

RSA, Digital signature 구현과제

12141595 이용준

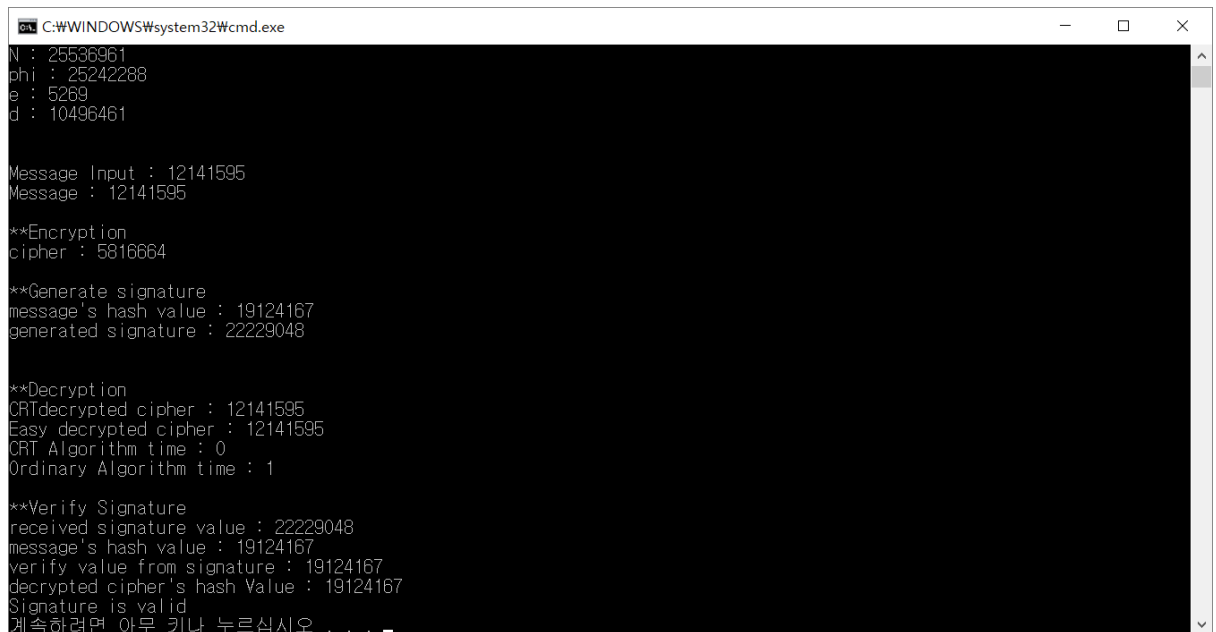
구현 환경 : OS- Window 10

IDE – Visual Studio 2017

구현 언어 : C++

결과 화면

1. Digital signature 가 유효.



```
C:\WINDOWS\system32\cmd.exe
N : 25536961
phi : 25242288
e : 5269
d : 10496461

Message Input : 12141595
Message : 12141595

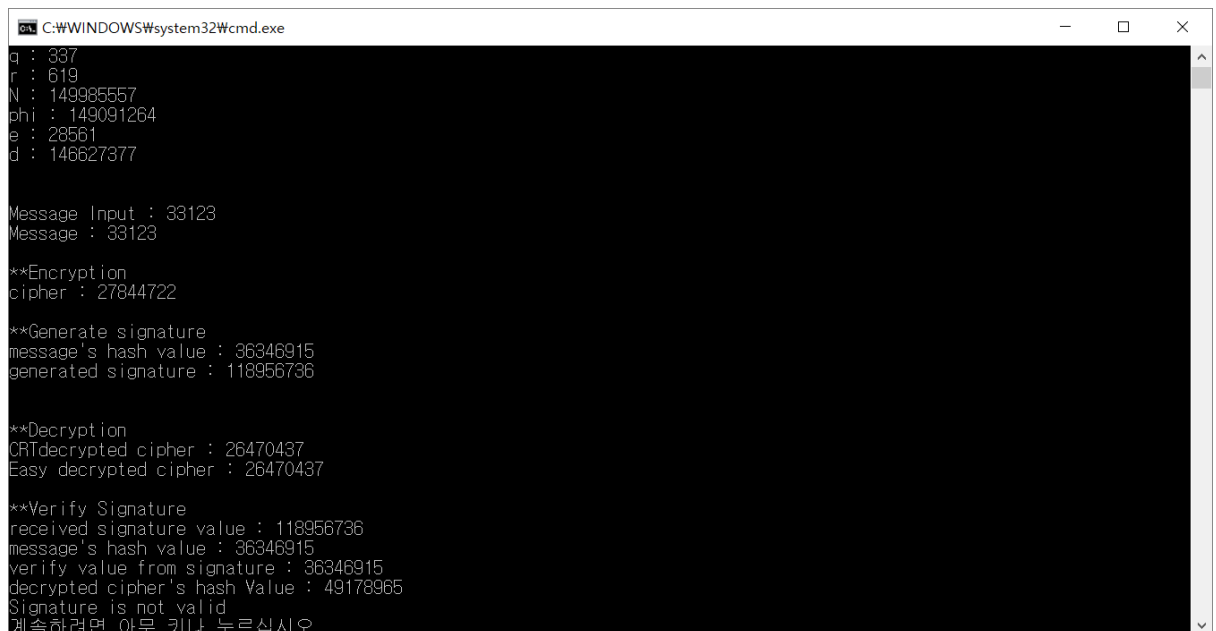
**Encryption
cipher : 5816664

**Generate signature
message's hash value : 19124167
generated signature : 22229048

**Decryption
CRTdecrypted cipher : 12141595
Easy decrypted cipher : 12141595
CRT Algorithm time : 0
Ordinary Algorithm time : 1

**Verify Signature
received signature value : 22229048
message's hash value : 19124167
verify value from signature : 19124167
decrypted cipher's hash Value : 19124167
Signature is valid
계속하려면 아무 키나 누르십시오 . . .
```

2. 변조로 인한 Digital signature 무효



```
C:\WINDOWS\system32\cmd.exe
d : 337
r : 619
N : 14998557
phi : 149091264
e : 28561
d : 146627377

Message Input : 33123
Message : 33123

**Encryption
cipher : 27844722

**Generate signature
message's hash value : 36346915
generated signature : 118956736

**Decryption
CRTdecrypted cipher : 26470437
Easy decrypted cipher : 26470437

**Verify Signature
received signature value : 118956736
message's hash value : 36346915
verify value from signature : 36346915
decrypted cipher's hash Value : 49178965
Signature is not valid
계속하려면 아무 키나 누르십시오 . . .
```

코드 별 설명

-p,q,r,e 난수 생성

10bit난수를 생성(1024미만의 수)하고 miller- rabin 알고리즘을 통해 소수임을 확인하고 아니면 다시 난수를 생성하게끔 코드를 구성하였다. E의 경우 오일러-phi함수와 서로소인 수를 찾았다. 그래야 역원이 존재하기 때문이다(Zn그룹에서 n과 서로소인 경우가 unit이다). 그렇다면 핵심은 miller-rabin알고리즘을 통한 소수 검증이 된다,

```
bool miller_rabin(int n, int a) { //true -> probably prime
    int d = n - 1;
    while (d % 2 == 0) {
        if (SquareandMultiply(a, d, n) == n - 1)
            return true;
        d /= 2;
    }
    int tmp = SquareandMultiply(a, d, n);
    return tmp == n - 1 || tmp == 1;
}
```

$n-1 = 2^s \cdot d$ 로 표현되는데 s가 높은 승수부터 검증한다. $a^{(2^s \cdot d)} \bmod n$ 이 $n-1$ 과 합동일 경우 소수일 가능성이 있다. 2로 모두 나누어 $n-1$ 이 홀수인 어느 수가 될 경우 $a^d \bmod n$ 이 1과 합동일 경우 소수일 가능성이 있다.

N은 검증하고 싶은 숫자이고 a는 $n-1$ 이하의 임의의 수이다. Miller rabin 알고리즘을 20회 수행하면서 a의 값을 적절히 사용하면 된다. 검증하고 싶은 숫자가 1024보다 작은 수인데, $n < 1,373,653$ 일 경우 $a = 2,3$ 만 검증해봐도 accuracy는 충분히 높다. 그렇기 때문에 20회를 2부터 순차적으로 검사했을 때 소수로 판별된다면 소수로 봐도 무방하다.

-RSA 암호화 복호화

RSA암호화는 Square and Multiplication for mod를 구현하면 쉽게 계산할 수 있다.

```
int SquareandMultiply(int a, int x, int n) {
    if (x == 0)
        return 1;
    else if (x == 1)
        return a%n;
    else {
        unsigned long long r=1, p=a;
        while (x)
        {
            if (x & 1) // 1bit set이면 곱셈
                r = (r*p) % n;
            p = (p*p) % n;
            x >>= 1;
        }
        return r;
    }
}
```

Square and Multiplication 알고리즘은 비교적 간단하다. 지수를 2진수로 표현해 1이면 곱셈과 제곱을 0이면 제곱만 수행하여 준다. bit연산을 통해 쉬프트해가며 연산한다.

메시지가 암호화 되었으면 복호화를 해주어야 할텐데, 복호화는 Square and Multiplication으로 간단하게 secret key를 지수승 하는 방법과 CRT_RSA로 빠르게 복호화하는 방법 두 가지가 있다. 캡처에 속도 측정을 한 결과가 있는데 속도는 CRT가 어떠한 경우에도 더 빠르게 나타났다.

```
int Decryption(int C, int d, int p, int q, int r) { //CRTdecryption ,p,q,r are prime
    unsigned long long m1, m2, m3,x;
    unsigned long long c1 = SquareandMultiply(C, d % (p - 1), p),
        c2 = SquareandMultiply(C, d % (q - 1), q),
        c3 = SquareandMultiply(C, d % (r - 1), r);
    unsigned long long n = p * q * r;
    m1 = c1 * q * r * (ExtendedEuclideanAlgo(q * r, p)); // ps + tqr = 1인 t
    m2 = c2 * p * r * (ExtendedEuclideanAlgo(p * r, q));
    m3 = c3 * p * q * (ExtendedEuclideanAlgo(p * q, r));
    x = (m1+m2+m3) % n;
    return x;
}
```

CRT_RSA는 $n=pqr$, $m = c^d \bmod n$ 으로부터 $m = c^{dp} \bmod p = c^{dq} \bmod q = c^{dr} \bmod r$ ($d_p = d \bmod (p-1), d_q = d \bmod (q-1), d_r = d \bmod (r-1)$) 을 만족한다. P,q,r이 소수임으로 $\bmod(p-1)$ 등으로 취해도 문제없다. Chinese Remainder Theorem에 의해 $0 < m < pqr$ 인 m 이 유일하게 존재한다. 그리고 m 은

$$m = [(c^{dp}qr[(qr)^{-1} \bmod p] + c^{dq}pr[(pr)^{-1} \bmod q] + c^{dr}[(pq)^{-1} \bmod r]) \bmod n$$

을 만족한다. 역원 확장 유클리디안 알고리즘으로 구할 수 있다. Overflow가 자주 발생했는데 unsigned long long을 사용하면 overflow는 없었다.

-Digital Signature

RSA기반 digital Signature는 해쉬된 값을 secret key로 digital signature를 만들고 cipher와 쌍을 이룬다.

검증자는 cipher를 복호화하여 hash하고, 후첨된 digital signature를 공개키로 연산하여 복호화된 메시지의 해쉬값과 비교함으로 검증이 가능하다. 주어진 xxHash가 64bit hash함수임으로 반드시 64bit로 인자를 넣어 줘야한다. 처음에 int로 하였을 때 계속 값이 이상했었으나 unsigned long long으로 문제를 해결하였다. 서명 생성과 검증연산은 Square and mulpicatoin으로 쉽게 가능하다

```
int v1 = DShash(decryptedCipher, N);      ->복호화된 메시지를 해싱
int v2 = SquareandMultiply(sig, e, N);    -> 후첨된 전자서명을 공개키로 연산
cout << "Wnmessage's hash value : " << hashValue << "Wnverify value from signature : " << v2 <<
"Wndecrypted cipher's hash Value : " << v1;
if (v1 == v2) // 검증
    cout << "WnSignature is validWn";
else
    cout << "WnSignature is not validWn";
```