Problem Set 1

Name : *Ziqi Liu*(*zl*3*kh*)

Problem 1

(a)

0393ceffaa8b8fa50bc665099c2ac05635ab6dfa550b2eb9ed41583c8d8965a1

(b)

0.0001 BTC (0.02 USD)

(c)

The total output is 3,945.62668066 BTC

(d)

I think about an hour. Because when I made another transactions, it takes about 50 minutes to get 3 confirmation. Since the fee of the new transaction is smaller than this one, I estimate it should be about an hour.

Problem 2

(a)

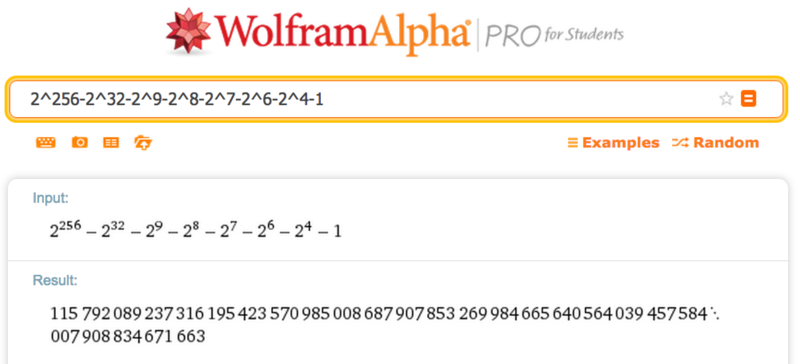
1BHD4CRjp1yLPhZDQY31A2vPRXUBKFt4JF
1PQHK5ivZGyRf83TDZsvhnmecxKgCK4JXa
1DuA2rvcJmgBLs9TVE14chN94mwsnW8xEe

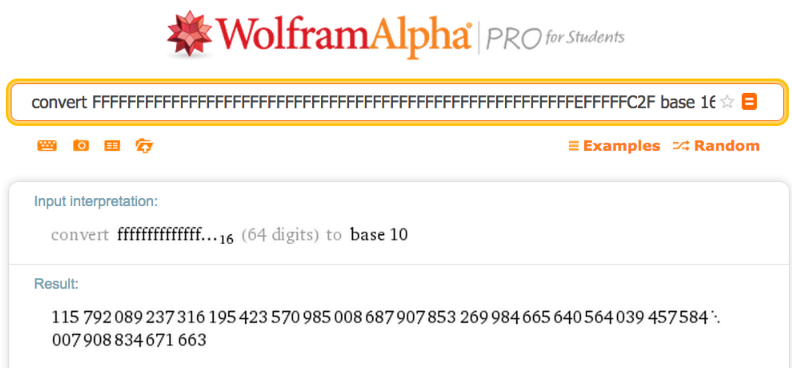
(b)

I keep tracking back the sender of the transactions...
And I think I got the purchase:
where the selller(sell bitcoins) is 1GymPp2dave9ByFRcFAwkPX3bMeVyoZkPq
and the buyer is 19WmbY4nDcjAEv6wb5rcd5E6MutVMXBZzy

(c)

I think I can guess the it's on the east coast. On the transaction info webpage, I find a "Network Propagation" section, which shows Florida.





Problem 3

(a) Place a backdoor which sent a private key to the developer;
Or secretly replace the receiver address to the developer's address. (b)I'm confident because through some research, I found that many people use it, without finding any evil thing happens.

Problem 4

I used wolfram/Mathmaticas to calculate them (half-)manually.
Compare the two results, we can see they are exactly the same.

Problem 5

1. I need to believe that the private key is purely random. i.e. It will not generate the key that the developer assigns.
2.The program won't send the key to someone else
3. The elliptic curve is configured correctly.

4.The multiplication function is working correctly (NewPrivateKey()), the calculation is right.
5. The btcec library is not malicious and working appropriately.

Problem 6

```
func generateVanityAddress(pattern string) (*btcec.PublicKey, *btcec.PrivateKey) {
    pub, priv := generateKeyPair()
addr := generateAddr(pub)
matched,err := regexp.MatchString(pattern,addr.String())
if err != nil {
// There was an error. Log it and bail out
log.Fatal(err)
}
for !matched {
    pub, priv = generateKeyPair();
    addr = generateAddr(pub);
    matched,err = regexp.MatchString(pattern,addr.String())
        if err != nil {
            log.Fatal(err)
        }
fmt.Printf("This is the associated Bitcoin address:\t[%s]\n", addr.String())
}

return priv.PubKey(),priv


}
```

Problem 7

My pattern is " $\wedge 1 * Z + i + q + i +$ "
And this is my result:

```
This is a private key in hex: [48eb31d5cc40acb96f86ba1969558508e54f1867db5643cb2342025e8b775
This is a public key in hex: [035c181f4fae08bfabf480cd4c9f109078fd590c700e970b99524cada5fd41
This is the associated Bitcoin address: [1ZiqicRL5HPXhatcMcqwh8vH96Uf7YZ9k]
```

Problem 8

They are on the same secure level.
Because the way I wrote the function, is keep generating the new key pair and

its address until it founds an address that's qualified.
So they are just the same as the previous one.
Unless, limiting the address to the pattern may cause the private key to be less random.

Problem 9

The transaction ID is
20993362fb866b8f1dbae8a044a236585f8906e1864356cfe21ff579dbd4725a

Problem 10

The api always return error, even though I have only use transaction id once.

```
 go run spend.go -privkey "48eb31d5cc40acb96f86ba1969558508e54f1867db5643cb2342025e8b775c4a'
Here is your raw bitcoin transaction:
01000000015a72d4db79f51fe2cf564386e106895f5836a244a0e8ba1d8f6b86fb62339920000000006b4830450
Sending transaction to: https://insight.bitpay.com/api/tx/send
The sending api responded with:
Transaction rejected by network (code -26). Reason: 64: non-canonical
```

Problem 11