A.J. Varshneya

CS 4501

9.15.15

Problem Set 1

Problem 1

A) 98ec9d91e02292955b5f20e4ddd69767278ed02083b1a36c874407176c2fb799

B) 0.0001 BTC, 0.02 USD

C) 1755.15 BTC, ~401455 USD

This was the “Estimated Transaction Volume” on blockchain.info.

D)

Received time: 2015-08-28 15:12:52

Confirmation 1: 2015-08-28 15:17:16

Confirmation 2: 2015-08-28 15:17:59

Confirmation 3: 2015-08-28 15:33:05

The transaction was confirmed by blocks 371914, 371915, and 371916. The time from block 371916 minus the received time is the amount of time it took to confirm the transaction 3 times. So it took 20 minutes and 13 seconds to confirm three times.

Problem 2

A)

1DkHaFKyrXhE28KsybBntCxoYeSoBumPfU
15X2qKbsXxPHbgfFTc6UgXmETVe7fpBUzX
1KEz1rFhEoy2vQV4zQcUCFWPYy7EkJyRcM

B) I traced back the coin to the address 19WmbY4nDcjAEv6wb5rcd5E6MutVMXBZzy. At this address it appears another address with ‘dave’ in it (Prof. Evans) added some coin and possibly made a purchase from an exchange. The reason I think he may have purchased the coin here is the previous transactions from which the 0.2 BTC in this wallet came from are fairly large splits of coin with one large input and many small outputs. If you go back further, you find a set of addresses with many inputs (exchange buying coin) and then many outputs (exchange selling coin).

C) The blockchain does not store any information about IP addresses, however the ‘taint analysis’ tool on blockchain.info has a list of IPs which may give some information about where it was relayed from. This could give you an idea of the country/region a person is from.

Problem 3

A) They could quietly log your master private key and send it to themselves. When you have accumlated a certain number of bitcoin they could recover your wallet using that key and send all of your coin to themselves in various wallets.

B) I am reasonably confident for the following reasons:
- The wallet I use was recommended by the Bitcoin Project website.
- The website I downloaded it from uses TLS and has a certificate issued by DigiCert.
- Multibit HD is open source—if there were malicious code someone could find it and tell everyone.
- The wallet I use seems to be widely used.
- The wallet's files are encrypted with AES.

Some steps I could take if I wanted to be more careful:
- I could use an offline wallet (paper wallet, USB drive, hardware wallet, etc.)
- I could make several encrypted backups in case my devices with access to my bitcoin wallet are destroyed.
- I could use a better password to encrypt my wallet so that even if someone did get their hands on my wallet they'd have a harder time cracking it.

Problem 4

I wrote a Go program to verify the result.

```
package main

import (
	"math/big"
	"fmt"
)

func main() {
	mod, _ := new(big.Int).SetString(
				"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F", 16)
	result := new(big.Int).Exp(big.NewInt(2), big.NewInt(256), nil) // initialize as 2^256
	result.Sub(result, new(big.Int).Exp(big.NewInt(2), big.NewInt(32), nil)) // subtract 2^32
	result.Sub(result, new(big.Int).Exp(big.NewInt(2), big.NewInt(9), nil)) // subtract 2^9
	result.Sub(result, new(big.Int).Exp(big.NewInt(2), big.NewInt(8), nil)) // subtract 2^8
	result.Sub(result, new(big.Int).Exp(big.NewInt(2), big.NewInt(7), nil)) // subtract 2^7
	result.Sub(result, new(big.Int).Exp(big.NewInt(2), big.NewInt(6), nil)) // subtract 2^6
	result.Sub(result, new(big.Int).Exp(big.NewInt(2), big.NewInt(4), nil)) // subtract 2^4
	result.Sub(result, big.NewInt(1)) // subtract 1

	if mod.Cmp(result) == 0 {
		fmt.Println("mod == result")
	} else {
		fmt.Println("mod != result")
	}
	fmt.Println("mod: \t\t", mod.String())
	fmt.Println("result: \t", result.String())
}
```

The output:

```
mod  == result
mod:        115792089237316195423570985008687907853269984665640564039457584007908834671663
result:     115792089237316195423570985008687907853269984665640564039457584007908834671663
```

Problem 5

You have to trust btcd not to be collecting keys for obvious reasons. You have to trust that the private key btcec generates is as random as possible. You have to trust that btcec is a correct/untainted implementation of ECC so that it produces a legitimate public key from which you cannot easily determine the private key. In general, you have to trust that the keys it generates are not predictable (except determining the public key via the private key) because if they are then an evil person might have a way to determine your keys and steal your coin. It also might be desirable to ensure that btcd is not logging any kind of personal information to protect your anonymity.

Problem 6

```
func generateVanityAddress(pattern string) (*btcec.PublicKey, *btcec.PrivateKey) {
        for {
                pub, priv := keypair.GenerateKeyPair()
                address := keypair.GenerateAddr(pub).String()
                re, _ := regexp.Compile(pattern)
                if re.MatchString(address) {return pub, priv}
        }
}
```

Problem 7

I used the regular expression: ^1(AJV|ajv)1

Generated address: 1AJV1wk2nC6ov8TXAbKGdKXG1ujKsgV2nc

As I added more characters to search for it seemed to take exponentially longer time. For each unique character I add to my search pattern, it should take ~58 times as long to compute an address that satisfies the constraints.

Problem 8

My vanity address is as secure as any other address because I computed it myself. This is because we simply compute many many addresses and pick the ones that satisfy the constraints for our vanity addresses. To determine the private key associated with my vanity address requires the same amount of computation as determining the private key associated with any other address.

If you use a service to compute a vanity address, it may not be as secure. An evil person that owns the service could easily log your private key when you generate such an address.

Problem 9

d9823c3aa2a87b9d7b96c6bc747b52be67318b8150476c38df4a258e05aed453

Problem 10

Console output:

```
Here is your raw bitcoin transaction:

010000000153d4ae058e254adf386c4750818b3167be527b74bcc6967b9d7ba8a23a3c82d9000000006b
48304502210092ef639abd7207a59bf5fc846a9abfdbd58eb0c62d7ad0d8d5c1e997eb4cc5180220453eb7
742c40b33605d91fa6a10cca53f5e94f438b7c9d3e354b4833b43bd9f8012103b5937a91f9fc1e8e1a0d98
02678cffb46906cf88397e8b6fe813007dbcc5b2ffffffff01905f0100000000001976a914e12f07a9b7b2c
a56d81623ad90636186ddcc6e6e88ac00000000

Sending transaction to: https://insight.bitpay.com/api/tx/send

The sending api responded with:

{"txid":"d1b7b6ff457ed6d6b3ba0814edb1bca699ab8c0a1226e6564d7a25d568dc8513"}
```

Problem 11

Code is attached in submission.

Transaction ids:

26582cc479623d56df4b2c99200deb9d4cf438f547205b8800f7a9c0da68c3e6

70c247aa8968161fcc5d0e630b55bb7f3612a701b6eb2035e761f475552dba7b