☐ cfh9en-ps1.md

Problem Set 1

Carter Hall | CS 4501 | 2015-09-15

1. Transactions

    i. 892e3f6bae9f354473b66d297835eebe8ef949f3a8fb6e4014b64c3268af484e
    ii. 0.0001 BTC
    iii. 14,487.20964744 BTC
    iv. About 3.5 hours; blockchain.info says the transaction happened 3.5 hours later than MultiBit says.

2. Sleuthing

    i. 1fiWBi5u7oDzQrMNjeNLbmAvvHv545oy9, 1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY, 1N96tMSyeeANTRFApr3zA6ML3Qow1gZR9A
    ii. I was able to trace the bitcoin back to the address 19WmbY4nDcjAEv6wb5rcd5E6MutVMXBZzy before the bitcoin appeared to come from multiple sources.

3. Security

    i. If I wanted to implement a bitcoin wallet that stole as much as possible without getting caught, I would basically implement a 'transaction fee' and not tell the user about it. For every outgoing transaction, I would have the wallet send money to the intended address, as well as a small amount to my own wallet somewhere else.

    ii. I have done almost nothing to verify that my money is safe, because I do not intend to put any significant amount of funds in my wallet. If I wanted to keep all of my income in a bitcoin wallet, I would find a dedicated laptop with no known security vulnerabilities to use only for that purpose, install Arch Linux, install only the software necessary to use my wallet, find an open-source wallet implementation for which I could personally verify the integrity of the source, set up 2-factor authentication for the laptop and the wallet, encrypt the hard drive, and put the hard drive and the laptop in separate safes while not in use.

4. Modulus verification

```
In[1]:= 2^256 - 2^32 - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1

Out[1]= 115 792 089 237 316 195 423 570 985 008 687 907 853 269 984 665 640 564 039 457 584 007 908 834 \
        671 663

In[2]:= 16^^FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F

Out[2]= 115 792 089 237 316 195 423 570 985 008 687 907 853 269 984 665 640 564 039 457 584 007 908 834 \
        671 663
```

5. Go compiler, all of keypair.go, all imported code, operating system, all running programs, all connected computer hardware, all connected network devices.

6. Vanity address generation code

```
func generateVanityAddress(pattern string) (*btcec.PublicKey, *btcec.PrivateKey) {
    for {
        pub, priv := generateKeyPair()
        addr := generateAddr(pub)
        matched, err := regexp.MatchString(pattern, addr.String())
        if matched && err == nil {
            return pub, priv
        }
    }
}
```

7. 13mexN9MUTxZk78Yq8RNN1quM8pFpREKTw ('carter' was taking too long, so I settled for 'REKT')
8. Assuming I trust my Go program and MultiBit equally, my vanity address is just as secure as the initial one I generated. The fact that it says 'REKT' doesn't make it any easier to find the corresponding key pair.
9. 3c6d5ebf28bdda7eaf850274278a924a590e74456e3d65c0879bb7e2349d2063
10. d48e8a01a6062e124b1cd5f3b25588b2fdc3edc867bda6493184a87db7ef588e
11. [modified spend.go attached]