제 5 장. 전자 서명

컴퓨터시스템보안

금오공과대학교 컴퓨터공학부

최태영

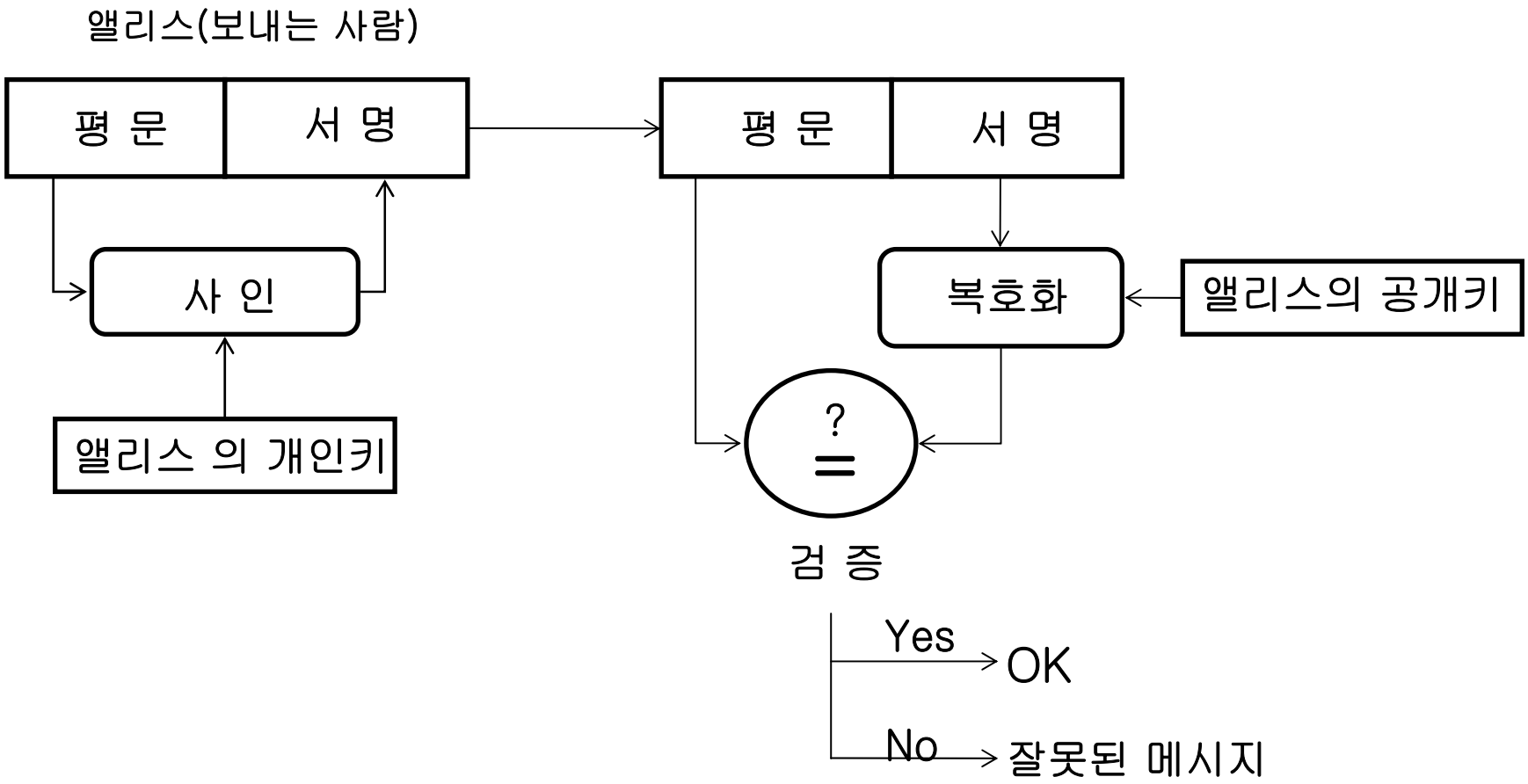
무결성 및 인증의 필요성

- 무결성 보장 필요
  - Ex) 인터넷 뱅킹의 계좌이체 시 패킷 내의 금액 변경
- 인증 보장 필요
  - Ex) 공격자가 사용자인척하여 계좌이체
- 암호화된 패킷의 무결성/인증 보장여부
  - 복호화한 것이 읽을 수 있는 평문이거나,
    - 컴퓨터가 판독하기에는 어려울 수 있음
  - 평문이 특정한 형태를 가지고 있거나,
  - 제한된 알파벳만을 사용하는 경우
  - 공격자가 위장을 하는 경우 위의 방법으로 막을 수 없음.
- 서명
  - 기존 문서 체계에서 그 문서의 작성자임을 입증하는 방법
- 기존 서명의 디지털화
  - 컴퓨터에 마우스나 전자펜 등으로 서명을 하면 효력이 있는가?

공개키를 사용한 전자 서명

- 서명에 대한 다른 관점
  - 서명한 필체는 유일하다. → 당사자가 소유한 유일한 것
  - 개인키 : 노출이 안된 유일한 전자적 정보
- 공개키 시스템에서의 서명
  - 문서를 개인키로 암호화하는 행위
- 공개키 시스템에서의 검증
  - 서명된 문서를 서명한 사람의 공개키로 복호화하여 확인하는 행위
  - 평문이 읽을 수 있는 문서이면 그 문서를 개인키로 암호화하고 공개키로 복호화하여 읽을 수 있는지 여부를 파악하면 됨
  - 평문이 읽을 수 없는 문서이면 비교할 정보가 추가로 필요함

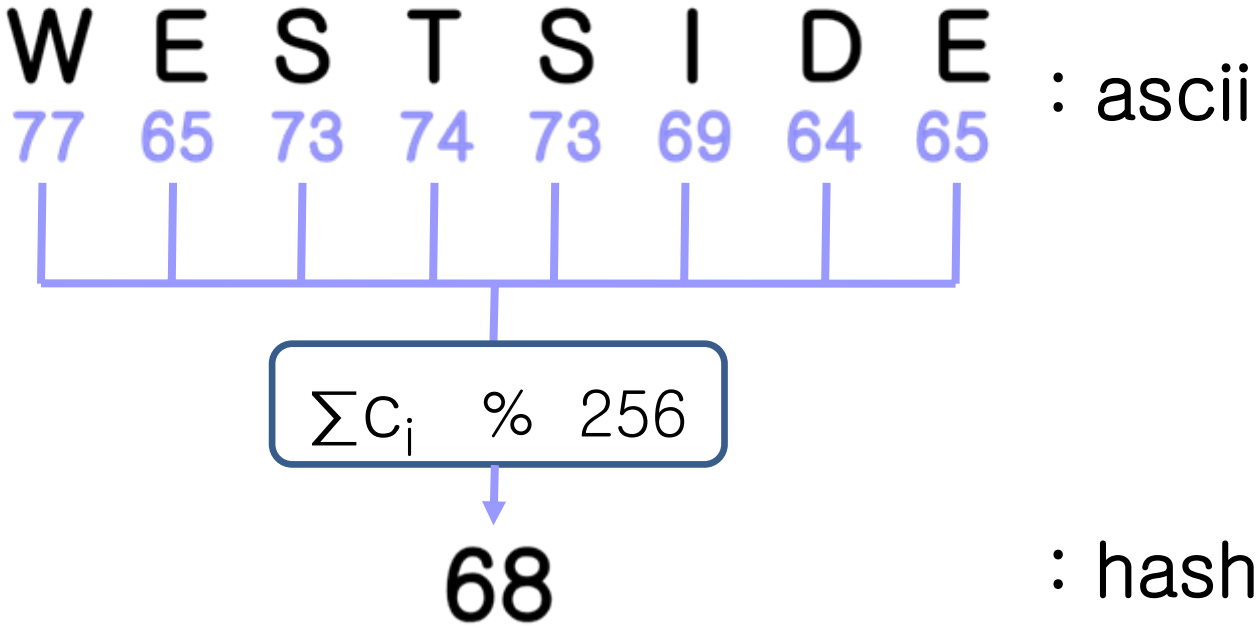
공개키 전자 서명 및 검증 과정



해싱

- 큰 메시지의 개인키 암호화는 긴 수행 시간을 요구함
- 해싱 (hashing) : 주어진 임의 크기의 메시지를 정해진 크기의 요약이나 지문으로 만드는 과정
- Message digest : 해싱에 의해 만들어진 일정 크기의 정보, 해시 (hash)라고도 불림
- CRC (Cyclic Redundancy Check)와 유사
  - CRC는 우발적인 변형 확인이 목적
- 다양한 공격에 견딜 수 있도록 설계되어야 함

해싱의 간단한 예제



컴퓨터 보안을 위한 해시의 조건

- 해시 함수 h()는 다음 조건들을 만족하는 것이 바람직하다.

1. **함수**: 메시지 m = n 이면 해시 h(m) = h(n)
2. **효율성**: 메시지 m에 대해 해시 h(m) 생성이 용이
3. **단방향성**: 해시 h(m)에 대해 메시지 m 구성이 곤란
4. **약한 충돌 방지**: 메시지 m과 해시 h(m)에 대해, h(m) = h(n)이고 m ≠ n인 n을 찾기가 곤란
5. **강한 충돌 방지**: h(m) = h(n)이고 m ≠ n을 만족하는 m과 n을 찾기가 곤란

단방향성과 충돌방지

- 단방향성
  - 해시는 원본 데이터의 내용이 거의 대부분 파괴되었기 때문에 복구하는 것은 매우 어려움
  - 하지만 원본의 패턴으로부터 추측에 의한 복구가 가능
- 약한 충돌 방지
  - '단방향성'과는 달리 원본과 다른 내용을 가지지만 같은 해시를 가지는 데이터를 만드는 것을 막는 것이 목적
  - Ex) 정식 문서가 있는 상황에서 같은 해시를 가지는 위조 문서를 만드는 것이 목적
- 강한 충돌 방지
  - 원본이 정해져 있지 않으며, 공격자가 원본과 위조를 모두 만들 수 있는 지위에 있을 때 이러한 공격을 막는 것이 목적
  - 위의 공격은 원본 문서가 정해져 있을 경우보다 공격이 매우 쉬움
  - 생일 문제 (birthday problem)은 이 공격에 대한 인용으로 많이 사용됨

생일 문제

- 몇 명이 모였을 때 필요한 확률만큼 생일이 같은 사람의 쌍이 생길 수 있는가에 대한 문제
- N명이 있을 때, 앨리스와 생일이 같은 사람이 있을 확률

$$1-\left(\frac{364}{365}\right)^N$$

- 앨리스와 생일이 같은 사람이 있을 확률이 50%를 넘기려면 몇 명 이상이 있어야 하나?

$$1-\left(\frac{364}{365}\right)^N \geq \frac{1}{2}$$

- $N \geq 252.7$
- 생일이 같은 사람이 있을 확률

$$1-\frac{364}{365}\frac{363}{365}\cdots\frac{365-N+1}{365} \geq \frac{1}{2}$$

- $N \geq 23$

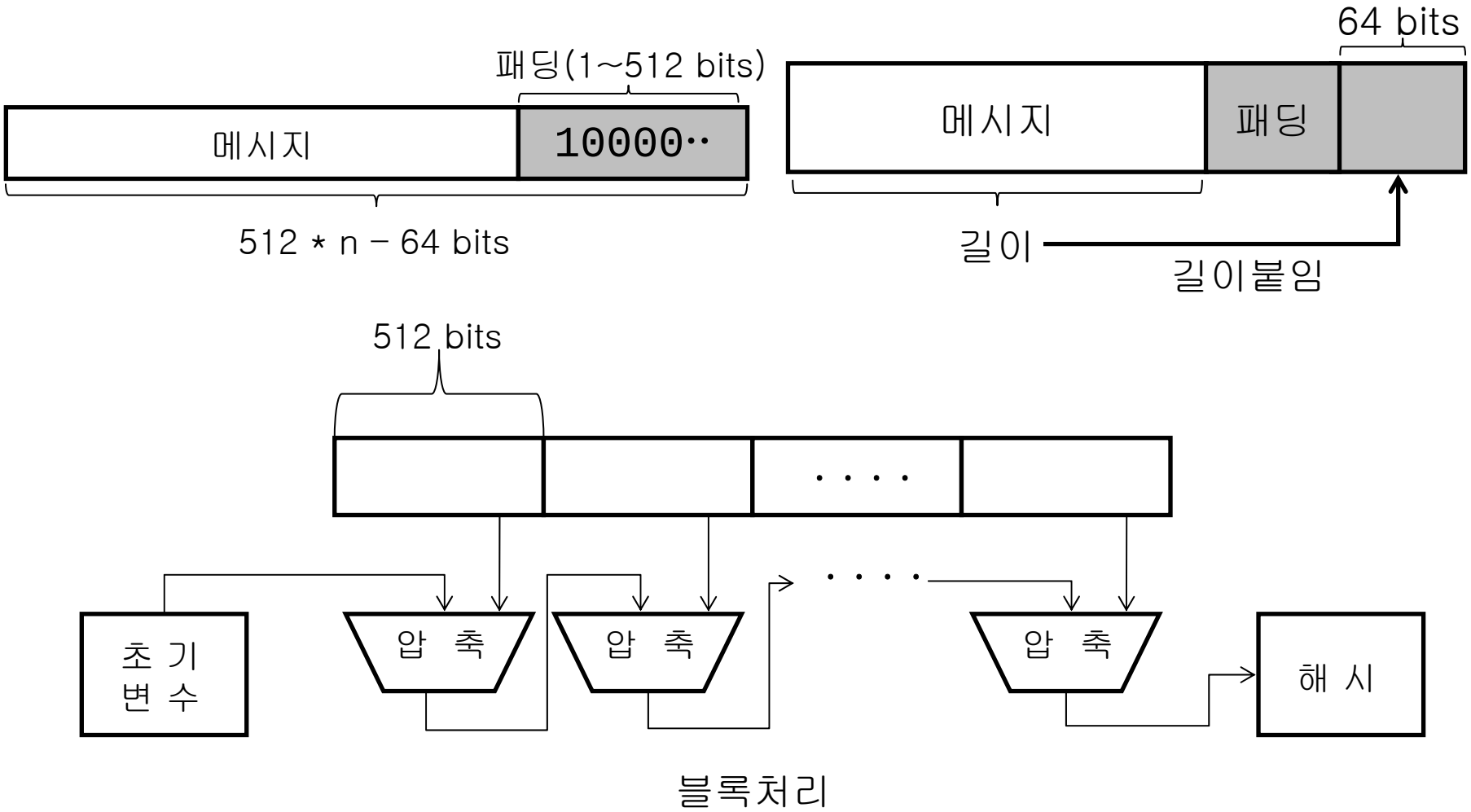
생일 문제 (계속)

- N : 특정인과 생일을 비교하는 경우의 수
- ${}_{N}C_{2}$: 특정인 없이 생일을 비교하는 경우의 수
  - $N(N-1)/2 \approx N^2$
- 같은 데이터에 대해 경우의 수가 제곱이 된다는 것은 필요한 데이터의 개수가 제곱근이라는 말이 된다.
- 해시에 적용 (크기가 k 비트인 해시)
  - 가능한 경우의 수는 2^k
  - 평문 m과 그 해시 h(m)에 대해 같은 해시를 가지는 다른 평문 시도 회수 : 평균 2^{k-1}
  - 해시가 같은 평문 2개 시도 회수 : 평균 $2^{k/2-1}$

MD5

- 1991년에 Ronald Rivest가 고안
- MD, MD2, MD4의 후속 해시 알고리즘
- 32 비트 연산들로 구성
- 128 비트 크기의 해시를 생성
- 입력 메시지는 512 비트 단위로 처리
- 메시지 길이에 제한은 없음
- 최근에 심각한 결함이 발견됨
- 동작
  - 패딩 (padding)
  - 길이 붙임 (append length)
  - 블록 분할 및 변수 초기화
  - 블록 처리

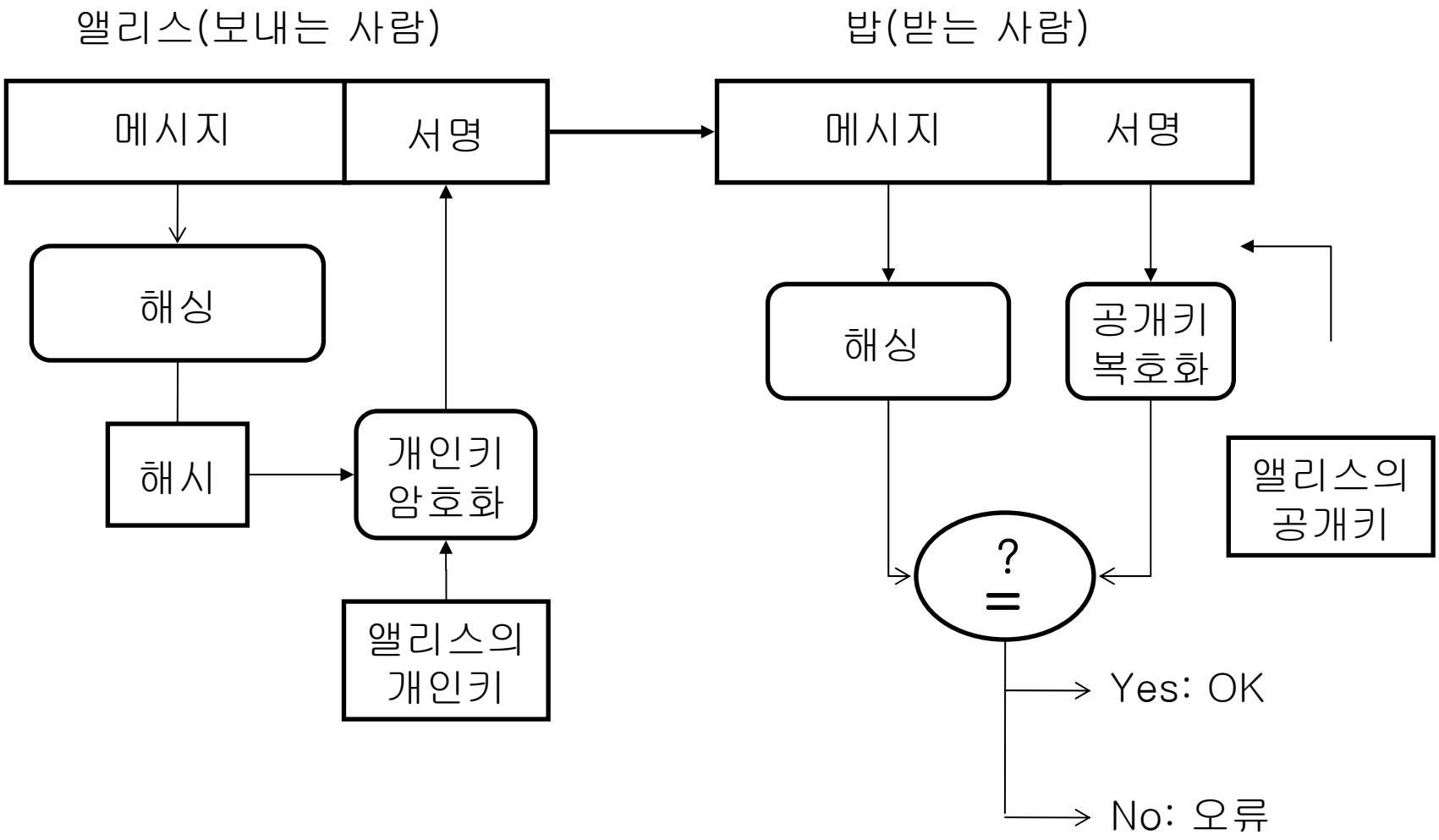
MD5 (계속)


패딩(1~512 bits)
메시지
10000··
512 * n – 64 bits
64 bits
메시지
패딩
길이
길이붙임
512 bits
초 기
변 수
압 축
압 축
압 축
해 시
블록처리

SHA (Secure Hash Algorithm)

- 1993년 미국 NIST에서 발표한 미국 해시 표준
  - SHS (Secure Hash Standard)로도 불림
- SHA-0, SHA-1, SHA-2 세 종류가 있음
- 최대 $2^{64}-1$ 비트 길이의 메시지 처리
- SHA-0/1은 160 비트 해시 생성
  - SHA-2는 224, 256, 384, 512 크기의 해시 생성
- SHA-1은 MD와 유사한 구조이지만 4번째의 블록처리 부분이 다름
- SHA-2는 다른 구조로 설계됨

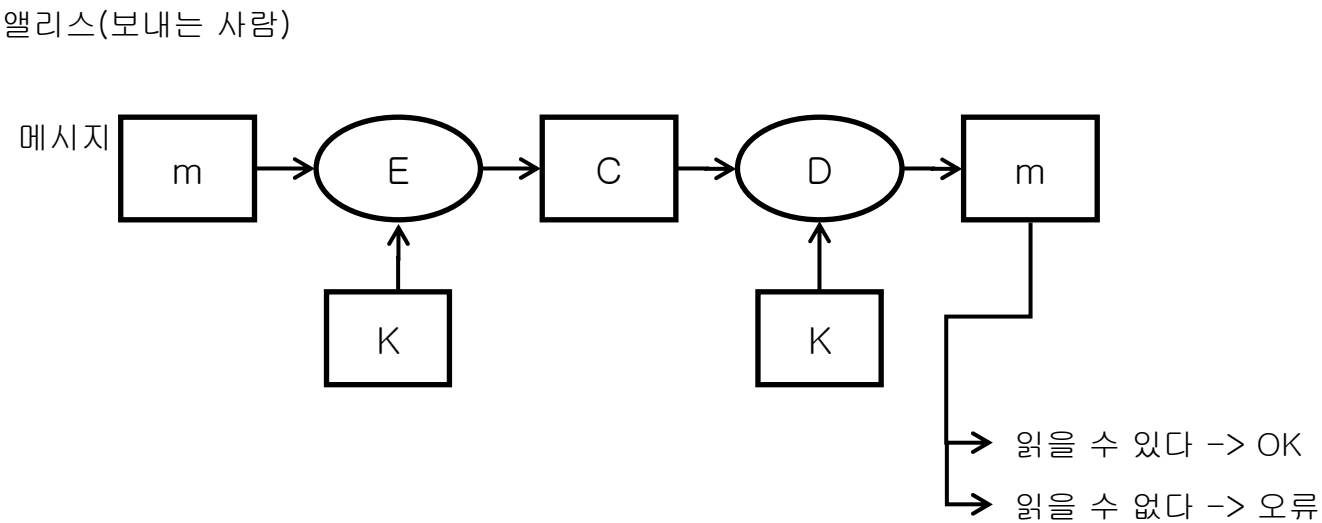
공개키와 해싱을 사용한 전자 서명



부인방지

- 메시지를 만든 사람이 이후에 자신이 만들지 않았다고 *부인*하는 것을 알아차릴 수 있는 방법
- 문서에 대한 부인 예제
  - 앨리스와 밥은 서로간의 서명된 문서를 주고받음
  - 앨리스는 밥에게 A사의 주식을 매입하도록 요청 (문서 생성)
  - 밥의 A사의 주식 매입 뒤 A사의 주식 폭락
  - 앨리스는 자신의 지시 없이 밥이 A사의 주식을 사서 손해를 보았다고 주장 (부인)
  - 밥은 매입 요청 문서를 제시
  - 앨리스는 자신이 그 문서를 제시하지 않았다고 부인
- 개인키로 서명을 작성한 경우
  - 앨리스만이 자신의 개인키를 소요할 수 있으므로 앨리스만 서명할 수 있음
  - 이의 검증에는 요청문서, 그 서명, 그리고 앨리스의 공개키가 필요

단순 비밀키 암호화를 사용



- 암호화된 문장이 변경되면 복호화한 문장은 반 이상이 변경되는 성질을 이용
- 메시지는 읽을 수 있어야 함.
- 받는 쪽이 기계이면 오류 여부를 구분을 못할 수 있음
- 부인 방지가 지원되지 않음

해싱과 비밀키를 사용한 전자 서명

1. E(M, K) | h(M)
2. E(M | h(M), K)
3. E(M, K) | h(E(M, K))
   - M : 메시지
   - K : 비밀키
   - | : concatenate 연산 (첫 번째 피연산자 뒤에 두 번째 피연산자를 붙임)
   - h : 해시 연산

- 3번째 방식은 전자 서명으로 적합하지 않음

MAC (Message Authentication Code)

- 메시지를 인증하기 위해 사용하는 짧은 정보
- 비밀키가 포함된 해시
- Hash를 이용한 MAC 생성 방법
  - h(K | M)
  - h(M | K) →다음 슬라이드
- HMAC (Hash-based Message Authentication Code) →다음 슬라이드
- 대칭키 암호 알고리즘 이용
  - CBC residue를 MAC으로 사용 →다음 슬라이드

Hash를 이용한 MAC

- h(K | M)
  - 공격자는 비밀키 K를 모르더라도 M 뒤에 첨부한 MAC을 구할 가능성이 있음
  - 위조된 문장 : M | m
  - 위조된 MAC : h(K | M | m) = hh((K | M) | m)
- h(M | K)
  - 해시가 같은 메시지 M'에 같은 MAC이 만들어짐
  - h(M | K) = h(h(M) | K) = h(h(M') | K)
  - 공격자는 비밀키 K를 몰라도 동일한 해시를 가진 메시지를 구함으로써 위조 가능

HMAC

- 앞의 두 MAC 방식의 단점을 해결한 방식
- 해시 알고리즘이 안전할 경우 HMAC도 안전함을 증명하였음
- HMAC(K, M) = h(($K \oplus$ opad) | h(($K \oplus$ ipad) | M))
  - M : 메시지
  - K : 비밀키
  - h : 해시 함수
  - opad = 0x5c5c5c...5c5c
  - ipad = 0x363636...3636

CBC residue를 이용한 MAC

- CBC residue : 블록 암호에서 CBC mode를 적용하여 얻은 마지막 블록
- 보내는 편 : 메시지와 CBC residue를 보낸다.
- 받는 편의 검증
  1. 받은 메시지를 CBC mode를 사용하여 CBC residue를 얻은 후,
  2. 계산된 residue와 받은 residue를 비교한다.
  3. 같은 경우 승인, 다른 경우 거부
- 메시지를 암호화한 경우 메시지 암호에 쓰인 비밀키와 CBC residue에 쓰인 비밀키가 달라야 함

암호화용 키와 CBC residue 키가 같을 경우

- CBC residue 계산
  - $C_1 = E(IV \oplus P_1, K)$
  - $C_2 = E(C_1 \oplus P_2, K)$
  - $C_3 = E(C_2 \oplus P_3, K)$
  - $MAC = C_3$
- 공격자가 메시지를 바꾸는 경우
  - C_3을 C'_3으로 MAC을 C'_3으로 변경
  - 받는 편은 C'_3을 복호화하여 P'_3을 얻음
  - 받는 편은 P_1, P_2, P'_3로 CBC residue를 구해 C'_3을 얻음,
  - MAC과 CBC residue가 동일하므로 승인
- 이러한 공격을 피하기 위해서는 암호화와 CBC residue 계산에 각각 다른 키를 사용해야 함.

OpenSSL을 이용한 전자 서명

- 해싱과 개인키 암호화를 통합한 함수셋을 제공
  - EVP_SignInit_ex() : 해시 함수 선택
  - EVP_SignUpdate() : 주어진 메시지를 해싱
  - EVP_SignFinal() : 개인키를 입력하여 서명 수행
- 검증 과정도 유사한 함수셋을 사용
  - EVP_VerifyInit_ex() : 해시 함수를 결정
  - EVP_VerifyUpdate() : 주어진 메시지를 해싱
  - EVP_VerifyFinal() : 공개키를 입력하여 검증 수행

전자 서명 과정

```
pkey = EVP_PKEY_new();
result = EVP_PKEY_set1_RSA(pkey, rsaPriv);
assert(result);

EVP_MD_CTX_init(&ctx);
result = EVP_SignInit_ex(&ctx, EVP_sha1(), NULL);
assert(result);
result = EVP_SignUpdate(&ctx, plaintext, plsize);
assert(result);
result = EVP_SignFinal(&ctx, sign, signSize, pkey);
assert(result);
assert(*signSize <= MAXSIGNSZ);
EVP_MD_CTX_cleanup(&ctx);
EVP_PKEY_free(pkey);
```

서명 검증 과정

```
  pukey = EVP_PKEY_new();
  result = EVP_PKEY_set1_RSA(pukey, rsaPub);
  assert(result);

  EVP_MD_CTX_init(&ctx);
  result = EVP_VerifyInit_ex(&ctx, EVP_sha1(), NULL);
  assert(result);
  result = EVP_VerifyUpdate(&ctx, plaintext, plsize);
  assert(result);
  result = EVP_VerifyFinal(&ctx, sign, signSize,
pukey);
  EVP_MD_CTX_cleanup(&ctx);
  EVP_PKEY_free(pukey);
```

프로젝트 P3 (보통)

- 전자 봉투와 전자 서명이 포함된 메시지 전송을 하는 채팅 프로그램을 구현하시오.
  - 두 사용자는 자신의 개인키 파일을 가지고 있고, 상대방의 공개키 파일을 각각 가지고 있다.
  - 메시지를 전달할 때마다 비밀키를 만들고, 만들어진 비밀키로 메시지를 암호화한다.
  - 비밀키는 상대방의 공개키로 암호화한다.
  - 암호화된 메시지와 암호화된 비밀키에 대해 자신의 개인키를 사용하여 전자서명을 만든다.
  - Sender는 암호문, 암호화된 비밀키, 그리고 전자서명을 상대방 Receiver에게 보낸다.
  - Receiver는 전자서명을 검증하고, 암호화된 비밀키를 복호화하고, 암호문을 풀어서 화면에 메시지를 출력한다.