보안모듈구현실습

RSA 프리미티비

장남수

RSA 프리미티비

RSA 키 생성의 이해

- RSA 키 생성
  - 공개키 : (n,e)
  - 비밀키 : (p,q,d)
  - 키 생성의 구분
    - 공개키 e를 랜덤하게 생성하는 경우
    - 공개키 e를 임의의 값으로 고정하는 경우

RSA키는 공개키와 비밀키로 구분됩니다. 이때 공개키는 n과 e이며, 비밀키는 p,q,d입니다. RSA 키 생성은 공개키 e를 고정하는 경우와 랜덤하게 생성하는 경우 파라미터 생성 순서가 다소 달라집니다. 각각의 경우에 따른 파라미터 생성 방법을 살펴보도록 하겠습니다.

RSA 표준

- PKCS #1 v2.2
- FIPS PUB 186-4

RSA 키 생성의 이해

- RSA 키 생성

공개키 e를 랜덤하게 생성하는 경우

1. 두 개의 큰 소수 p와 q를 생성한다.

2. n=pq를 계산하여 n을 구한다.

3. 주어진 n에 대하여 φ(n)=(p-1)(q-1)을 계산한다.

4. gcd(φ(n),e)=1을 만족하는 정수 e(1<e<φ(n))를 선택한다.

5. 유클리드 알고리즘 등 역원 계산 알고리즘을 이용하여
ed=1 mod φ(n)을 만족하는 d를 구한다.

먼저 랜덤하게 생성하는 경우를 살펴보겠습니다. 가장 먼저 안전성을 위하여 큰 소수 p와 q를 랜덤하게 생성합니다. 그리고 p와 q의 곱을 n이라 정의하면, 오일러 함수에 의하여 φ(n)=(p-1)(q-1)가 됩니다. 그리고 φ(n)보다 작고 φ(n)과 서로소인 공개키 e를 선택합니다. 그러면 오일러 정리에 의하여 비밀키 d가 결정이 되게 됩니다. 이때 d는 복호화를 위하여 ed=1 mod φ(n)을 유일하게 만족하는 값입니다. 이와 같은 키 생성 과정이 완료되면, 비밀키 (p,q,d)와 공개키 (n,e)을 얻게 됩니다. 이와 같은 방법이 일반적인 방법입니다.

RSA 키 생성의 이해

- RSA 키 생성

공개키 e를 임의의 값으로 고정하는 경우

1. 공개키 **e**를 입력 받는다. **(65537=2^16+1)**
2. **gcd(p-1,e)=1**을 만족하는 **p**를 생성한다.
3. **gcd(q-1,e)=1**을 만족하는 **q**를 생성한다.
4. **n=pq**를 계산하여 **n**을 구한다.
5. 주어진 **n**에 대하여 **φ(n)=(p-1)(q-1)**을 계산한다.
6. 유클리드 알고리즘 등 역원 계산 알고리즘을 이용하여
   ed=1 mod φ(n)을 만족하는 **d**를 구한다.

다음으로 임의의 e값을 사용하는 경우에 대하여 살펴보겠습니다. 가장 먼저 e를 입력을 받습니다. 그리고 e와 p-1이 서로소가 되는 p를 랜덤하게 생성합니다. 마찬가지로 q 또한 e와 q-1이 서로소가 되는 q를 랜덤하게 생성합니다 그리고 p와 q의 곱을 n이라 정의하면, 오일러 함수에 의하여 φ(n)=(p-1)(q-1)가 됩니다. 그러면 오일러 정리에 의하여 비밀키 d가 결정이 되게 됩니다. 이때 d는 복호화를 위하여 ed=1 mod φ(n)을 유일하게 만족하는 값입니다. 현재 대부분 고속화를 위하여 공개키 e는 $2^{16}+1$을 사용하고 있기 때문에 공개키 e를 랜덤하게 선택하는 경우 보다 고정하여 사용하는 경우가 대부분입니다.

RSA 키 생성의 이해

- **RSA public key**
  - n the RSA modulus, a positive integer
  - e the RSA public exponent, a positive integer

- RSA public exponent e : 3과 n-1 사이의 값으로 t를 LCM(p-1,q-1)라 하면, GCD(e, t) = 1을 만족한다.

다음으로 pkcs #1에 정의된 공개키 타입을 살펴보겠습니다. 공개키는 잘 아시는 바와 같이 두 소수 p와 q의 곱인 n과 지수 e를 공개키라 정의합니다. 이때, 공개키 e는 3과 n-1 사이의 값으로 t를 p-1과 q-1의 최소공배수라 할 때, e와 t는 서로소이어야 합니다.

RSA 키 생성의 이해

- **RSA public key**
  - *d* the RSA private exponent, a positive integer
  - p the first factor, a positive integer
  - q the second factor, a positive integer
  - dP the first factor's CRT exponent, a positive integer
    - $e \cdot dP = 1 \pmod{(p-1)}$
  - dQ the second factor's CRT exponent, a positive integer
    - $e \cdot dQ = 1 \pmod{(q-1)}$
  - qInv the (first) CRT coefficient, a positive integer
    - $q \cdot qInv = 1 \pmod{p}$

- RSA 복호화 및 서명생성을 고속화하기 위하여 CRT를 사용하는데 이때 사용하기 위하여 추가적으로 dP, dQ, qInv을 추가적으로 정의하여 사용할 수 있습니다.

다음으로 pkcs #1에 정의된 비밀키 타입을 살펴보겠습니다. 비밀키는 두 소수 p, q와 지수 e를 비밀키라 정의합니다. 이때, 지수 d는 phi(n)에 대하여 e의 역원을 d라 합니다. RSA 복호화 및 서명생성을 고속화하기 위하여 CRT를 사용하는데 이때 사용하기 위하여 추가적으로 dP, dQ, qInv을 추가적으로 정의하여 사용할 수 있습니다. dp는 p-1에 대한 e의 역원이고 dq는 q-1에 대한 e의 역원이며 Qinv는 p에 대한 q의 역원입니다. CRT를 사용하지 않는 경우 추가 파라미터는 정의할 필요가 없습니다. 이후에 자세히 설명을 하겠지만 CRT를 사용하는 경우 이론적으로 4배 정도 속도가 빨라지므로 고속화를 위하여 위의 파라미터를 정의하고 꼭 CRT를 사용해야만 합니다.

RSA 암복호화의 이해

- RSA 고속화
  - CRT를 사용하여 RSA 복호화 및 서명생성을 (고속화)할 수 있으며, 이를 위하여 (비밀키) 쌍에 dP, dQ, qInv을 추가적으로 정의한다.
  - PKCS#1에는 암복호화 및 전자서명의 메시지 인코딩을 위하여 추가적으로 (MGF)를 정의하여 사용하며, (MGF)는 (해쉬함수) 기반으로 구성되어 있다.
  - 1024비트 정수 곱셈의 속도와 512비트 정수 정수 곱셈의 속도는 이론적으로 (4배) 차이가 나며, (CRT)를 이용하면 1024비트의 곱셈을 512비트로 줄여서 계산할 수 있으므로 더 빠르게 암복호화 및 전자서명을 수행할 수 있다.

RSA 암복호화의 이해

- PKCS #1의 암호화 스킴
  - 암호화 스킴은 암호화 연산과 복화화 연산으로 구성된다.
  - RSA 표준 PKCS#1에는 두 개의 암호화 스킴 RSAES-OAEP와 RSAES-PKCS1-v1_5 가 있다.
    - RSAES는 RSA Encryption Scheme을 의미합니다.
  - 표준에서는 RSAES-OAEP를 권장하고 있으며 RSAES-PKCS1-v1_5은 이전버전과의 호환성을 위해 제공될 뿐 사용을 권장하지 않는다.
    - 국내 인증제도인 KCMVP : 2011년 까지는 1.5 버전을 보호함수로 포함하였으나 현재는 OAEP버전만을 보호함수로 지정하고 있다.

RSA 암복호화의 이해

- RSA 암복호화 프리미티브

[입력] (n,e) : RSA 공개키
m : n보다 작은 메시지
[출력] c : n보다 작은 암호문
[에러] : 메시지가 범위를 벗어난 경우

단계
1. 만약에 메시지가 범위를 벗어난 경우 에러를 출력하고 종료한다.
2. $c = m^e \bmod n$.
3. 출력(c)

[입력] (n,d) : RSA 공개키 n과 비밀키 d
c : n보다 작은 암호문
[출력] m : n보다 작은 메시지
[에러] : 암호문이 범위를 벗어난 경우

단계
1. 만약에 암호문이 범위를 벗어난 경우 에러를 출력하고 종료한다.
2. $m = c^d \bmod n$.
3. 출력(m)

문제점

키가 동일한 경우 같은 평문에 대해서는 항상 같은 암호문이 생성됨

RSA 암복호화의 이해

- RSA-OAEP 암호화

옵션 : 해쉬함수(hLen : 해쉬함수 출력의 바이트 수, MGF(mask generation function)
입력 : 공개키 (n, e), M : 메시지 (mLen:메시지의 바이트 수, mLen ≤ k – 2hLen – 2),
L : 옵션레이블(제공되지 않는 경우 empty 스트링)

출력 : 암호문 C(k 바이트의 암호문)
에러 : 메시지가 또는 레이블이 범위를 벗어난 경우

Steps:

1. 길이 체크 :
   a. 만약 레이블 L이 해쉬함수의 입력 길이보다 긴 경우 에러 (2^{61}-1보다 큰 경우)
   b. 만약 mLen > k – 2hLen – 2인 경우 에러
2. EM = EME-OAEP encoding(M, mLen, L, lLen)
3. RSA 암호화 :
   a. m = OS2IP (EM).
   b. c = RSAEP ((n, e), m).
   c. C = I2OSP (c, k).
4. 암호문 C 출력

$$OAEP\_encode(m)^e \bmod n = c$$

RSA 암복호화의 이해

- RSA-OAEP 복호화

옵션 : 해쉬함수(hLen : 해쉬함수 출력의 바이트 수, MGF(mask generation function)
입력 : 비밀키 K (k : 모듈러 n의 바이트 수), C : 암호문 (2hLen+2 ≤ 암호문의 바이트 수),
L : 옵션레이블(제공되지 않는 경우 empty 스트링)

출력 : 메시지 M(mLen ≤ k – 2hLen – 2)
에러 : 암호문 또는 레이블이 범위를 벗어난 경우

Steps:

1. 길이 체크 :
   a. 만약 레이블 L이 해쉬함수의 입력 길이보다 긴 경우 에러
   b. 만약 암호문의 길이가 모듈러 보다 큰 경우 에러
2. RSA 암호화 :
   a. c = OS2IP (C).
   b. m = RSADP (K, c).
   c. EM = I2OSP (m, k).
3. M = EME-OAEP decoding(EM, L, lLen)
4. 메시지 M 출력

$c^d \bmod n = EM \rightarrow OAEP\_decode(EM) = m$

RSA 프리미티브 실습

RSA 키 생성 실습

- 2048비트 RSA 키 쌍을 만들고 암호화를 통하여 검증하라.