Programming Assignment 3.1(bitcoin) Description

Group number :

two

Group member :

方舟, 张帅, 任中杰, 邱磊, 乔鹏宇, 王天蔚

1. Overview of this project

First of all, this is a very comprehensive project, which relates to several cryptography knowledge such as digital signature system, public key and private key system, stream cipher system. therefore we need to understand all of them then we can solve this project. Let's take a look at this project.

Firstly, we have 1000 (c1, signatute) and 1000 (pk$_i$||sk$_i$, signatutre) Paris, but only one of them are signed sb SK$_C$ which is kept by teacher, so we need to use the PK$_C$ is given to us already to verify those 1000 paris to find the c$_i$ and pk$_i$||sk$_i$, In order to verify, we need to hash function SHA256 the message (such as C$_i$, pk$_i$||sk) and get the resualt then use the signature to verify. once we get the unique resualt of those 1000 pairs, then we can do the next step.

Secondly, we get the C$_i$and Sk$_i$ though the preview step, then we obey the RSA-OAEP encryption algorithm , which input the C$_i$and Sk$_i$then we can get the othutput K

Thirdly , we already received the Cipher text CM , and we get the K which is treated as the stream cipher key last step. Therefore we can decrypt the CM used the K which obeyed the RC4 algorithm

Finally, we get the massage then we can do the Bitcoin Minning. which is let the 24 MSB of SHA256(M||r) be 0 . Which can be soved by a "while" statement

2. Code

prestatement

- include lib :

```python
import hashlib
from ecdsa import VerifyingKey, ellipticcurve, util, curves, keys
from functools import partial
from Crypto import Random
from Crypto.PublicKey import RSA
from Crypto.Math.Numbers import Integer
from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1_v1_5
from Crypto.Cipher import PKCS1_OAEP as Cipher_OAEP
```

- include language :
    - python
    - go (1 file used to salve RC4)

part one

In this part, we do the work of reading the 2000 pairs from the file, then we verify them use the PK $_i$, then we get the two unque pars , this the code below:

- read_and_verify.py

```python
"""
提取1000条记录，并且解出通过验证的记录
"""

import hashlib
from ecdsa import VerifyingKey, ellipticcurve, util, curves, keys
# 构造特殊的curve 类
x = 0xA374F7774070CD8BECCDD5B01450FFBC0033EE5FFBFEC7829C10DDD1
y = 0xF484F9AF3B9DD12C93F5DA7971333C3A0DCDBD20865CCE59145D9180
_r = 26959946667150639794667015087019625940457807714424391721682722368061
# order
# 构造公钥的pk(在给定的点的情况下)
public_point = ellipticcurve.Point(curves.NIST224p.curve, x, y, _r)
# 构造vk用于验证
vk = VerifyingKey.from_public_point(public_point, curve=curves.NIST224p,
hashfunc=hashlib.sha256)
the_order = vk.pubkey.order


def read_rc4():
    with open("record/RC4KeyAndSignature.txt") as file_obj:
        the_lines = file_obj.readlines()
    return the_lines
```

```
def get_ci():
    """度读取ci文件"""
    records = read_rc4()
    for record in records:
        spilt_record = record.split(',')
        c = spilt_record[0]
        r_int = int(spilt_record[1], 16)
        s_int = int(spilt_record[2], 16)
        sig = util.sigencode_string(r_int, s_int, the_order)
        try:
            vk.verify(sig, bytearray.fromhex(c))
        except keys.BadSignatureError:
            continue
        print("the break Ci is:" + c)
        break


def read_rsa():
    """读取rsa文件"""
    with open("record/RSAKeyAndSignature.txt") as file_obj:
        the_lines = file_obj.readlines()
    return the_lines


def get_sk_and_pk():
    """读取sk和pk文件"""
    records = read_rsa()
    for record in records:
        spilt_record = record.split(',')
        n = spilt_record[0]+','
        e = spilt_record[1]+','
        d = spilt_record[2]+','
        # if len(d) % 2:
        #     d = "0" + spilt_record[2]
        p = spilt_record[3]+','
        q = spilt_record[4]
        r_int = int(spilt_record[5], 16)
        s_int = int(spilt_record[6], 16)
        # 构造签名
        mes = n+e+d+p+q
        sig = util.sigencode_string(r_int, s_int, the_order)
        try:
            vk.verify(sig, mes.encode("ascii"))
        except keys.BadSignatureError:
            continue
        print("n:" + n)
        print("e:" + e)
        print("d:" + d)
        print("p:" + p)
        print("q:" + q)
        break


if __name__ == "__main__":
    # verify the sig
    get_ci()
    get_sk_and_pk()
```

Due to the "Good encapsulation" of library ecdsa, we **cannot generate the EC-PK by call the library function directly** , which means that we need to rebuild the SK$_C$ by ourself , therefore we need to read the source code and get the original object to rebuild ,**which is the obstacle one in this project**:

```
# Construct the object of ellipticcurve.Point uesd P(x, y)
x = 0xA374F7774070CD8BECCDD5B01450FFBC0033EE5FFBFEC7829C10DDD1
y = 0xF484F9AF3B9DD12C93F5DA7971333C3A0DCDBD20865CCE59145D9180
_r = 26959946667150639794667015087019625940457807714424391721682722368061
 # order of 224p
public_point = ellipticcurve.Point(curves.NIST224p.curve, x, y, _r)
# Construct the object of VerifyingKey by the object of ellipticcurve.Point
vk = VerifyingKey.from_public_point(public_point, curve=curves.NIST224p,
hashfunc=hashlib.sha256)
the_order = vk.pubkey.order
```

we can get the ponit (x, y) by the material already know. then we find the order of curve NIST224p in the source code, which can be used to build the object of ellipticcurve.Point. then we use this Point object to build the Vk object which can be used to verify the signature.

those steps can only be done after we understand the source code which is quite diffcult

After we construst the object of vk , we can do the verify:

The detail step is to read from the file by line , and break the line by "," , then we can get the the signature and the verify text. Here comes to our **obstacle two in this project** :

the signature in our project is consist of two parts which are r and s. However due to the "Good encapsulation" of this library, the accepttd input of verify is a processed signature which is combine the r with s and form one complete signature. **Therefore we need to read the source code and undestand how the library combine r and s, and recombine the r ,s in our project to pass** :

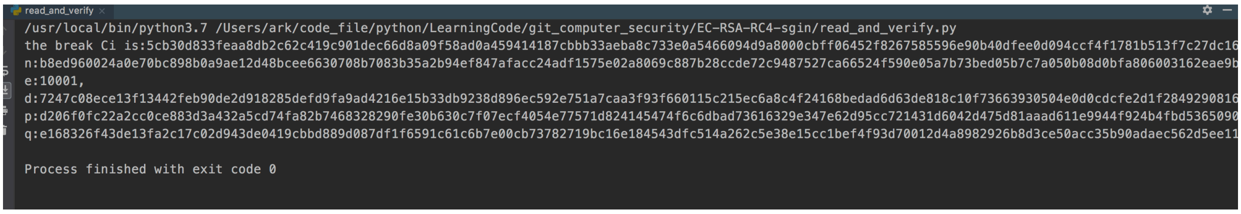
```
sig = util.sigencode_string(r_int, s_int, the_order) # combine work
    try:
    vk.verify(sig, bytearray.fromhex(c))
    except keys.BadSignatureError:
        continue
```

Finally, we can get the resualt :

```
Ci =
5cb30d833feaa8db2c62c419c901dec66d8a09f58ad0a459414187cbbb33aeba8c733e0a546
6094d9a8000cbff06452f8267585596e90b40dfee0d094ccf4f1781b513f7c27dc165784eb7
9badb36d86da272276f07fd0c7778901b828b178b646b1eac3c5913bb0c72bcc8aa67783719
5fb1c642cb7bc2ff87d19b4e7cb853fb6e8fbc5d547a359b867bda72b83e2c1546ce4e2c565
0cf6277a33db3c92d202da589b215c34f6f3e7db63152fd20c90c84c5699620ce254cb88f8f
b2839532e3e343ce283b5a1d6af748a728eb7e3adf2ad53bbe8cf755308b5bda1fc5c4a96b4
6159b0fe4b8f6d04224c8ed80a65ffbe295e98808326e4774b05e86c769055
```

```
n =
b8ed960024a0e70bc898b0a9ae12d48bcee6630708b7083b35a2b94ef847afacc24adf1575e
02a8069c887b28ccde72c9487527ca66524f590e05a7b73bed05b7c7a050b08d0bfa8060031
62eae9b5ce55504a10a893a9c577b865925a58b1f30d1b83a5d066496806c83676a75b908b4
aad77eb8e7eb18a8087b3a7d63cbd00ded3e6341bd517def4e5e93d95856ad5b15b50713744
058fa8f999eeb3965daba9b172f8fd51c06d668e38b769097f5a3c31eb4f1391a4c59c36a71
7806c4ba3d11b2d444fa1fa0b2376b4d890b19ceda0bff48b6312b99b070ef94743f3643c00
5f932907222c839d60b26132182ff35380a7bd773194a6e389d1c078dc3ac5
e = 10001
d =
7247c08ece13f13442feb90de2d918285defd9fa9ad4216e15b33db9238d896ec592e751a7c
aa3f93f660115c215ec6a8c4f24168bedad6d63de818c10f73663930504e0d0cdcfe2d1f284
929081652ce16400a152cc0dfa4e05d21f235df47a32fb71dba271d3a349ada89aa2c9df057
55cc5e2f7e82b67600e7c6397ed690e778e8920e78de804ac42c38f1fdeabb4f0ba1c005123
3f4db3beeca20da2eec793634b7be6374c93cffbec899faa2ca188daeef4954b10fc12fbbb4
a21ef91f776fea7e8f9158dc95ed840a4b59402d2ce3e7b765e5d2085c474e23b8a03c4891e
1d9df1493d9d35aece0900c0fab7d495691992352cf9faef29fa29ffddf301
p =
d206f0fc22a2cc0ce883d3a432a5cd74fa82b7468328290fe30b630c7f07ecf4054e77571d8
24145474f6c6dbad73616329e347e62d95cc721431d6042d475d81aaad611e9944f924b4fbd
53650900a463c39af2ae3f9f065f6e5c66838c5113289bc951c6b5a120a826d13714f57c25e
5ab4a913ab205276474593f10175631
q =
e168326f43de13fa2c17c02d943de0419cbbd889d087df1f6591c61c6b7e00cb73782719bc1
6e184543dfc514a262c5e38e15cc1bef4f93d70012d4a8982926b8d3ce50acc35b90adaec56
2d5ee11e19f35350e4b940a414f07a9e8b43d53f058798ee88e6e00d25f2992d0c9ec50ba4b
051d16851a40cf522aa4f172db4c4d5
```

The part of result :



```
/usr/local/bin/python3.7 /Users/ark/code_file/python/LearningCode/git_computer_security/EC-RSA-RC4-sgin/read_and_verify.py
the break Ci is:5cb30d833feaa8db2c62c419c901dec66d8a09f58ad0a459414187cbbb33aeba8c733e0a5466094d9a8000cbff06452f8267585596e90b40dfee0d094ccf4f1781b513f7c27dc16
n:b8ed960024a0e70bc898b0a9ae12d48bcee6630708b7083b35a2b94ef847afacc24adf1575e02a8069c887b28ccde72c9487527ca66524f590e05a7b73bed05b7c7a050b08d0bfa806003162eae9b
e:10001,
d:7247c08ece13f13442feb90de2d918285defd9fa9ad4216e15b33db9238d896ec592e751a7caa3f93f660115c215ec6a8c4f24168bedad6d63de818c10f73663930504e0d0cdcfe2d1f2849290816
p:d206f0fc22a2cc0ce883d3a432a5cd74fa82b7468328290fe30b630c7f07ecf4054e77571d824145474f6c6dbad73616329e347e62d95cc721431d6042d475d81aaad611e9944f924b4fbd5365090
q:e168326f43de13fa2c17c02d943de0419cbbd889d087df1f6591c61c6b7e00cb73782719bc16e184543dfc514a262c5e38e15cc1bef4f93d70012d4a8982926b8d3ce50acc35b90adaec562d5ee11

Process finished with exit code 0
```

part 2

in this part we get the C$_i$ and Sk$_i$ though the preview step, then we obey the RSA-OAEP encryption algorithm , which input the C$_i$ and Sk$_i$ then we can get the othutput K :

- get_the_message.py

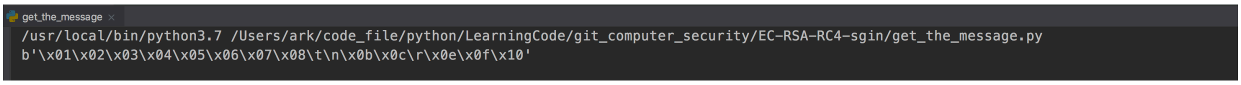
```
Cipher_key =
"5cb30d833feaa8db2c62c419c901dec66d8a09f58ad0a459414187cbbb33aeba8c733e0a54
66094d9a8000cbff06452f8267585596e90b40dfee0d094ccf4f1781b513f7c27dc165784eb
79badb36d86da272276f07fd0c7778901b828b178b646b1eac3c5913bb0c72bcc8aa6778371
95fb1c642cb7bc2ff87d19b4e7cb853fb6e8fbc5d547a359b867bda72b83e2c1546ce4e2c56
50cf6277a33db3c92d202da589b215c34f6f3e7db63152fd20c90c84c5699620ce254cb88f8
fb2839532e3e343ce283b5a1d6af748a728eb7e3adf2ad53bbe8cf755308b5bda1fc5c4a96b
46159b0fe4b8f6d04224c8ed80a65ffbe295e98808326e4774b05e86c769055"
n =
0xb8ed960024a0e70bc898b0a9ae12d48bcee6630708b7083b35a2b94ef847afacc24adf157
5e02a8069c887b28ccde72c9487527ca66524f590e05a7b73bed05b7c7a050b08d0bfa80600
3162eae9b5ce55504a10a893a9c577b865925a58b1f30d1b83a5d066496806c83676a75b908
b4aad77eb8e7eb18a8087b3a7d63cbd00ded3e6341bd517def4e5e93d95856ad5b15b507137
44058fa8f999eeb3965daba9b172f8fd51c06d668e38b769097f5a3c31eb4f1391a4c59c36a
717806c4ba3d11b2d444fa1fa0b2376b4d890b19ceda0bff48b6312b99b070ef94743f3643c
005f932907222c839d60b26132182ff35380a7bd773194a6e389d1c078dc3ac5
n_int = int(str(n), 10)
e = 0x10001
e_int = int(str(e), 10)
d =
0x7247c08ece13f13442feb90de2d918285defd9fa9ad4216e15b33db9238d896ec592e751a
7caa3f93f660115c215ec6a8c4f24168bedad6d63de818c10f73663930504e0d0cdcfe2d1f2
84929081652ce16400a152cc0dfa4e05d21f235df47a32fb71dba271d3a349ada89aa2c9df0
5755cc5e2f7e82b67600e7c6397ed690e778e8920e78de804ac42c38f1fdeabb4f0ba1c0051
233f4db3beeca20da2eec793634b7be6374c93cffbec899faa2ca188daeef4954b10fc12fbb
b4a21ef91f776fea7e8f9158dc95ed840a4b59402d2ce3e7b765e5d2085c474e23b8a03c489
1e1d9df1493d9d35aece0900c0fab7d495691992352cf9faef29fa29ffddf301
d_int = int(str(d), 10)
p =
0xd206f0fc22a2cc0ce883d3a432a5cd74fa82b7468328290fe30b630c7f07ecf4054e77571
d824145474f6c6dbad73616329e347e62d95cc721431d6042d475d81aaad611e9944f924b4f
bd53650900a463c39af2ae3f9f065f6e5c66838c5113289bc951c6b5a120a826d13714f57c2
5e5ab4a913ab205276474593f10175631
p_int = int(str(p), 10)
```

```
q =
0xe168326f43de13fa2c17c02d943de0419cbbd889d087df1f6591c61c6b7e00cb73782719b
c16e184543dfc514a262c5e38e15cc1bef4f93d70012d4a8982926b8d3ce50acc35b90adaec
562d5ee11e19f35350e4b940a414f07a9e8b43d53f058798ee88e6e00d25f2992d0c9ec50ba
4b051d16851a40cf522aa4f172db4c4d5
q_int = int(str(q), 10)
# rsa.newkeys(1024)
# the_sk = rsa.PrivateKey(n, e, d, p, q)
the_sk = rsa.PrivateKey(n_int, e_int, d_int, p_int, q_int)
the_pk = rsa.PublicKey(n_int, e_int)

Cipher_key = Cipher_key.encode("ascii")
# print(Cipher_key)
# Cipher_key = bytes.fromhex(Cipher_key)
message = rsa.decrypt(Cipher_key, the_sk)
# message = rsa.decrypt(Cipher_key.encode("ascii"), the_sk)
# message = rsa.decrypt(Cipher_key, the_sk)

print(message)
```

in this part we use the library of RSA , and construct the_sk by the function of rsa.PrivateKey () . and pass the n, e, d , p, q which we get in the last step. then we get the the key of stream cipher RC4

```
 RC4_key =
b'\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10'
```

Here is the result:



```
/usr/local/bin/python3.7 /Users/ark/code_file/python/LearningCode/git_computer_security/EC-RSA-RC4-sgin/get_the_message.py
b'\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10'
```

part 3

we already received the Cipher text CM , and we get the K which is treated as the stream cipher key last step. Therefore we can decrypt the CM used the K which obeyed the RC4 algorithm

here comes ourt obstacle three in this project : the RC4 algorithm in supported badly in python, however is well support in Go, In the other side, the process of read a bin file in python is easiler then go , After weighing the pros and cons then we decide to read the bin file in pyhton, and get the binary data used in the go to execute the RC4 algorithm

- read bin file RC4.py

  ```
  from functools import partial


  class Converter:
  ```

```
    @staticmethod
    def to_ascii(h):
        list_s = []
        for i in range(0,len(h),2):
            list_s.append(chr(int(h[i:i+2].upper(),16)))
        return ''.join(list_s)
    @staticmethod
    def to_hex(s):
        list_h = []
        for c in s:
            list_h.append(str(hex(ord(c)))[-2:]) #取hex转换16进制的后两位
        return ''.join(list_h)


if __name__ == "__main__":
    i = 0
    f = open('record/RC4Ciphertext.bin', 'rb')
    f2 = open('f.txt', 'w')
    # print(f.read())
    records = iter(partial(f.read, 2), b'')  # 每次2字节
    for r in records:
        j = 0
        # r_int = int.from_bytes(r, byteorder='big')  # 将 byte转化为
int
        i += 1
        print('i={0}:{1}'.format(i, r))
        f2.write(str(r) + ' ')
        if 8 == i:
            f2.write('\n')
            i = 0
            break
    f.close()
    f2.close()
```

then we can get the binary message :

```
    # 二进制文件
the_msg =
b'\xd9\xa8\xa2\xfd\x12\xfcj\x82\xde\xf2\\\xf0\xa2\x8ah\xb6r\x11\xab\xe
0\xb0|s\x1c\x0b\xaa\xf7\x84~y\xfbox\xfb\x96\x889X\x80\xcc4\xf6\x16\x1e
\x92\x14\xb8\x9d\xd04\xccA\xc6X`\xeb\xae\x13\xf6\xe5\xba\x13\x1d\x14i\
xe8\xcd'
```

However the pass parametric of go should be the type of decimal num, so we transform the btyes to decimal num

```
    # 二进制文件
the_msg =
b'\xd9\xa8\xa2\xfd\x12\xfcj\x82\xde\xf2\\\xf0\xa2\x8ah\xb6r\x11\xab\xe
0\xb0|s\x1c\x0b\xaa\xf7\x84~y\xfbox\xfb\x96\x889X\x80\xcc4\xf6\x16\x1e
\x92\x14\xb8\x9d\xd04\xccA\xc6X`\xeb\xae\x13\xf6\xe5\xba\x13\x1d\x14i\
xe8\xcd'
    the_hex_from = the_msg.hex()
    #二进制文件转为hexto10
    the_length = len(the_hex_from)
    i = 0
    while i+2 < the_length:
        the_hex = the_hex_from[i:i+2]
        print(int(the_hex, 16))
        i = i+2
```

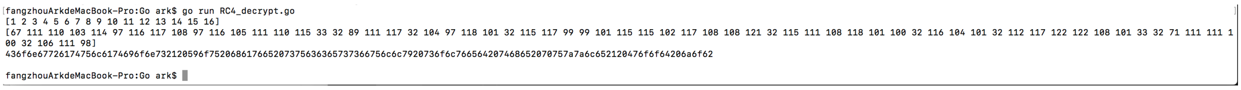
After we get those decimal numbres we can use the go language word for us

- RC4_decrypt.go

```
package main
import "fmt"
import "crypto/rc4"

func main() {
     var key []byte = []byte{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14 ,15 ,16} //初始化用于加密的KEY
     fmt.Println(key)
     rc4obj1, _ := rc4.NewCipher(key) //返回 Cipher
     rc4str1 := []byte{217,
168,
162,
253,
18,
252,
106,
130,
222,
242,
92,
240,
162,
138,
104,
182,
114,
17,
171,
224,
176,
124,
115,
28,
11,
170,
247,
132,
126,
121,
251,
111,
120,
251,
150,
136,
57,
88,
128,
204,
52,
246,
22,
30,
146,
20,
184,
157,
208,
52,
204,
65,
198,
88,
96,
235,
174,
19,
246,
229,
186,
19,
29,
20,
105,
232,
}
     plaintext := make([]byte, len(rc4str1)) //
     rc4obj1.XORKeyStream(plaintext, rc4str1)
```

```
        fmt.Println(plaintext)
        stringinf1 := fmt.Sprintf("%x\n", plaintext) //转换字符串
        fmt.Println(stringinf1)

    }
```

we can run this go file in the bash , here is the result:



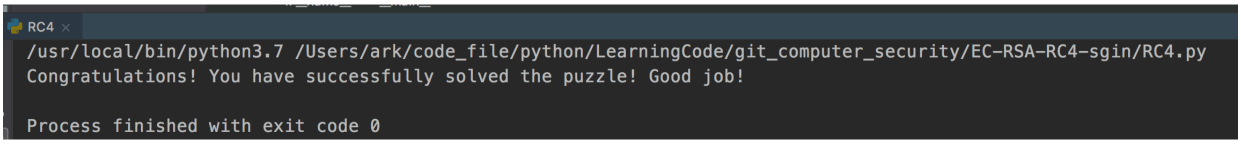
```
fangzhouArkdeMacBook-Pro:Go ark$ go run RC4_decrypt.go
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]
[67 111 110 103 114 97 116 117 108 97 116 105 111 110 115 33 32 89 111 117 32 104 97 118 101 32 115 117 99 99 101 115 115 102 117 108 108 121 32 115 111 108 118 101 100 32 116 104 101 32 112 117 122 122 108 101 33 32 71 111 111 1
00 32 106 111 98]
436f6e67726174756c6174696f6e732120596f752068617665207375636365737366756c6c7920736f6c76656420746865207075 7a7a6c652120476f6f64206a6f62

fangzhouArkdeMacBook-Pro:Go ark$ 
```

Obviously, these numbers are the hexadecimal form of the final result.

then we use python to transformn them into ascii :

```
    #  Go 语言解密出来的16进制数
    mes =
 "436f6e67726174756c6174696f6e732120596f752068617665207375636365737366756c6c
 7920736f6c76656420746865207075 7a7a6c652120476f6f64206a6f62"

    # ANS !!!!!!!!!!!
    print(Converter.to_ascii(mes))
```

Finally , we get the final message !



```
/usr/local/bin/python3.7 /Users/ark/code_file/python/LearningCode/git_computer_security/EC-RSA-RC4-sgin/RC4.py
Congratulations! You have successfully solved the puzzle! Good job!

Process finished with exit code 0
```

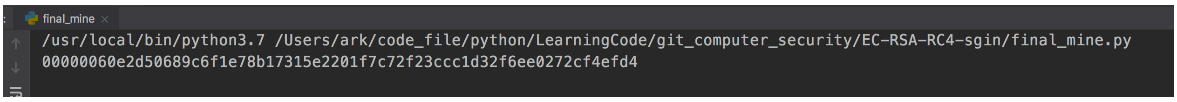
part 4

we get the massage then we can do the Bitcoin Minning. which is let the 24 MSB of SHA256(M||r) be 0 . Which can be soved by a "while" statement s

- Final_mine.py

```
"""
最后的挖矿程序
"""
import hashlib

mes = "Congratulations! You have successfully solved the puzzle! Good
job!"

# 寻找答案
i = 0x01
while 1:
    str_hex = hex(i)[2:]
    len_str = len(str_hex)
    if len_str % 2 is 1:
        str_hex = "0" + str_hex
    bytes_mes = mes+str_hex
    if hashlib.sha256(bytes_mes.encode('ascii')).hexdigest()[:6] ==
"000000":
        print(str_hex)
        break
    i += 1

# hex
ans = "0214295e"
print
# dec
#ans = "34875742"
```

this is the final resual after hashing:



```
/usr/local/bin/python3.7 /Users/ark/code_file/python/LearningCode/git_computer_security/EC-RSA-RC4-sgin/final_mine.py
00000060e2d50689c6f1e78b17315e2201f7c72f23ccc1d32f6ee0272cf4efd4
```

So far, we have completed the assignment.

3. conclusion

In this project, I learn how to implement the digital signature system, public key and private key system, stream cipher system, and figure out the mutual relationship betweent those algorithm such as RC4 , RSA , ECESA and so on Furthermore we use the konwleage which we learn in the class, and apply it into the program, which can increase our skill in programing and also give us second thinking about the knowledge, and give large improvement about our ability!