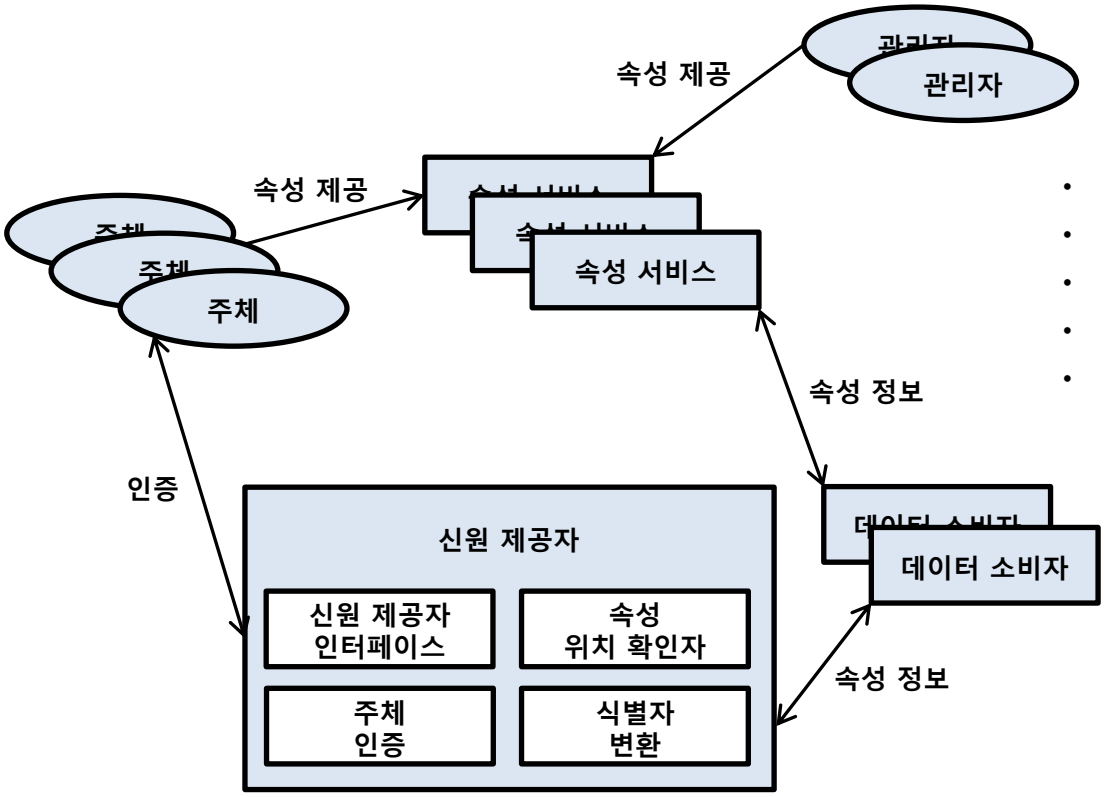
- **신원관리(Identity management)**
  - **신원관리**
    - 기업에 직원이나 권한을 가진 개인이 자원에 접근하는 절차를 자동화하는 방법
    - 사용자의 신원을 정의하고 속성을 신원에 연관 짓고 사용자가 신원을 인증
  - **싱글사인온(SSO: Single Sign-on)**
    - 신원관리 시스템의 중심적 개념으로 사용자가 한 번만 인증하면 네트워크의 모든 자원에 접속할 수 있게 하는 것

통합신원관리

- **신원관리(Identity management)**
  - **신원관리의 원칙**
    - 인증(Authentication): 제공한 사용자 이름과 사용자가 일치하는지를 확인하는 것
    - 허가(Authorization): 인증에 근거해서 특정 서비스나 자원에 접근을 허락하는 것
    - 계정(Accounting): 로그인 접근과 허가 절차
    - 제공(Provisioning): 시스템에 사용자 등록
    - 작업절차 자동화(Workflow automation): 비즈니스 프로세스에서 데이터의 이동
    - 관리위임(Delegated administration): 허가부여를 위한 역할-기반 접근 통제 사용
    - 패스워드 동기화(Password synchronization): 싱글사인온을 생성하여 사용자가 일회의 인증으로 네트워크의 모든 자원에 접근할 수 있도록 함
    - 셀프-서비스 패스워드 리셋(Self-service password reset): 사용자가 자신의 패스워드를 갱신하도록 함
    - 통합(Federation): 인증과 허가절차를 한 시스템에서 다른 시스템으로 전달하여 사용자가 매번 인증받을 필요가 없음

통합신원관리

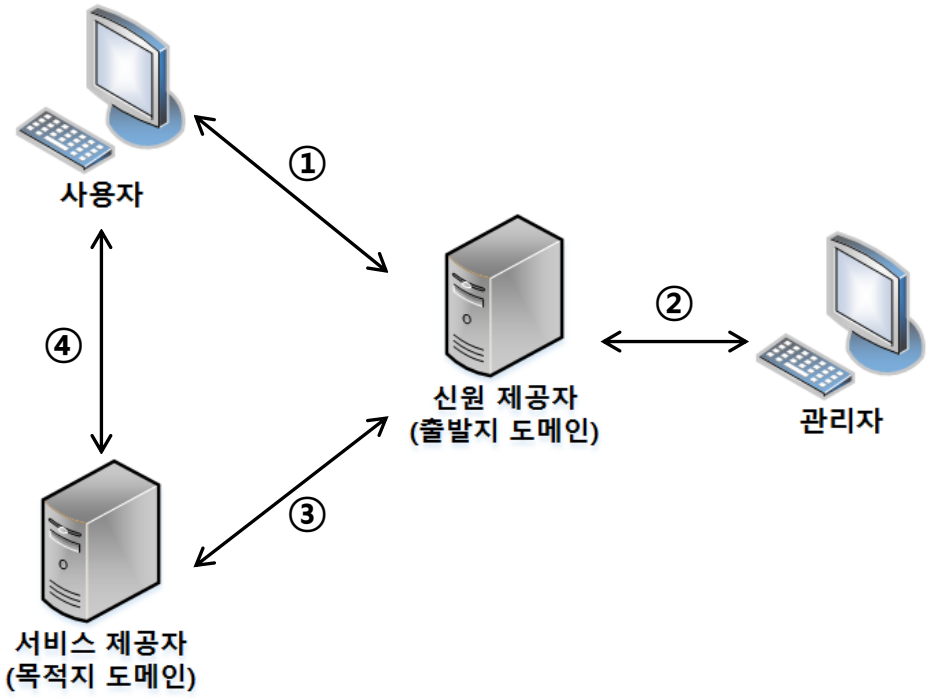
- **신원관리(Identity management)**
  - **일반적인 신원관리 구조**



- 주체(principal): 신원 소지자
- 신원 제공자(identity provider): 인증정보를 주체와 연관
- 속성 서비스(attribute service): 속성정보 생성/유지/관리
- 관리자(administrator): 사용자에게 속성을 부여
- 데이터 소비자(data consumers): 신원 제공자 또는 속성 서비스가 관리하는 데이터를 받아 사용하는 개체

통합신원관리

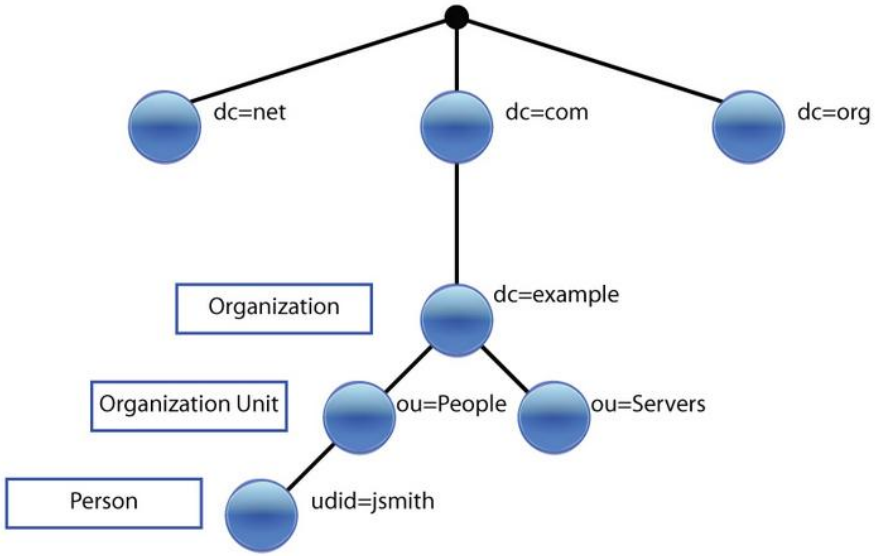
- **신원 통합(Identity federation)**
  - 다른 도메인으로 신원관리를 확장하고 디지털 신원을 공유하여 한 번의 사용자 인증으로 다수의 도메인에 있는 응용 프로그램이나 자원에 접근할 수 있도록 하는 것



① 동일 도메인 내의 신원 제공자에게 사용자는 사용자의 신원관 연관된 속성 값을 제공
② 동일 도메인 내의 관리자가 제공하는 신원과 연관된 속성 제공
③ 원격 도메인 서비스 제공자에게 사용자의 신원 정보와 인증정보 제공
④ 서비스 제공자는 원격 사용자와 세션을 생성하고 사용자의 신원 및 속성에 기반한 제한요건에 맞춰 접근 통제

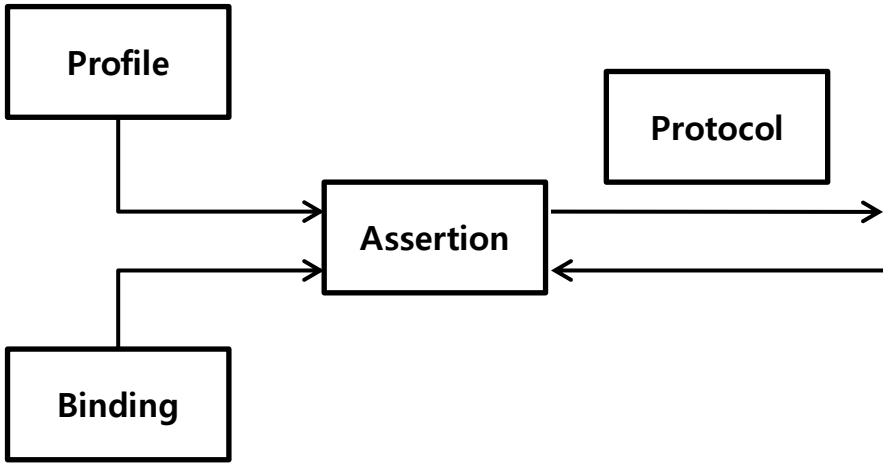
통합신원관리

- **신원 통합 표준**
  - **LDAP(Lightweight Directory Access Protocol)**
    - 디렉터리 서비스 액세스를 위한 클라이언트-서버 프로토콜



통합신원관리

- **신원 통합 표준**
  - **SAML(Security Assertion Markup Language)**
    - 보안 도메인 간에 인증정보와 권한부여에 관련된 자료를 교환할 수 있도록 설계된 XML 기반의 표준
  - **SAML 구조**

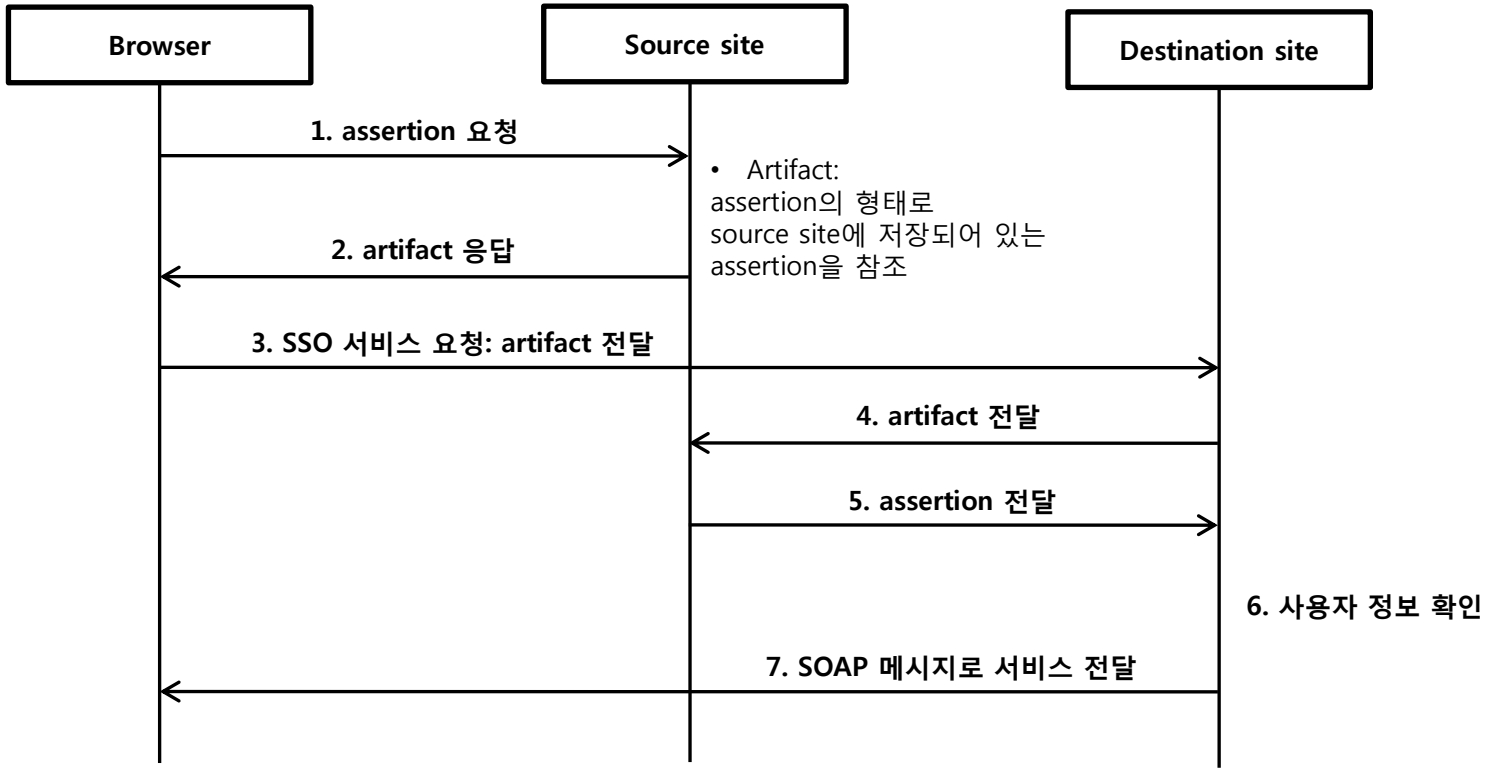


- Assertion: 사용자가 해당하는 웹사이트에 대한 접근 여부 요청에 응답
- Protocol: 요청과 응답의 포맷 결정
- Binding: 요청과 응답을 전송하는 방식
- Profile: 메시지 안의 사용자 정보 설명

  - **SOAP(Simple Object Access Protocol)**
    - 네트워크 상에서 웹 서비스가 통신할 수 있도록 하는 메시지 프로토콜

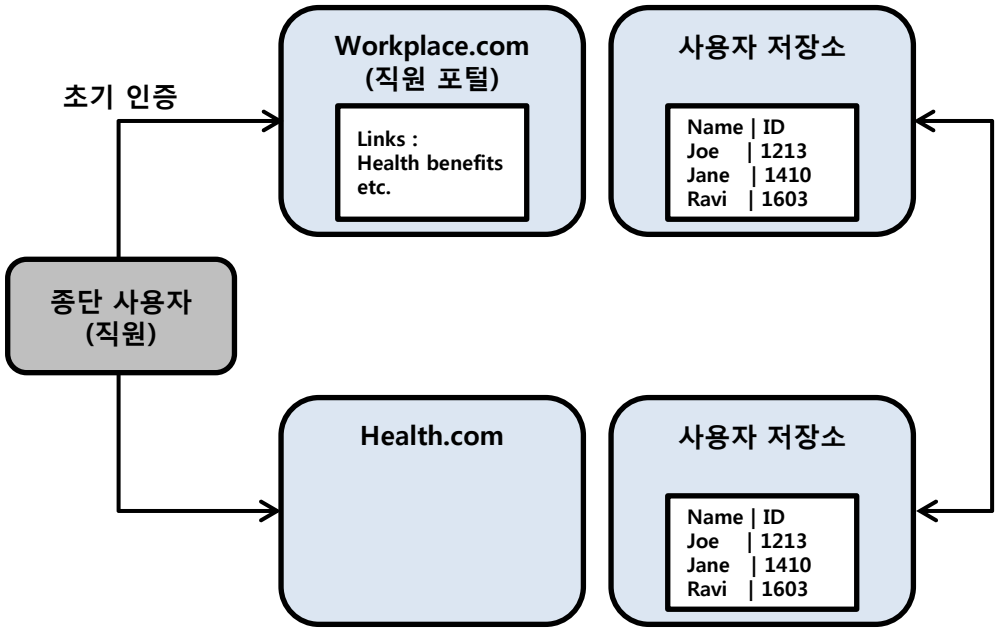
통합신원관리

- **신원 통합 표준**
  - **SAML을 이용한 SSO 서비스 시나리오**



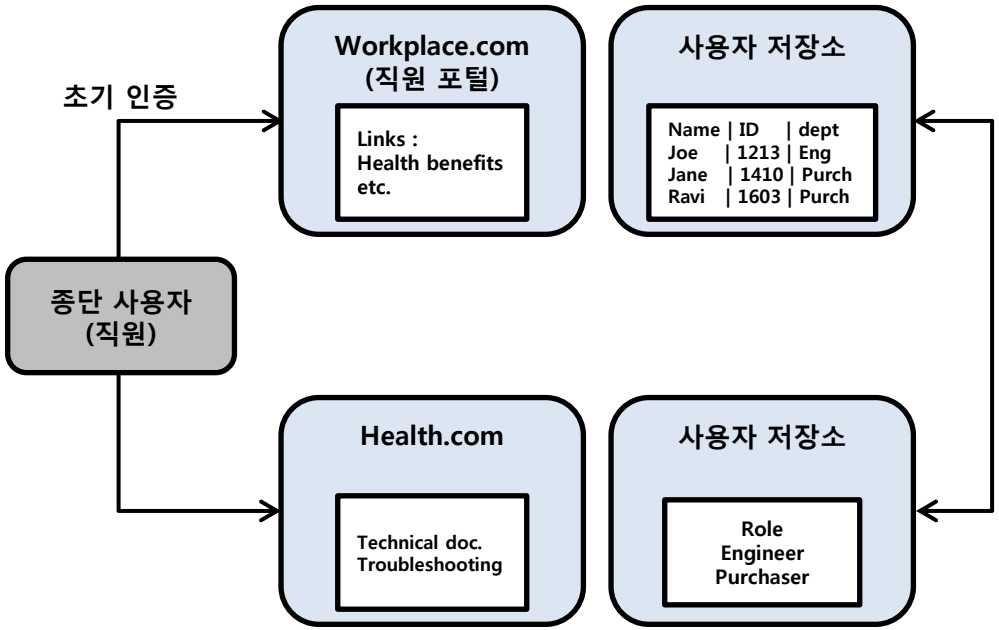
통합신원관리

- 신원 통합 시나리오
  - 계정 연결기반 통합



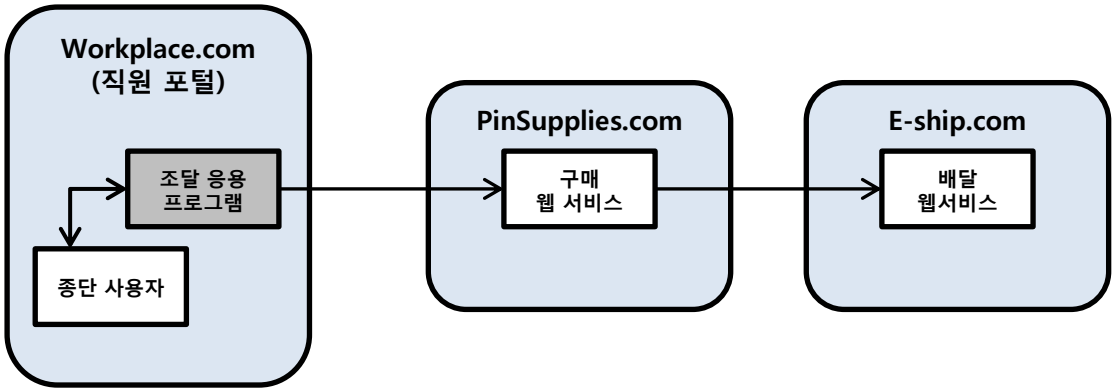
통합신원관리

- 신원 통합 시나리오
  - 역할 기반 통합



통합신원관리

- 신원 통합 시나리오
  - 연쇄 웹 서비스



네트워크 보안 에센셜

끝!