Public Key Cryptography

Prof. Dong-Chan Kim
Department of Information Security, Cryptology, and Mathematics
Kookmin University
dckim@kookmin.ac.kr

Updated: 2023년 3월 8일

제 1 장

IFP-based Primitives

1977년 Ron Rivest, Adi Shamir, Leonard Adleman는 인수분해의 어려움을 기반으로 한 공개키암호 알고리듬을 제안했다 [4]. 이를 RSA 암호라고 한다.

1 도메인 파라미터

n비트 키를 사용하는 RSA-n의 도메인 파라미터(domain parameter)는 다음과 같다.

(1) p, q: 비트 길이가 (거의) $n/2$인 서로 다른 소수

(2) $N(= pq)$: n비트 양의 정수로 RSA-n은 $[0, N)$에 속한 정수를 암호화함.

표 1.1은 주요 안전성 수준에 해당하는 RSA 키길이를 보여준다[1].

표 1.1: 안전성 수준 별 RSA 파라미터 [1]

안전성 수준 (비트)	파라미터
80	RSA-1024
112	RSA-2048
128	RSA-3072
192	RSA-7680
256	RSA-15360

메모 1.1. KCMVP(Korea Cryptographic Module Validation Program, 한국 암호모듈검증 프로그램)에서는 RSA-2048/3072만 검증대상이다[2]. (2023년 기준)

[1] Recommendation for Key Management: Part 1 – General https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

[2] 암호모듈 검증대상 암호알고리즘, https://seed.kisa.or.kr/kisa/kcmvp/EgovVerification.do

2 Textbook RSA

2.1 키 생성

알고리듬 1은 RSA 키생성 과정을 보여준다.

알고리듬 1 RSA: GenKeyPair

입력: A security parameter n

출력: A public key $pk = \langle N, e \rangle$ and a secret key $sk = \langle N, p, q, d \rangle$

procedure GenKeyPair(1^n)
  Choose two distinct random $n/2$-bit strong primes p and q ▷ Primality Test
  $N \leftarrow p \times q$ ▷ Multiplication, $\mathsf{bitlen}(N) \approx n$
  Choose $e \in \{2, \ldots, \phi(N) - 1\}$ such that $\gcd(e, \phi(N)) = 1$ ▷ E.A.
  Calculate $d \in \{2, \ldots, \phi(N) - 1\}$ such that $ed \equiv 1 \pmod{\phi(N)}$ ▷ E.E.A.
  return A public key $pk = \langle N, e \rangle$ and a secret key $sk = \langle N, p, q, d \rangle$
end procedure

메모 1.2. (1) NIST FIPS 186-5 DSS 표준에서는 공개키 e를 먼저 결정하고, $\gcd(e, \phi(N)) = 1$를 만족하는 소수 p와 q를 생성한다[3]. 전자서명에서 공개키 연산은 서명 검증 연산이기 때문에 이를 고속화하기 위해 일반적으로 e를 2^{16}에서 2^{256} 사이의 수로 설정한다 [2].

(2) 강한 소수(strong prime) p란 $p \pm 1$의 모든 소인수가 충분히 큰 소수이다. 만일 작은 크기의 소인수를 가진다면 Pollard의 $p - 1$ 인수분해와 William의 $p + 1$ 인수분해 알고리듬으로 $N = pq$의 인수분해가 현실적으로 가능할 수 있다. NIST FIPS 186-5 DSS 표준은 각 소인수의 최소 비트 크기를 제시하고 있다 [2].

(3) 소수는 소수임이 확실한 증명가능한(provable) 소수와 소수일 가능성이 매우 높은 확률적(probable) 소수로 구분한다. 일반적으로 증명가능한 소수는 Shawe-Taylor 방법과 AKS 방법으로 생성하고, 확률적 소수는 Miller-Rabin 방법과 Lucas 방법으로 생성한다 [2].

[3] FIPS 186-5 Digital Signature Standard (DSS) https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

2.2 암호 연산과 복호 연산

알고리듬 2는 RSA의 암호 연산과 복호 연산 과정을 보여준다.

알고리듬 2 RSA: ENCRYPT and DECRYPT

입력: A public key $pk = \langle N, e \rangle$ and a plaintext $m \in [0, N)$
출력: A ciphertext $c \in [0, N)$ or `error`

procedure ENCRYPT($pk; m$)
  if $m > N$ **then**
    return `error`
  end if
  $c \leftarrow m^e \bmod N$
  return c
end procedure

입력: A secret key $sk = \langle N, p, q, d \rangle$ and a ciphertext $c \in [0, N)$
출력: A plaintext $m \in [0, N)$ or `error`

procedure DECRYPT($sk; c$)
  if $c > N$ **then**
    return `error`
  end if
  $m \leftarrow c^d \bmod N$
  return m
end procedure

$$m \in [0, N) \xrightarrow{\text{ENCRYPT w. } pk} c = m^e \bmod N \xrightarrow{\text{DECRYPT w. } sk} (m^e \bmod N)^d \bmod N = m.$$

정리 1.1. *정의역과 공역이 모두* $[0, N)$*인 함수* $f(x) := x^e \bmod N$*은 일대일대응 함수(permutation)이다.*

증명 함수 f는 정의역과 공역이 같기 때문에 일대일 함수임을 증명하면 된다. 즉,

$$m_1^e \equiv m_2^e \pmod{N},\ m_1, m_2 \in [0, N) \quad \Rightarrow \quad m_1 = m_2.$$

두 정수 $m_1, m_2 \in [0, N = pq)$가 $m_1^e \equiv m_2^e \pmod{N}$를 만족하면, $m_1^e \equiv m_2^e \pmod{p}$도 만족한다. 이때 $p \mid m_1$이면, $p \mid m_2$이어야 하기 때문에 m_1과 m_2가 모두 p의 배수이거나 모두 아닌 경우로 나누어 생각한다.

(1) m_1과 m_2가 모두 p의 배수인 경우 $p \mid m_1 - m_2$이다.

(2) m_1과 m_2가 모두 p의 배수가 아닌 경우에는, $\gcd(e, p-1) = 1$이므로 $ek_1 + (p-1)k_2 = 1$를 만족하는 어떤 정수 $k_1, k_2 \in \mathbb{Z}$가 존재한다. $(m_1^e)^{k_1} \equiv (m_2^e)^{k_1} \pmod{p}$이기 때문에 다음을 얻는다.

$$\begin{aligned}(m_1^e)^{k_1} = m_1^{1-(p-1)k_2} \equiv m_1(m_1^{-(p-1)k_2}) \equiv m_1(m_1^{p-1})^{-k_2} \equiv m_1 \pmod{p},\\ (m_2^e)^{k_1} = m_2^{1-(p-1)k_2} \equiv m_1(m_2^{-(p-1)k_2}) \equiv m_1(m_2^{p-1})^{-k_2} \equiv m_2 \pmod{p}.\end{aligned}$$

상기식으로부터 $p \mid m_1 - m_2$임을 알 수 있다.

동일한 논리로 $q \mid m_1 - m_2$임도 알 수 있다. 따라서 $pq \mid m_1 - m_2$이기 때문에 $m_1 = m_2$가 된다. □

Proof of Decryption. 메시지 $m \in [0, N)$은 다음의 두 가지 경우 중 하나를 만족한다.

$$\text{(경우 1) } \gcd(m, N) = 1, \qquad \text{(경우 2) } 1 < \gcd(m, N) < N \ (\Leftrightarrow \gcd(m, N) \in \{p, q\}).$$

(경우 1) $ed \equiv 1 \pmod{\phi(N)}$이므로 $ed = k\phi(N) + 1$를 만족하는 어떤 정수 $k \in \mathbb{Z}$가 존재한다. 따라서 Euler 정리에 의해 다음이 성립한다.

$$m^{ed} = m^{k\phi(N)+1} = (m^{\phi(N)})^k m \equiv m \pmod{N}.$$

(경우 2) $\gcd(m, N) = p$이면 m이 p의 배수이기 때문에 $m^{ed} \equiv m \pmod{p}$이 자명하다. $ed \equiv 1 \pmod{\phi(N)}$이므로 $ed = k(p-1)(q-1)+1$를 만족하는 어떤 정수 $k \in \mathbb{Z}$가 존재한다. 따라서 $\gcd(m, q) = 1$이기 때문에 다음과 같이 FLT에 의해 $m^{ed} \equiv m \pmod{q}$를 얻는다.

$$m^{ed} \equiv m^{k(p-1)(q-1)+1} \equiv (m^{q-1})^{k(p-1)} m \equiv m \pmod{q}.$$

$\gcd(m, N) = q$인 경우도 유사하기 때문에 결과적으로 다음이 유도된다.

$$m^{ed} \equiv m \pmod{p}, \quad m^{ed} \equiv m \pmod{q}.$$

$p \mid (m^{ed} - m)$, $q \mid (m^{ed} - m)$이므로 $pq \mid (m^{ed} - m)$가 되어 $m^{ed} \equiv m \pmod{N}$을 얻는다.

□

예제 1.1 (RSA-8)**.**

$$pk = \langle N = 143, e = 37 \rangle\,,\ sk = \langle N = 143, p = 11, q = 13, d = 13 \rangle \leftarrow \textsc{GenKeyPair}(1^8).$$

$ed = 37 \times 13 = 481 = 120 \times 4 + 1 \equiv 1 \pmod{\phi(N) = 120}$.

```
p = 13 = 0xd
q = 11 = 0xb
N = 143 = 0x8f
phi = 120 = 0x78
e = 37 = 0x25
d = 13 = 0xd
0x00->0x00->0x00: True(*)
0x01->0x01->0x01: True(*)
0x02->0x6a->0x02: True
0x03->0x2a->0x03: True
0x04->0x52->0x04: True
0x05->0x87->0x05: True
0x06->0x13->0x06: True
0x07->0x48->0x07: True
0x08->0x70->0x08: True
0x09->0x30->0x09: True
0x0a->0x0a->0x0a: True(*)
0x0b->0x0b->0x0b: True(*)
0x0c->0x0c->0x0c: True(*)
0x0d->0x75->0x0d: True
0x0e->0x35->0x0e: True
0x0f->0x5d->0x0f: True
0x10->0x03->0x10: True
0x11->0x1e->0x11: True
0x12->0x53->0x12: True
0x13->0x7b->0x13: True
0x14->0x3b->0x14: True
0x15->0x15->0x15: True(*)
0x16->0x16->0x16: True(*)
0x17->0x17->0x17: True(*)
0x18->0x80->0x18: True
0x19->0x40->0x19: True
0x1a->0x68->0x1a: True
0x1b->0x0e->0x1b: True
0x1c->0x29->0x1c: True
0x1d->0x5e->0x1d: True
0x1e->0x86->0x1e: True
0x1f->0x46->0x1f: True
0x20->0x20->0x20: True(*)
0x21->0x21->0x21: True(*)
0x22->0x22->0x22: True(*)
0x23->0x8b->0x23: True
0x24->0x4b->0x24: True
0x25->0x73->0x25: True
0x26->0x19->0x26: True
0x27->0x34->0x27: True
0x28->0x69->0x28: True
0x29->0x02->0x29: True
0x2a->0x51->0x2a: True
0x2b->0x2b->0x2b: True(*)
0x2c->0x2c->0x2c: True(*)
0x2d->0x2d->0x2d: True(*)
0x2e->0x07->0x2e: True
0x2f->0x56->0x2f: True
0x30->0x7e->0x30: True
0x31->0x24->0x31: True
0x32->0x3f->0x32: True
0x33->0x74->0x33: True
0x34->0x0d->0x34: True
0x35->0x5c->0x35: True
0x36->0x36->0x36: True(*)
0x37->0x37->0x37: True(*)
0x38->0x38->0x38: True(*)
0x39->0x12->0x39: True
0x3a->0x61->0x3a: True
0x3b->0x89->0x3b: True
0x3c->0x2f->0x3c: True
0x3d->0x4a->0x3d: True
0x3e->0x7f->0x3e: True
0x3f->0x18->0x3f: True
0x40->0x67->0x40: True
0x41->0x41->0x41: True(*)
0x42->0x42->0x42: True(*)
0x43->0x43->0x43: True(*)
0x44->0x1d->0x44: True
0x45->0x6c->0x45: True
0x46->0x05->0x46: True
0x47->0x3a->0x47: True
0x48->0x55->0x48: True
0x49->0x8a->0x49: True
0x4a->0x23->0x4a: True
0x4b->0x72->0x4b: True
0x4c->0x4c->0x4c: True(*)
0x4d->0x4d->0x4d: True(*)
0x4e->0x4e->0x4e: True(*)
0x4f->0x28->0x4f: True
0x50->0x77->0x50: True
0x51->0x10->0x51: True
0x52->0x45->0x52: True
0x53->0x60->0x53: True
0x54->0x06->0x54: True
0x55->0x2e->0x55: True
0x56->0x7d->0x56: True
0x57->0x57->0x57: True(*)
0x58->0x58->0x58: True(*)
0x59->0x59->0x59: True(*)
0x5a->0x33->0x5a: True
0x5b->0x82->0x5b: True
0x5c->0x1b->0x5c: True
0x5d->0x50->0x5d: True
0x5e->0x6b->0x5e: True
0x5f->0x11->0x5f: True
0x60->0x39->0x60: True
0x61->0x88->0x61: True
0x62->0x62->0x62: True(*)
0x63->0x63->0x63: True(*)
0x64->0x64->0x64: True(*)
0x65->0x3e->0x65: True
0x66->0x8d->0x66: True
0x67->0x26->0x67: True
0x68->0x5b->0x68: True
0x69->0x76->0x69: True
0x6a->0x1c->0x6a: True
0x6b->0x44->0x6b: True
0x6c->0x04->0x6c: True
0x6d->0x6d->0x6d: True(*)
0x6e->0x6e->0x6e: True(*)
0x6f->0x6f->0x6f: True(*)
0x70->0x49->0x70: True
0x71->0x09->0x71: True
0x72->0x31->0x72: True
0x73->0x66->0x73: True
0x74->0x81->0x74: True
0x75->0x27->0x75: True
0x76->0x4f->0x76: True
0x77->0x0f->0x77: True
0x78->0x78->0x78: True(*)
0x79->0x79->0x79: True(*)
0x7a->0x7a->0x7a: True(*)
0x7b->0x54->0x7b: True
0x7c->0x14->0x7c: True
0x7d->0x3c->0x7d: True
0x7e->0x71->0x7e: True
0x7f->0x8c->0x7f: True
0x80->0x32->0x80: True
0x81->0x5a->0x81: True
0x82->0x1a->0x82: True
0x83->0x83->0x83: True(*)
0x84->0x84->0x84: True(*)
0x85->0x85->0x85: True(*)
0x86->0x5f->0x86: True
0x87->0x1f->0x87: True
0x88->0x47->0x88: True
0x89->0x7c->0x89: True
0x8a->0x08->0x8a: True
0x8b->0x3d->0x8b: True
0x8c->0x65->0x8c: True
0x8d->0x25->0x8d: True
0x8e->0x8e->0x8e: True(*)
```

Code 1.1: Textbook RSA (Sage)

```
def show_dec_hex(str, x):
    print ("{} = {} = 0x{:x}".format(str,x,x))

#  generation of two distinct random primes p and q
bnd = 2^4
while True:
    p = random_prime(bnd-1, lbound=bnd>>1)
    q = random_prime(bnd-1, lbound=bnd>>1)
    if p != q:
        N, phi = p*q, (p-1)*(q-1)
        break

#  generation of a public key e and a secret key d
while True:
    e = ZZ.random_element(2,phi)
    if 1 == gcd(e,phi):
        d = (e.xgcd(phi)[1])%phi
        if e != d:
            break

# show RSA parameter
show_dec_hex("p", p); show_dec_hex("q", q);
show_dec_hex("N", N); show_dec_hex("phi", phi);
show_dec_hex("e", e); show_dec_hex("d", d);

for m in [0..N-1]:
    c  = power_mod(m,e,N)
    m1 = power_mod(c,d,N)
    if m == c:
        print ("0x{:02x}->0x{:02x}->0x{:02x}: {}(*)".format(m, c, m1, m == m1))
    else:
        print ("0x{:02x}->0x{:02x}->0x{:02x}: {}".format(m, c, m1, m == m1))
```

질문 1.1. (1) $N = pq$를 인수분해할 수 있다면 무엇을 할 수 있는가?

(2) 만일 $p = q$이면 어떤 문제가 있는가?

참고 문헌

[1] Nist special publication 800-57 part 1 recommendation for key management (revision 5), 2020.

[2] Fips pub 186-5 federal information processing standards publication digital signature standard (dss), February 2023. U.S.Department of Commerce/National Institute of Standards and Technology.

[3] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[4] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.