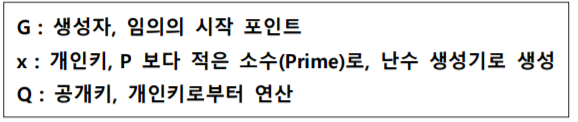
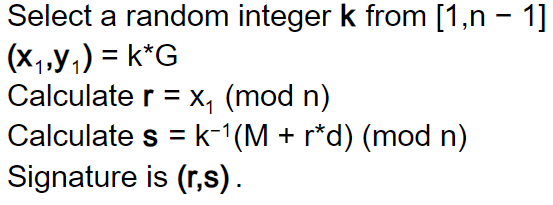
2019 암호경진대회 : 3번 문제 답안

ECDSA는 DSA에 ECC알고리즘을 적용한 디지털 서명방식이다. ECC는 타원곡선의 이산로그 문제에 기반한 공개키 암호화 방식이다.

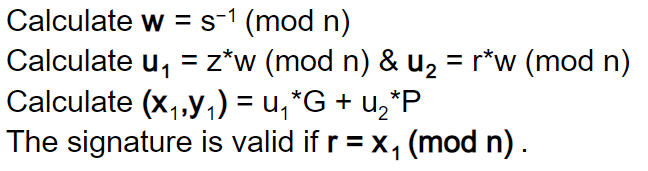
ECDSA의 키생성 과정은



서명 생성과정 :



서명 검증과정 :



ECDSA의 키생성 및 검증과정의 공식은 이러하다.

그렇다면 이 문제에서 어떻게 어떤 취약점으로 두개의 서명을 가지고 비밀키를 찾아낼 수 있을까?

**취약점**

이 문제의 취약점은 서로다른 두서명의 앞 32byte가 같은 값을 가진다는 것에 있다. Signature(r,s)의 r값이 동일하다는 것인데, 이는 같은 난수인 k를 2개의 서명에 사용하게 됨으로써 나오는 취약점이다.

위 식에 의해서 서로 다른 서명 S1, S2는 아래와 같은 식을 가지게 된다

**S1 = (M1 + r \* x) / k (mod n)**

**S2 = (M2 + r \* x) / k (mod n)**

// 이 식은 곧

**S1 \* k = M1 + r \* x (mod n) --- (1)**

**S2 \* k = M2 + r \* x (mod n)**

// 이러한 두 식을 연립하여 아래와 같은 식을 만들 수 있다.

**(S1 – S2) \* k = M1 – M2 (mod n)**

**k = (M1 – M2) / (S1 – S2) (mod n)**

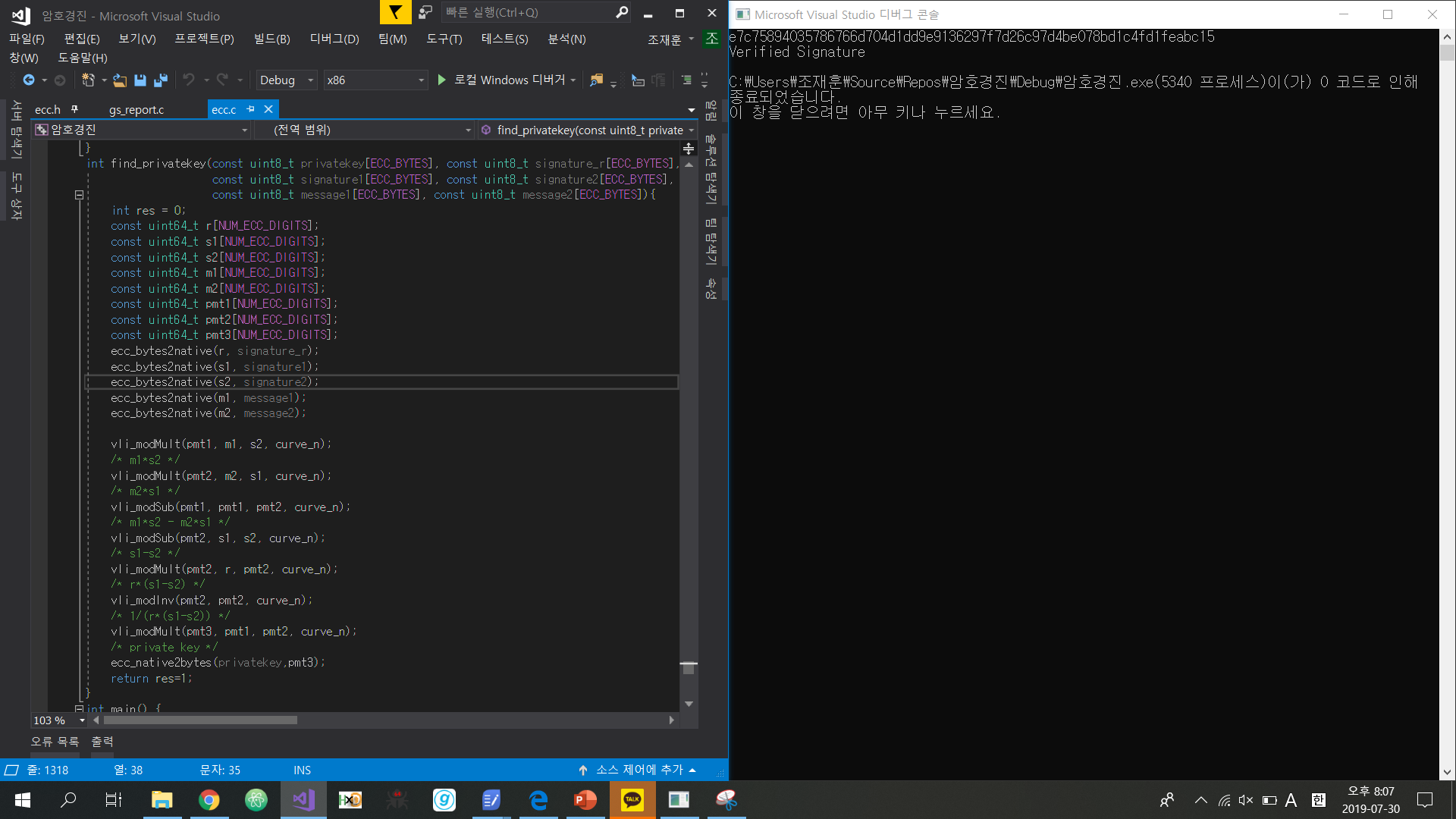
// 이로써 k값을 구한 후 (1)번식에 대입하게 되면

**x = (S1 \*(M1 – M2) / (S1 – S2) – M1) / r (mod n)**

**x = (M1 \* S2 – M2 \* S1) / (r \* (S1 – S2)) (mod n)**

// 이렇게 x값을 구할 수 있다.

**결과 화면**



**실제 구현 코드**

실제 주어진 문제의 생성시 사용한 ECDSA library에 구현된 함수들을 이용하여 보다 편하게 연산할 수 있었다.

int find\_privatekey(const uint8\_t privatekey[ECC\_BYTES], const uint8\_t signature\_r[ECC\_BYTES],

const uint8\_t signature1[ECC\_BYTES], const uint8\_t signature2[ECC\_BYTES],

const uint8\_t message1[ECC\_BYTES], const uint8\_t message2[ECC\_BYTES]){

int res = 0;

const uint64\_t r[NUM\_ECC\_DIGITS];

const uint64\_t s1[NUM\_ECC\_DIGITS];

const uint64\_t s2[NUM\_ECC\_DIGITS];

const uint64\_t m1[NUM\_ECC\_DIGITS];

const uint64\_t m2[NUM\_ECC\_DIGITS];

const uint64\_t pmt1[NUM\_ECC\_DIGITS];

const uint64\_t pmt2[NUM\_ECC\_DIGITS];

const uint64\_t pmt3[NUM\_ECC\_DIGITS];

ecc\_bytes2native(r, signature\_r);

ecc\_bytes2native(s1, signature1);

ecc\_bytes2native(s2, signature2);

ecc\_bytes2native(m1, message1);

ecc\_bytes2native(m2, message2);

vli\_modMult(pmt1, m1, s2, curve\_n);

/\* m1\*s2 \*/

vli\_modMult(pmt2, m2, s1, curve\_n);

/\* m2\*s1 \*/

vli\_modSub(pmt1, pmt1, pmt2, curve\_n);

/\* m1\*s2 - m2\*s1 \*/

vli\_modSub(pmt2, s1, s2, curve\_n);

/\* s1-s2 \*/

vli\_modMult(pmt2, r, pmt2, curve\_n);

/\* r\*(s1-s2) \*/

vli\_modInv(pmt2, pmt2, curve\_n);

/\* 1/(r\*(s1-s2)) \*/

vli\_modMult(pmt3, pmt1, pmt2, curve\_n);

/\* private key \*/

ecc\_native2bytes(privatekey,pmt3);

return res=1;

}

**<정답>**

**[e7c75894035786766d704d1dd9e9136297f7d26c97d4be078bd1c4fd1feabc15]**