Chapter 2-2 : Cryptographic Tools

암호화 알고리즘

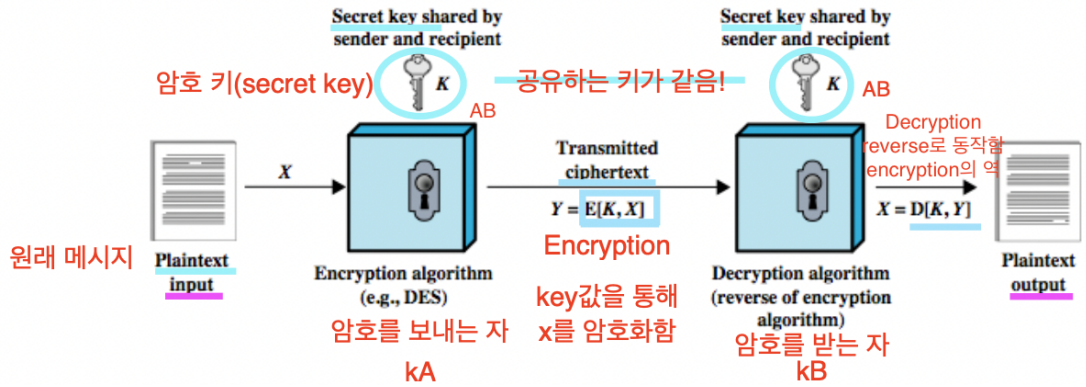
Objectives

cryptographic algorithms의 다양한 종류

- 대칭 암호화 (Symmetric encryption)
- Public-Key 암호화 (Asymmetric encryption) + Hash Algorithm
- Random Number

🔑 Symmetric Encryption

대칭 암호화 기법 - 대칭키 이용, 키가 하나임!



- (= = **conventional encryption**), (== **single-key encryption**) 이라고도 불린다.
- **Confidentiality를 보장함**. (shared-key를 갖고 있어야 하기 때문.)
  - **Plaintext(평문)** : input, original 데이터 또는 메시지

    - **Encryption algoritm(암호화 알고리즘)** : plaintext에 대해 다양한 변환과 치환를 수행함. → 치환 규칙만 활용하기 때문에 상대적으로 알고리즘이 간단함.
    - **Secret Key (”k”, 비밀키)** : 알고리즘이 수행하는 치환(subtitutions), 변환(transformations)은 이 키에 따라 달라짐. 또한 seed(입력)값에 따라 각 치환 및 변환값이 정해진다.
    - **Ciphertext(암호문)** : plaintext, secret key에 의해 생성된 output (암호화된 메시지)
    - **Decryption algorithm(복호화 알고리즘)** : 암호화 알고리즘을 역으로 실행 (입력 : 암호문, 비밀키, 출력 : 평문)

- **대칭 암호화의 안전한 사용을 위한 필요 조건**
    - 알고리즘이 복잡해야한다.
        - 입력 X가 보이지 않도록 해야 함. (black box로 된 알고리즘 강하게 만들기)
    - A와 B가 동시에 갖고 있는 key가 안전한 방식으로 공유돼야 함. (키가 길어야 하고, 정해진 방식은 없으나 어떻게 공유할지?)

아무 정보 없을 때의 공격 방식 ?

- 암호화된 결과값만 갖고 공격 (알고리즘이 강하지 않을 경우)
- 내가 만든 메시지를 직접 넣어 추론. 선택된 메시지 값으로 Ciphertext 만들어 추론하기

👊 Attacking Symmetric Encryption

대칭 암호화 기법을 공격하는 일반적인 방법

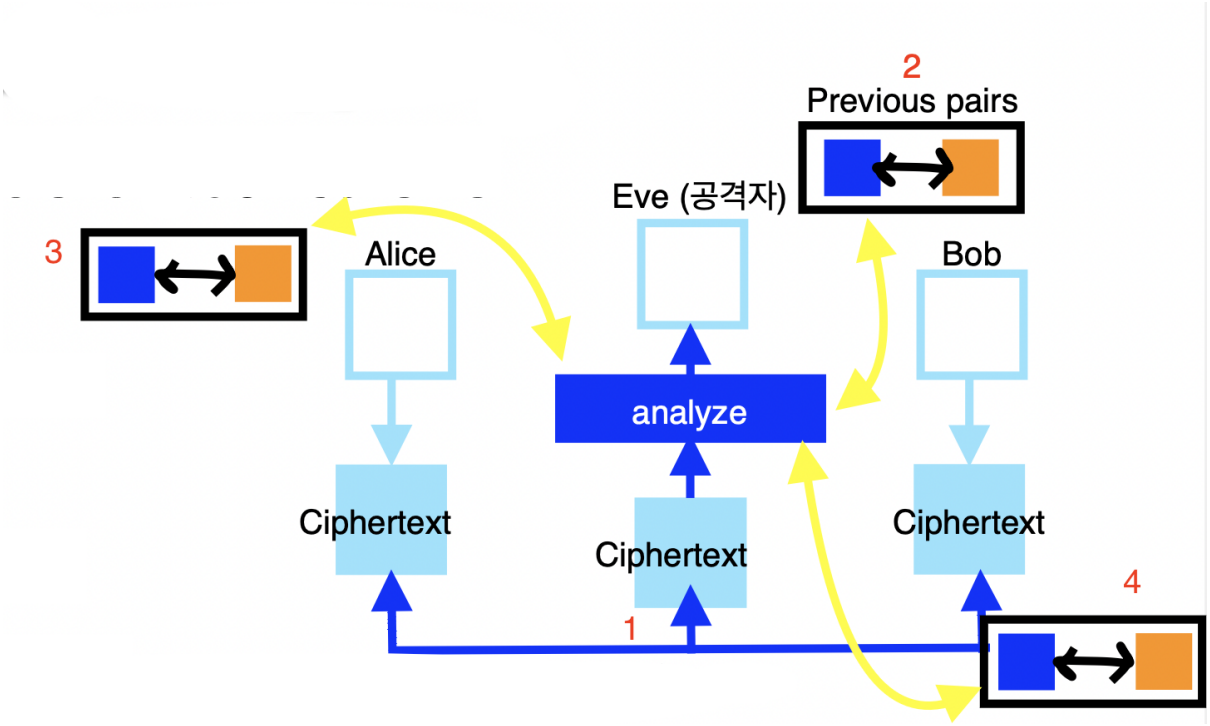
Brute-force Atack

- 무차별 공격
- Ciphertext에 관한 모든 키 값을 다 시도함 (경우의 수 모두 시도)
- 해석 가능한 문장이 얻어질 때까지

Cryptanlaysis

- 암호학적 방식(수학적)
- 알고리즘의 특징 + plaintext의 특징 두 가지 이용.
- 일부 예시인 “평문 - 암호문“ 몇 쌍이 유용한 샘플이 될 수 있음.
- 알고리즘의 기본적 특성을 이용하여 특정 평문 또는 키를 유추

Symmetric Encryption 공격 유형



1. **Ciphertext-only Attack (COA)**
    - 암호문 단독 공격
    - **가정** : 도청된 암호문만 주어짐
    - 가능한 **모든 키 적용** : 가장 간단, 키가 길면 유효하지 못함
    - **목적** : 공개된 암호문 및 복호화 **알고리즘에 대한 취약점 분석이 주된 공격대상**
2. **Known-Plaintext Attack (KPA)**
    - 기지 평문 공격
    - **가정** : 몇 쌍의 평문과 대응하는 암호문이 주어짐
    - **목적** : 도청된 암호를 해독하거나 적용된 비밀키 분석

3. **Chosen-Plaintext Attack (CPA)**
    - 선택 평문 공격
    - **가정** : 암호분석가가 구조 파악이 예상되는 평문을 선택
    - **목적** : 비밀키 분석
    - 가장 선호되는 분석 환경. 이 공격에서 안전하면 가장 이상적인 알고리즘(defense)이다.
4. **Chosen Ciphertext Attack (CCA)**
    - 선택 암호문 공격
    - **가정** : 암호 분석가가 선택한 암호문에 해당하는 평문을 얻을 수 있다. + 암호 분석가가 복호화 장치에 접근 가능한 상황
    - **목적** : 다른 관측된 암호문에 해당하는 평문을 도출함

Exhaustive Key Search

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s$ = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s$ = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ years	6.4×10^{6} years

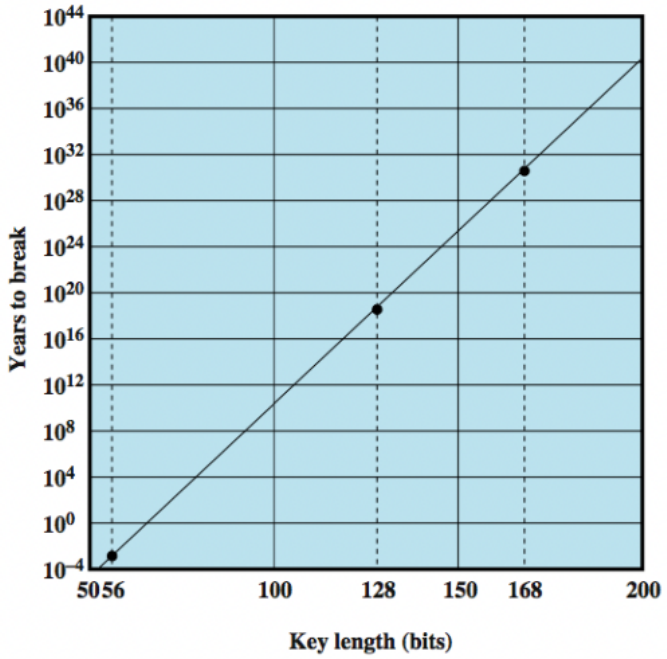
빨리 깨지는 알고리즘이 됨

< Time involved for various key sizes >

키 사이즈가 작을수록 빨리 깨지는 알고리즘이 된다

안전한 알고리즘이란?

- 가치만큼 깨지지 않는 알고리즘
- 모든 것이 어차피 깨질 수 밖에 없음.
- 최대한 늦게 깨지도록 막기 위해 bits 수 조정
- key를 최대한 길게 하면 가능 (key값이 중요) —> 하지만, 알고리즘의 복잡도를 고려해 key값이 너무 길어도 안됨.
- 128-bit가 default이고, 그보다 크게 쓰는 곳도 있다.



1년도 안 걸림!

< Time taken to crack a DES-style algorithm>

56-bit의 키는 더 이상 안전한 키 길이가 아니다.

Symmetric Encryption Algorithm

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

3*56, 2*56

DES = Data Encryption Standard
AES = Advanced Encryption Standard

- **대칭 암호화 알고리즘**에서 가장 많이 쓰이는 방식은 **Block Ciphers**(블록 암호화 - 고정된 크기의 블록 단위로 평문을 분할하고, 각 블록을 독립적으로 암호화)인 세 가지이다.
  - **DES(Data Encryption Standard)**
    - 2개의 안전한 알고리즘의 조건을 만족하지 못함.
    - 사용자는 64-bit 평문 블록과 56-bit 키를 사용 → 64-bit 암호문 블록 생성.

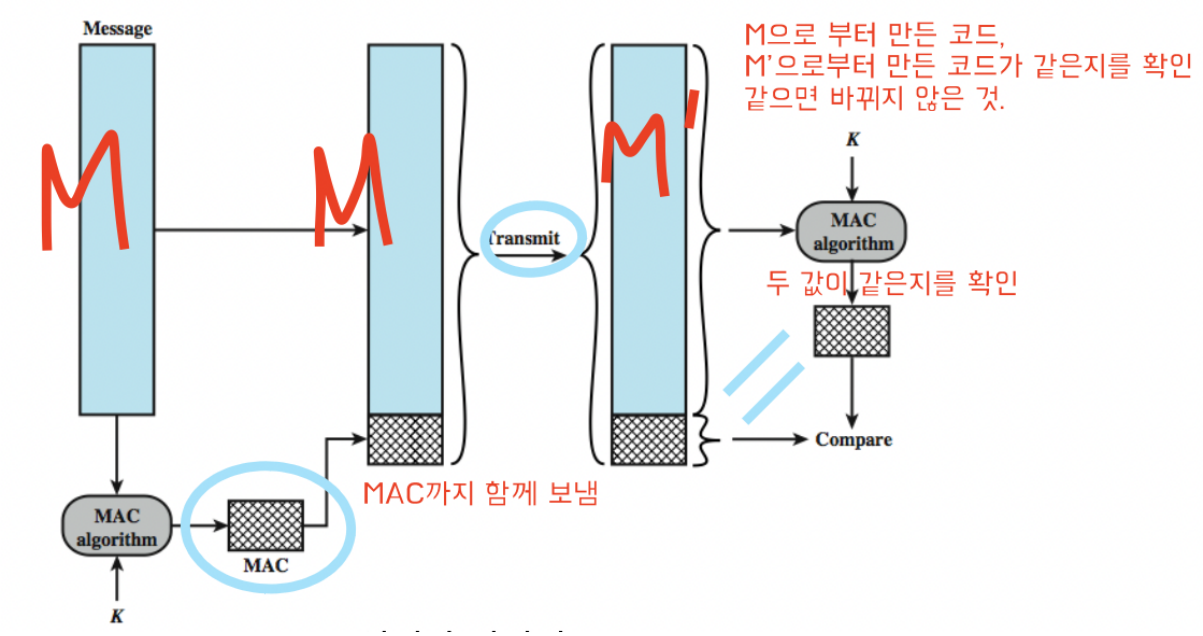
        - 알고리즘 & 56-bit 키 사용에 대한 우려 존재함.
    - **triple DES**
        - DES를 3번 반복해 사용해 조금 더 안전한 키 이지만, 3배로 속도가 느려진다.
    - **AES(Advanced Encryption Standard)**
        - 진보된 암호화 표준
        - 128-bit data & 128/192/256-bit keys
        - key도 늘리고, 알고리즘도 안전하게 만든다.

적용 - Message Authentication ✉️

- **Active attack**(공격자가 직접적인 변경 또는 손상을 가하는 것)으로부터 보호한다.
    - Ex) falsification of data and tansactions
- 받은 메시지가 진정성(진정한)있는 메시지인가?를 검증한다.
    - 내용이 변경되지 않도록 오류 탐지코드 등을 포함
    - 메시지가 '진짜' 송신자로부터 왔는지
    - 메시지가 본래의 순서를 맞춰 제대로 왔는지
- 메시지의 **시간성(timeless)**과 **순서(sequence)**를 검증하려면 전통적인 암호화 기법 or 별도의 인증 메커니즘을 사용할 수 있음
    - 오직 sender, receiver만 key가 필요함
    - 반면, 암호화를 의존하지 않는 별도의 인증(Authentication) 메커니즘을 사용하는 경우, 인증 태그를 일반 텍스트 메시지에 추가함.
        - Ex) 통신에서 사용. 메시지에 대한 originality를 검증하기 위한 알고리즘 구조. AES-CCM(Counter with CBC-MAC) : TLS에서 8바이트의 인증 태그 포함 (authentication tag)

Message Authentication Methods

MAC (Message Authentication Code)



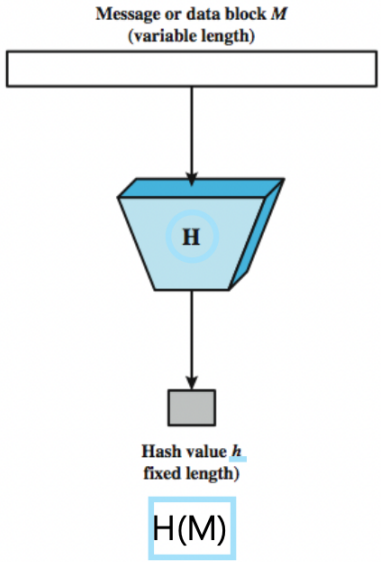
- 메시지 인증 코드
- 메시지 인증 기술 : 비밀 키를 사용해 메시지에 추가되는 일정한 길이의 작은 블록 데이터를 생성함
- A, B라는 쌍방의 통신 장치가 공통 비밀 키인 “Kab”를 공유함
- 작동 절차
  - A가 B에게 보낼 메시지가 있을 때, 메시지와 키를 함수로 계산하여 MAC을 계산

$$MAC_M = F(K_{AB}, M)$$

  - 메시지와 코드를 수신자에게 전송
  - 수신자는 수신된 메시지에 대해 동일한 비밀 키를 사용해 계산을 수행하여 새로운 인증 코드를 생성함
  - 수신된 코드가 계산된 코드와 일치하는지 비교
- 수신된 코드가 계산 코드와 일치하면
  - 수신자는 메시지가 변경되지 않았음을 보장함
  - 수신자는 메시지가 주장하는 송신자로부터 왔음을 보장함

One-Way Hash Function

단일방향 해쉬함수



MAC 대신 H(M)을 붙이기
- MAC과는 다르게 key를 기반으로 하는 동작 X.

- 해쉬 함수의 목적 → **메세지의 'fingerprint'** 지문 생성
- 메세지 인증 코드와 마찬가지로 **가변 크기의 메시지 'M'**을 입력받아, **고정 크기의 메시지 다이제스트 H(M)**을 출력
- MAC과는 달리, 해쉬 함수는 비밀키 입력 없음. (키 기반 동작 x)
- 메시지 인증을 위해 메시지 digest가 메시지와 함께 전송되어 인증되어야 함

Hash Funtion Requirements

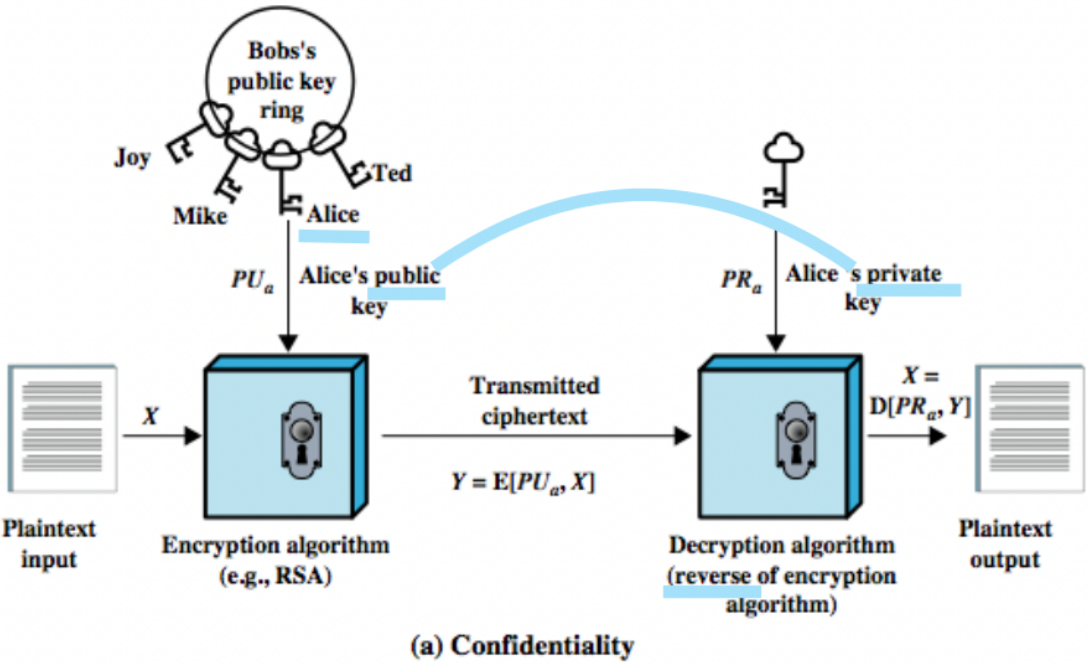
해쉬 함수 요구사항

- any size data에 적용 가능해야 함
- H → **고정 길이 출력**을 생성해야 함
- H(x) 계산이 비교적 쉬워야 함
- one-way property (단일 방향성)
  - **H(x) = h**를 만족하는 x를 찾는 것이 계산적/현실적으로 불가능함
- weak collision resistance (약한 충돌 저항)

    - *H(y) = H(x)인 y ≠ x* 를 찾는 것이 계산적/현실적으로 불가능함
        - → 서로 다른 입력값을 같이 계산할 수 없음
- strong collision resistance (강한 충돌 저항)
    - *H(x) = H(y)인 모든 쌍 (x, y)*를 찾는 것이 계산적/현실적으로 불가능함!

🔑🗝️ Public Key Encryption

공개 키 암호화



(a) Confidentiality

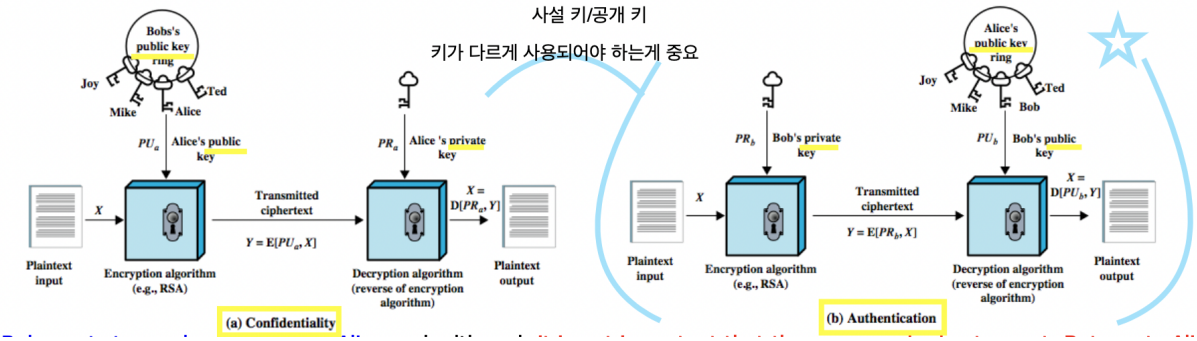
- 비트 패턴에 대한 간단한 연산 수행 x, 수학 함수에 기반
- Asymetric (비대칭적), 두 개의 별개 키를 사용함
    - 공개키 → 다른 사람이 사용가능하게 공개
    - 개인키 → 소유자만 알고있음
- **두개의 키 사용**으로 **confidentiality(기밀성), key distribution(키 분배), authentication(인증)**에 깊은 영향.
    - Public(A) → Private(A) : **Confidentiality**(기밀성)

    - Pricate(B) → Public(B) : **Aythentication**(인증) - B가 보낸 사람이라는 것을 인증해줌.

- 포함 요소
    - **Plaintext(평문)** : 입력, 읽기 가능한 메시지나 데이터
    - **Encryption Algorithm(암호화 알고리즘)** : 평문에 대해 다양한 변환을 수행함
    - **Public Key(공개키) / Private Key(개인키)** : 한 쌍이 됨. 하나가 암호화에 사용 - 다른 하나는 복호화에 사용
    - **Ciphertext(암호문)** : 평문과 키에 따라 생성된 암호화된 메세지
    - **Decryption Algorithm(복호화 알고리즘)** : 암호화 알고리즘을 역으로 실행. 암호문과 키를 사용해 원래의 평문을 생성. 복호화는 Private key로만 가능함.

Public Key Authentication

공개 키 인증



- **가정** : Bob은 Alice에게 메시지를 보내고, 메시지가 비밀로 유지되어야 할 필요는 없지만, 밥이 메시지를 보냈다는 것을 앨리서가 확실히 인증하길 원함.
- **Authentication (인증)**
    - User가 Bob의 public 키를 사용해 Ciphertext(암호문)을 성공적으로 복원할 수 있다면, 이것은 Bob 만이 평문을 암호화한 것을 나타내므로 인증됨.
- **Authentication / Data Integrity (인증, 데이터 무결성)**
    - Bob 외의 다른 사람은 Bob의 private 키로 평문을 암호화할 수 없으므로, 아무도 평문 수정이 불가능함.
    - 이는 밥만이 해당 메시지의 **무결성 (integrity)를 보장함을** 의미함. → 메시지가 전송되는 동안 누구도 메시지를 변경할 수 없음을 보증

- 전체 메시지를 암호화하는 방법은 Author, Contents를 모두 확인함. But **많은 저장 공간 및 추가 처리 비용**이 필요하다.
- 더 효율적으로 동일한 결과를 내려면, 문서의 함수로 이루어진 작은 블록 (Authenticator)를 암호화 하는 것
    - 인증자가 발신자의 private key로 암호화 된 경우, 이는 **발신자, 내용, 순서를 검증(verifies origin, content, sequencing)하는 signature(서명)으로 작동**함
    - SHA-1과 같은 안전한 해쉬 코드 (hash code)를 함수로 사용 가능
    - **Signature**는 **기밀성 (confidentiality)을 제공하지 않는다는 것을 강조하는** 것이 중요함. (전송되는 메시지는 변경으로부터 안전한 반면 **도청으로부터는 안전하지 않다!**)

공개 키 암호화의 필요조건

- Pub(공개키), Prb(개인키)의 키 쌍을 생성하는 것이 계산적으로 쉬워야 함
    - 계산하는 것은 쉽게, 복호화하는 것은 어렵게
- **sender**가 PUb를 알고 있다면, 메시지를 **암호화하는 계산이 쉬워야 함**
    - ***C=E(PUb, M)***
- **receiver**가 PRb를 알고 있다면, 암호문을 **복호화하는 계산이 쉬워야 함**
    - ***M=D(PRb, C)***
- 공개 키(PUb)를 알고 있더라도, 개인 키(PRb)를 결정하는 계산이 **불가능해야** 함
- 공개 키(PUb)와 암호문(C)를 알고 있더라도, 원래의 메시지 M을 복원하는 계산이 **불가능해야** 함
- 두 키중 하나를 사용해 암호화하고, 다른 하나 사용해 복호화할 수 있는 경우에 유용

Public Key Algorithms

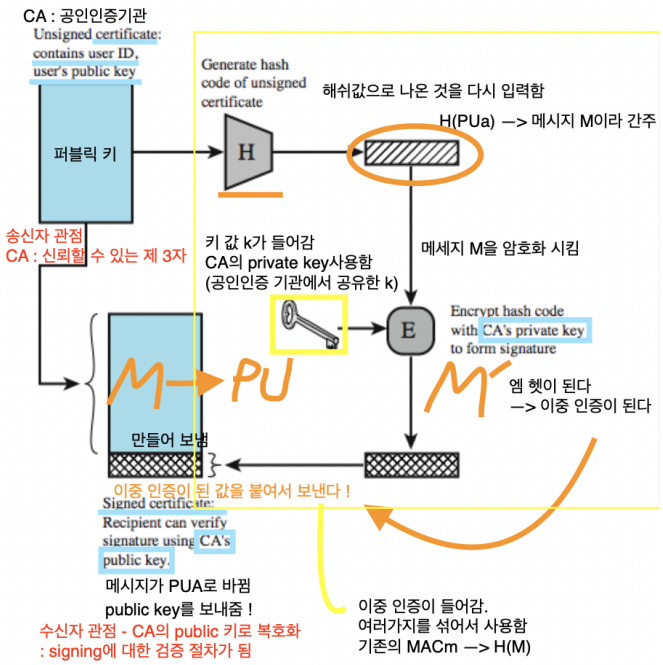
공개 키 알고리즘

- **Diffie-Hellman 키 교환 알고리즘 (DH)**
    - Confidentiality에서 주로 사용.
    - 1976년 개발

    - 비밀 키만 교환 가능함
- **RSA (Rivest, Shamir, Adleman)**
    - 1977년 개발
    - 널리 인정받는 공개 키 암호화 알고리즘. 현재 사용됨
    - 평문, 암호문이 0 ~ n-1까지의 정수인 블록을 암호화
    - 현재는 1024비트의 키 크기 (약 300자리)로 강력함 - 키 값이 커야 암호화 강도 높음
- **Elliptic Curve Cryptography (ECC)**
    - 타원 곡선 암호
    - RSA와 같은 보안성, 더 작은 키 크기
    - ECC는 아직 필드에서 검증되지 못함 (신뢰 그닥)
    - 키 값은 짧지만 좌표평면을 다르게 만들어 작은 키 사이즈로 동일한 완강도를 제공
- **디지털 서명 표준 (DSS)**
    - NIST가 FIPS PUB 186 발행
    - SHA-1과 함께 디지털 서명 기능만 제공.
    - Hash 표준화 알고리즘과 같음. 현재는 SHA-3

적용 1 - Public Key Certificates (공인인증서)

공인인증서 (Unsigned Certificate)

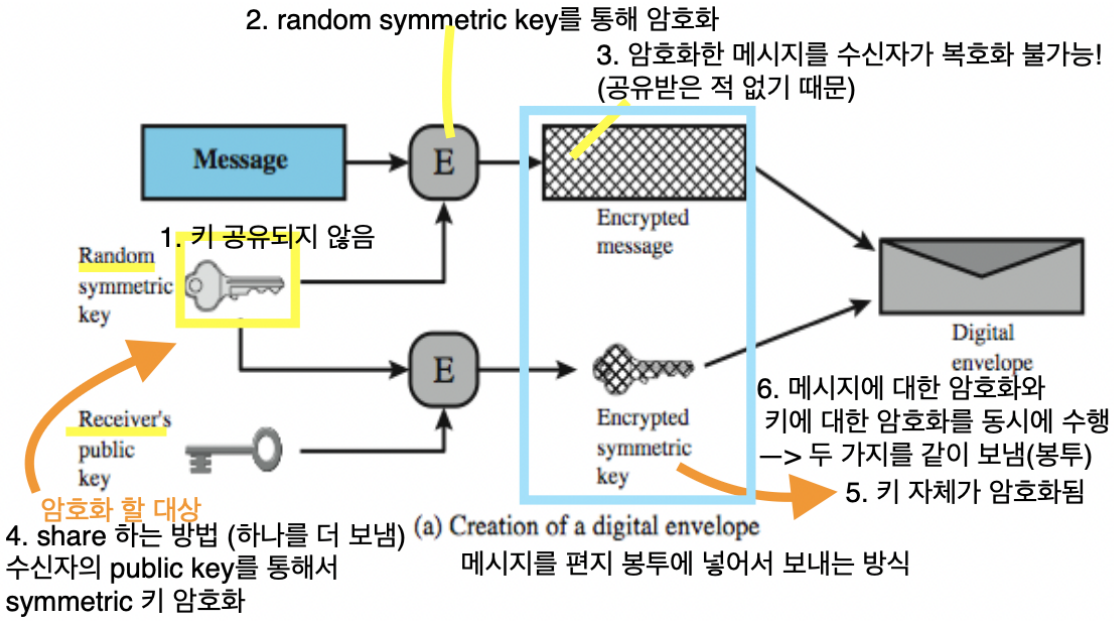


- 공인인증 기관에서 Signed Certificate를 만들어 보냄. 해쉬 값에서 하나만 쓰지 않고 섞어 사용한다
- 키 배포 문제를 해결하기 위한 두 가지 측면
  - 공개 키의 진위성
    - 이유 - 공개 키는 공개적이지만, 특정 사용자의 공개 키를 갖고 있는지 / 위조된 키를 갖고 있는지 알아내는 것이 문제
    - 해결책 - **공개 키 인증서**는 신뢰가능한 제 3자가 서명한 **공개 키**와 키 소유자의 **사용자 ID**로 구성된 공캐 키 블록. 사용자는 자신의 공개 키를 인증기관에 안전한 방법으로 제출하고 인증서를 받을 수 있음. 또한 사용자는 인증서를 공개할 수 있음
      - 공인인증 기관을 통해 생성된 공인인증서에 Public Key를 집어넣어, 해당 인증서가 진짜 Bob의 public key인지를 인증해줌
  - 공개 키 암호화를 사용해 **비밀 키** 배포
- Ex) 제 3자/공인인증기관 (CA: - Bob, Alice가 모두 신뢰할 수 있는지)
  - Bob, Alice는 CA의 Public key가 공유됨
  - CA는 자신의 Private key를 통해 암호화 가능

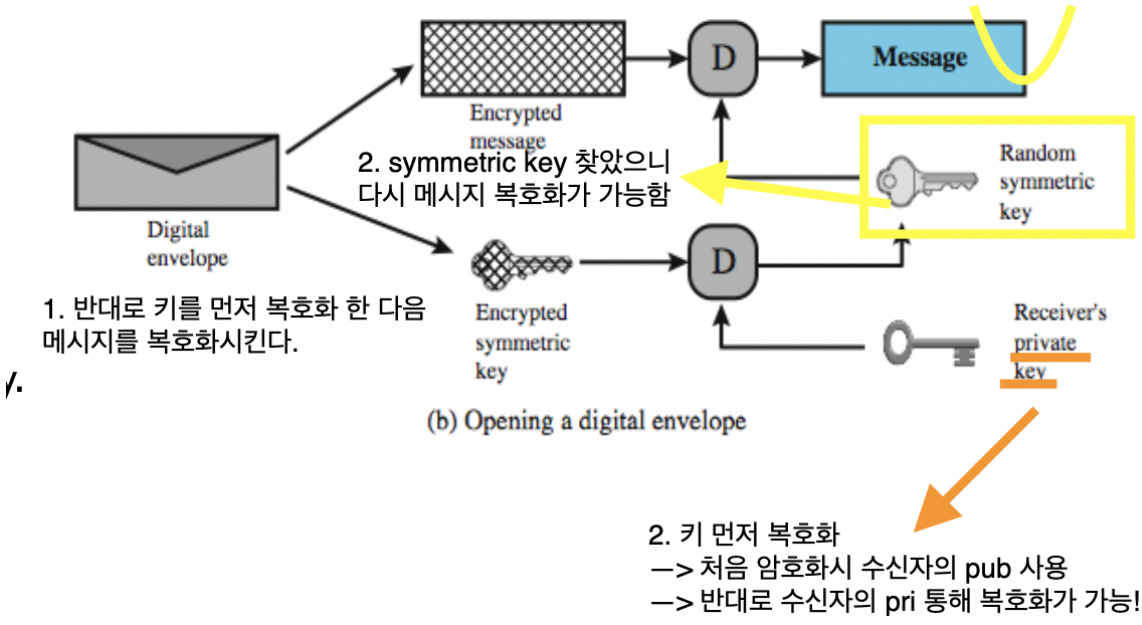
    → Bob과 Alice는 공유된 Public Key를 통해 메시지를 복호화 할 수 있음!

적용 2 - Digital Envelopes ✉️

public key의 분배에 관하여 - public key, symmetric key가 동시에 동작하도록 만듦. (이중 암호화)



암호화 : 디지털 봉투 만들기



복호화 : 디지털 봉투 열기

- 핵심 - 메시지 M을 복원할 수 있어야 한다

키 배포 문제를 해결하는 두 가지 측면

- **공개 키의 진위성을 보장하는 것**
- **디지털 봉투 - 공개키 암호화를 사용해 비밀 키를 배포하는 것**
  - sende, receiver간 동일한 비밀 키를미리 정해놓을 필요 없이 메시지 보호 가능
  - Ex) **Bob이 Alice에게 기밀 메시지**를 보내려고 할 때, Bob은 다음과 같은 과정 수행
    1. 메시지 준비
    2. 일회용 session key로 메시지를 전통적인 암호화 방식으로 암호화
    3. Alice의 공개 키를 사용해 session key를 공개 키 암호화로 암호화함
    4. 암호화된 session key를 메시지에 첨부하고 Alice에게 보냄
