Problem Set #1

Sam Prestwood - swp2sf

Problem 1.

a. What is the transaction ID?

I received two transactions:

1. 6d8600ed8a057af24ac0d97d8d3aace21ba1fd5e578b430198116f0b4f903ef9
2. 1875f11186bc1db7d7976ed6d810222cdbbee9d6feaad3210abb7eacd74b8a79

b. What was the transaction fee for the transaction? (Give your answer in BTC, as well as current approximate US dollar value.)

For both transactions, the fee was 0.0001 BTC ($0.02).

c. What was the total value of all the transactions in the block containing your transfer? (Note: https://blockchain.info provides this info conveniently, although you could compute it yourself)

The first transaction was in block #371919, which contained 3,450.09327553 BTC. The second transaction was in block #371933, which contained 10,164.98226155 BTC.

d. How long did it take from when the transaction was received until it had 3 confirmations? (Include an explanation of how you estimated this in your answer.)

The first transaction was received at 2015-08-28 15:39:41

- The first block it was included in was confirmed at 2015-08-28 15:40:30
- The second block it was included in was confirmed at 2015-08-28 15:57:55
- The third block it was included in was confirmed at 2015-08-28 16:29:41

(2015-08-28 16:29:41) - (2015-08-28 15:39:41) = 40 minutes

The second transaction was received at 2015-08-28 17:59:23

- The first block it was included in was confirmed at 2015-08-28 18:06:08
- The second block it was included in was confirmed at 2015-08-28 18:06:53
- The third block it was included in was confirmed at 2015-08-28 18:11:04

(2015-08-28 18:11:04) - (2015-08-28 17:59:23) = 11 minutes, 41 seconds

Problem 2.

a. Identify the bitcoin addresses of what are likely to be other students in the class (you could potentially find all of them, but it is enough to find 3).

12YwFZNmUoexnQAMKFvRJSSydxTZUZPWx8
1QLT7GNBnKNrGnKi7HNnZTYSDbPvUaVoZs
1NpYM8Pa5xt26A4sLqcChMoEke9jATw4A5
1GMrGqvF8FwbgCSShCdzijcSUHzmNR1VCz

b. Trace back the source of the bitcoin as far as you can. Bonus points if you can figure out from which exchange the bitcoin was purchased and when.

Address 1JKnDhzvAab7YkPL4wg1Jz24qr1YvbCbWS appears to be the source of the bitcoin. The transactions before this are of significantly greater volume, which implies to me that they are from a bitcoin exchange.

c. (Bonus) Can you learn anything about where the send of the bitcoin is located geographically? (In this case, you have external information to know I'm in Charlottesville, but what could you learn about the sender's probable location just from the information in the blockchain?)

The blockchain appears to log the IP address of whoever made each transaction. IP addresses can be correlated with geographical location, although this could be obfuscated with the use of a VPN or Tor.

Problem 3.

Suppose a malicious developer wanted to distribute a bitcoin wallet implementation that would steal as much bitcoin as possible from its users with a little chance as possible of getting caught.

(a) Explain things a malicious developer might do to create an evil wallet.

Behind the scenes, the developer could make a second set of keys whenever the user makes a new wallet receiving address. Then, the malicious wallet would display the developer's key instead of the user's key, whenever the user tried to display their receiving address. This would allow the developer to route any incoming bitcoin first to a hidden wallet address she controls,

possibly removing tiny amounts of bitcoin from each transaction, and then ferrying the transaction onto the user's wallet.

To make sure that the user doesn't double-check the block chain, the developer could add a block chain explorer in the wallet implementation that sneakily doesn't display any transactions to or from the user's wallet. In other words, the explorer intentionally prevents the user from seeing the hidden bitcoin addresses that it creates.

(b) How confident are you your money is safe in the wallet you are using, and what would you do to increase your confidence if you were going to store all of your income in it?

I'm relatively confident that my money is safe because the wallet software seems to be trustworthy.

First, I wouldn't store all of my income in Bitcoin. ...but if I were to do that, I would store it in a way that would keep the wallet's private key never available to the internet (i.e. in a hardware or paper wallet).

Problem 4.

Verify that the modulus used as secp256k1.P in btcec.go is correct.

The following code verifies the modulus by manually calculating 2^256 - 2^32 - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1.

```
package main

import(
    "fmt"
    "math/big"
)

func main() {
    a := new(big.Int)
    b := new(big.Int)

    a.Exp(big.NewInt(2), big.NewInt(256), nil)

    b.Exp(big.NewInt(2), big.NewInt(32), nil)
    a.Sub(a, b)

    b.Exp(big.NewInt(2), big.NewInt(9), nil)
    a.Sub(a, b)

    b.Exp(big.NewInt(2), big.NewInt(8), nil)
    a.Sub(a, b)

    b.Exp(big.NewInt(2), big.NewInt(7), nil)
    a.Sub(a, b)

    b.Exp(big.NewInt(2), big.NewInt(6), nil)
    a.Sub(a, b)

    b.Exp(big.NewInt(2), big.NewInt(4), nil)
    a.Sub(a, b)

    b.Exp(big.NewInt(2), big.NewInt(0), nil)
    a.Sub(a, b)

    fmt.Printf("%016X\n", a)
}
```

Problem 5.

What are all the things you need to trust if you are going to send money to the key generated by running keypair.go?

- keypair.go must be using the correct values for G
- keypair.go must be using the correct secp256k1 modulus
- k must be generated using a cryptographically-secure random number generator

- all of the mathematical operations used to generate the key pair from k and G must work correctly
- keypair.go must not leak any data about my key over the network
- the key pair should be generated in a secure physical environment that is free from any snooping by third parties
- I need to trust that every component (software, hardware, or otherwise) has not been compromised

Problem 6.

Define a function, func generateVanityAddress(pattern string) (*btcec.PublicKey, *btcec.PrivateKey) where pattern is a regular expression.

Below is the code for this function. The general idea is to keep generating random keys until one of them matches the string pattern

```
func generateVanityAddress(pattern string) (*btcec.PublicKey, *btcec.PrivateKey) {
    publicKey, privateKey := generateKeyPair()
    isMatch, err := regexp.MatchString(pattern, generateAddr(publicKey).String())
    if err != nil {
        log.Fatal(err)
    }
    for !isMatch {
        publicKey, privateKey = generateKeyPair()
        isMatch, err = regexp.MatchString(pattern, generateAddr(publicKey).String())
        if err != nil {
            log.Fatal(err)
        }
    }
    return publicKey, privateKey
}
```

Problem 7.

Use your generateVanityAddress function to create your own vanity address.

address: 1MuFGDztkvU7xEU**SamPR**3zFPDhPjvQ5ZSj

Problem 8.

Is your vanity address more or less secure than the first address you generated?

Assuming that the randomness my computer generates stays constant throughout the address generation process, the vanity address will be just as secure as the first address I generated. This is because I'm simply repeating the original address generation process several times until I find an address that matches the vanity pattern.

Problem 9.

Make a small (e.g., 1 mBTC) transfer from your wallet address to your vanity address.

Transaction id: 528457cb1a48e06571f6adc97e0acfa07e19bb2209b104028f3b21fb30375d15

Problem 10.

Transfer some bitcoin from your vanity address to someone else in the class (you can use one of the addresses you identified in Problem 2).

Transaction id: 68d089faa9a485b725c9347ae0589fe08bf36126aa82ccc56239601807ecbda5

Problem 11.

The provided spend.go code sends the full amount of the input transaction (less the network fee) to the destination address. Modify the program to add an -amount <value> flag that takes the amount to transfer in satoshi.

~~I underestimated how much time Problem 11 would take to complete and started working on this problem set too late. The code for this problem spend_swp2sf.go is incomplete.~~

UPDATE: 2015-09-15 08:44 PM (15 minutes late)

I was able to complete the alterations to spend.go. Below is a link to the transaction that I used to test the code:

id: e6b696a65bf4a406045d156030645fa70a43501762537daafb7fe80fe32f34e3