교과목 : 정보보호

4. Public Key Crypto

2023학년도 2학기
Suk-Hwan Lee

References

Textbook
- Mark Stamp, Information Security: Principles and Practice, Second edition, & Lecture Note
- William Stallings, Cryptography and Network Security, Seventh Edition

참조
- Mark Stamp, Lecture Note
- 부산대, Computer Security, Lecture Note
- 순천향대학교, 통신망 정보보호, Lecture Note

- Wikipedia
- Cryptographics, https://cryptographics.info/all-cryptographics/#
- etc……

Contents

1. Concept of Public-key cryptography

2. Knapsack

3. RSA

4. Diffie-Hellman

5. ECC : Elliptic Curve Cryptography

6. ElGamal EC Encryption

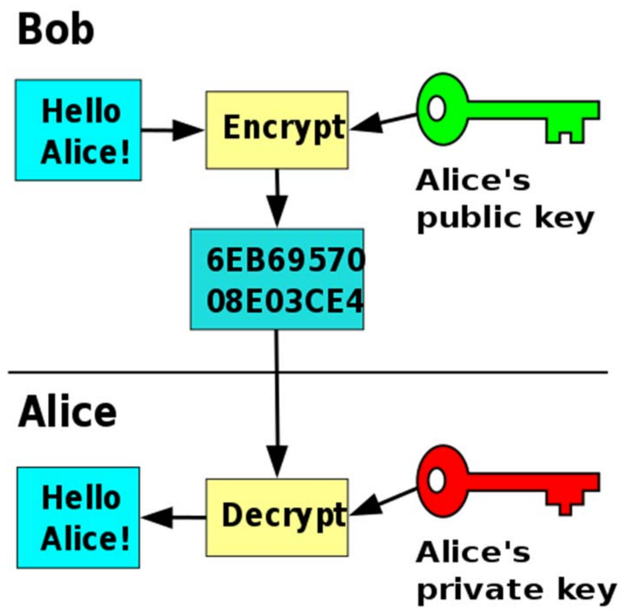
7. Uses for Public Key Crypto

8. Public Key Infrastructure

9. etc

1. Public-key cryptography

- Public-key cryptography, or asymmetric cryptography
  - ✓ Uses pairs of keys
    - ➢ public keys, which may be disseminated widely
    - ➢ private keys, which are known only to the owner
  - ✓ Any person can encrypt a message using the receiver's public key, but that encrypted message can only be decrypted with the receiver's private key.
  - ✓ The generation of such keys depends on cryptographic algorithms based on mathematical problems to produce one-way functions.
  - ✓ Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.

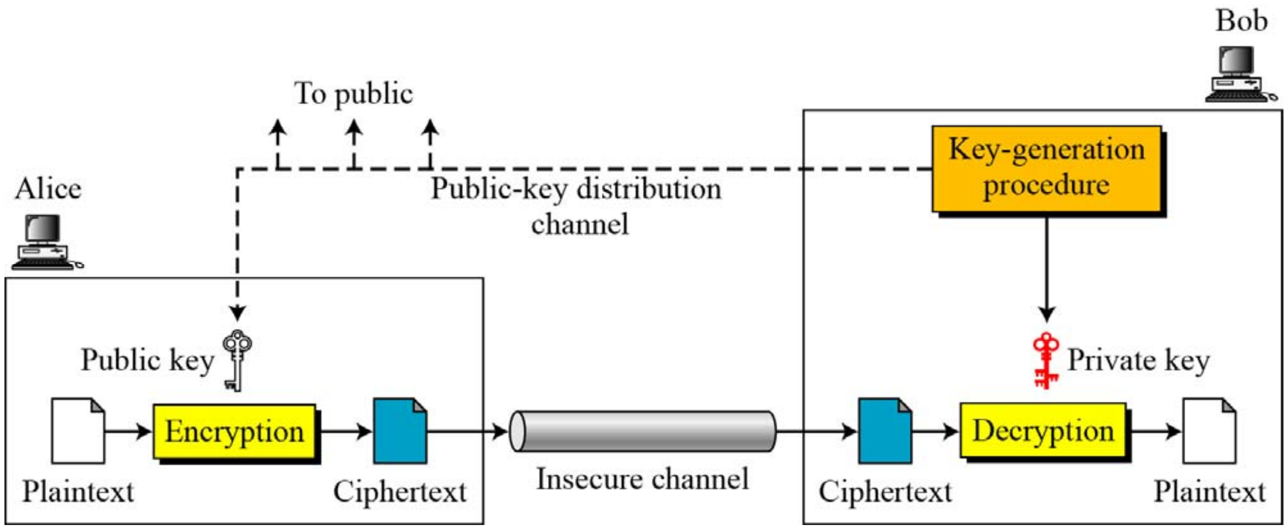


In an asymmetric key encryption scheme, anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt.
Security depends on the secrecy of the private key.

[Wikipedia]

1. Public-key cryptography

- Public key algorithms are fundamental security ingredients in modern cryptosystems, applications and protocols assuring the *confidentiality*, *authenticity* and *non-repudiability* of electronic communications and data storage.
- They underpin various Internet standards, such as Transport Layer Security (TLS), S/MIME, PGP, and GPG.
- Some public key algorithms provide key distribution and secrecy (e.g., Diffie–Hellman key exchange), some provide digital signatures (e.g., Digital Signature Algorithm), and some provide both (e.g., RSA).



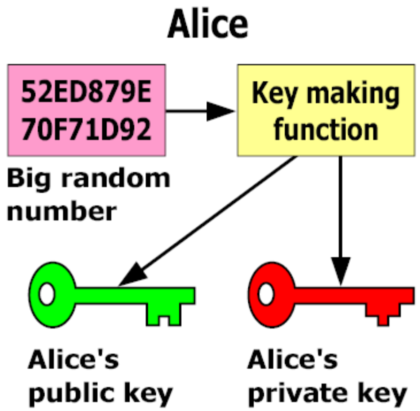
[Wikipedia]

Figure 10.2 비대칭키 암호시스템의 일반적 아이디어

Key Generation of PKC

- Making two keys: Based on trap door one way function
  - ✓ Easy to compute in one direction
  - ✓ Hard to compute in other direction
  - ✓ "Trap door" used to create keys
  - ✓ Example: Given p and q, product N=pq is easy to compute, but given N, it is hard to find p and q

- A message encrypted by the public key can decrypted only with the corresponding private key



An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

❖ [참고] Trapdoor 설명

Key Generation of PKC

Note

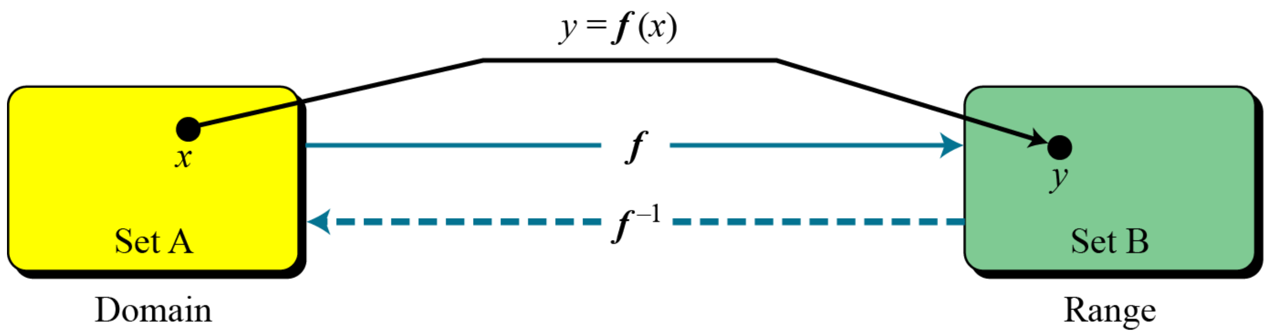


Figure 10.3 정의역과 공역 사이의 대응규칙인 함수

일방향 함수 (*One-Way Function (OWF)*)

1. f 는 계산이 쉽다.
2. f^{-1} 1. 는 계산이 어렵다.

트랩도어 일방향 함수 (*Trapdoor One-Way Function (TOWF)*)

*3. y*와 트랩도어(trapdoor)(비밀)가 주어지면 *x* 를 쉽게 계산할 수 있다.

Key Generation of PKC

Note

Example 10. 1

n 이 매우 큰 수라면, $n = p \times q$ 는 일방향 함수이다.

주어진 p 와 q 로부터 n 을 계산하는 것은 매우 쉽다. 하지만 주어진 n 으로부터 p 와 q 를 계산하는 것은 매우 어렵다. 이것이 바로 ***소인수분해*** 문제이다.

Example 10. 2

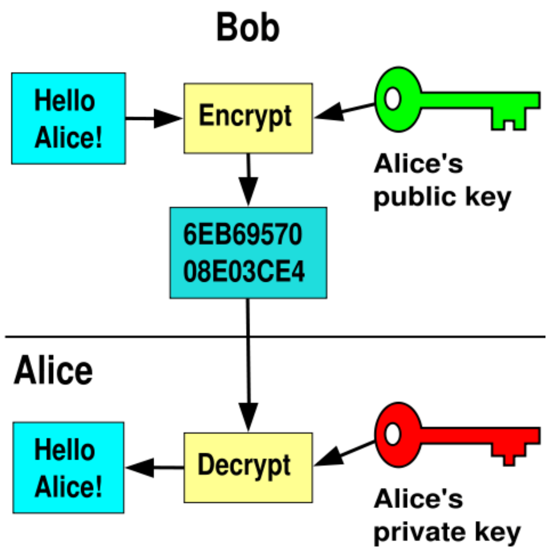
n 이 매우 큰 수라면, 함수 $y = x^k \bmod n$ 은 하나의 ***트랩도어 일방향 함수***이다.

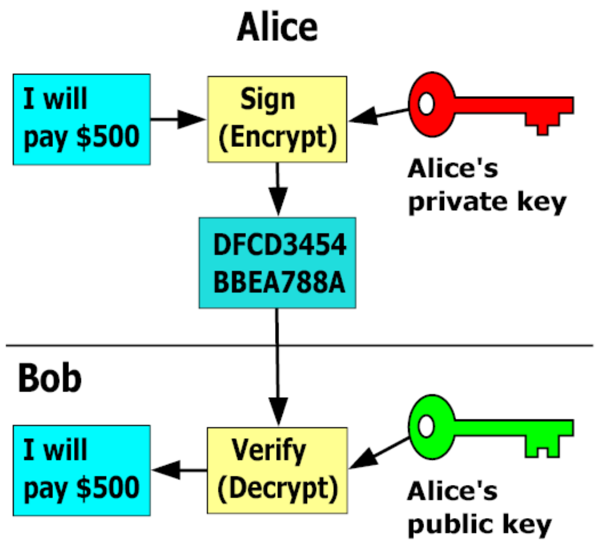
주어진 x, k, 와 n에 대해 우리가 y 를 계산하는 것은 쉽다. 하지만 y, k, 와 n이 주어졌을 때, x 를 계산해내는 것은 매우 어렵다. 이것이 이산대수문제이다.

그러나 만약에 우리가 $k \times k' = 1 \bmod f(n)$이 되는 트랩도어 k' 를 알고 있다면, $x = y^{k'} \bmod n$ 을 이용하여 x를 구할 수 있다. 이것이 바로 유명한 RSA이다.

Two main branches of PKC

- Public key Encryption
  - ✓ Suppose we encrypt M with Alice's public key
  - ✓ Only Alice's private key can decrypt to find M
- Digital Signature
  - ✓ Sign by "encrypting" with private key
  - ✓ Anyone can verify signature by "decrypting" with public key
  - ✓ But only private key holder could have signed
  - ✓ Like a handwritten signature (and then some)





PKCs to Discuss

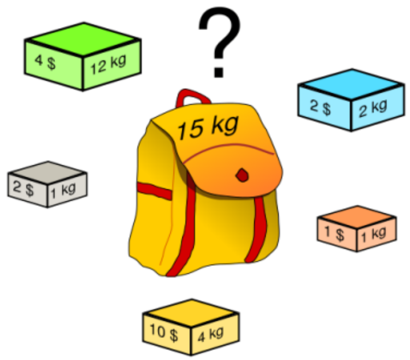
- Knapsack
  - ✓ The first proposed PKC
  - ✓ It is insecure
- RSA
  - ✓ Problem of factoring large numbers
- Diffie-Hellman Key Exchange
  - ✓ Discrete log problem
- ECC(Elliptic Curve Cryptography)
  - ✓ Based on the algebraic structure of elliptic curves over finite fields

Knapsack

Knapsack Problem

- Given a set of n weights $W_0, W_1, \ldots, W_{n-1}$ and a sum S, is it possible to find $a_i \in \{0,1\}$ so that
  $S = a_0W_0 + a_1W_1 + \ldots + a_{n-1}W_{n-1}$
  (technically, this is "subset sum" problem)
- Example
  - ✓ Weights (62,93,26,52,166,48,91,141)
  - ✓ Problem: Find subset that sums to S=302
  - ✓ Answer: 62+26+166+48=302
- The (general) knapsack is NP-complete

NP complete (Nondeterministic Polynomial)



Knapsack Problem

- General knapsack (GK) is hard to solve
- But superincreasing knapsack (SIK) is easy
  - SIK : each weight greater than the sum of all previous weights
- Example
  - ✓ Weights (2,3,7,14,30,57,120,251)
  - ✓ Problem: Find subset that sums to S=186
  - ✓ Work from largest to smallest weight
  - ✓ Answer: 120+57+7+2=186

Superincreasing : such that every element of the sequence is greater than the sum of all previous elements.

Knapsack Cryptosystem

1. Generate superincreasing knapsack (SIK)
2. Convert SIK into "general" knapsack (GK)
   Public Key: GK
   Private Key: SIK plus conversion factors

- Easy to encrypt with GK
- With private key, easy to decrypt (convert ciphertext to SIK)
- Without private key, must solve GK (???)

Knapsack Cryptosystem

1. Let (2,3,7,14,30,57,120,251) be the SIK Private key
2. Choose m = 41 and n = 491 with m, n rel. prime (m,n 서로 소인 정수, gcd(m,n)=1) and n greater than sum of elements of SIK

   Then General knapsack can be computed;

   $2 \cdot 41 \mod 491 = 82$
   $3 \cdot 41 \mod 491 = 123$
   $7 \cdot 41 \mod 491 = 287$
   $14 \cdot 41 \mod 491 = 83$
   $30 \cdot 41 \mod 491 = 248$
   $57 \cdot 41 \mod 491 = 373$
   $120 \cdot 41 \mod 491 = 10$
   $251 \cdot 41 \mod 491 = 471$

   GK[i] = (SIK[i] • m) mod n
3. General knapsack: (82,123,287,83,248,373,10,471) Public key

Knapsack Example

- Private key (SIK): (2,3,7,14,30,57,120,251) (복호)
- Public key (GK): (82,123,287,83,248,373,10,471) (암호)

  GK[i] = (SIK[i] • m) mod n
- Plaintext (binary) : 10010110

$$S' = \sum_i \mathrm{p}[i]\mathrm{GK}[i]$$

- Encrypt : S'=sum_i (p[i]GK[i])

  = 1x82+0x123+0x287+1x83+0x248

  +1x373+0x10+0x471

  = 82 + 83 + 373 + 10

  = 548 = S' (Ciphertext)

- To decrypt, m^{-1} 계산 (m=41, n=491)

  m^{-1} : m • x mod n = 1에서 x를 구함

  (어떤 연산을 했을 때 항등원이 나오는 역원을 구함)

  41 • x mod 491 = 1, m^{-1} = x = 12

- Decrypt
  - ✓ (Ciphertext · m^{-1}) mod n" 결과값을 개인키(SIK)에 찾음
  - ✓ (S' · m^{-1}) mod n = (548 · 12) mod 491 = 193 = S

  - ✓ Solve (easy) SIK with S = 193
  - ✓ S = 193 = 2+0+0+14+0+57+120+0
  - ✓ Obtain plaintext 10010110

Knapsack Weakness

- Trapdoor: Convert SIK into "general" knapsack using modular arithmetic
- One-way: General knapsack easy to encrypt, hard to solve; SIK easy to solve
- This knapsack cryptosystem is insecure
  - ✓ Broken in 1983 with Apple II computer
  - ✓ The attack uses lattice reduction
  - ✓ "General knapsack" derived from SIK is not general enough!
  - ✓ This special knapsack is easy to solve!

Note

Knapsack Cryptosystem

정의(Definition)

$a = [a_1, a_2, \ldots, a_k]$ *and* $x = [x_1, x_2, \ldots, x_k]$.

$$\boxed{s = knapsackSum\ (a, x) = x_1a_1 + x_2a_2 + \cdots + x_ka_k}$$

주어진 a와 x로부터 s를 계산하는 것은 매우 쉽다. 하지만 s와 a가 주어졌다고 했을 때, x를 구하는 것은 어렵다.

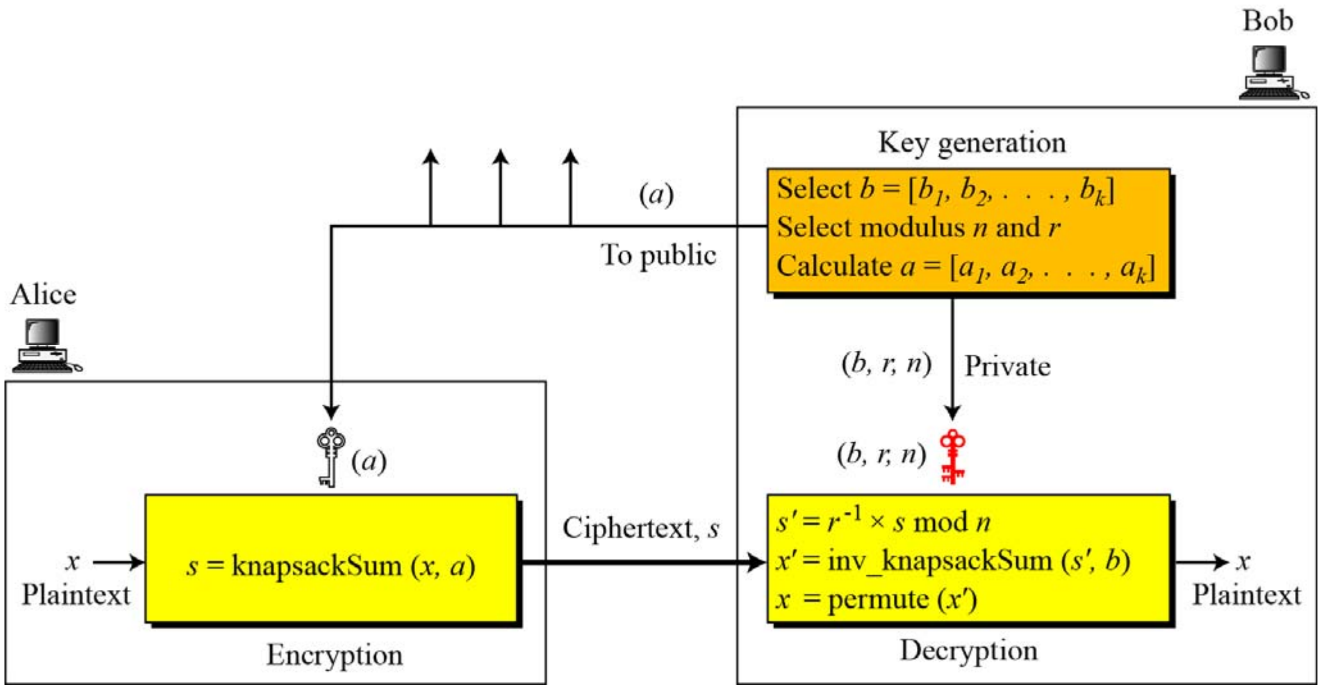
초증가 순서짝 (*Superincreasing Tuple)*

$$\boxed{a_i \geq a_1 + a_2 + \ldots + a_{i-1}}$$

Algorithm 10.1 *knapsacksum and inv_knapsackSum for a superincreasing k-tuple*

knapsackSum (x [1 … k], a [1 … k]) { $s \leftarrow 0$ for (i = 1 to k) { $s \leftarrow s + a_i \times x_i$ } return s }	**inv_knapsackSum** (s, a [1 … k]) { for (i = k down to 1) { if $s \geq a_i$ { $x_i \leftarrow 1$ $s \leftarrow s - a_i$ } else $x_i \leftarrow 0$ } return x [1 … k] }

Secret Communication with Knapsacks



1. 키 생성:
   a. 밥은 초증가 순서짝 $b = [7, 11, 19, 39, 79, 157, 313]$ 을 만든다.
   b. 밥은 모듈로로 $n = 900$,$r = 37$ 을 선택한다. 치환표로는 $[4\ 2\ 5\ 3\ 1\ 7\ 6]$ 을 택한다.
   c. 밥은 순서짝 $a = permute(t) = [543,\ 407,\ 223,\ 703,\ 259,\ 409]$ 를 계산한다.
   d. 밥은 순서짝 $t = [259,\ 407,\ 703,\ 259,\ 781,\ 409]$ 를 계산한다.
   e. 밥은 a 를 공개한다. 그리고 n, r, b 는 비밀로 한다.

2. 앨리스가 단 하나의 문자 *"g"* 만을 밥에게 보낸다고 가정을 해보자.
   a. 앨리스는 7-비트 **ASCII** 코드를 이용하여 *"g"*를 $(1100111)_2$ 로 부호화한다. 그 다음 순서짝 $x = [1, 1, 0, 0, 1, 1, 1]$ 을 생성한다. 이것이 평문이다.
   b. 앨리스는 $s = knapsackSum(a, x) = 2165$를 생성한다. 이것이 밥에게 보낼 암호문이다.

3. 밥은 암호문 $s=2165$ 를 복호화 할 수 있다.
   a. 밥은 $s' = x \times r^{-1} \bmod n = 2165 \times 37^{-1} \bmod 900 = 527$ 을계산한다.
   b. 밥은 $x' = inv_knapsackSum(s', b) = [1, 1, 0, 1, 0, 1, 1]$를 계산한다.
   c. 밥은 $x = permute(x') = [1, 1, 0, 0, 1, 1, 1]$을 계산하여 $(1100111)_2$를 얻는다. 이것을 **ASCII** 코드를 이용하여 문자로 환원하면 문자 *"g"*를 얻는다.

RSA

RSA

- The most difficult computation?

Addition	Multiplication	Factorization
Easy		Difficult
123 + 654 --------- 777	123 x 654 ---------- 492 615 738 ------------ 80442	221 = ?x? 221/2 = 221/3 = 221/5 = 221/7 = 221/11 = 221/13 = 221 = 13 x 17

RSA

❖ Invented by Cocks (GCHQ), independently, by **R**ivest, **S**hamir and **A**dleman (MIT, 1978)

- Let p and q be two large prime numbers
- Let $N = pq$ be the modulus
- Choose e relatively prime (서로 소) to $(p-1)(q-1)$
- Find d s.t. $ed = 1 \bmod (p-1)(q-1)$
- Public key is (N, e)
- Private key is d

RSA

- Public key = {N, e}, Private key = {d}

$N=pq$, $\gcd(e,(p-1)(q-1))=1$,
$ed=1 \bmod (p-1)(q-1)$ (곱셈에 대한 역원)

- To encrypt message M compute

$C = M^e \bmod N$

- To decrypt C compute

$M = C^d \bmod N$ $(M^e)^d \bmod N = M^{ed} \bmod N$

- If attacker can factor N(=pq), he can use e to easily find d since $ed = 1 \bmod (p-1)(q-1)$
- Factoring the modulus breaks RSA
- It is not known whether factoring is the only way to break RSA

Does RSA Really Work?

If $n = p \times q$, $a < n$, and k is an integer, then $a^{k\times\phi(n)+1} \equiv a \pmod{n}$.

$P_1 = C^d \bmod n = (P^e \bmod n)^d \bmod n = P^{ed} \bmod n$
$ed = k\phi(n) + 1$ // d and e are inverses modulo $\phi(n)$
$P_1 = P^{ed} \bmod n \rightarrow P_1 = P^{k\phi(n)+1} \bmod n$
$P_1 = P^{k\phi(n)+1} \bmod n =$ P mod n // Euler's theorem (second version)

- Given $C = M^e \bmod N$, we must show
  - ✓ $M = C^d \bmod N = M^{ed} \bmod N$ where $M < N$
- We'll use Euler's Theorem (오일러 정리)
  - ✓ If M is relatively prime to N, then $M^{\varphi(N)} = 1 \bmod N$
- Facts:
  - ✓ $ed = 1 \bmod (p-1)(q-1)$
  - ✓ By definition of "mod", $ed = k(p-1)(q-1) + 1$
  - ✓ $\varphi(N) = \varphi(pq) = (p-1)(q-1)$
  - ✓ Then $ed - 1 = k(p-1)(q-1) = k\varphi(N)$
- $M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)} = M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N = M \bmod N$

φ(N) : N보다 적고 N과 서로소인 양의 정수가 되는 함수
(p,q가 소수일 경우, $\varphi(pq) = (p-1)(q-1)$)

❖ [참고] Euler's Theorem 설명

Simple RSA Example

- Example of RSA
  - ✓ Select "large" primes $p = 11, q = 3$
  - ✓ Then $N = pq = 33$ and $(p-1)(q-1) = 20$
  - ✓ Choose $e = 3$ (relatively prime to 20)
  - ✓ Find d such that $ed = 1 \bmod 20$, we find that $d = 7$ works
- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$

- Suppose message $M = 8$
- Ciphertext C is computed as
  - ✓ $C = M^e \bmod N = 8^3 = 512 = 17 \bmod 33$
- Decrypt C to recover the message M by
  - ✓ $M = C^d \bmod N$

    $= 17^7 = 410{,}338{,}673$

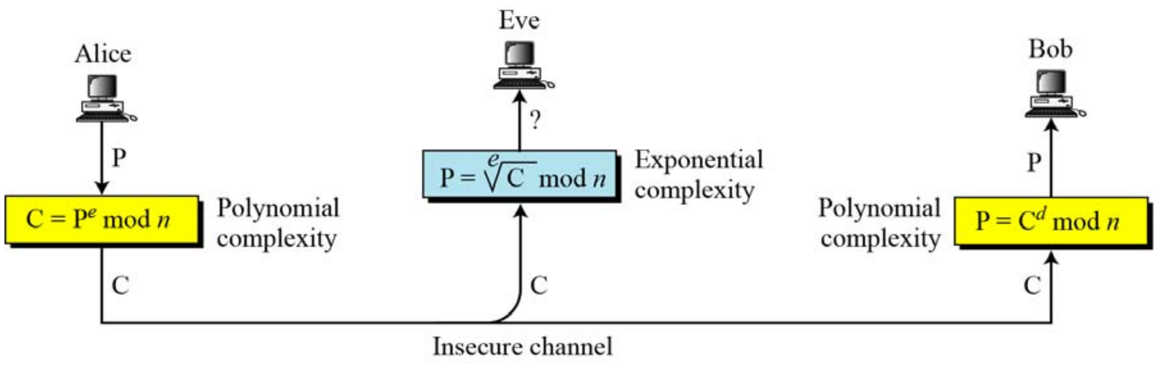
    $= 12{,}434{,}505 \times 33 + 8$

    $= 8 \bmod 33$

Note

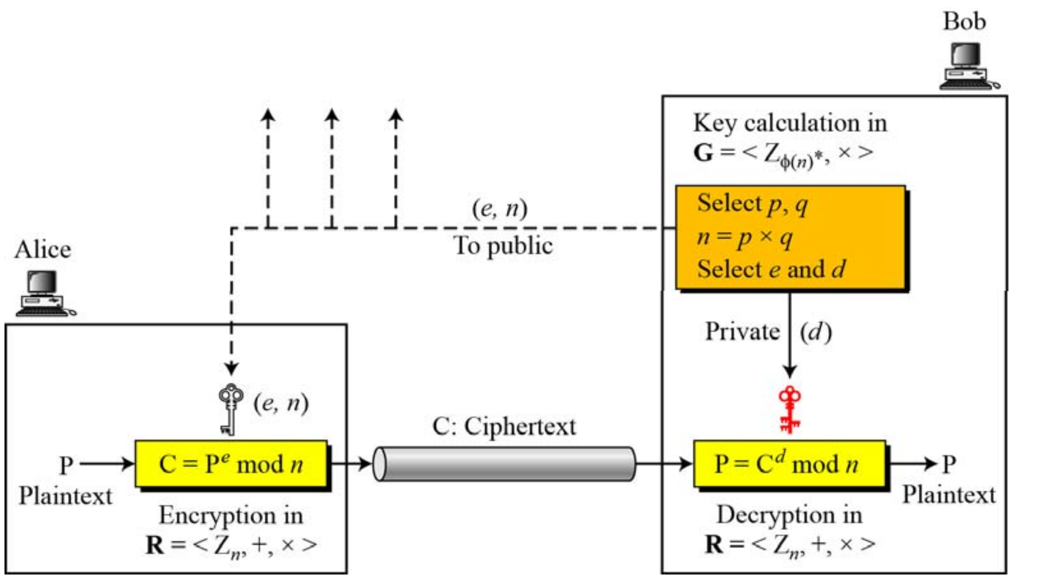
RSA Key Generation

RSA 연산의 복잡도



- RSA는 모듈로 지수계산을 이용해서 암호화/복호화 하므로 이것을 공격하려면 이브는 $\sqrt[e]{C} \bmod n$ 을 계산해야 한다.

RSA 암호화, 복호화 키 생성



Note

Two Algebraic Structures

암호화/복호화 환
Encryption/Decryption Ring:

$$\boldsymbol{R} = \langle \boldsymbol{Z_n}, +, \times \rangle$$

키 생성 군
Key-Generation Group:

$$\boldsymbol{G} = \langle \boldsymbol{Z_{\phi(n)}}^{*}, \times \rangle$$

RSA uses two algebraic structures:
a public ring R = $\langle Z_n, +, \times \rangle$ and a private group G = $\langle Z_{\phi(n)}^{*}, \times \rangle$.

In RSA, the tuple (e, n) is the public key; the integer d is the private key.

Note

RSA Key Generation

Algorithm 10.2 *RSA Key Generation*

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p - 1) \times (q - 1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$

Public_key $\leftarrow$ (e, n) // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

Note

RSA Encryption & Decryption

Algorithm 10.3 *RSA encryption*

```
RSA_Encryption (P, e, n)                  // P is the plaintext in Z_n and P < n
{
    C ← Fast_Exponentiation (P, e, n)     // Calculation of (P^e mod n)
    return C
}
```

In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

Algorithm 10.4 *RSA decryption*

```
RSA_Decryption (C, d, n)                  //C is the ciphertext in Z_n
{
    P ← Fast_Exponentiation (C, d, n)     // Calculation of (C^d mod n)
    return P
}
```

More Efficient RSA

- Modular exponentiation example
  - ✓ $5^{20} = 95367431640625 = 25 \bmod 35$
- A better way: repeated squaring
  - ✓ 20 = 10100 base 2
  - ✓ (1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)
  - ✓ Note that
    $2 = 1 \cdot 2,\ 5 = 2 \cdot 2 + 1,\ 10 = 2 \cdot 5,\ 20 = 2 \cdot 10$
  - ✓ $5^1 = 5 \bmod 35$
  - ✓ $5^2 = (5^1)^2 = 5^2 = 25 \bmod 35$
  - ✓ $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 = 10 \bmod 35$
  - ✓ $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \bmod 35$
  - ✓ $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \bmod 35$
- Never have to deal with huge numbers!

암호화와 복호화 계산량을 줄이는 방법

- 모듈러 연산의 특징 이용
  $[(a \bmod n) \times (b \bmod n)] \bmod n$
  $= (a \times b) \bmod n$

⇒ 중간 결과를 축소

- 효율적인 지수 승
  직접적인 방법 (x^{16} 일 경우)
  : $x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$

⇒ 15번의 곱셈

  효율적인 방법
  : x^2, x^4, x^8, x^{16} ⇒ 4번의 곱셈

Note

RSA에 대한 가능한 공격들