Sugat Poudel
CS 4501: Crypto Currency Cabal

**Problem Set 1: Bitcoin Transactions**

**Problem 1**

a. Transaction ID (TxID)
   b94f41a9f0873e3faa6fdf1e08378af1fad61c06f1b485dc569247f1ee2608df
b. Transaction fee
   0.0001 BTC = 0.02 USD
c. Total value of block containing transaction
   14,487.20964744 BTC = 3,339,301.82 USD
d. Confirmation refers to the number of blocks that have been verified since the transaction. This transaction is included in block 37238, which has a timestamp of 2015-08-30 00:58:05, the next two blocks have the respective timestamps of 01:24:17 and 01:39:58. Looking at the difference between the 3rd and 1st block confirmation, it took around 41 minutes and 53 seconds for this transaction to receive 3 confirmations.

**Problem 2**

a. Bitcoin addresses of other students
   Following the input and output addresses, it is possible to determine who else the bitcoins were sent to (most likely to be other students in the class). Here are three such addresses
   1. 1ExEJE581mstkFPe5n5onKYHK9urFHCTTe
   2. 1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY
   3. 1fiWBi5u7oDzQrMNjeNLbmAvvHv545oy9
b. Trace of bitcoin source
   I kept following the input address for each transaction such that the transaction amount came from one source. The last address I found until there were several input addresses for the transaction was 1AxtTbSeLopt9LkacZ9w8kmzqxLaFobhyj, which originally contained 4000 BTC. The previous transaction with this address as one of the outputs, has no transaction fee and occurred on Aug 6, 2015 3:03:11 PM. It was probably purchased from an exchange and coinbase is one exchange that has no transaction fees.
c. Location of sender
   The blockchain does not store information regarding the sender's geographic location. The only information in the transaction pertaining to location are the display of network propagations, which is not closely related to the sender's location.

**Problem 3**

a. Malicious developer's evil wallet
   There are many things a bitcoin developer might do if intending to steal money from its users. For instance, the developer might intercept any private keys the user creates. Since a light client needs to connect to the internet to connect to the bitcoin network, they can easily transmit the keys and have access to the bitcoin balance of all of their users because they can attain the public keys and the address from the private key alone. They could

also add parameters to any transaction such that a small amount is always transferred to the developer's own wallet and disguise it as a transaction fee.

b. I am fairly confident that multibit is a safe wallet because it was referred to me by a trusted authority such as the official organization site for bitcoin (bitcoin.org). If I was to store all my money in the form of bitcoins, I would only use wallet implementations that store locally rather than ones that rely on online storage and ones that are recommended and linked by trusted financial authorities such as banks and monetary funds. Creating your own bitcoin wallet would also guarantee safety to some degree because you alone would know the internal workings of the wallet.

**Problem 4**

Written in the file verification.go

**Problem 5**

What needs to be trusted?

The program keypair.go generates a private and public key pair and returns the corresponding address. In order for one to completely trust the generated address, we need to be able to trusts that the sub-routines to perform their function as expected. The method generateKeyPair() and subsequently btcec.NewPrivateKey() should return a valid private key with the appropriate level of entropy for rand.Reader such that it is not predisposed to certain values that would make it easy for an attacker to guess. The NewPrivateKey() method should be trusted to using elliptic curve cryptography that would make it incredibly difficult for an attacker to reverse the public key to find my private key. Finally the function generateAddr() needs to be trusted to use a strong hash function such that for any given address h, it is hard to find the public key m such that H(m) = h and consequently there is little chance of finding collisions, these properties would ensure that the attacker would not be able to find another address that would also work with my public key, in which case the attacker could hijack my transactions.

**Problem 6**

Written in the file vanity.go

**Problem 7**

The method generateVanityAddress() relied on methods generateKeyPair() and generateAddr() from keypair.go. With this method I was able to find an address that contained a form of my name:

vanity address: 1MXfNrpMshKeHN6sUgAtCR87C6cPNBdu6v

private key: [f3d467d525e7b07a97c5145d964d4153b2e2f3514093caab8c0c943daa16ce41]

public key: [02c8df8654a71c9bedfed2f56bd9452cc57006ee539842d5e07fbf7d82cb46eebc]

**Problem 8**

The generated vanity address above is fundamentally the same as any other generated address the only difference being it is composed of specific and recognizable characters. In terms of changes in security, the cryptographic principles are still used however since it is a recognizable address, it is harder for attackers to substitute their own address and divert transactions.

However, in the case where the vanity address contains only a small recognizable string, it is easy for an attacker to generate a very similar address and divert transactions their way. If a vanity address was to be nearly or wholly composed of recognizable characters, then the attacker couldn't feasibly compute a similar address making any diversions extremely difficult. Such addresses, consequently, are computationally difficult. Vanity addresses are a double edged sword in terms of security and over all are the same if not worst in some cases than regular addresses.

**Problem 9**

1 mBTC transferred to above vanity address
TxID: a645ae5a998900492ced697c03e4879d65552dce39f71268c3485c5424b90d25

**Problem 10**

Some bitcoin was transferred to address 1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY using spend.go. The id of the resulting transaction was
c80100e3a464eb2b4350b788d6a27e78cc417c53bf8dc0980eaf3b13d9cd19bf

```
d-172-26-22-67:ps1 sugat$ go run spend.go -privkey
"f3d467d525e7b07a97c5145d964d4153b2e2f3514093caab8c0c943daa16ce41" -
toaddress "1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY" -txid
"a645ae5a998900492ced697c03e4879d65552dce39f71268c3485c5424b90d25" -
vout 1
Here is your raw bitcoin transaction:
0100000001250db924545c48c36812f739ce2d55659d87e4037c69ed2c490089995aae
45a6010000006b483045022100d4dce04770306dcef9e01a6470b924b30457104d6034
f112e9453b3d0a43d58802200704183b77615a3eb2a634d4fd24449c00b7a4b298c191
e410d887f2252a019e012102c8df8654a71c9bedfed2f56bd9452cc57006ee539842d5
e07fbf7d82cb46eebcffffffff01905f0100000000001976a914f0fa9453e9b8e07087
cc509d673964e312bf8b8088ac00000000
Sending transaction to: https://insight.bitpay.com/api/tx/send
The sending api responded with:
{"txid":"c80100e3a464eb2b4350b788d6a27e78cc417c53bf8dc0980eaf3b13d9cd1
9bf"}
```

**Problem 11**

0.0005 BTC or 0.5 mBTC was transferred to address [1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY] from my vanity address using a modified form of spend.go. The id of the resulting transaction was d032d0f8a66196c94a19d8c320e685e70ddd852b9a1f17f164271a7cdb5919d5

```
d-172-26-22-67:ps1 sugat$ go run custom_spend.go -privkey
"f3d467d525e7b07a97c5145d964d4153b2e2f3514093caab8c0c943daa16ce41" -
toaddress "1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY" -txid
"aaeda37aeb98d68321837315aaa2d437745d55870595c432e8e3ed2a6feefd9b" -
vout 0 -amount 50000
base addr: 1MXfNrpMshKeHN6sUgAtCR87C6cPNBdu6v
50000 satoshi being sent to [1NyBMbtqZLAmB3CSbjHzDHvedRJJJ7CupY]
Here is your raw bitcoin transaction:
```

```
01000000019bfdee6f2aede3e832c4950587555d7437d4a2aa1573832183d698eb7aa3
edaa000000006a47304402203befc1d2fdb1c0fff892ee58ec48241c2bed5c51275322
6948672c1117ab555a0220321deac1fc6abca9702a5d7eb14f923e9db8094e1a1b0075
375651cd4dec8a5a012102c8df8654a71c9bedfed2f56bd9452cc57006ee539842d5e0
7fbf7d82cb46eebcffffffff0250c30000000000001976a914f0fa9453e9b8e07087cc
509d673964e312bf8b8088ac409c0000000000001976a914e12f07a9b7b2ca56d81623
ad90636186ddcc6e6e88ac00000000
Sending transaction to: https://insight.bitpay.com/api/tx/send
The sending api responded with:
{"txid":"d032d0f8a66196c94a19d8c320e685e70ddd852b9a1f17f164271a7cdb591
9d5"}
```

Code in custom_spend.go

Problem 12

When I tried the same transaction with different addresses each time. The first transaction went through and I received a TxID of ce31db19e7a7ae00af9c7f1f7da77ea772a0996e9a0cbdc998631b5643ea051e however the second time the BTC network responded with the error message: Generic error (code -25).