1. (a) The transaction id is 8e2a5df4b5ce2fb042dfba5761a2e5ec2b9564fd81ea6cfb1bd5552494fa5e49.

   (b) The transaction fee was 0.0001 BTC. This is about 0.02 USD.

   (c) The total value of transactions in this block was 7,531.39109632 BTC. This is about 1,722,655.09 USD.

   (d) It took 19 minutes and 33 seconds for 3 confirmations.

2. (a) Here are 3 addresses: 1QLT7GNBnKNrGnKi7HNnZTYSDbPvUaVoZs, 12YwFZNmUoexn-QAMKFvRJSSydxTZUZPWx8, 1MDjCqjwnKmMWPxawpgN1UuDfbMUUgqnWw.

   (b) The bitcoins seem to have been purchased as 0.2 BTC by the following address 19WmbY4nDcjAEv6 At which point they were systematically distributed out to the class in amounts of 4.50 mBTC.

   (c) The total value of transactions in this block was 7,531.39109632 BTC. This is about 1,722,655.09 USD.

   (d) 3 blocks: 2015-08-28 15:34:34, received at: 2015-08-28 15:13:32

3. (a) There are numerous things a malicious developer could do. A simple approach would be to write a bitcoin wallet that would send each keypair for a user to a server somewhere where the developer could access and read them; the tricky thing is that for a user to use the wallet, it would most likely need to be open source. In this case, a high level of obfuscation and ingenuity would be needed to prevent themalicious code from being discovered. A more effective approach would be to get a group of like-minded malicious developers together and create the malicious wallet and have the people in the group all vouch for the legitimacy of the wallet. In this way, the average user who would only google for the validity of the wallet would easily fall prey to the wallet.

      One could also generate the randomness used to generate the private keys from a non-random source with a given probability distribution making it much easier to brute force the keys.

      One could also leave an intentional yet incredibly obscure security breach with the storing of the keypairs and then write a piece of malware to specifically target that vulnerability.

   (b) I am relatively confident that the wallet I am using is secure given that it is widely used and open source. Given the popularity of the software, more likely than not, there are multiple developers who have verified the legitimacy of the wallet. If I really wanted to verify the security of the wallet, I would sift through the source code and verify that there were no security breaches and a cryptographically secure source of randomness. I would also probably want to store my wallet on a computer that only performed wallet operations to prevent any other sorts of malware.

4. ```
   var curve *btcec.KoblitzCurve = btcec.S256()
   var x *big.Int = big.NewInt(0)
   var two = big.NewInt(2)
   x.Sub((new(big.Int)).Exp(two,big.NewInt(256),nil),(new(big.Int)).Exp(two,bi
   x.Sub(x,(new(big.Int)).Exp(two,big.NewInt(9),nil))
   x.Sub(x,(new(big.Int)).Exp(two,big.NewInt(8),nil))
   x.Sub(x,(new(big.Int)).Exp(two,big.NewInt(7),nil))
   ```
   ```

```
x.Sub(x,(new(big.Int)).Exp(two,big.NewInt(6),nil))
x.Sub(x,(new(big.Int)).Exp(two,big.NewInt(4),nil))
x.Sub(x,big.NewInt(1))
x.Sub(x,curve.P)
fmt.Printf("The difference is %v \n",x)
```

5. You need to trust that keypair.go does not store/send any of the keypairs generated. You also need to trust that the version of btcd you obtained does not store/send keypairs, that it uses a suitable source of randomness, and that it properly implements the bitcoin protocols.

6.
```
func generateVanityAddress(pattern string) (*btcec.PublicKey, *btcec.Privat

  r, _ := regexp.Compile(pattern)

  // Generate a private key, use the curve secpc256k1 and kill the program
  // any errors
  for i:=0; i>-1; i++ {
    priv, _ := btcec.NewPrivateKey(btcec.S256())
    var address string = generateAddr(priv.PubKey()).String()
    if(r.MatchString(address)){
      return priv.PubKey(), priv
      i = -2
    }
  }

  return nil,nil

}
```

7. I searched for the regular expression: [0-9]vig[0-9]

   This is the associated Bitcoin address: [1JfAKb9cS46RAexTxwqzGYpR8vig3JqSV1]

8. The address is equivalently secure. This is because there is no better way than brute force guessing to find the private key given a bicoin address.

9. The transaction ID is 9c2183314384f4fa11e49f9f2c5cdd4e02059ff92bf2b936fc17a8cf50e20dde

10. Code is attached in submission.