정보보호개론(CSE441)
www.facebook.com/KoreaTech.Crypto

암호알고리즘 개요 1부

NOTE 02

한국기술교육대학교 컴퓨터공학부 김상진

sangjin@koreatech.ac.kr
www.facebook.com/sangjin.kim.koreatech

교육목표

- 암호알고리즘에 대한 기초적 이해 (종류, 특성, 용도)
- 대칭 암호알고리즘
- 비대칭 암호알고리즘
  - 공개키 기반구조
  - 인증서 기반, 신원기반, DPKI

암호알고리즘 (1/2)

- 암호알고리즘(cryptographic algorithm, cipher):
  - 암호화와 복호화 과정에 사용하는 수학 함수 (좁은 의미)
    - 비밀성 서비스 제공이 목적
  - 암호기술에서 사용하는 모든 알고리즘 (넓은 의미)
    - 다양한 목적을 위해 사용함
- 현대 암호화 함수와 복호화 함수는 모두 키(key)를 사용
  - 표기법

$$E.K(M) = C \quad D.K(C) = M$$

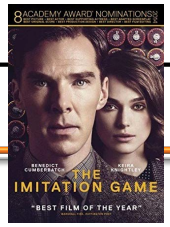
    - 암호프로토콜 기술에서는 $E.K(M)$ 대신에 $\{M\}.K$를 사용
  - 암호키(cryptographic key): 암호화/복호화에 사용하는 키
    - 현대 암호알고리즘의 안전성은 키에 의존함. 알고리즘 자체는 공개함
  - 정확성(correctness) 요구사항: $D.K\big(E.K(M)\big) = M$

제한적 알고리즘

open design principle
vs.
security by obscurity

- 현대 암호알고리즘의 안전성은 키에 의존해야 함.
  (케르크호프스(Kerckhoffs)의 원리, 1883)
  - 키 대신에 알고리즘의 비밀성에 의존할 경우에는 제한적 알고리즘 (restricted algorithm)이라 함

	제한적 알고리즘	키 의존 알고리즘
장점	• 해독하기가 상대적으로 더 힘듦 	• 알고리즘 공유가 가능 • 키가 노출되면 키 자체만 변경 가능 • 알고리즘이 공개되어 있으므로 허점의 발견이 용이함 ◦ 알고리즘 개선에 사용 가능
단점	• 알고리즘을 공유할 수 없음 • 역공학(reverse engineering)의 가능성이 존재함 • 노출될 경우에는 알고리즘 자체를 변경해야 함	• 알고리즘이 공개되어 있으므로 허점의 발견이 용이함 ◦ 악의적인 목적으로 사용 가능

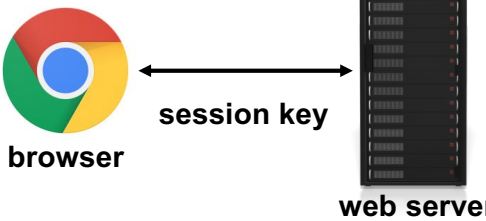


키 용어 (1/2)

암호키(cryptographic key)	암호알고리즘에 사용되는 모든 종류의 키
암호화키(encryption key)	암호화 과정에 사용되는 암호키
복호화키(decryption key)	복호화 과정에 사용되는 암호키
비밀키(secret key)	대칭 암호알고리즘에 사용되는 암호키
개인키(private key)	비대칭 암호알고리즘에서 각 개인이 비밀로 하는 암호키
공개키(public key)	비대칭 암호알고리즘에서 각 개인이 공개하는 암호키

키 용어 (2/2)

- 사용 기간 또는 횟수에 의한 분류
  - 단기간키, 세션키, 일회용키(short-term, session, one-time, single use)
    - 매우 제한적으로 사용하는 키
    - 생성 후 일시적으로 사용한 후에 폐기하는 키
    - 비휘발성 메모리에 유지할 필요가 없는 키
  - 장기간키, 다중사용키(long-term, many time, multi use)
    - 여러 번 사용하는 키로 비휘발성 메모리에 유지해야 하며, 안전한 키 관리가 중요함



인터넷/모바일 Banking

공개키 쌍: 장기간 키

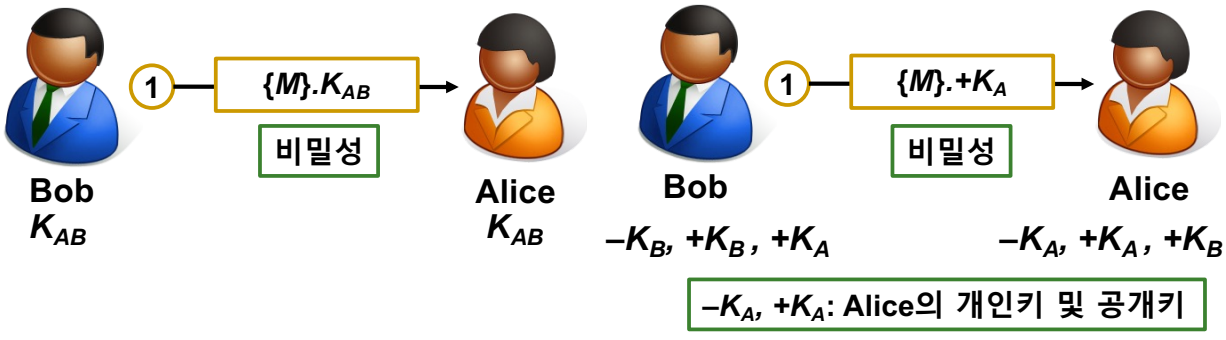
HDD ⟹ USB ⟹ Smart Phone ⟹ USIM

암호알고리즘의 분류 (1/2)

- 결정적(deterministic) vs. 확률적(probabilistic)
  - 결정적 알고리즘: 같은 입력이 주어지면 항상 같은 과정을 거쳐 같은 결과를 주는 알고리즘
  - 확률적 암호알고리즘: 같은 입력이 주어지더라도 다른 계산 과정을 거칠 수 있는 알고리즘 (다른 말로 무작위(randomized) 알고리즘)
    - 같은 입력에 대해 다른 계산 과정을 거치더라도 결과는 같을 수 있음
  - 암호알고리즘에서는 계산 과정보다는 출력 결과에 주목함
    - 안전성 측면에서 같은 입력에 대해 매번 다른 출력을 주는 것이 효과적임
      - 암호화 알고리즘은 확률적 알고리즘이 되어야 바람직함
      - 복호화 알고리즘은 출력이 달라지는 확률적 알고리즘이 될 수 없음
  - 결정적 암호알고리즘에 랜덤 요소를 추가하여 확률적 암호알고리즘으로 변경 가능
    - 안전성이 보장되도록 추가하는 것이 간단하지는 않음

암호알고리즘의 분류 (2/2)

- 대칭(symmetric) vs. 비대칭(asymmetric)
  - 대칭 암호알고리즘은 암호화할 때 사용하는 암호키와 복호화할 때 사용하는 암호키가 같은 암호알고리즘
  - 비대칭 암호알고리즘은 암호화할 때 사용하는 암호키와 복호화할 때 사용하는 암호키가 다른 암호알고리즘



대칭 암호알고리즘

- 대칭 암호알고리즘(symmetric, conventional, secret key algorithm): 암호화키 = 복호화키
  - 비밀키 암호알고리즘

비밀키의 생성: 암호학적 안전한 의사랜덤 함수를 이용하여 생성

- 암호화하는 사람과 그것을 복호화하는 사람은 같은 키를 가지고 있어야 함
  - 원격에 있는 두 사용자가 대칭 암호알고리즘을 이용하기 위해서는 사전에 안전하게 대칭키를 공유하여야 함 (어떻게 공유?)
- 종류
  - 스트림(stream) 암호방식: 평문의 각 바이트를 하나씩 암호화하는 방식으로 기본적으로 XOR 연산 이용
    - 대표 알고리즘: Salsa20

$C = P \oplus K,\ P = C \oplus K$

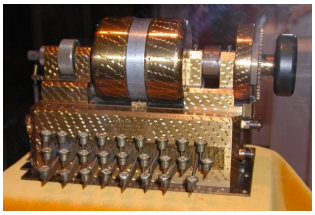
  - 블록(block) 암호방식: 평문을 일정한 블록 크기로 나누어, 각 블록을 암호화하는 방식
    - 암호화 함수 E는 PRP(Pseudo Random Permutation)임
      - $E: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X}$, n비트 문자열을 또 다른 n비트 문자열로 매핑
    - 대표 알고리즘: AES(Advanced Encryption System)

대칭 암호알고리즘의 역사

- Caesar Cipher: 기원전
  - 수에토니우스(Suetonius)의 저서 황제전에서 언급됨
  - A ⟹ D, B ⟹ E: shift 3, CRYPT ⟹ FUBSW
- Vigenere Cipher: 16세기
  - 키를 이용한 교체 암호

Plaintext:	ATTACKATDAWN
Key:	LEMONLEMONLE
Ciphertext:	LXFOPVEFRNHR

- Rotor Machine: 19세기
  - Hebern Rotor Machine
  - Enigma: 2차세계대전에서 독일군이 사용



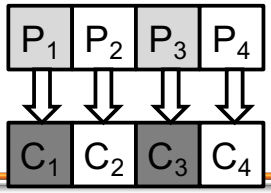
- One time pad: 1917 Vernam
  - $C = P \oplus K$
- 1970년대: DES, 키 길이 56비트
- 2010년:
  - AES(키 길이: 128비트), Salsa20

합성 암호 방식

- 현대 대칭 암호알고리즘은 합성 암호(product cipher) 방식을 사용함
- 합성 암호란 내부적으로 교체(substitution)와 자리바꿈(transposition) 연산을 모두 사용하는 것을 말함
  - 교체 연산: 한 값을 다른 값으로 바꿈
    - 예) LOVE ⇒ MPWF
    - 영문자를 다른 영문자로 교체하는 암호를 사용할 경우 가능한 키의 개수는?
  - 자리바꿈: 평문을 구성하는 요소의 위치를 바꿈
    - 예) LOVE ⇒ EVLO
- 이와 같은 연산을 여러 번 해야 안전성을 높일 수 있기 때문에 동일한 과정을 여러 번 반복하게 되며, 한 과정을 라운드라 한다.

암호화 모드

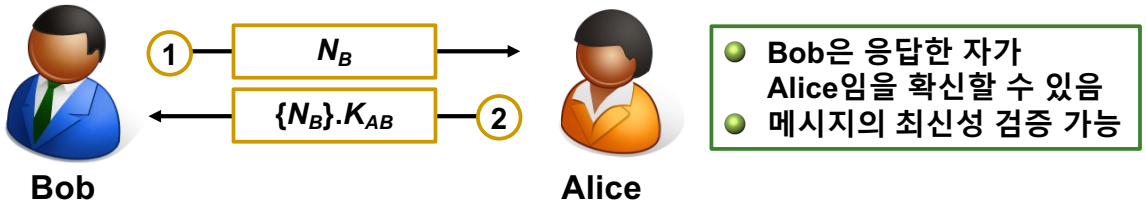
- 블록 암호방식을 사용하는 대칭 암호알고리즘은 항상 고정된 크기의 블록을 입력으로 사용함
  - 평문크기<블록크기: 평문크기를 블록크기로 만들기 위한 채우기(padding)가 필요함
  - 평문크기>블록크기: 평문을 블록크기 단위로 나누어 암호화하여야 함. 이 경우에도 마지막 블록은 채우기가 필요할 수 있음
- 암호화 모드(cryptographic mode)란 블록 암호방식에서 블록 크기보다 큰 평문을 암호화하는 방법을 말함 (Note 08)
- 가장 단순한 암호화 모드는 나누어진 개별 평문 블록들을 독립적으로 암호화하는 것이며, 이 모드를 ECB(Electronic CodeBook) 모드라 함
  - 이 모드는 결정적 암호알고리즘에서는 동일 평문 블록은 동일 암호문 블록으로 암호화되는 문제점이 있음 (정보 노출 문제)



if $P_1=P_3$ then $C_1=C_3$

대칭 암호알고리즘의 사용 용도

- 공개 채널로 전달하는 메시지에 대한 비밀성 보장
  - 송신자와 수신자가 공유하고 있는 비밀키로 메시지를 암호화하여 교환함으로써 제3자가 도청하여도 내용을 알 수 없도록 만듦
- 기억장치에 저장하는 데이터에 대한 비밀성 보장
  - 다른 사용자가 저장된 데이터를 열람할 수 없도록 데이터를 암호화하여 저장할 수 있음. 키의 백업이 중요함 (랜섬웨어)
- 개체 인증
  - Alice와 Bob이 비밀키 K_{AB}를 공유하고 있을 경우 다음과 같은 프로토콜을 통해 인증할 수 있음



N_B: nonce(Number used just ONCE)
$\{N_B\}.K_{AB}$: N_B를 K_{AB}로 암호화

법 강화를 위한 키 위탁

- 정부는 비밀 통신을 감청할 능력을 유지하고 싶음
  - 방법 1. 암호화 사용 금지
    - 예) 고수준 암호화 모듈 포함 수출 금지
  - 방법 2. 암호알고리즘을 해독할 수 있는 능력 보유
  - 방법 3. 키 위탁(key escrow): 사용자들이 사용하는 모든 키를 강제로 보관
- 세 번째 방법도 문제이지만 1과 2는 가능하지 않는 방법
- 세 번째 방법을 사용하기 위한 요구사항
  - 정부의 권한 남용을 방지하기 위한 대책 필요
    - 임계 기반 비밀 공유 기법 사용하여 제공 가능 (Note 01. 조건부 프라이버시)
- 키 위탁은 키 분실하였을 때 키를 복구(recovery)하기 위해서도 유용하게 사용될 수 있음

암호기술이 항상 좋은 목적으로만 사용하는 것은 아님

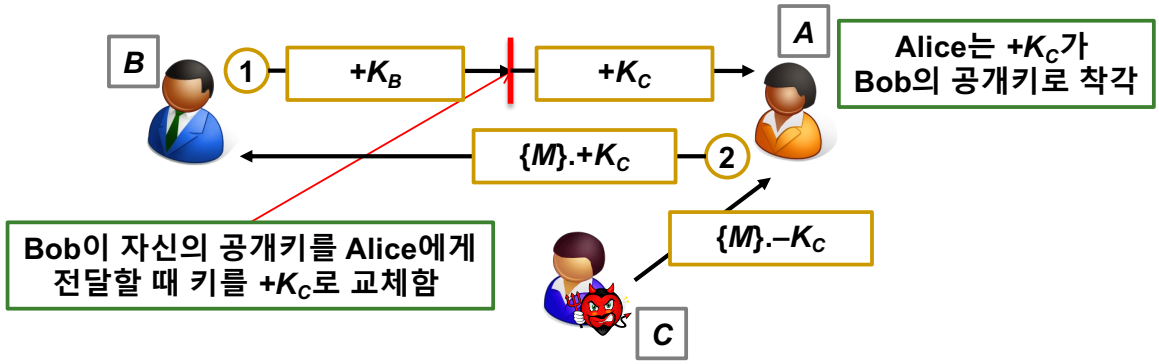
비대칭 암호알고리즘

- 비대칭 암호알고리즘(asymmetric, public key algorithm): 암호화키 ≠ 복호화키
  - 두 개의 키를 공개키(public key)와 개인키(private key)라 함
  - 각 사용자는 자신의 공개키는 공개하고(누구나 사용할 수 있음), 개인키는 비밀스럽게 유지함
  - 다른 말로 공개키 암호알고리즘이라 함
- 1976년 Diffie와 Hellman이 처음으로 개념 소개
  - 대칭 암호알고리즘에 비해 역사가 상대적으로 짧음
- 대칭 암호알고리즘과 달리 사전에 공유된 키가 없어도 메시지를 암호화하여 교환할 수 있음
  - 공개키의 인증이 가장 중요
- 대표 알고리즘: RSA(인수분해), ElGamal(이산대수), 타원곡선(이산대수)

> - 보통 암호학적 안전한 의사랜덤 함수를 이용하여 개인키를 생성한 후 그것을 이용하여 공개키를 생성함.
> - 거꾸로 생성할 수 있나? (슬라이드 25. 신원기반)

공개키 인증

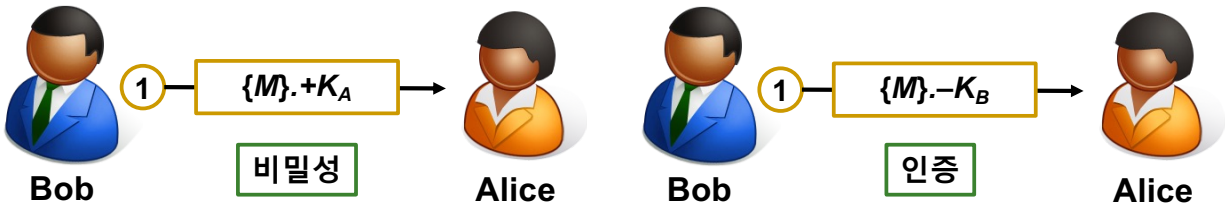
- 공개키 암호알고리즘을 활용하기 위해서는 반드시 보장되고 선행되어야 하는 것은 공개키의 인증임
  - 공개키의 인증이란 주어진 공개키가 누구의 공개키인지 확인하는 것을 말함
- 예) 공개키 인증 관련 문제



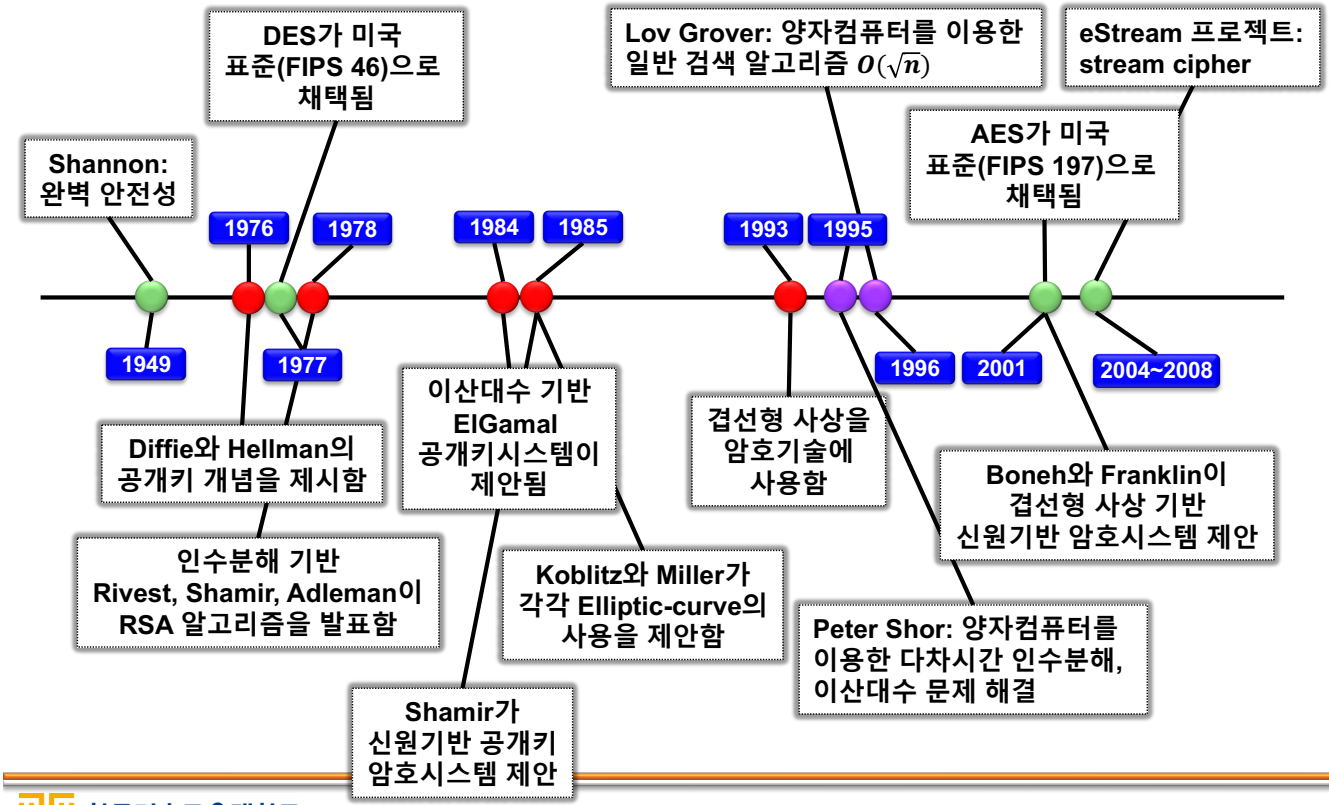
- 공개키의 인증을 제공하는 가장 대표적이고 널리 사용하는 메커니즘이 인증서(certificate)임

공개키 암호알고리즘의 사용용도

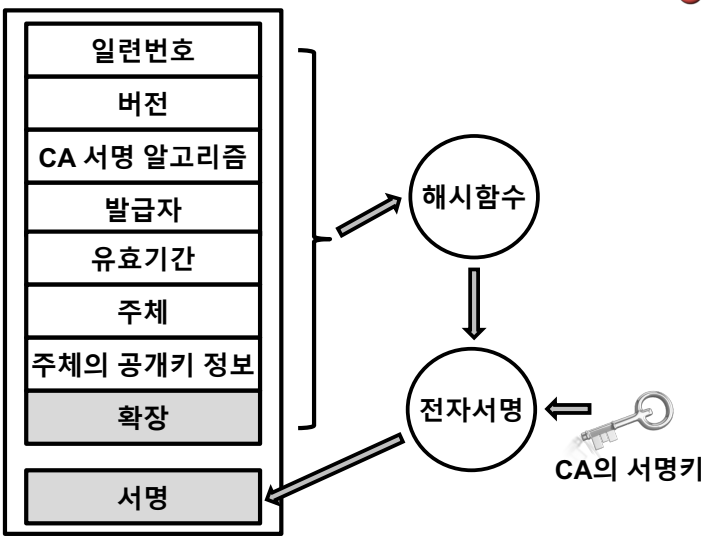
- 공개키 암호알고리즘은 대칭 암호알고리즘과 같은 용도로 사용 가능하지만 상대적 성능문제 때문에 보통 메시지 자체를 암호화하는데 사용하지 않음
- 이 때문에 다음과 같은 하이브리드(hybrid) 방식을 많이 사용함
  - $\{M\}.K$, $\{K\}.+K_A$
- 공개키와 개인키를 모두 사용하여 같은 알고리즘으로 메시지를 암호화할 수 있는 것이 있고(예: RSA), 그렇지 않은 알고리즘도 있음
  - 개인키를 사용하여 메시지를 암호화할 수 있으면 전자서명 알고리즘으로 활용 가능함. 개인키 암호화는 인증 서비스를 제공하기 위해 널리 사용함
  - 그렇지 않은 경우 키는 같지만 서로 다른 알고리즘을 사용함 (ElGamal, DSA)



암호알고리즘의 주요 milestone



인증서



- 인증서(certificate)는 공개키와 그것의 소유자를 바인딩시켜 주는 전자문서
  - Kohnfelder가 1978년에 처음으로 제안
  - 보통 신뢰할 수 있는 인증기관 (CA, Certification Authority)이 전자서명하여 생성함
  - 인증기관이 공개키를 공증해준다고 생각하면 됨
  - 표준: X.509 Version 3

공개키: 인증서에 유지함
개인키: 패스워드 기반 대칭 암호알고리즘으로 암호화되어 유지함

X.509 인증서는 DER라는 인코딩 방식으로 이진 파일에 저장하여 유지함
그런데 DER를 올바르게 parsing하는 것이 쉽지 않음.
이 부분을 악용하여 공격하는 경우도 있음

인증서의 검증

- 사용자들은 다른 사용자의 공개키를 사용하기 전에 인증서를 검증하여 공개키의 사용자를 확인해야 함
- 인증서를 검증하기 위한 절차
  - 단계 1. 인증서의 서명 확인
    - 발급 인증기관의 공개키 이용
  - 단계 2. 인증서의 유효기간 확인
  - 단계 3. 인증서의 사용 용도 확인
    - 예) 범용 vs. 전용, 서명 vs. 암호화
  - 단계 4. 인증서의 폐지 여부

- 위 네 가지 단계를 통해 한 번 확인된 인증서는 보통 캐시(또는 파일 시스템)에 보관하며, 이 인증서를 같은 용도로 다시 사용해야 할 경우에는 위 네 가지 단계를 다시 반복하지 않고 유효기간과 폐지 여부만 확인함

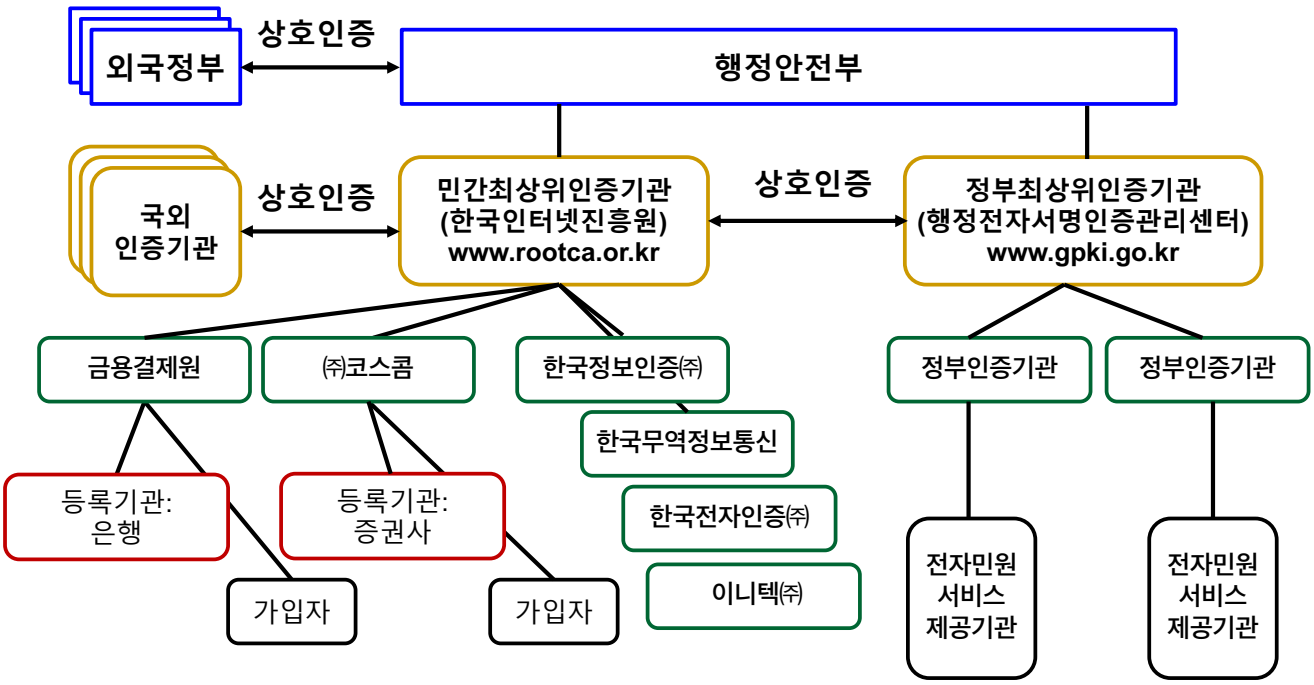
PKI의 신뢰 모델

- 인증서를 검증할 때 이 인증서를 발급한 기관의 공개키를 이용함
- 발급 기관의 인증서를 이용하기 전에 발급 기관의 인증서를 먼저 검증해야 함
- 그러면 발급기관의 인증서는 누가 발급하나?
  - 발급기관의 인증서를 발급하여 주는 상위 인증기관이 있음
- 인증서를 발급하는 기관들을 트리 형태로 표현할 수 있으며,
  이때 트리의 루트에 위치한 인증기관을 최상위 인증기관이라 함
  - 최상위 인증기관은 자체 서명 인증서를 사용함
- 전 세계의 모든 기관을 단일 계층구조로 관리할 수 없음
- 따라서 동등 관계에 있는 인증기관들은 상호 보증하는 형식을 취함
  - 이것을 전문 용어로 상호 인증(cross certification)이라 함

우리는 보통 **PKI**를 생각하면 (구)공인인증서를 생각하게 됨
하지만 해외에서는 **TLS(SSL, HTTPS)** 인증서를 생각함
웹은 계층구조 신뢰 모델을 사용하고 있지 않음
각 브라우저에는 브라우저가 신뢰하는 인증기관의 인증서 목록을 포함하고 있음

국내 인증체계

2020년 12월 10일 이전에는 일반 민간기업에서는
인증서를 발급할 수 없었음 ⟹ (구)공인인증서

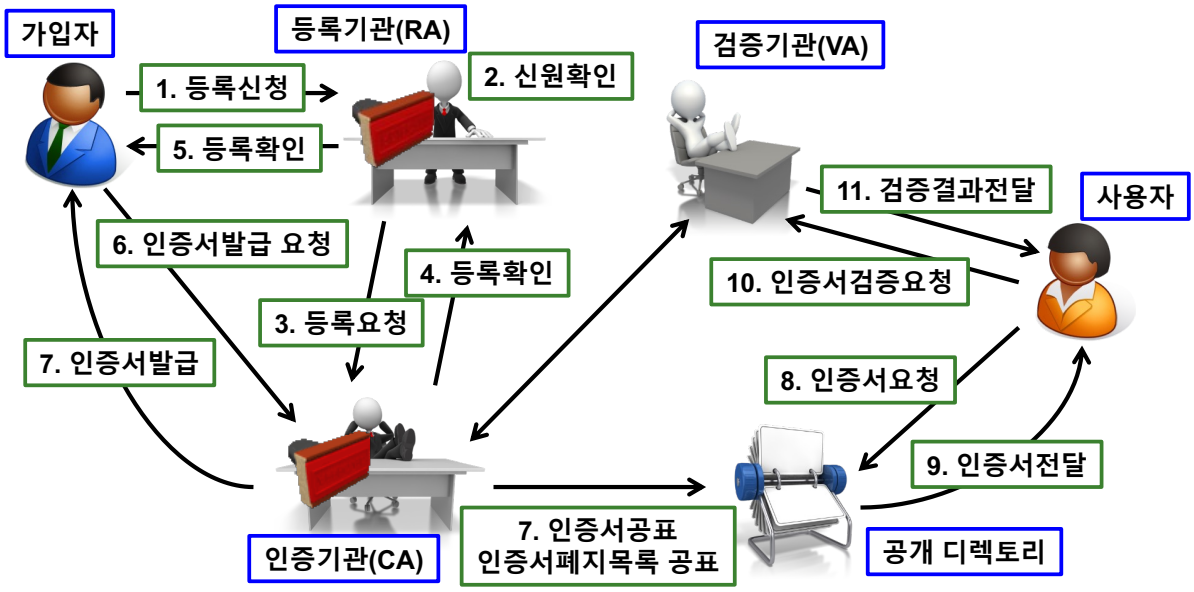


인증서 폐지

- 예) 신용카드
  - 신용카드도 유효기간이 만료되기 전에 분실/도난 등의 이유로 폐지해야 하는 경우가 있음
  - 초기에는 각 상점에게 폐지된 신용카드 번호 목록을 분배하였으며, 상점은 신용카드 거래를 승인하기 전에 이 목록에서 확인해야 함 (CRL)
  - 현재는 온라인으로 신용카드의 폐지 여부를 확인함 (OCSP)
- 인증기관은 주기적으로 폐지된 인증서 목록을 발급함
  - 인증서 폐지 목록(CRL, Certification Revocation List)
    - bad-list 기반 (비고. good-list, white-list)
  - 인증서와 마찬가지로 목록의 유효성을 확인할 수 있도록 서명하여 발급함
  - data aging: 유효기간이 지나면 목록에서 자동 삭제
- CRL ⇒ OCSP ⇒ OCSP stapling
- OCSP stapling: 인증서를 제공할 때 제공자가 OCSP 서버 확인서를 같이 첨부하는 형태

공개키 기반구조

- 인증서를 사용하기 위해서는 필요한 하드웨어, 소프트웨어, 인력, 정책, 절차를 공개키 기반구조(PKI, Public Key Infrastructure)라 함



신원기반 암호시스템 (1/2)

- 공개키 암호알고리즘의 사용에서 공개키 인증은 매우 중요하며, 이에 대한 표준 해결책으로 인증서를 사용하고 있음
- 인증서를 사용하기 위해서는 공개키 기반구조가 잘 확립되어 있어야 하며, 사용 비용이 그렇게 저렴한 것은 아님
- 인증서 외에 다른 방법에 대한 고민이 있었으며, 그 중에 가장 근접한 대안이 신원기반 공개키 암호시스템(identity-based public-key cryptosystem)임
- 신원기반 공개키 암호시스템은 1984년 Shamir가 처음 제안함
- 2001년에 Boneh와 Franklin은 겹선형 사상을 이용한 신원기반 시스템을 제안하였으며, 이 이후 연구가 다시 활발하게 진행되었음
- 하지만 신원정보 사용의 문제, 공개키 철회 문제 등 때문에 인증서 기반의 대안으로는 현장에서 활용하고 있지 못함
  - 지금은 블록체인 기반 DPKI가 대안으로 (슬라이드 31)

신원기반 암호시스템 (2/2)

- 기본 생각은 전자우편주소나 주민등록번호처럼 사용자의 잘 알려진 독특한 신원정보로부터 그 사용자의 공개키를 유도하여 사용함
  - 장점
    - 사용자는 다른 사용자의 공개키를 그 사용자의 신원정보를 이용하여 직접 생성할 수 있음
    - 기존 공개키 시스템과 달리 공개키와 사용자를 바인딩하여 주는 인증서가 필요 없음
- 신원기반 시스템에서는 통신하고자 하는 상대방이나 제3의 서버에 문의하지 않고 상대방의 신원을 알고 있다면 상대방의 공개키를 생성할 수 있음
  - 신원기반 암호시스템의 가장 큰 장점/목표

김상진 공개키 = F("sangjin@koreatech.ac.kr")

PKG

- 사용자는 자신의 개인키를 직접 만들 수 없음
  - 직접 만들 수 있다면 자신의 개인키 뿐만 아니라 다른 사용자의 개인키도 생성할 수 있다는 것을 의미함
- 이 때문에 신원기반 시스템에서는 사용자의 개인키를 발급하여 주는 PKG(Private Key Generator)라는 신뢰기관을 사용함
- 사용자의 개인키는 누구나 생성할 수 있는 해당 사용자의 공개키와 PKG가 비밀로 유지하는 마스터키를 이용하여 생성함
- PKG는 모든 사용자의 개인키를 만들 수 있으므로 모든 암호문을 복호화할 수 있고, 모든 사용자의 서명을 위조할 수 있음
  - 부인방지에 취약함
  - 해결책. 임계 기반 비밀 공유 기법 활용
    - 여러 기관에 발급 권한을 분배하여 n명의 기관 중 t명 이상이 협조해야 사용자의 개인키를 만들 수 있도록 함

김상진 개인키 = G(PKG 마스터 키, 김상진 공개키)

사용자의 신원정보 (1/2)

- 신원기반 암호시스템에서 사용자의 공개키는 그 사용자의 신원정보를 이용하여 생성하고, 대응되는 개인키는 PKG의 개인키(마스터키)와 사용자의 공개키를 이용하여 생성함
- 공개키를 생성하기 위해 사용하는 신원정보는 각 사용자마다 독특해야 함
  - 예) 주민번호, 전자우편주소 (이 두 정보의 차이점은?)
- 독특한 것이 필요조건이지만 충분조건은 아님
- 여권번호나 주민번호는 독특하지만 다른 사용자의 이와 같은 정보를 외우고 있지 않으며, 쉽게 얻을 수 없음
- 독특한 신원정보의 또 다른 특성은 보통 이런 정보들은 불변 정보임
  - 바꿀 수 없는 정보(주민번호)이거나
    바꾸기가 매우 불편한 정보(핸드폰 번호)임

사용자의 신원정보 (2/2)

- 이런 불변 정보만을 이용하면 나중에 공개키를 바꾸어야 할 때 다음 둘 중에 하나를 변경해야 함
  - 불변 정보인 사용한 신원 정보
  - PKG의 개인키

  하지만 전자는 본질적으로 변경할 수 없으며, 후자는 모든 사용자의 공개키 쌍의 변경이 필요함

김상진 개인키 = G(PKG 마스터 키, 김상진 공개키)

- 이 문제를 해결하기 위해 사용자의 공개키를 생성할 때 불변 정보 외에 변경이 가능한 동적 정보를 함께 포함할 수 있음
  - 예) 발급 시간
    - 하지만 상대방 공개키의 발급 시간을 알고 있어야 하는 문제점이 있음
    - 해결책. 고정된 발급시간을 사용함. 주기적 발급과 주기 내에서 변경이 힘듦

김상진 공개키 = F("sangjin@koreatech.ac.kr"||2023-03-01)

중앙집중 인증서 기반 PKI VS. 신원기반

	중앙집중 인증서 기반 PKI	신원기반
공개키 쌍의 생성	• 보통 사용자가 키 쌍을 생성 • RA에서 대신 생성할 수 있음	• 공개키: 누구나 신원정보로부터 계산할 수 있음 • 개인키: PKG가 발급
인증서의 필요성	• 공개키와 그것의 사용자를 바인딩하기 위해 절대적으로 필요 • 인증서가 필요하기 때문에 관련된 PKI 요소 필요	• 공개키 자체로 인증이 되므로 인증서가 불필요 • 인증서와 관련된 여러 PKI 요소가 필요 없음
부인방지 기능	• 사용자가 직접 개인키를 생성하면 강력한 부인방지를 제공할 수 있음	• PKG는 언제든지 사용자의 개인키를 계산할 수 있으므로 부인방지가 약함 • 해결책: threshold 기법 사용
철회 기능	• 필요 (CRL, OCSP 등)	• 필요하지만 신원기반의 장점을 퇴색하지 않으면서 철회 기능을 제공할 방법이 아직 없음
기타		• 온라인으로 개인키를 발급하기 위해서는 PKG는 사용자를 인증할 수 있어야 함. 어떻게?

DPKI(Decentralized PKI) (1/4)

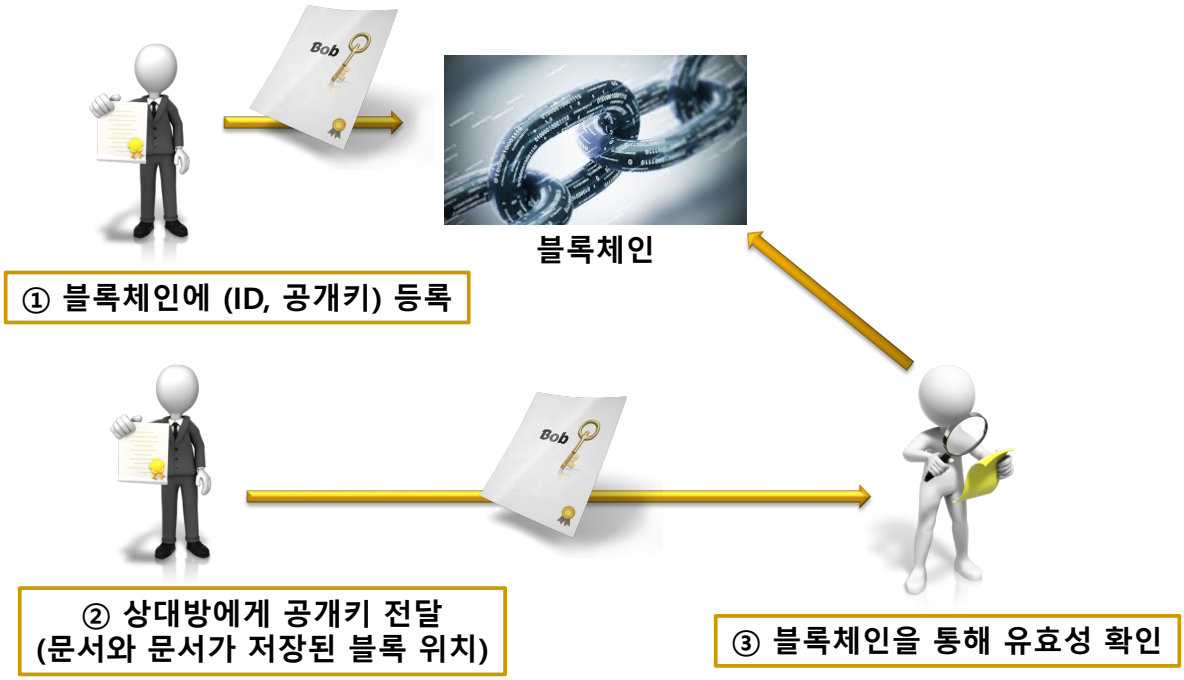
- 탈중앙 PKI
  - 중앙집중 인증기관이 인증서를 발급하는 것이 아님
  - 각 개체가 스스로 인증서를 발급하고 관리하는 형태임
    - 기존 PKI의 CA에 권한 집중 문제를 해결하는 것이 목적
    - 단일 실패점이 사라짐
- 동작 원리
  - 블록체인에 일정한 형식을 갖춘 ID와 공개키를 연결하는 문서를 저장함
    - 블록체인이 기존 PKI에서 공개 디렉토리 역할을 함
  - ID와 실제 개체와의 연결은 다른 방법을 사용해야 함
    - 보통 신뢰 관계를 점진적으로 확장하는 방법 사용 (web of trust)
      - 오프라인에서 서로 교환, 기관 홈페이지에 기관 ID 게시 등

WOT(Web of Trust) 모델은 이메일 보안을 위해 개발된 **PGP**에서 사용한 모델

TOFU(Trust On First Use) 모델: 처음에 무조건 신뢰, 그 이후에는 최초 키와 비교
>> 첫 신뢰 형성 때만 공격 가능

블록체인의 특징
1) 불가역성 (첨삭 전용 – 추가만 가능)
2) 분산 중복 저장
3) 합의 기술을 이용하여 일관성 유지

DPKI(Decentralized PKI) (2/4)



DPKI (3/4)

CPKI에서는 하나의 CA에 대한 공격에 성공하면 모든 사용자로 위장 가능
DPKI는 개별 사용자에 대한 공격만 가능

- 블록체인을 사용하는 이유? 역할?
  - 자동 분산 저장됨 (단일 실패점 제거)
  - ID와 바인딩된 공개키를 조작하기 힘듦
  - 조회의 용이성: 효과적으로 ID와 바인딩된 공개키를 얻을 수 있음
- 아무나 등록 가능하면 문제가 없나?
  - 분산 합의 기술에 의해 분산 합의된 데이터만 저장
    - 예) 중복된 ID의 저장이 가능하지 않음
- 갱신은 어떻게?
  - 블록체인은 첨삭 전용이므로 새 데이터의 추가를 통해 갱신
  - 개인키가 없으면 가능하지 않음 (TOFU)
- 폐지는 어떻게?
  - null 데이터를 등록하여 폐지함 (update를 통해)
- 개인키 분실이 가장 큰 문제임. 복구 기술이 필수임

DPKI (4/4)

목적. 유효 공개키 검색을 효과적으로 하기 위한 용도
여러 블록을 검색할 필요 없이 최종 블록에 포함된 축적값을 통해 확인할 수 있음

- Cryptographic Accumulator
  - 암호학적으로 안전한 집합 자료구조
    - 어떤 값이 축적값에 포함되어 있는지 확인할 수 있음
    - 축적되어 있는 값을 축적값에서 제거할 수 있음
    - 축적값에 포함되어 있지 않은 값이 포함되어 있다고 증명하는 것은 계산적으로 힘듦
  - 최종 블록에 모든 유효한 (ID, 공개키) 쌍이 포함되어 있는 축적값을 포함
    - 키 갱신: 축적값에서 기존 쌍 삭제 후 새 (ID, 공개키) 쌍 추가
    - 키 폐지: 축적값에서 (ID, 공개키) 쌍 삭제
  - 모든 사용자는 이 값을 이용하여 획득한 (ID, 공개키) 쌍이 유효한지 확인할 수 있음
    - 폐지 여부를 확인하는 용도로 활용 가능

FIDO

- FIDO(Fast Identity Online) 인증
  - FIDO에서 개발한 빠르고 간결하고 안전한 인증 방법의 집합
  - FIDO Alliance 참여 기업: 구글, MS, Mozilla, Yubico 등
- 사용자는 각 서비스마다 등록 과정에서 새로운 공개키 쌍을 생성하여 공개키를 서버에 안전하게 등록함
- 사용자의 클라이언트 SW는 개인키를 안전하게 보관함
  - 개인키 접근에 대한 다양한 인증 메커니즘을 사용할 수 있음
    - 현재 국내 많은 서비스는 지문을 활용하고 있음
- 등록된 공개키에 대응되는 개인키를 알고 있음을 증명하여 사용자를 인증함
- 앞서 살펴본 PKI와 달리 FIDO는 각 서비스마다 다른 공개키 쌍을 사용함