Chapter 9
Public-Key Cryptography & RSA

정보보안

Public-Key Cryptography

- Public-key cryptography 공개키
  (a.k.a. asymmetric cryptography) 비대칭형 암호
  - Public-key cryptography refers to a cryptographic system requiring two separate keys, one of which is secret and one of which is public. Although different, the two parts of the key pair are mathematically linked.
  - A party (Alice) generates a public key along with a matching private key (a.k.a. secret key).
  - The public key is widely distributed and is assumed to be known to anyone (Bob) who wants to communicate with Alice.
  - Alice's public key is also known to the attacker! 공격자도 public key 알고있음
  - Alice's private key remains secret.
  - Bob may or may not have a public key of his own.

Private- vs. Public-Key Cryptography I

- Disadvantages of private-key/symmetric cryptography
  - Keys need to be securely shared. 미리공유해야함
    - What if this is not possible?
    - You need to know in advance the parties with whom you will communicate.
    - It can be difficult to distribute/manage keys in a large organization.
  - $O(t^2)$ keys are needed for person-to-person communication in a t-party network.
    - All these keys need to be stored securely.

사람수가 많아지면 키 개수가 많아짐

  - Private-key cryptography is inapplicable in open systems.
    - e.g., e-commerce 모든 사람 힘듦

잠그는 열쇠 모두

Private- vs. Public-Key Cryptography II

우린 왜 대칭형 AES, DES 공부했나

- Why do we study private-key cryptography at all?
  - Private-key cryptography is orders of magnitude more efficient.
  - Private-key cryptography still has domains of applicability.
    - Military settings, disk encryption, …
  - Private-key cryptographic primitives can be combined with public-key techniques to get the best of both (for encryption).
  - Keys may be distributed using trusted entities.
    - e.g., KDC (Key Distribution Center)

대칭키가 속도 1000배 빠름

잠그는 사람과 여는 사람이 같음

Public + Private 같이 씀

Cryptographic Primitives

대칭 비대칭

	Symmetric cryptography (Private-key cryptography)	Asymmetric cryptography (Public-key cryptography)
기밀성 Confidentiality 의도된 사람만 열어봐라	Private-key encryption (e.g., AES) DES	Public-key encryption (e.g., RSA-OAEP)
Integrity 무결성	Message authentication code (MAC)	Digital signature (e.g., RSA-PSS)

중간에 위변조X

그대로 내용이 잘 왔는지

전자서명

메시지를 쓴이가 옳음

Public-Key Encryption

public secret

- Key generation algorithm:
  - Randomized algorithm that outputs (pk, sk).

input에 Random number
같이 넣어줌

숨기고자 하는 메시지 암호문

- Encryption algorithm:
  - Takes a public key and a message (plaintext), and outputs a ciphertext.
  - $c = \mathrm{E}(pk, m)$
- Decryption algorithm:
  - Takes a private key and a ciphertext, and outputs a message.
  - $m = \mathrm{D}(sk, c)$

RSA Background

예전엔 1024 bit
요즘은 3072 bit

- Euler's Theorem 지수에 φ(n) 이 있는것과 같음
  - For m and n that are relatively prime (i.e., $m \in Z_n^*$), $m^{\phi(n)} \equiv 1 \pmod{n}$
  - If $ed \equiv 1 \pmod{\phi(n)}$, then for all m it holds that $(m^e)^d \equiv m^{ed} \equiv m \pmod{n}$.

큰 Prime nuber의 곱 encryption 암호문

- $n = pq$, where p and q distinct, odd primes. e를 알더라도 d를 모름
  - $\phi(n) = (p-1)(q-1) = |Z_n^*|$ n값을 알아도 n의 소인수 p,q를 모르면 φ(n) 계산 불가
  - Easy to compute $\phi(n)$ given the factorization of n.
  - Hard to compute $\phi(n)$ without the factorization of n.

m이랑 서로소인 것의 개수

- We have an asymmetry!
  - Given d (which can be computed from e and the factorization of n), it is possible to compute m from $c = m^e \bmod n$.
  - Without the factorization of n, there is no apparent way to compute m from $c = m^e \bmod n$.

얼마라나는 나머지 집합에서 곱셈에 대한 역원을 알아야한다.

RSA Key Generation

- Generate random odd primes p, q of sufficient length.
- Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
- Compute e and d such that $ed \equiv 1 \pmod{\phi(n)}$.
  - e must be relatively prime to $\phi(n)$, i.e., $\gcd(e, \phi(n)) = 1$.
  - d can be computed by the extended Euclidean algorithm.
- Public key = (e, n); private key = (d, n).

e로 잠그고, d로 풀

곱셈에대한 역원이 존재할때 까지. e와 n이 서로소일때까지 다시 뽑음

e를 뽑고, d는 e의 역원.

"Textbook RSA" Encryption

- Public key (e, n); private key (d, n).
- To encrypt a message $m \in Z_n^*$, compute $ed = 1.$

$$c = m^e \bmod n. \qquad C = M^e \bmod n$$

- To decrypt a ciphertext c, compute $m = c^d \bmod n$. $C^d = M^{ed}$
  - The CRT can speed up the decryption process by approximately four times.

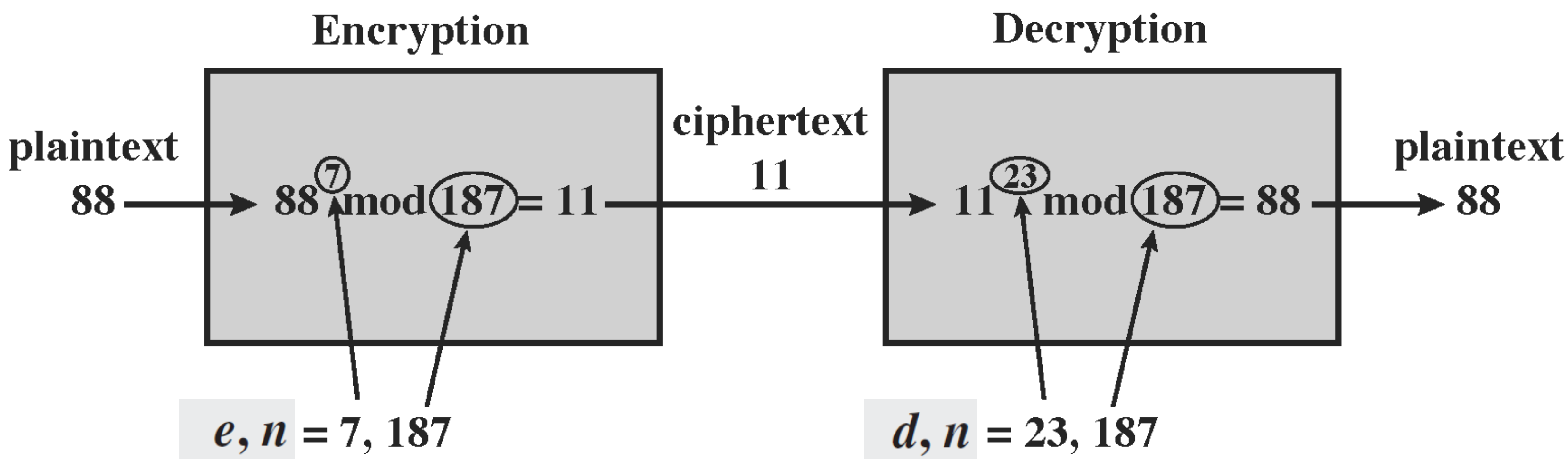
pq는 지워 버린다. 시간이 오래걸림 AES보다

- What about security?
  - It is deterministic! 펭귄그림 | 랜덤한것처럼 바꿔줌
  - Furthermore, it can be shown that the ciphertext leaks specific information about the plaintext.
    - e.g., the Jacobi symbol $\left(\frac{c}{n}\right) = \left(\frac{m^e}{n}\right) = \left(\frac{m}{n}\right)^e = \left(\frac{m}{n}\right)$ because e is odd.

- [Rivest, Shamir, Adleman - The RSA Algorithm Explained]

"Textbook RSA" Example

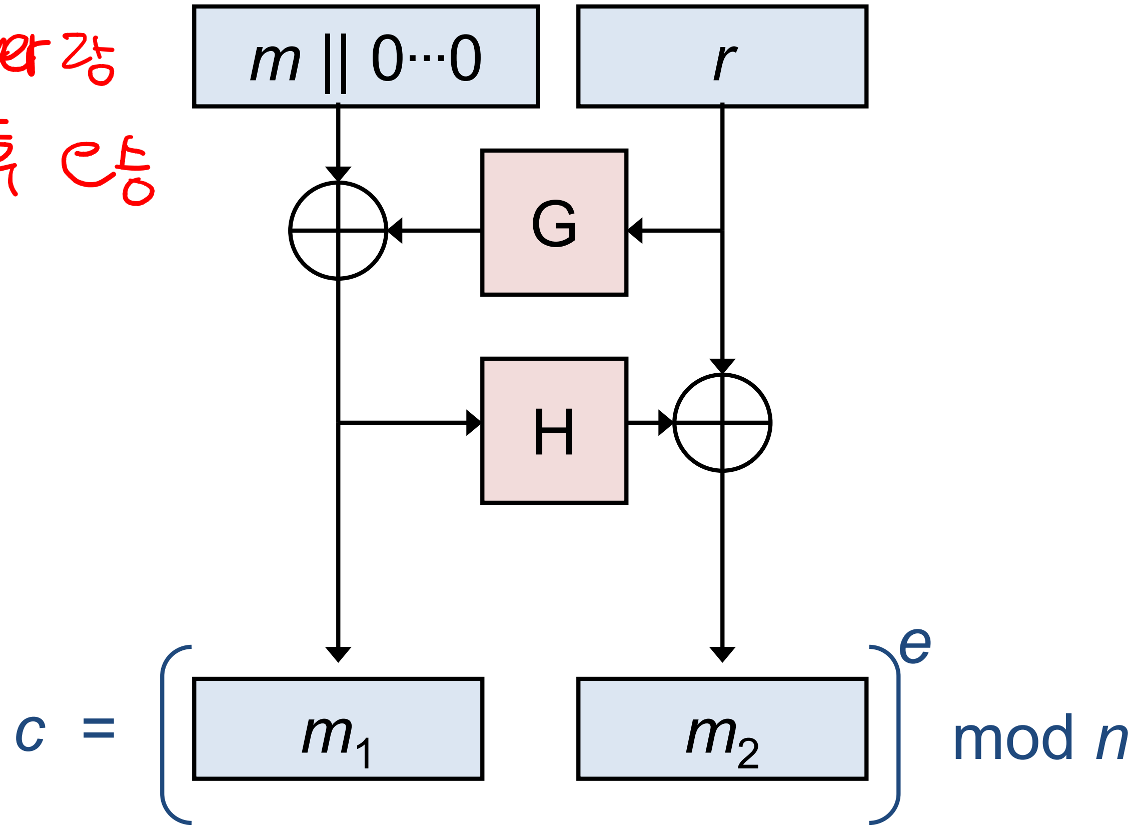
1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 4).



RSA-OAEP (PKCS #1 v2.1)

- OAEP (Optimal Asymmetric Encryption Padding)

M+ rand number 값
어떠한 계산 후 e승



  - In theory, G and H are random oracles.
  - In practice, G and H are cryptographic hash functions.

Hybrid Encryption

장점만 모아놓음. 대칭키, 공개암호키. 소도수

- Public-key encryption is slow.

ALice

Me,d

(n,e)

Bob 할라면

연산가능

연산어려움

$k = c^d \bmod n$ ← $c = k^e \bmod n$

AES key

- Hybrid encryption
  - A hybrid encryption scheme uses public-key encryption to encrypt a random symmetric key, and then proceeds to encrypt the message with that symmetric key.
  - The receiver decrypts the symmetric key using the public-key encryption scheme and then uses the recovered symmetric key to decrypt the message.
  - Hybrid encryption gives the functionality of public-key encryption at the (asymptotic) efficiency of private-key encryption!

Euler

$m^{\phi(n)} \bmod n = 1$

$a^{b \bmod \phi(n)}$

$(p-1)(q-1)$

$\bmod (n)$ = pq

$ed \bmod \phi(n) = 1$

Another consideration is the efficiency of exponentiation, because with RSA, we are dealing with potentially large exponents. To see how efficiency might be increased, consider that we wish to compute x^{16}. A straightforward approach requires 15 multiplications:

자승

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

However, we can achieve the same final result with only four multiplications if we repeatedly take the square of each partial result, successively forming (x^2, x^4, x^8, x^{16}). As another example, suppose we wish to calculate $x^{11} \bmod n$ for some integers x and n. Observe that $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$. In this case, we compute $x \bmod n$, $x^2 \bmod n$, $x^4 \bmod n$, and $x^8 \bmod n$ and then calculate $[(x \bmod n) \times (x^2 \bmod n) \times (x^8 \bmod n)] \bmod n$.

10진수가 1000개면 loop 너무 많음 $x^{11} = x^8 \cdot x^2 \cdot x$

곱셈을 10번이면 해결가능

$x^{1011} = x^{1000(2)} \cdot x^{10(2)} \cdot x^{1(2)}$ 2진수 상에서 1되는 애들을 뽑아옴

Repeated Square-and-Multiply Algorithm

Algorithm Right-to-left binary exponentiation 오른쪽부터 봄. 지수를 2진수로. 지수를4.8진수

INPUT: an element $g \in G$ and integer $e \geq 1$.
OUTPUT: g^e. 지수를 e라고하면

1. $A \leftarrow 1$, $S \leftarrow g$. $\log_2 e$ (전부1일때)
2. While $e \neq 0$ do the following:
   1이면 곱. 2.1 If e is odd then $A \leftarrow A \cdot S$. 하지만 0과 반반
   flow function 2.2 $e \leftarrow \lfloor e/2 \rfloor$. 1000 자리대신 4번만 $= 1.5 \log_2 e$ 하면된다.
   2.3 If $e \neq 0$ then $S \leftarrow S \cdot S$.
3. Return(A). 짝.홀 중에 뭐냐?

답

$e = (e_t e_{t-1} \ldots e_1 e_0)_2$ 지수를 2진수로 표현

1. A $\leftarrow$ 1, S $\leftarrow$ g —스퀘어
2. For i from 0 up to t do the following:
   2.1 If $e_i = 1$, then A $\leftarrow$ A·S
   2.2 S $\leftarrow$ S·S 제곱 제곱 제곱 : tS
3. Return(A)

답을 고름 : A

Example (*right-to-left binary exponentiation*) The following table displays the values of A, e, and S during each iteration for computing g^{283}. Note that $e = 283 = 100011011_{(2)}$. □

이5

S계산=8번곱

D.B번

A	1	g	g^3	g^3	g^{11}	g^{27}	g^{27}	g^{27}	g^{27}	g^{283}
e	283	141	70	35	17	8	4	2	1	0
S	g	g^2	$(g^4)^2$	g^8	g^{16}	g^{32}	g^{64}	g^{128}	g^{256}	—

S는 계속 제곱 1 1 0 1 제곱한번더 1 0 0 0 1 그냥계산하면 283번, 지수승은8번.

Repeated Square-and-Multiply Algorithm

Algorithm Right-to-left binary exponentiation

INPUT: an element $g \in G$ and integer $e \geq 1$.
OUTPUT: g^e.

1. $A \leftarrow 1$, $S \leftarrow g$.
2. While $e \neq 0$ do the following:
   2.1 If e is odd then $A \leftarrow A \cdot S$.
   2.2 $e \leftarrow \lfloor e/2 \rfloor$.
   2.3 If $e \neq 0$ then $S \leftarrow S \cdot S$.
3. Return(A).

$e = (e_t e_{t-1} \ldots e_1 e_0)_2$

1. A ←1, S ←**g**
2. For i from 0 up to t do the following:
   2.1 If $e_i = 1$, then A ← A·S
   2.2 S ← S·S
3. Return(A)

Example (*right-to-left binary exponentiation*) The following table displays the values of A, e, and S during each iteration for computing g^{283}. Note that $e = 283 = 100011011_{(2)}$. □

A	1	g	g^3	g^3	g^{11}	g^{27}	g^{27}	g^{27}	g^{27}	g^{283}
e	283	141	70	35	17	8	4	2	1	0
S	g	g^2	g^4	g^8	g^{16}	g^{32}	g^{64}	g^{128}	g^{256}	—

Repeated Square-and-Multiply Algorithm

Algorithm Left-to-right binary exponentiation

INPUT: $g \in G$ and a positive integer $e = (e_t e_{t-1} \cdots e_1 e_0)_2$.
OUTPUT: g^e.

왼쪽에서 오른쪽으로

1. $A \leftarrow 1$.
2. For i from t down to 0 do the following:
   2.1 $A \leftarrow A \cdot A$. 답을 계속 제곱.
   2.2 If $e_i = 1$, then $\boxed{}$
3. Return(A).

2.2 If $e_i = 1$, then $A \leftarrow A \cdot g$

미리 곱해야는 높은 자리수 필요할 때만 ×g

상위 비트가 1이면 X5

아래는 S=S·S

i	8	7	6	5	4	3	2	1	0
e_i	1	0	0	0	1	1	0	1	1
A									

g g^2 g^4 g^8 g^{17} g^{35} g^{70} g^{141} g^{283}

0이면 앞에거를 제곱하고, 1이면 제곱 ×g,

Repeated Square-and-Multiply Algorithm

Algorithm Left-to-right binary exponentiation

INPUT: $g \in G$ and a positive integer $e = (e_t e_{t-1} \cdots e_1 e_0)_2$.
OUTPUT: g^e.

1. $A \leftarrow 1$.
2. For i from t down to 0 do the following:
   2.1 $A \leftarrow A \cdot A$.
   2.2 If $e_i = 1$, then $\boxed{}$
3. Return(A).

2.2 If $e_i = 1$, then $A \leftarrow A \cdot g$

i	8	7	6	5	4	3	2	1	0
e_i	1	0	0	0	1	1	0	1	1
A									

g g^2 g^4 g^8 g^{17} g^{35} g^{70} g^{141} g^{283}