2021 암호분석경진대회			
3번 문제 풀이			
구분	일반부	팀명	HB

0. 개요

문제 풀이를 위한 전체 코드는 python3.8으로 작성하였다. 문제 풀이를 위한 코드는 1,2,3번 순으로 3-1.py, 3-2.py, 3-3.py 이며 문제 해결을 위한 모듈로 SimpleStack.py, ECDSA.py, ExtendedStack.py을 작성하였다. 또한 첨부된 실행 파일 3-1.exe, 3-2.exe, 3-3.exe은 1,2,3번 문제에서 요구하는 SW를 실행 파일으로 만든 것이며, 정상 동작하는 경우와 정상 동작하지 않는 경우 (ex. 입력이 잘못된 경우, 서명 검증에 실패한 경우 등)를 적용할 수 있다. 실행 파일 생성은 pyinstaller (ver 3.6)을 활용하였다.

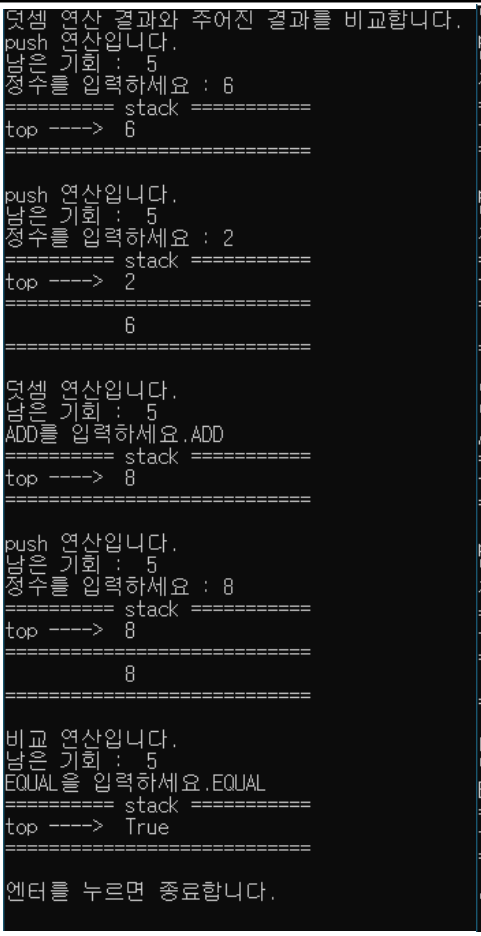
1. 3-1 풀이

스택이란, FILO (First In, Last Out) 구조를 가진 자료구조로, 가장 위에 쌓인 top 위치의 자료가 pop(제거)되지 않으면 아래의 자료들은 스택에 계속 남겨져 있고, top 위치의 자료가 pop되면, top의 바로 아래에 위치한 자료가 top이 되는 구조이다. 또한 새로운 자료가 push(삽입)되면 해당 자료가 top이 된다.

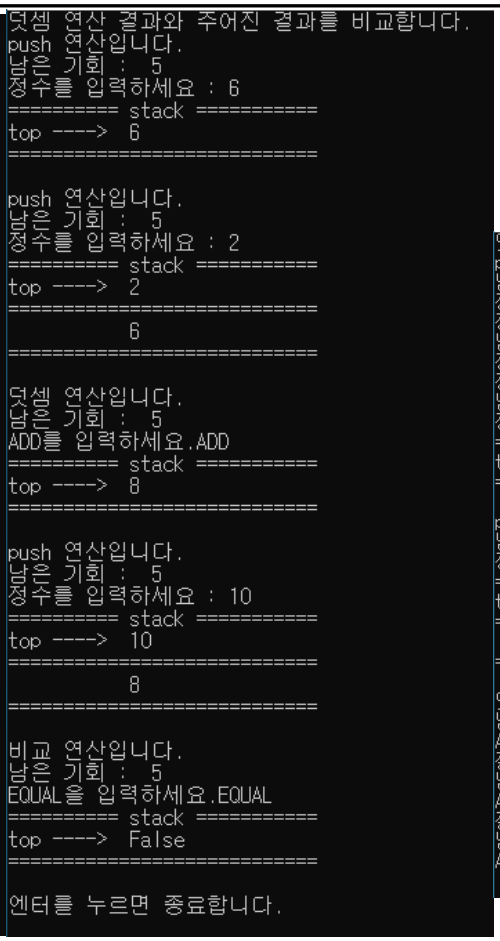
스택을 기반으로 두 정수를 더하는 간단한 덧셈 연산과 주어진 결과 값과 연산 결과값을 비교하는 스택 프로세서를 구현하기 위해 SimpleStack.py 모듈을 만들었으며, 해당 실행 결과는 3-2.exe를 통해 확인 가능하다. SimpleStack.py 모듈의 구성 함수는 다음과 같은 역할을 한다.

- init() : stack을 나타내기 위하여 python의 list 자료구조를 활용하였다. init() 함수가 호출되면 빈 stack과 입력의 기회 cnt가 리턴 된다. 잘못된 입력을 계속해서 받을 경우 무한 루프에 빠질 수 있기 때문에 명령어와 값의 입력의 기회를 5번으로 제한하였다. cnt가 0이 되면 해당 프로그램은 종료된다.
- print_cnt(cnt) : 남은 기회를 출력한다.
- print_stack(stack) : stack의 전체 내용을 출력한다. stack이 비어있을 경우 “stack is empty”를 출력한다.
- push(stack, cnt) : 정수값이 입력되면 stack에 push하는 함수이다. 정수값이 입력되지 않을 시 cnt가 1씩 감소한다.
- add(stack, cnt) : 덧셈 연산을 수행하는 함수이다. stack의 pop 연산을 2번 실행하여 해당 결과를 덧셈 연산한 뒤 stack에 push한다. 해당 실행파일에서 add(stack, cnt)를 올바르게 수행하기 위한 입력값은 ADD, Add, add, + 중 하나이다. 올바르지 않은 값이 입력되면 cnt가 1씩 감소하며, stack의 pop 연산을 2번 실행할 수 없는 경우, IndexError로 인해 프로그램이 종료된다.
- equal(stack, cnt) : 비교 연산을 수행하는 함수이다. stack의 pop 연산을 2번 실행하여 stack 내의 2개의 정수 자료가 일치하는지 확인한다. 일치할 시 stack에 “True”를 push, 일치하지 않을 시 “False”를 push 한다. 해당 실행파일에서 equal(stack, cnt)를 올바르게 수행하기 위한 입력값은 EQUAL, Equal, equal, ==, = 중 하나이다. 올바르지 않은 값이 입력되면 cnt가 1씩 감소하며, stack의 pop 연산을 2번 실행할 수 없는 경우, IndexError로 인해 프로그램이 종료된다.

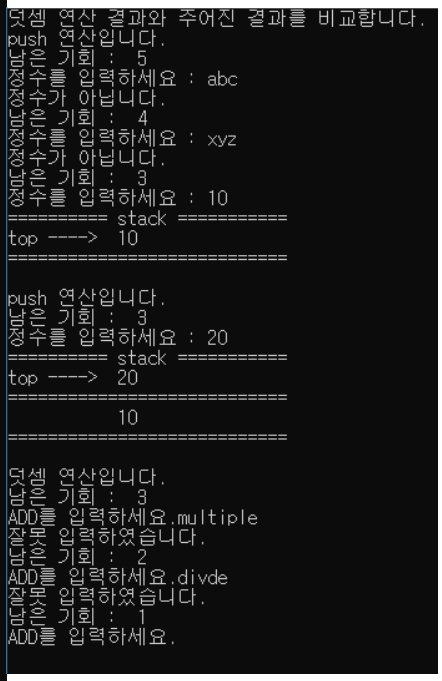
아래는 3-1.exe의 덧셈 후 비교 연산 성공, 실패, 입력값이 잘못 입력된 경우 3가지에 대한 실행 이미지이다.



(덧셈 후 비교 성공)



(덧셈 후 비교 실패)



(잘못된 입력)

2. 3-2 풀이

ECDSA는 타원곡선암호(ECC)를 이용한 전자서명 알고리즘이다. ECDSA의 서명 생성 및 검증 알고리즘은 다음과 같다.

G : 타원곡선의 generator, n : G의 차수 (n*G=O(무한원점, 타원곡선의 identity), n은 충분히 큰 소수), m : 메시지, d : 서명 검증을 받고자 하는 자의 개인키 (random값, [1, n-1]의 정수), Q : 서명 검증을 받고자 하는 자의 공개키 (Q=dG)

서명 생성

1. $e = H(m)$이며 H는 암호학적 해쉬함수
2. z는 e의 binary 값에서 L_n번째까지 자른 값 (L_n : n의 bit length)
3. 난수 k를 [1, n-1]에서 고른다.
4. $(x,y) = dG$를 계산한다.
5. $r = x \pmod n$을 계산한다. r=0 이면 3번으로 돌아가 k를 다시 고른 뒤 순서대로 진행한다.
6. $s = k^{-1}(z+rd) \pmod n$을 계산한다. s=0 이면 3번으로 돌아가 k를 다시 고른 뒤 순서대로 진행한다.
7. 완성된 서명은 (r,s)이다.

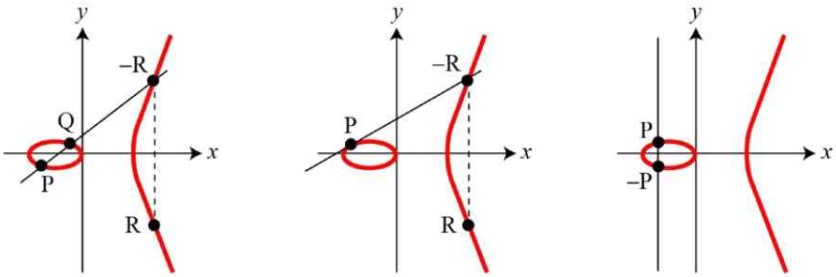
서명 검증 - 곡선 위의 점 인증

1. Q가 무한원점이 아니면서 곡선 위의 점인지 확인한다.
2. $nQ = O$ 인지 확인한다.

서명 검증 - 서명 유효성 인증

1. $r, s \in [1, n-1]$ 인지 확인한다. 아니면 서명은 무효이다.
2. $e = H(m)$을 계산한다.
3. z는 e의 binary 값에서 L_n번째까지 자른 값 (L_n : n의 bit length)
4. $w = s^{-1} \pmod n$을 계산한 다음, $u_1 = zw, u_2 = rw \pmod n$을 계산한다.
5. $(x, y) = u_1 G + u_2 Q$를 계산한다. (using Shamir's trick) $(x, y) = O$이면 서명은 무효이다.
6. $r = x \pmod n$일 때 서명은 유효하다. 아니면 서명은 무효이다.

또한 F(p) (p는 큰 소수, char(F) > 3)에서 잘 정의된 타원곡선은 $y^2 = x^3 + ax + b$이며 타원곡선 상의 두 점 P, Q에 대한 덧셈 연산은 다음과 같다.



$$P(x_1, y_1) + Q(x_2, y_2) = R(x_3, y_3)$$

1. $P = Q$

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1\,,\ y_3 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)^2 (x_1 - x_3)$$

2. $P \neq Q$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2\,,\ y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 (x_1 - x_3)$$

그리고 ECC P256r1 (ordinary curve)의 parameter는 NIST SEC 2: Recommended Elliptic Curve Domain Parameters (Ver 2.0)에서 추천하는 값(secp256r1)을 사용하였다. 해당 parameter(hex)는 다음과 같다.

$y^2 = x^3 + ax + b$ (over F_p)

p = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF

a = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC

b = 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B

S : seed for random , $G = (G_x, G_y)$: generator of EC , n : order of G

S = C49D3608 86E70493 6A6678E1 139D26B7 819F7E90

G_x = 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296

G_y = 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5

n = FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551

그리고 메시지 해싱 과정에 사용된 해쉬함수는 LSH256을 사용하였고, KISA에서 제공하는 LSH 소스코드 (ver 1.0.2)를 활용하였다. LSH256 해쉬함수에 의해 임의의 길이의 메시지가 256bit 값으로 해싱된다. 해당 소스코드 파일 및 사용 매뉴얼을 문제 풀이 소스코드 및 실행파일과 같이 첨부하였다.

다음으로 ECDSA 서명 생성 및 검증을 위한 함수가 담긴 ECDSA.py 모듈에 대한 설명은 다음과 같다.

- inverse(num, mod) : modulo mod에서 num의 곱셈에 대한 역원을 계산한다.
- doubling(Qx, Qy, Fp, Ea) : 타원곡선이 F(p)에서 정의되고, 타원곡선의 x의 계수가 a일 때 (Qx, Qy) + (Qx, Qy) 의 연산 결과를 리턴한다.
- add(Px, Py, Qx, Qy, Fp) : 타원곡선이 F(p)에서 정의될 때 (Px, Py) + (Qx, Qy) 의 연산 결과를 리턴한다.
- multiple(Qx, Qy, k, Fp, Ea) : Q=(Qx, Qy) 일 때, kQ의 결과를 리턴한다.
- make_key() : [1, n-1]의 random값을 개인키로 하여 개인키와 공개키를 생성하고 리턴한다.
- Hashing(msg) : 메시지 msg를 LSH256 해쉬함수에 적용한 해쉬값을 리턴한다.
- make_signature(msg, d) : 위 서명 생성 과정을 통해 개인키 d를 사용하여 msg에 대한 서명을 생성한다. 서명이 (r, s) 일 때 (r, -s) 또한 올바른 서명임을 확인하기 위하여 2개의 서명을 리턴한다.
- sig_verifiaction(sig, Q, msg) : 서명 sig가 메시지 msg에 대한 올바른 서명임을 공개키 Q를 통해 검증한다. 위 서명 검증 과정을 거쳐 올바른 서명일 때 “signature is valid”를 리턴하고, 6번 과정을 만족하지 못할 시 “signature is invalid”를 리턴한다. 이전 과정에서 서명이 무효화되면 해당 오류 사항을 리턴한다.

아래는 3-2.exe파일을 통한 서명 검증과 검증 실패에 대한 결과를 나타낸다. 실제 프로토콜 과정에서는 개인키가 유출되면 안 되지만, 생성 확인을 위하여 개인키도 같이 출력을 하였다.

```
==============================
==== ECDSA 서명 생성 및 검증 ====
==============================
메시지를 입력하세요.
Enter : 2021cryptocontest


==============================
키를 생성합니다.

개인키 :
d :  0xc6d561d758c8a6dd446088184b03d94732df029bf1e1add951308570d13e1472


공개키 :
Qx :  0x6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278
Qy :  0xea12a100483ebbe30c137ad986709df04709a19f02f4a26e563115b4f004bca3


==============================
서명을 생성합니다.

서명 = (r1, s1), (r2, s2)
r1 :
0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s1 :
0x4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15


r2 :
0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s2 :
0xbbee30762a539ff6631c11fa5b149c3a85b56bdc9d7e7f79d977c4c901ee0a3c


====== (r1, s1) 서명 검증 =======
서명 검증을 합니다.

signature is valid
서명 검증에 성공하였습니다.


====== (r2, s2) 서명 검증 =======
서명 검증을 합니다.

signature is valid
서명 검증에 성공하였습니다.
```

(서명 검증에 성공)

(기존 공개키로 다른 서명을 검증하였을 때 검증 실패)

(기존 서명을 다른 공개키로 검증하였을 때 검증 실패)

3. 3-3 풀이

1번의 stack 프로세서를 확장하고 2번의 ECDSA 보안 프로토콜을 사용하여 주어진 동작을 수행하는 stack 프로세서를 구현하자. 확장된 stack 프로세서는 인증을 요청한 사람의 공개키 값을 복사 후 해당 값에 대한 hash 값과 이후 추가로 같은 공개키의 hash값의 일치 여부를 확인하여 메시지 트랜잭션(서명, 공개키)에 대한 소유권을 지정하며(Locking), 동일한 hash값일 때 서명 검증을 통하여 소유권 확인(Unlocking)을 진행한다.

확장된 stack 프로세서에서 수행할 명령어 :
<서명값>, <공개키 값>, 복사명령어, hash명령어, <공개키 값에 대한 hash값>, 동일한지 검증하는 명령어, 서명값을 검증하는 명령어

1번의 SimpleStack.py 모듈을 확장하여 ExtendedStack.py 모듈을 생성하였다. stack 프로세서에서 명령어를 수행하기 전 다음과 같은 단계를 거친다. 먼저 값(서명, 공개키, hash 등)과 명령어를 입력이 요구하는 대로 input을 시킨다. 이후 해당 input을 명령어 단위로 분리하여 옳은 명령일 시 queue에 해당 Opcode와 값을 넣고 옳지 않은 명령일 시 전체 기회 cnt 값을 1씩 감소하고 queue는 초기화 후 새로운 입력을 받도록 처리한다. 그리고 모두 옳은 명령일 때, queue의 front 자료를 pop하여 stack에 차례로 넣고 해당 명령어를 수행함으로써 stack 프로세서를 동작시킨다.

이러한 방법으로 명령어를 처리한 이유는 비트코인의 Script Language를 참고하였을 때, 값을 stack에 넣기 위해서 해당 값의 byte크기를 Opcode로 사용한다. (ex. 0x14 0xAABB.... : 0x14 이후 20byte에 해당하는 값(0xAABB....)을 스택에 push) 이 과정에서 byte크기에 해당하는 Opcode가 새로 삽입되고, 새로 삽입되는 Opcode를 처리하기 위하여 큐를 활용하였다. 즉, 사용자의 입력을 queue가 대체하는 것이다. (ex. 명령 : <서명값=(0xAA, 0xBB)> <공개키값=(0xCCDD, 0xEEFF)> 복사명령어 일 때, queue : 0x02 <0xAA, 0xBB> 0x04 <0xCCDD, 0xEEFF> 0x76 이 된다. (복사명령어 Opcode=0x76))

확장된 스택 프로세서에서 수행할 명령어에는 복사명령어(OP_DUP), hash명령어(OP_HASH), 동일한지 검증하는 명령어(OP_EQUALVERIFY), 서명값을 검증하는 명령어(OP_CHECKSIG) 가 존재하며 해당 명령의 Opcode는 비트코인 Script Language를 참고하여 다음과 같이 설정하였다.

OP_DUP : 0x76, OP_HASH : 0xAA, OP_EQUALVERIFY : 0x88, OP_CHECKSIG : 0xAC

또한 정상적으로 메시지 트랜잭션에 대한 소유권 지정 (hash값 일치 확인, Locking) 및 소유권 확인 (서명 검증, Unlocking)에 대한 내용이 작동하는 지 확인하기 위하여 총 5가지 모드로 동작하도록 설계하였다.

해당 모드는 다음과 같다.

mode 1 : 올바르게 생성된 서명과 공개키로 서명 검증 확인 (인증 성공)
mode 2 : 임의의 공개키에 대한 해시값 생성 (인증 실패)
mode 3 : 임의의 공개키를 생성하여 서명 검증 (인증 실패)
mode 4 : 임의의 서명 생성하여 서명 검증 (인증 실패)
mode 5 : 직접 입력 (올바른 입력 시 인증 성공, 입력 형식이 맞지 않을 시 cnt 값 1씩 감소)

이와 같이 설계한 이유는 다음과 같은 경우가 발생하기 때문이다. A가 만든 서명을 B가 검증하고 C는 A라고 주장하는 공격자라고 하자.

mode 1 : 인증 성공
A의 서명을 B가 검증하기 위해 A는 자신의 서명과 공개키에 대한 트랜잭션을 B에게 보낸다. B는 A의 공개키를 복사하고(OP_DUP) A의 공개키에 대한 hash값을 생성(OP_HASH)하여 해당 트랜잭션에 대한 소유권을 지정한다.(Locking) hash값 일치 시(OP_EQUALVERIFY) A의 공개키로 A의 서명을 검증(OP_CHECKSIG)을 통하여 소유권을 확인한다.(Unlocking) 최종적으로 A는 인증에 성공한다.

mode 2 : 인증 실패 (1)
A의 서명을 B가 검증하기 위해 A는 자신의 서명과 공개키에 대한 트랜잭션을 B에게 보낸다. B가 A의 공개키를 복사 후(OP_DUP) A의 공개키에 대한 hash값 생성하고(OP_HASH) 해당 hash값을 stack에 넣기 전 공격자 C가 자신의 공개키에 대한 hash값을 stack에 넣는다.(Locking) A의 공개키에 대한 hash값과 C의 공개키에 대한 hash값이 일치하지 않으므로 Locking 상태가 유지된다. 최종적으로 A는 인증에 실패한다.

mode 3 : 인증 실패 (2)
A의 서명을 B가 검증하기 위해 A는 자신의 서명과 공개키에 대한 트랜잭션을 B에게 보낸다. 공격자 C가 해당 트랜잭션을 탈취하여 A의 서명과 C의 공개키에 대한 트랜잭션을 B에게 보낸다. B는 C의 공개키를 보고 C의 공개키를 복사 후(OP_DUP) C의 공개키에 대한 hash값을 생성하여(OP_HASH) 소유권을 지정한다.(Locking) hash값이 일치하므로(OP_EQUALVERIFY) A의 서명을 C의 공개키로 검증한다.(OP_CHECKSIG) A의 서명은 C의 공개키로 인증이 불가능하기에 Unlocking에 실패한다. 최종적으로 A는 인증에 실패한다.

mode 4 : 인증 실패 (3)
A의 서명을 B가 검증하기 위해 A는 자신의 서명과 공개키에 대한 트랜잭션을 B에게 보낸다. 공격자 C가 해당 트랜잭션을 탈취하여 C의 서명과 A의 공개키에 대한 트랜잭션을 B에게 보낸다. B는 A의 공개키를 보고 A의 공개키를 복사 후(OP_DUP) A의 공개키에 대한 hash값을 생성하여(OP_HASH) 소유권을 지정한다.(Locking) hash값이 일치하므로(OP_EQUALVERIFY) C의 서명을 A의 공개키로 검증한다.(OP_CHECKSIG) C의 서명은 A의 공개키로 인증이 불가능하기에 Unlocking에 실패한다. 최종적으로 A는 인증에 실패한다.

stack 프로세서를 동작시키기 위한 ExtendedStack.py 모듈의 함수의 기능은 다음과 같다.

- init() : stack 프로세서에 필요한 stack과 전체 입력 기회 횟수 cnt가 리턴된다. cnt=5를 기본값으로 설정하여 잘못된 입력에 대한 무한 루프를 방지하였다.
- print_cnt(cnt) : 남은 기회를 출력한다.
- print_stack(stack) : stack의 내용을 출력한다. stack이 비어있을 경우 “stack is empty”를 출력한다.
- make_script(cnt, sig, pubkey, msg, mode) : 명령을 입력(script)받고 비트코인 Script의 Locking Script에 해당하는 ScriptPubKey와 Unlocking Script에 해당하는 ScriptSig를 리턴한다.
- scripts_to_queue(scripts, cnt) : 입력(script)을 큐로 대체하고 올바른 입력일 때만 stack 프로세서에서 어떠한 명령을 수행할지에 대한 Opcode가 넣어진 올바른 큐를 리턴한다.
- oper(stack, msg, scripts) : scripts(큐)의 수행할 명령어의 Opcode를 확인하여 해당 Opcode에 맞는 명령을 수행한다. 수행은 다음과 같이 동작한다.

Opcode ∈ [0x01, 0x4B] : 이후 Opcode 만큼의 바이트 값을 stack에 push한다.
Opcode = 0x76 : stack의 top(공개키)을 복사하고 해당 값을 push한다.(OP_DUP) stack이 empty인 상황이면 IndexError에 의해 프로그램이 종료된다.
Opcode = 0xAA : stack의 pop 값(stack의 기존 top값, 복사된 공개키 값)의 hash값을 구하고 stack에 push한다.(OP_HASH) stack이 empty인 상황이면 IndexError에 의해 프로그램이 종료된다.
Opcode = 0x88 : 기존의 stack의 top(새로운 hash값)과 공개키 hash값을 pop하여 일치하는지 확인한다.(OP_EQUALVERIFY) 일치하면 True를 리턴, 일치하지 않으면 False를 리턴한다. stack의 pop이 2번 일어나지 않는 상황이면 IndexError에 의해 프로그램이 종료된다.
Opcode = 0xAC : stack의 top(공개키)과 서명을 pop하여 검증을 한다.(OP_CHECKSIG) 검증에 성공하면 True를 리턴, 실패하면 False를 리턴한다.

ExtendedStack.py 모듈은 ECDSA.py 모듈과 LSH256 라이브러리를 호출하여 위 함수을 수행한다. 아래는 3-3.exe를 이용하여 위 5가지 mode에 대한 테스트이다. 실제 프로토콜 환경에서는 개인키가 유출되면 안 되지만 확인을 위하여 개인키와 공개키 서명 hash값 모두를 출력하였다. 또한 mode 별로 동일한 테스트를 위하여 Seed 값을 고정시켜 개인키는 같은 값을 갖도록 하였고, 메시지 또한 동일한 메시지 “2021cryptocontest“를 사용하였다.

mode 1

```
==========================================
메시지를 입력하세요.
2021cryptocontest


키를 생성합니다.

서명을 생성합니다.

생성된 메시지, 키, 서명은 다음과 같습니다.

메시지 :  2021cryptocontest


개인키 :  0xc6d561d758c8a6dd446088184b03d94732df029bf1e1add951308570d13e1472


공개키 : Qx, Qy
Qx :  0x6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278
Qy :  0xea12a100483ebbe30c137ad986709df04709a19f02f4a26e563115b4f004bca3


서명 : r, s
r :  0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s :  0x4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15


공개키의 해시 값 : LSH(Qx), LSH(Qy)
LSH(Qx) :  0xa26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad
LSH(Qy) :  0xc067eaf91863a2c32c64ece1021439c6000976b2d5f05d9bc744b1684ebfbb53
```

```
==========================================
확장된 스택 프로세서 입니다.
메시지 소유권을 지정하고 확인합니다.

모드를 입력하세요.
생성된 서명과 공개키로 서명 검증 확인 (검증 성공) : 1
임의의 공개키에 대한 해시값 생성 (검증 실패) : 2
임의의 공개키를 생성 (검증 실패) : 3
임의의 서명 생성 (검증 실패) : 4
직접 입력 (올바른 입력 시 검증 성공) : 5
mode : 1
남은 기회 :  5
==============================================
명령어는 다음과 같이 입력됩니다.
<서명(hex)> <공개키(hex)> 복사명령어 해쉬명령어 <공개키 해쉬값(hex)> 동일검증명령어 서명검증명령어
<r,s> <Qx,Qy> OP_DUP OP_HASH <HashX,HashY> OP_EQUALVERIFY OP_CHECKSIG
ex) <abcd,0123> <01234,abcde> OP_DUP OP_HASH <0000,ffff> OP_EQUALVERIFY OP_CHECKSIG
==============================================
==========================================
scriptPubKey : OP_DUP OP_HASH <a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad,c067eaf91863a2c32c64e
ce1021439c6000976b2d5f05d9bc744b1684ebfbb53> OP_EQUALVERIFY OP_CHECKSIG

scriptSig : <e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0,4411cf88d5ac600a9ce3ee05a4eb63c537318ed
109991f0b1a4205f9fa751b15> <6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278,ea12a100483ebbe30c137ad9
86709df04709a19f02f4a26e563115b4f004bca3>

==========================================
```

```
push 입니다.
========== stack ==========
top ----> ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

push 입니다.
========== stack ==========
top ----> ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

duplicate 입니다.
========== stack ==========
top ----> ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

hash 입니다.
========== stack ==========
top ----> ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

push 입니다.
========== stack ==========
top ----> ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================
```

```
equal and verify 입니다.
hash 값이 일치합니다.
========== stack ==========
top ----> ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

check signature 입니다.
signature is valid
서명 검증에 성공하였습니다.
========== stack ==========
top ----> True
============================



========================================
최종 결과
서명 검증에 성공하였습니다.
UnLocking Success
========== stack ==========
top ----> True
============================

엔터를 누르면 종료합니다.
```

mode 2

```
==========================================
메시지를 입력하세요.
2021cryptocontest


키를 생성합니다.

서명을 생성합니다.

생성된 메시지, 키, 서명은 다음과 같습니다.

메시지 :  2021cryptocontest


개인키 :  0xc6d561d758c8a6dd446088184b03d94732df029bf1e1add951308570d13e1472


공개키 : Qx, Qy
Qx :  0x6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278
Qy :  0xea12a100483ebbe30c137ad986709df04709a19f02f4a26e563115b4f004bca3


서명 : r, s
r :  0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s :  0x4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15


공개키의 해시 값 : LSH(Qx), LSH(Qy)
LSH(Qx) :  0xa26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad
LSH(Qy) :  0xc067eaf91863a2c32c64ece1021439c6000976b2d5f05d9bc744b1684ebfbb53
```

```
==========================================
확장된 스택 프로세서 입니다.
메시지 소유권을 지정하고 확인합니다.

모드를 입력하세요.
생성된 서명과 공개키로 서명 검증 확인 (검증 성공) : 1
임의의 공개키에 대한 해시값 생성 (검증 실패) : 2
임의의 공개키를 생성 (검증 실패) : 3
임의의 서명 생성 (검증 실패) : 4
직접 입력 (올바른 입력 시 검증 성공) : 5
mode : 2
남은 기회 :  5
==============================================
명령어는 다음과 같이 입력됩니다.
<서명(hex)> <공개키(hex)> 복사명령어 해쉬명령어 <공개키 해쉬값(hex)> 동일검증명령어 서명검증명령어
<r,s> <Qx,Qy> OP_DUP OP_HASH <HashX,HashY> OP_EQUALVERIFY OP_CHECKSIG
ex) <abcd,0123> <01234,abcde> OP_DUP OP_HASH <0000,ffff> OP_EQUALVERIFY OP_CHECKSIG
==============================================
==========================================
scriptPubKey : OP_DUP OP_HASH <5922a62f62a6400755d9eceec8b47d87513150f6422e0ac020e526f2aadfd236,7cd97d8d01054b53e34f5
57dcf920a09c5a83239bf33778f911ab9b20176892a> OP_EQUALVERIFY OP_CHECKSIG

scriptSig : <e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0,4411cf88d5ac600a9ce3ee05a4eb63c537318ed
109991f0b1a4205f9fa751b15> <6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278,ea12a100483ebbe30c137ad9
86709df04709a19f02f4a26e563115b4f004bca3>

==========================================
```

```
push 입니다.
========== stack ===========
top ---->  ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

push 입니다.
========== stack ===========
top ---->  ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

duplicate 입니다.
========== stack ===========
top ---->  ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================
```

```
hash 입니다.
========== stack ===========
top ----> ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

push 입니다.
========== stack ===========
top ----> ('5922a62f62a6400755d9eceec8b47d87513150f6422e0ac020e526f2aadfd236', '7cd97d8d01054b53e34f557dcf920a09c5a8
3239bf33778f911ab9b20176892a')
============================
          ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================
```

```
equal and verify 입니다.
hash 값이 일치하지 않습니다.
========== stack ===========
top ----> ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================



========================================
최종 결과
서명 검증에 실패하였습니다.
UnLocking Fail
========== stack ===========
top ----> ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

엔터를 누르면 종료합니다.
```

mode 3

```
========================================
메시지를 입력하세요.
2021cryptocontest


키를 생성합니다.

서명을 생성합니다.

생성된 메시지, 키, 서명은 다음과 같습니다.

메시지 :  2021cryptocontest


개인키 :  0xc6d561d758c8a6dd446088184b03d94732df029bf1e1add951308570d13e1472


공개키 : Qx, Qy
Qx :  0x6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278
Qy :  0xea12a100483ebbe30c137ad986709df04709a19f02f4a26e563115b4f004bca3


서명 : r, s
r :  0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s :  0x4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15


공개키의 해시 값 : LSH(Qx), LSH(Qy)
LSH(Qx) :  0xa26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad
LSH(Qy) :  0xc067eaf91863a2c32c64ece1021439c6000976b2d5f05d9bc744b1684ebfbb53
```

```
========================================
확장된 스택 프로세서 입니다.
메시지 소유권을 지정하고 확인합니다.

모드를 입력하세요.
생성된 서명과 공개키로 서명 검증 확인 (검증 성공) : 1
임의의 공개키에 대한 해시값 생성 (검증 실패) : 2
임의의 공개키를 생성 (검증 실패) : 3
임의의 서명 생성 (검증 실패) : 4
직접 입력 (올바른 입력 시 검증 성공) : 5
mode : 3
남은 기회 :  5
============================================
명령어는 다음과 같이 입력됩니다.
<서명(hex)> <공개키(hex)> 복사명령어 해쉬명령어 <공개키 해쉬값(hex)> 동일검증명령어 서명검증명령어
<r,s> <Qx,Qy> OP_DUP OP_HASH <HashX,HashY> OP_EQUALVERIFY OP_CHECKSIG
ex) <abcd,0123> <01234,abcde> OP_DUP OP_HASH <0000,ffff> OP_EQUALVERIFY OP_CHECKSIG
============================================
========================================
scriptPubKey : OP_DUP OP_HASH <5922a62f62a6400755d9eceec8b47d87513150f6422e0ac020e526f2aadfd236,7cd97d8d01054b53e34f557dcf920a09c5a83239bf33778f911ab9b20176892a> OP_EQUALVERIFY OP_CHECKSIG

scriptSig : <e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0,4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15> <2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1,a394436f0348ffaf95a96d1d0a54c061aae397e4db90668effbe340e822e6995>

========================================
```

```
push 입니다.
========== stack ===========
top ----->  ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15')
============================

push 입니다.
========== stack ===========
top ----->  ('2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1', 'a394436f0348ffaf95a96d1d0a54c061aae397e4db90668effbe340e822e6995')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15')
============================

duplicate 입니다.
========== stack ===========
top ----->  ('2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1', 'a394436f0348ffaf95a96d1d0a54c061aae397e4db90668effbe340e822e6995')
============================
          ('2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1', 'a394436f0348ffaf95a96d1d0a54c061aae397e4db90668effbe340e822e6995')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15')
============================
```

```
hash 입니다.
========== stack ===========
top ----> ('5922a62f62a6400755d9eceec8b47d87513150f6422e0ac020e526f2aadfd236', '7cd97d8d01054b53e34f557dcf920a09c5a8
3239bf33778f911ab9b20176892a')
============================
          ('2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1', 'a394436f0348ffaf95a96d1d0a54c061aae3
97e4db90668effbe340e822e6995')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

push 입니다.
========== stack ===========
top ----> ('5922a62f62a6400755d9eceec8b47d87513150f6422e0ac020e526f2aadfd236', '7cd97d8d01054b53e34f557dcf920a09c5a8
3239bf33778f911ab9b20176892a')
============================
          ('5922a62f62a6400755d9eceec8b47d87513150f6422e0ac020e526f2aadfd236', '7cd97d8d01054b53e34f557dcf920a09c5a8
3239bf33778f911ab9b20176892a')
============================
          ('2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1', 'a394436f0348ffaf95a96d1d0a54c061aae3
97e4db90668effbe340e822e6995')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================
```

```
equal and verify 입니다.
hash 값이 일치합니다.
========== stack ===========
top ----> ('2f9a015fed27ac0301b605645490df6b8c50ed98558c7dfbdd0d50a7ffc1bad1', 'a394436f0348ffaf95a96d1d0a54c061aae3
97e4db90668effbe340e822e6995')
============================
          ('e26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0', '4411cf88d5ac600a9ce3ee05a4eb63c53731
8ed109991f0b1a4205f9fa751b15')
============================

check signature 입니다.
signature is invalid
서명 검증에 실패하였습니다.
========== stack ===========
top ----> False
============================



========================================
최종 결과
서명 검증에 실패하였습니다.
UnLocking Fail
========== stack ===========
top ----> False
============================

엔터를 누르면 종료합니다.
```

mode 4

```
========================================
메시지를 입력하세요.
2021cryptocontest


키를 생성합니다.

서명을 생성합니다.

생성된 메시지, 키, 서명은 다음과 같습니다.

메시지 :  2021cryptocontest


개인키 :  0xc6d561d758c8a6dd446088184b03d94732df029bf1e1add951308570d13e1472


공개키 : Qx, Qy
Qx :  0x6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278
Qy :  0xea12a100483ebbe30c137ad986709df04709a19f02f4a26e563115b4f004bca3


서명 : r, s
r :  0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s :  0x4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15


공개키의 해시 값 : LSH(Qx), LSH(Qy)
LSH(Qx) :  0xa26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad
LSH(Qy) :  0xc067eaf91863a2c32c64ece1021439c6000976b2d5f05d9bc744b1684ebfbb53
```

```
========================================
확장된 스택 프로세서 입니다.
메시지 소유권을 지정하고 확인합니다.

모드를 입력하세요.
생성된 서명과 공개키로 서명 검증 확인 (검증 성공) : 1
임의의 공개키에 대한 해시값 생성 (검증 실패) : 2
임의의 공개키를 생성 (검증 실패) : 3
임의의 서명 생성 (검증 실패) : 4
직접 입력 (올바른 입력 시 검증 성공) : 5
mode : 4
남은 기회 :  5
================================================
명령어는 다음과 같이 입력됩니다.
<서명(hex)> <공개키(hex)> 복사명령어 해쉬명령어 <공개키 해쉬값(hex)> 동일검증명령어 서명검증명령어
<r,s> <Qx,Qy> OP_DUP OP_HASH <HashX,HashY> OP_EQUALVERIFY OP_CHECKSIG
ex) <abcd,0123> <01234,abcde> OP_DUP OP_HASH <0000,ffff> OP_EQUALVERIFY OP_CHECKSIG
================================================
========================================
scriptPubKey : OP_DUP OP_HASH <a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad,c067eaf91863a2c32c64e
ce1021439c6000976b2d5f05d9bc744b1684ebfbb53> OP_EQUALVERIFY OP_CHECKSIG

scriptSig : <52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb,d36402cb083fba3d177a9c1c92f2de979fd6136
f375a06cacbf8ec8cf8d02ce3> <6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278,ea12a100483ebbe30c137ad9
86709df04709a19f02f4a26e563115b4f004bca3>

========================================
```

```
push 입니다.
========== stack ===========
top ---->  ('52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb', 'd36402cb083fba3d177a9c1c92f2de979fd6
136f375a06cacbf8ec8cf8d02ce3')
============================

push 입니다.
========== stack ===========
top ---->  ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb', 'd36402cb083fba3d177a9c1c92f2de979fd6
136f375a06cacbf8ec8cf8d02ce3')
============================

duplicate 입니다.
========== stack ===========
top ---->  ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb', 'd36402cb083fba3d177a9c1c92f2de979fd6
136f375a06cacbf8ec8cf8d02ce3')
============================
```

```
hash 입니다.
========== stack ===========
top ----> ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb', 'd36402cb083fba3d177a9c1c92f2de979fd6
136f375a06cacbf8ec8cf8d02ce3')
============================

push 입니다.
========== stack ===========
top ----> ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('a26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad', 'c067eaf91863a2c32c64ece1021439c60009
76b2d5f05d9bc744b1684ebfbb53')
============================
          ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb', 'd36402cb083fba3d177a9c1c92f2de979fd6
136f375a06cacbf8ec8cf8d02ce3')
============================
```

```
equal and verify 입니다.
hash 값이 일치합니다.
========== stack ===========
top ----> ('6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278', 'ea12a100483ebbe30c137ad986709df04709
a19f02f4a26e563115b4f004bca3')
============================
          ('52d273c67370456e98e774c6b0d3107cd125b3f8f42adb0ca1978f6229ff1cdb', 'd36402cb083fba3d177a9c1c92f2de979fd6
136f375a06cacbf8ec8cf8d02ce3')
============================

check signature 입니다.
signature is invalid
서명 검증에 실패하였습니다.
========== stack ===========
top ----> False
============================



========================================
최종 결과
서명 검증에 실패하였습니다.
UnLocking Fail
========== stack ===========
top ----> False
============================

엔터를 누르면 종료합니다.
```

mode 5 (검증에 실패하는 값 입력)

```
========================================
메시지를 입력하세요.
2021cryptocontest


키를 생성합니다.

서명을 생성합니다.

생성된 메시지, 키, 서명은 다음과 같습니다.

메시지 :  2021cryptocontest


개인키 :  0xc6d561d758c8a6dd446088184b03d94732df029bf1e1add951308570d13e1472


공개키 : Qx, Qy
Qx :  0x6c2ff30ebc64a00c2c37c4a435a03f8a1c87ae802106129ee543be7870399278
Qy :  0xea12a100483ebbe30c137ad986709df04709a19f02f4a26e563115b4f004bca3


서명 : r, s
r :  0xe26cf2cd46c14bfad2b61e5a4280094ed38f42d08750b178f54d9b67a3e291f0
s :  0x4411cf88d5ac600a9ce3ee05a4eb63c537318ed109991f0b1a4205f9fa751b15


공개키의 해시 값 : LSH(Qx), LSH(Qy)
LSH(Qx) :  0xa26de96d2234afe065f0880dbb3c614c869ada39384ee947390a163d98ef77ad
LSH(Qy) :  0xc067eaf91863a2c32c64ece1021439c6000976b2d5f05d9bc744b1684ebfbb53
```

```
========================================
확장된 스택 프로세서 입니다.
메시지 소유권을 지정하고 확인합니다.

모드를 입력하세요.
생성된 서명과 공개키로 서명 검증 확인 (검증 성공) : 1
임의의 공개키에 대한 해시값 생성 (검증 실패) : 2
임의의 공개키를 생성 (검증 실패) : 3
임의의 서명 생성 (검증 실패) : 4
직접 입력 (올바른 입력 시 검증 성공) : 5
mode : 5
남은 기회 :  5
================================================
명령어는 다음과 같이 입력됩니다.
<서명(hex)> <공개키(hex)> 복사명령어 해쉬명령어 <공개키 해쉬값(hex)> 동일검증명령어 서명검증명령어
<r,s> <Qx,Qy> OP_DUP OP_HASH <HashX,HashY> OP_EQUALVERIFY OP_CHECKSIG
ex) <abcd,0123> <01234,abcde> OP_DUP OP_HASH <0000,ffff> OP_EQUALVERIFY OP_CHECKSIG
================================================
명령어를 입력하세요 : <0123,abcd> <ffff,0000> OP_DUP OP_HASH <1111,2222> OP_EQUALVERIFY OP_CHECKSIG
========================================
scriptPubKey : OP_DUP OP_HASH <1111,2222> OP_EQUALVERIFY OP_CHECKSIG

scriptSig : <0123,abcd> <ffff,0000>

========================================
```

```
push 입니다.
========== stack ===========
top ---->  ('123', 'abcd')
============================

push 입니다.
========== stack ===========
top ---->  ('ffff', '0')
============================
           ('123', 'abcd')
============================

duplicate 입니다.
========== stack ===========
top ---->  ('ffff', '0')
============================
           ('ffff', '0')
============================
           ('123', 'abcd')
============================
```

```
hash 입니다.
========== stack ===========
top ----> ('1a8527007487fe4bd4ff5d7e0d120f0abd262f14c680c2f6e1ff59b43a252c60', '7f88ec8b06a2877313005dbad9e378733a76
8f32b4494c557d81e232da93822e')
============================
          ('ffff', '0')
============================
          ('123', 'abcd')
============================

push 입니다.
========== stack ===========
top ----> ('1111', '2222')
============================
          ('1a8527007487fe4bd4ff5d7e0d120f0abd262f14c680c2f6e1ff59b43a252c60', '7f88ec8b06a2877313005dbad9e378733a76
8f32b4494c557d81e232da93822e')
============================
          ('ffff', '0')
============================
          ('123', 'abcd')
============================
```

```
equal and verify 입니다.
hash 값이 일치하지 않습니다.
========== stack ===========
top ----> ('ffff', '0')
============================
          ('123', 'abcd')
============================



========================================
최종 결과
서명 검증에 실패하였습니다.
UnLocking Fail
========== stack ===========
top ----> ('ffff', '0')
============================
          ('123', 'abcd')
============================

엔터를 누르면 종료합니다.
```

위 결과로 mode 1을 제외하고 모두 서명 검증에 실패함을 알 수 있고, 정상 동작함을 확인할 수 있다.

참고자료

1. 비트코인 script language : https://en.bitcoin.it/wiki/Script
2. ECDSA protocol : https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm
3. ECC P256r1 curve : SEC 2: Recommended Elliptic Curve Domain Parameters - Certicom Research
4. LSH 알고리즘 : https://seed.kisa.or.kr/kisa/algorithm/EgovLSHInfo.do
5. LSH 소스코드 : https://seed.kisa.or.kr/kisa/Board/22/detailView.do