Acacia Dai
Cryptocurrency Cabal
Problem Set 1

Problem 1. Answer the following questions about the transaction where you received the bitcoin. If you received more than one transfer, include all the transaction IDs).

a. What is the transaction ID?

a46e8b2f3f17e7450eedd9ff0fe83c320b66779750241280d04ee2b5eb2be5d8

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

0.0001 BTC = 0.02 USD

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)

Block #372359

Estimated Transaction Volume: 2,732.47337582 BTC = $ 630,081.04

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.)

21 minutes = 7 minutes between Received Time and Inclusion in Blocks * 3

Problem 2. See how much can you figure out about the way bitcoin was transferred to students in the class, starting from your transactions.

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).

1BHD4CRjp1yLPhZDQY31A2vPRXUBKFt4JF

1FqQV1R3H8dftDupCjCoW7b6kSVxoBZBmh

1LveaSSVmvRvnbcwsvBdC8AvoqoLro5DG4

Found from following the "change" forward and looking for more sends of similar amounts.

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.

Traced back to Address 1PAyvggUiXLcQdj5gkHqx1sC1vW96p9FLT

b. (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?)

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 developed might do to create an evil wallet. (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?

The evil wallet would transfer very small amounts at a time to the malicious developer, and would make transfers to and through multiple accounts owned by the malicious developer, so that the user would take a long time to notice and it would be hard to figure out the identity of the developer. The evil wallet would not display any major changes in the balance so the user would only notice losses by going to look at their account on a ledger website.

I am very confident that my wallet is secure only because Multibit is widely used and it is used for this class. To increase my confidence, I would want to use a separate system connected to a public ledger to notify me every time a transaction happens from my account, in case it is not reflected in Multibit.

Problem 4. Verify that the modulus used as secp256k1.P inbtcec.go is correct. You can do this either using math/big, Go's bit integer library to do computations on such large numbers, or by computing it by hand. (For your answer, just show how you verified the modulus. Including a snippet of code is fine.)

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F

Decimal=
$15\times16^{63}+15\times16^{62}+15\times16^{61}+15\times16^{60}+15\times16^{59}+15\times16^{58}+15\times16^{57}+15\times16^{56}+15\times16^{55}+15\times16^{54}+15\times16^{53}+15\times16^{52}+15\times16^{51}+15\times16^{50}+15\times16^{49}+15\times16^{48}+15\times16^{47}+15\times16^{46}+15\times16^{45}+15\times16^{44}+15\times16^{43}+15\times16^{42}+15\times16^{41}+15\times16^{40}+15\times16^{39}+15\times16^{38}+15\times16^{37}+15\times16^{36}+15\times16^{35}+15\times16^{34}+15\times16^{33}+15\times16^{32}+15\times16^{31}+15\times16^{30}+15\times16^{29}+15\times16^{28}+15\times16^{27}+15\times16^{26}+15\times16^{25}+15\times16^{24}+15\times16^{23}+15\times16^{22}+15\times16^{21}+15\times16^{20}+15\times16^{19}+15\times16^{18}+15\times16^{17}+15\times16^{16}+15\times16^{15}+15\times16^{14}+15\times16^{13}+15\times16^{12}+15\times16^{11}+15\times16^{10}+15\times16^{9}+14\times16^{8}+15\times16^{7}+15\times16^{6}+15\times16^{5}+15\times16^{4}+15\times16^{3}+12\times16^{2}+2\times16^{1}+15\times16^{0}$ = **1.157920892373162e+77**

2^(256) - 2^(32) - 2^(9) - 2^(8) - 2^(7) - 2^(6) - 2^(4) – 1 = **1.1579209e+77**

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? You should assume that you are an ultra-paranoid multi-billionaire who intends to transfer her entire fortune to the generated address.

The private key is not a "commonly used" number because some hackers compile the addresses generated by easy numbers to use for stealing.

The random number generator for the private key is perfectly random and will not form clusters that thieves can take advantage of.

The generator has a very large range of possible private keys it can generate.

Problem 7. Use your `generateVanityAddress` function to create your own vanity address.

Vanity pattern was "WAT"

This is a private key in hex: [19f138b72a212b3be880828eba682cc99163f67c63dd291686f5049689698f37]

This is a public key in hex: [038f59ce3db645f9dbc697181be3527e293ddd68c7f2bb62fa73f1ba84a9736d9e]

This is the associated Bitcoin address: [1LWab61QwjxtsBmuWAT24Vk5ZwAmqL755n]

Problem 8. Is your vanity address more or less secure than the first address you generated?

The vanity address is not any more or less secure than the “random” address because only the public key, and not the private key, becomes less random. There are a lot of possible keypairs that can generate an address that would satisfy the vanity pattern, so it is still very difficult for an attacker to try and cycle through each combination to check.

Problem 9. Make a **small** (e.g., 1 mBTC) transfer from your wallet address to your vanity address

TID: 4fc3e46a76191a8d4b6a13622ed008acb83a61e1977d252925035cad76d98fe3

Problem 10. Transfer some bitcoin from your vanity address to someone else in the class

```
Here is your raw bitcoin transaction:
010000000168eb41404abf1f05cad92be2804641a7936e23c6c189faedb6b0c449d2a467cd010000006a473044
02207492435c0691b0ff49ad9a684964b1713984cd4df39d681c13adb70bd00a5d2702205196520f739a49a3a9
38f5514495ffc127d5ddb19656ba8e8f57d4141cc701620121038f59ce3db645f9dbc697181be3527e293ddd68c
7f2bb62fa73f1ba84a9736d9effffffff01bc610000000000001976a914c446799026edde4fcfc2eae671afaab353ec
6e6788ac00000000
Sending transaction to: https://insight.bitpay.com/api/tx/send
The sending api responded with:
{"txid":"6261b8ed07444d50bbcc0c439845c146ce2b0f5328cb31542800099348363395"}
```

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.

```
Sending 12774 Satoshis (0.03 USD) + 10000 Satoshi fee (0.02 USD)

Here is your raw bitcoin transaction:
01000000016632caea98e45bbd6d113d394a82841d054a0afe362ce248c0ce9c57dabb7401010000006b483045
022100e803679439263fd7cfaa378bc2c668d380a6a5237dc7032a06600e10ab67957502204493174 7bc15f330
7995788b865935a27232dc71e1ae7981ddc5422e0045186c0121038f59ce3db645f9dbc697181be3527e293ddd
68c7f2bb62fa73f1ba84a9736d9effffffff01f658000000000000001976a914c446799026edde4fcfc2eae671afaab35
3ec6e6788ac00000000
Sending transaction to: https://insight.bitpay.com/api/tx/send
The sending api responded with:
{"txid":"84317e741dbd680cd042b9a77fb3a1415661a2fcb859f8a2176fec879495986d"}
```