Chap 2.
Cryptographic Tools

연세대학교

Cryptographic Tools

- 암호 알고리즘(Cryptographic algorithms)
  - 보안 서비스의 중요한 요소
- 다양한 유형의 암호화 도구 구성요소 검토
  - symmetric encryption
  - public-key (asymmetric) encryption
  - digital signatures와 key management
  - secure hash functions

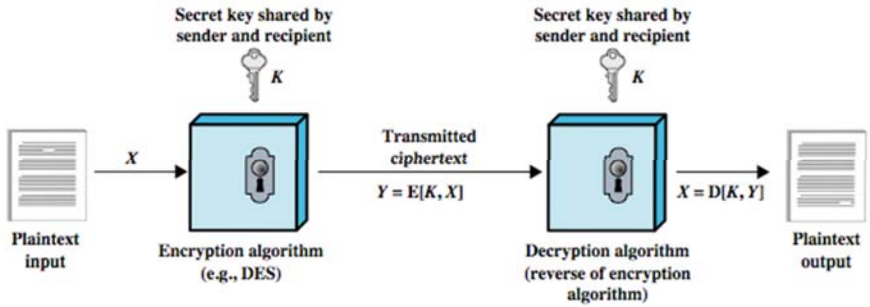
연세대학교

Symmetric Encryption

- 전송되는 또는 저장된 데이터에 대한 confidentiality를 제공하는 일반적인 기술
- 단일키 암호화(single-key encryption)
  - 암호와와 복호화에 같은 키를 사용
- 안전한 사용을 위한 두 가지 요구사항
  - 강력한 encryption algorithm
  - sender와 receiver가 비밀키 사본을 안전한 방식으로 확보하고 안전하게 유지해야 함

연세대학교

Symmetric Encryption



연세대학교

Attacking Symmetric Encryption

- 전수 공격 (Brute-Force Attack)
  - 키에 대한 모든 경우의 수를 시도함
    - 알기 쉬운 평문 번역이 얻어질 때까지
  - 성공할 때까지 평균적으로 모든 가능한 키의 반을 시도해야 함.
    - 시간이 매우 많이 소요됨
- 암호 분석 공격 (Cryptanalytic Attacks)
  - 다음을 사용하여 분석을 통한 암호해독 시도
    - 알고리즘의 특성(nature)
    - 평문의 일반적인 특성에 대한 지식
    - 샘플 평문-암호문 쌍
- 비밀키의 추론에 성공하면 해당 키로 암호화된 (과거와 미래의) 모든 메시지 해독 가능

Symmetric Encryption Algorithms

- 대칭키 암호화 알고리즘
  - DES (Data Encryption Standard)
  - triple DES
  - AES (Advanced Encryption Standard)

	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

DES = Data Encryption Standard
AES = Advanced Encryption Standard

Data Encryption Standard (DES)

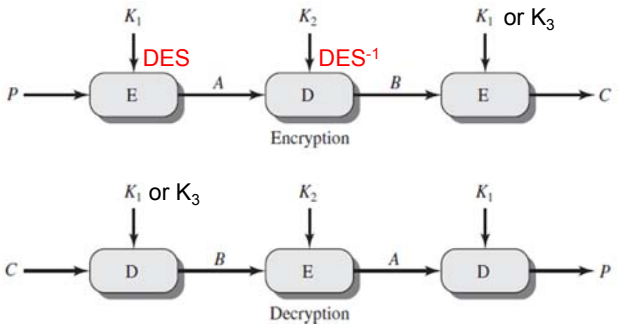
- 가장 널리 사용되는 암호화 방법
  - 사용되는 알고리즘을 Data Encryption Algorithm이라고 함
  - 64 비트 평문 블록과 56 비트 키를 사용하여 64 비트 암호문 블록 생성
- 보안성(strength)에 대한 우려
  - 알고리즘에 대한 우려
    - DES는 현재 가장 많이 연구되고 있는 암호화 알고리즘입니다.
  - 56 비트 키 사용 – 길이가 짧음
    - 1998 년 7월에 Electronic Frontier Foundation(EFF)은 DES 암호화를 3일 이내 (56 시간)에 깨뜨렸다고 발표

Average Time Required for Exhaustive Key Search

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Triple DES (3DES)

- 두 개 또는 세 개의 key를 사용하여 기본 DES 알고리즘을 세 번 반복합니다.



Triple DES

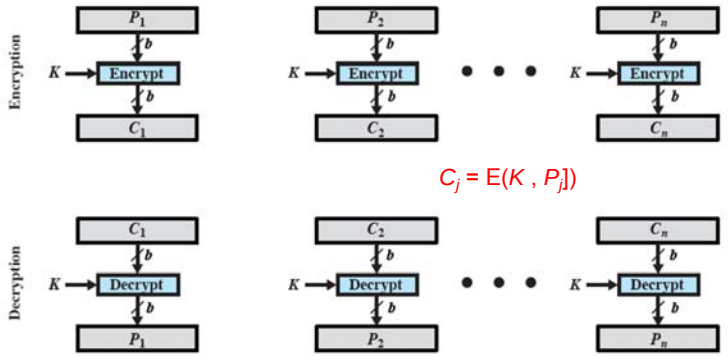
- 장점
  - 112/168 비트 키 길이가 DES의 전수공격에 대한 취약점을 극복
  - 기본 암호화 알고리즘은 DES와 같음
- 단점
  - 알고리즘이 소프트웨어에서 느리다.
  - 64 비트 블록 크기를 사용합니다.

Advanced Encryption Standard (AES)

- triple DES의 대체 알고리즘 필요성 대두
  - 3DES는 장기간 사용하기에 적절하지 않음
- NIST는 1997년에 새로운 AES에 대한 제안을 요구하여 공모
  - 3DES 이상의 우수한 보안 강도를 가져야 함
  - 크게 향상된 효율
  - 대칭키 블록 암호 – 128비트 블록 데이터
  - 128/192/256비트 키
- 2001년 11월에 Rijndael을 AES로서 선택
  - 벨기에 암호학자 Vincent Rijmen, Joan Daemen이 설계
  - FIPS(Federal Information Processing Standards) 197로 공표

Practical Security Issues

- 데이터 크기는 일반적으로 64 비트 또는 128 비트 블록보다 크다.
  - 여러 개의 블록 → 블록 단위로 암호화
- ECB(Electronic Code Book) 모드
  - 다중 블록 암호화에 대한 가장 간단한 접근 방법
  - 각 블록은 동일한 키를 사용하여 암호화됩니다.

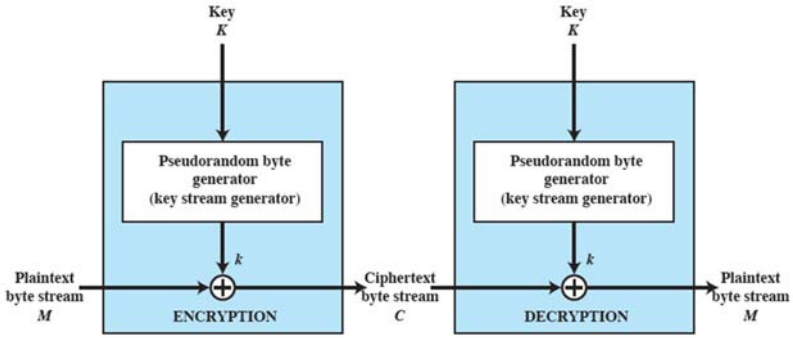


- ECB 모드의 단점
  - 같은 평문은 같은 암호문을 생성
  - 암호 해독에 평문의 규칙성을 이용가능

- ESB 단점을 해결하기 위한 동작 모드
  - 이전 블록의 암호문을 다음 블록 암호화에 사용하여 대규모 데이터 시퀀스의 보안을 향상시켜서 ECB 단점을 해결
    - CBC(Cipher Block Chaining) 모드 $C_j = E(K, [C_{j-1} \oplus P_j])$
    - CFB(Cipher Feedback) 모드 $C_i = P_i \oplus E(K, C_{i-1}]$

Stream Encryption

- pseudorandom number generator에서 key stream 생성
  - key stream의 각 key k와 평문의 각 byte를 XOR하여 암호문 생성 → 키가 계속적으로 바뀜
- key K 가 pseudorandom number generator의 입력으로 사용되어 key K 없이 key stream을 알 수 없도록 함.



Block & Stream Ciphers

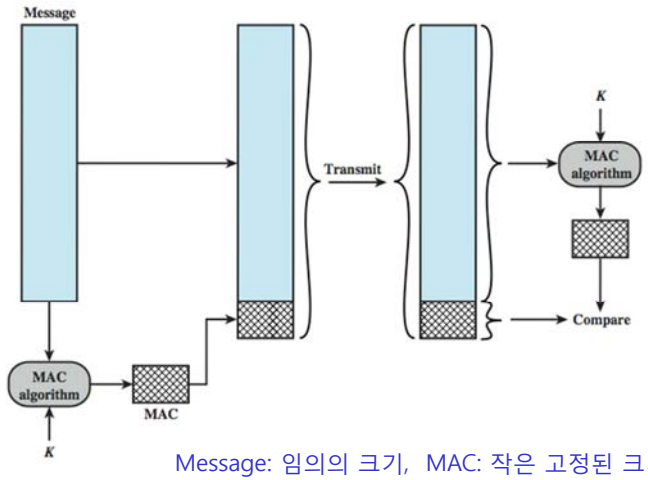
- Block Cipher
  - 입력을 블록 단위로 한 번에 처리
  - 각 입력 블록에 대해서 출력 블록을 생성
  - 키를 재사용 가능
  - 일반적으로 사용하는 방법

- Stream Cipher
  - 입력을 바이트 단위로 연속적으로 처리합니다.
  - 한 번에 한 바이트씩 암호화
  - 빠르며 훨씬 적은 코드를 사용
  - 의사 난수 스트림은 입력 키를 알지 못하면 예측할 수 없음

Message Authentication

- 수신 메시지가 진짜(authentic)임을 확인함
  - 내용이 변경되지 않음
  - 발신자(source)가 인증됨
  - 적시에, 올바른 순서로 전송됨
- active attack으로부터 보호함
- 기존의 암호화 방법을 사용하여 message authentication할 수 있음
  - 발신자와 수신자 만 키 공유

Message Authentication Codes (MAC)

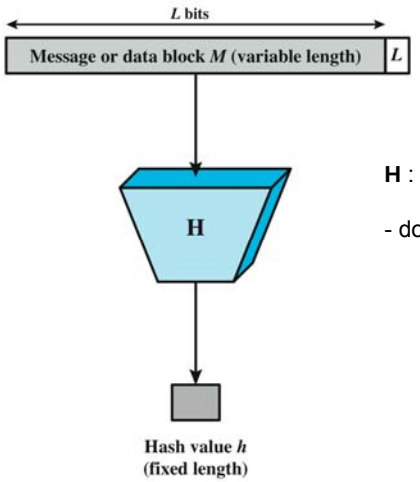


Message: 임의의 크기, MAC: 작은 고정된 크기

- 이론적으로 같은 MAC값을 갖는 메시지들이 존재함
- 실제로 이러한 메시지 쌍을 찾는 것은 불가능함
  → 충돌 회피성

- MAC 함수로 DES 사용 가능
  - 암호문의 마지막 16/32비트 숫자를 MAC 코드로 사용
  - 복호화는 불필요

Secure Hash Functions

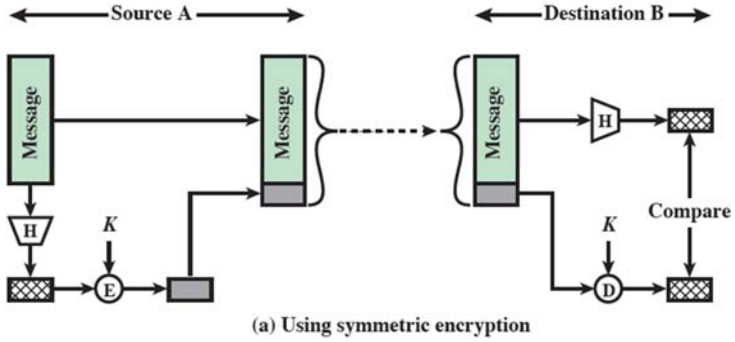


H : One-way hash function

- do not use a key

Message Authentication using a One-way Hash Function

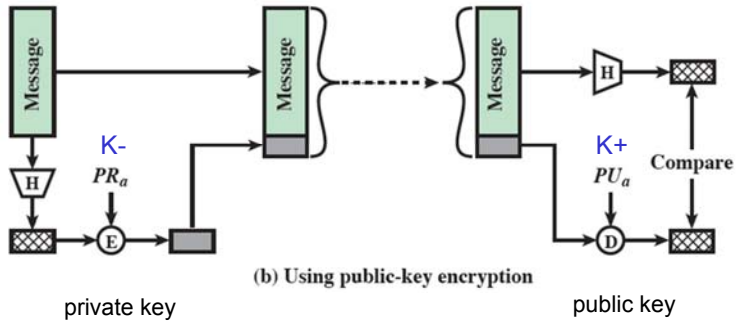
- Symmetric 암호화 사용



Message Digest : MD=E[K, H(M)]

D[K, MD]과 H(M) 비교

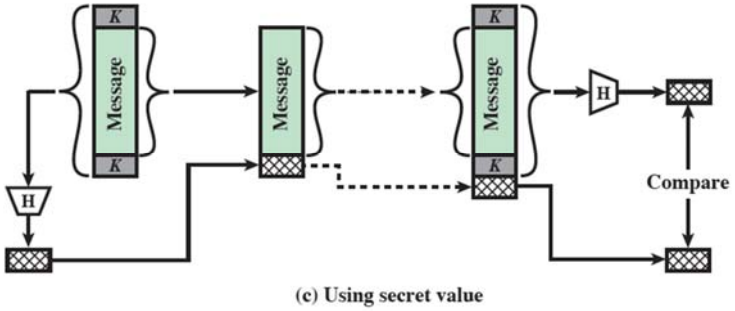
- Asymmetric 암호화 사용



Message Digest : MD=E[K-, H(M)]

D[K+, MD]과 H(M) 비교

- Keyed hash MAC



Message Digest : MD = H(K||M||K) → do not use encryption

Hash Function Requirements

- 임의의 크기의 데이터 블록에 적용 가능
  - 고정 길이 출력을 생성
- H(x)는 비교적 계산하기 쉬워야 함
  - 하드웨어/소프트웨어로 계산 가능
- 단방향 함수 : pre-image resistant
  - h에 대하여 H(x) = h가 되는 x를 찾는 것이 계산적으로 불가능함 (해시값 h를 갖는 메시지를 찾기 불가능)
- 약한 충돌 방지: second pre-image resistant => 약한 해시 함수
  - x에 대하여 H(y) = H(x)가 되는 y ≠ x인 y를 찾는 것이 불가능함 (메시지 x에 대한 대안 메시지 찾기 불가능 → 메시지 위조 방지)
- 강한 충돌 방지: collision resistant => 강한 해시 함수
  - H(x) = H(y)가 되는 한 쌍 (x, y) 찾는 것이 불가능 (B가 생성한 메시지 x에 A가 서명할 때에, B는 메시지 y로 바꾸어서 A의 서명을 첨부하는 공격이 불가능)

Security of Hash Functions

- secure 해시 함수를 공격하는 방법
  - 암호화 분석 : 알고리즘의 논리적 약점을 이용
  - 전수 공격 : 해시 함수의 안전성은 알고리즘에 의해 생성된 해시 코드의 길이(n)에만 의존합니다
    - pre-image resistant, second pre-image resistant : 2^n
    - collision resistant : $2^{n/2}$
- SHA(secure hash algorithm) : 널리 사용되는 secure hash algorithm
  - SHA-1 : 160-bit
  - SHA-2 : 224, 256, 384, 512 bit
- secure 해시 함수 추가 응용:
  - password : 운영체제는 password의 해시를 저장함
  - 침입 탐지 : 시스템의 각 파일 F에 대해 해시값 H(F)를 저장하고 이 값을 안전하게 보관함
    - H(F)을 재계산하고, 보관 값과 비교하여 파일 수정여부 탐지

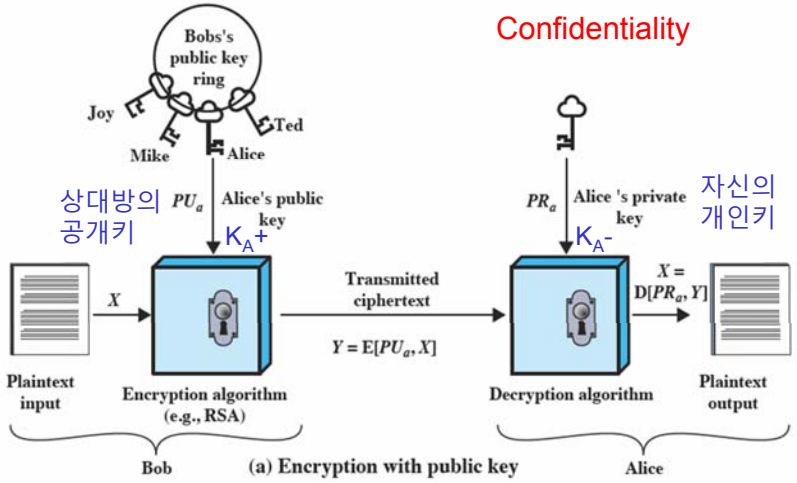
Public-Key Encryption

- 1976 년 Diffie와 Hellman이 공개적으로 제안
- 수학 함수에 기초
- 비대칭 암호화
  - 두 개의 키 사용 - **공개키**(K+) 및 **개인키**(K-)
  - 공개키는 다른 사람들이 사용할 수 있도록 공개하여 배포

  (cf) 대칭 암호화 – **비밀키**(K)
- 공개키 배포를 위해 배포를 위한 프로토콜이 필요하다

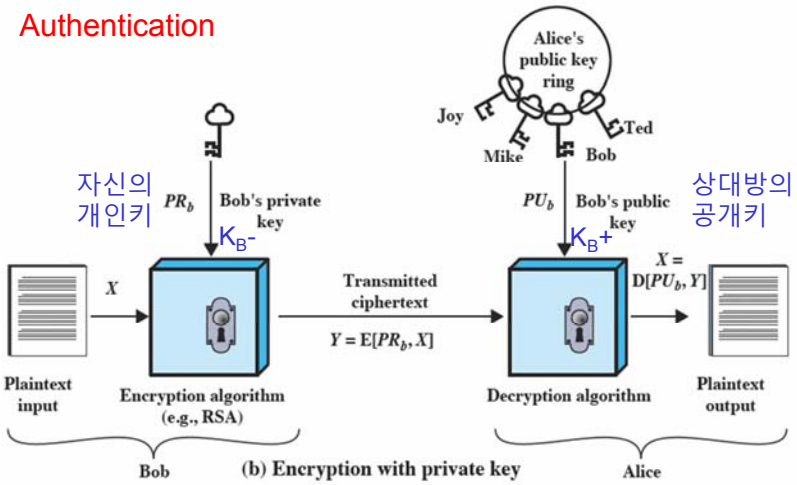
Public-Key Encryption

Confidentiality



Private-Key Encryption

Authentication



Requirements for Public-Key Cryptography

- 한 쌍의 키(공개키, 개인키) 생성이 계산상 쉽다.
- sender가 receiver의 공개키로 메시지를 암호화하는 것이 계산상 쉽다
- receiver가 자신의 개인키로 암호문을 해독하여 평문을 얻는 것이 계산상 쉽다.
- 공캐키에서 개인키를 얻는 것이 계산상 불가능하다.
- 상대방이 공개키와 암호문을 사용하여 평문을 복구하는 것이 계산상 불가능하다.
- 두 관련된 키가 각 역할로 사용될 수 있으면 유용함 (필수는 아님)
  - 한 키가 암호화에 사용되는 다른 키는 복호화에 사용됨
    (두 키중에서 배포하는 키가 공개키이고, 자신이 보유하는 키가 개인키임)

Asymmetric Encryption Algorithms

- RSA
  - 1977년에 개발, 공개키 암호화에 대부분 채택된 방식,
  - 평문과 암호문이 0과 n-1 사이의 정수로 된 블록 암호
  - Ron Rivest, Adi Shamir, Len Adleman에 의해 개발
- Diffie-Hellman 키 교환 알고리즘
  - 두 사용자가 공유 비밀키에 대해 안전하게 동의(교환)하게 함
  - 키를 교환하는 데 제약이 있음
- Digital Signature Standard (DSS)
  - 디지털 서명 기능 제공. SHA-1 이용
  - 암호화와 키 교환에 사용될 수 없다.
- 타원곡선 암호화(Elliptic curve cryptography: ECC)
  - 더 짧은 키를 사용하여 RSA와 동일한 보안성 제공
  - ECC의 신뢰수준이 아직 RSA만큼 높지 않음


연세대학교

Applications for Public-Key Cryptosystems

- 공개키 암호화 응용
  - 암호화
  - 전자서명
  - 키 관리 및 배포
    - 대칭키(비밀키) 전달
    - 임시 비밀키 생성을 위한 공개키 사용
    - 공개키의 안전한 배포

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes


연세대학교

Digital Signatures

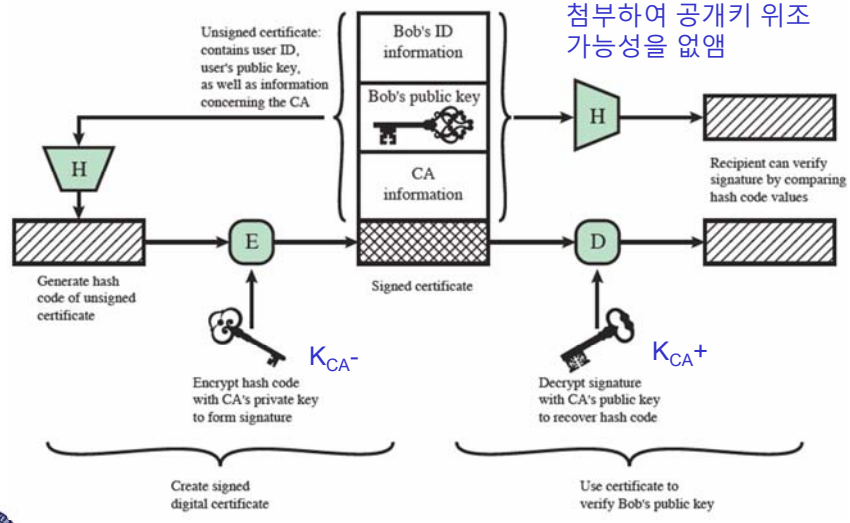
- **소스(상대방)의 인증**과 **데이터 무결성**을 확인하는 데 사용
  - 해시 값을 **개인키**로 암호화하여 전자서명 생성
  - 기밀성을 제공하지 않음 – 공개키로 해독 가능
- 메시지는 변경(alteration)에 대해서는 안전함 (데이터 무결성)
  도청(evesdropping)에 대해서는 안전하지 않음 (기밀성 없음)
- 전자 서명 절차
  - Bob은 메시지에 자신의 개인키를 사용한 전자서명을 추가하여 보냄
  - Alice는 메시지와 전자서명을 받으면
    (1) 메시지에 대한 해시값을 계산
    (2) Bob의 공개키를 사용하여 서명을 해독함
    (3) 계산된 해시값을 해독된 해시값을 비교하여 두 값이 같으면 Alice는 Bob이 메시지에 서명했음을 확인함

(Bob의 개인키는 혼자 알고 있으므로 제3자가 대신 서명할 수 없음)


연세대학교

Public Key Certificates

인증기관(CA)의 서명을 첨부하여 공개키 위조 가능성을 없앰




연세대학교

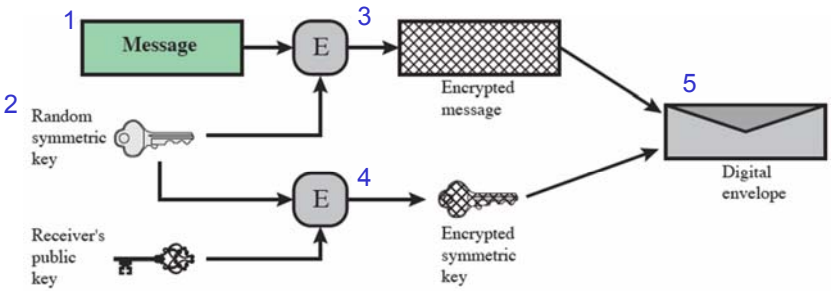
- 공개키 인증서 생성 및 사용 과정
  1) 사용자(client)는 한 쌍의 키 생성 – 공개키, 개인키
  2) **사용자 ID**와 **공개키**를 포함한 서명되지 않은 인증서 준비

  3) CA에게 서명되지 않은 인증서를 보안 규칙에 따라 제공
  4) CA는 인증서에 대한 서명 생성
     - 해시 코드 생성 후 **CA의 개인키**로 암호화
  5) 서명되지 않은 인증서에 서명 첨부 → 서명된 인증서 생성
  6) 서명된 인증서를 사용자(client)에게 제공

  7) 사용자는 자신의 서명된 인증서를 다른 사용자에게 제공
  8) 다른 사용자는 **CA의 공개키**를 이용하여 인증서의 유효성 검증

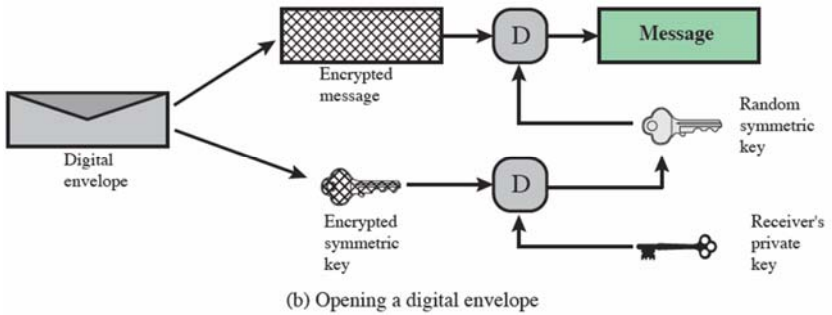
Digital Envelopes

- sender와 receiver가 수신자가 동일한 비밀키를 미리 준비할 필요 없이 메시지를 보호한다. (전자봉투에 비밀키를 함께 보냄)
- 서명되지 않은 편지를 포함한 봉인된 봉투와 같다.
- 전자 봉투 생성



(a) Creation of a digital envelope

- 전자봉투 개봉



(b) Opening a digital envelope

Random Numbers

- 난수(random number)의 용도
  - 공개키 알고리즘 용 키
  - 대칭 스트림 암호용 스트림 키
  - 대칭키 생성 – 임시 세션키, 전자봉투용 대칭키
  - 세션키 생성 - 키 배포 기관이나 신용기관에서 수행
  - 재전송(replay) 공격을 막기 위한 handshaking용 난수 (23장)

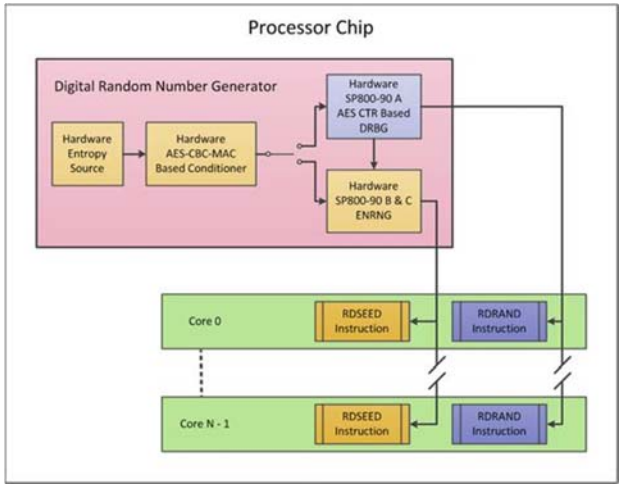
Random Number Requirements

- 무작위성(randomness)
  - 기준 :
    - 균일 분포 - 각 숫자의 출현 빈도는 거의 같아야 함
    - 독립성 – 난수열의 어떤 값도 다른 값에서 추론될 수 없음
- 예측 불가능성(unpredictability)
  - 각 숫자는 난수열의 다른 숫자와 통계적으로 독립적임.
  - 이전의 난수를 기초로 하여 난수열의 미래에 나올 난수들을 예측할 수 없어야 함.

Random vs. Pseudorandom

- 암호화 응용 프로그램은 일반적으로 **난수 생성 알고리즘**을 사용
  - 알고리즘은 결정적(deterministic)이어서 통계적으로 무작위가 아닌 일련의 수를 생성
- 의사 난수(pseudorandom number)
  - 통계적 무작위 검사를 만족시키는 난수열(시퀀스) 생성
  - 예측 가능한 특성→ determinstic 특성
- true random number generator (TRNG) :
  - 난수 생성에 비결정적 소스를 사용
  - 대부분 예측할 수 없는 자연 과정을 측정하여 작동
    - (예) 방사선, 가스 방전, 누설 커패시터, 열 잡음 등
  - 최근의 프로세서들에서 제공이 증가함
    - (예) 인텔 프로세서 digital random number generator

Intel Digital Random Number Generator



DRBG (deterministic random bit generator)
ENRNG (enhanced non deterministic random number generator)

Application: Encryption of Stored Data

- 네트워크를 통해 전송되는 데이터 - 암호화가 많이 사용됨
- 저장된 데이터 – 암호화가 많이 사용되지 않음
  - 도메인 인증과 운영체제 접근 통제 이외의 보호기능은 거의 없음
  - 데이터는 저장장치에 무기한 보관됨
  - 데이터가 삭제되더라도 디스크 섹터가 재사용될 때까지는 데이터를 복구 할 수 있음

  → 저장된 데이터의 암호화 필요성이 증대됨