FinTech_hw2

Jiaqiu Wang

February 18, 2017

FNCE 385/885 Assignment 2 Blockchain

Initialization & Preparation

```
rm(list = ls())
setwd("C:/Users/rache/Google Drive/Spring 2017 Academic/fintech/assignment/ass2")
library("digest") # You need this package to calculate Hash
```

```
## Warning: package 'digest' was built under R version 3.3.2
```

```
# read the file exp.module.R in working directory
source("exp.module.R")
```

Part1: Create Keys

Recall how public and private keys are created in class. Suppose that p = 7, q = 23, and e = 13. a) What is the public key? b) Let d be the smallest positive interger which can be used to create the private key. What should the private key be?

Suppose that p = 7, q = 23, and e = 13. (a)N = p*q = 161 public key is N = 161, e = 13.

```
#(b)
#Pick any relative prime number e=7, to (p-1)*(q-1)=6*22=132
#finding the smallest postive interger d>=55 such that (7^d)mod(132) = 1
d <- 55 # The first trial is 55
while (exp.module(7,132,d) != 1) {   # In the parenthese there is the condition under
  # which the CONTINUES. In this case this means that the d tested does not have
  # (3^d)mod(103) = 93
  d <- d+1 # We test d in the next iteration
}
print(d)
```

```
## [1] 60
```

```
print(exp.module(7,132,d))
```

```
## [1] 1
```

(b)The private key is N=161, d=60.

Part2: Sign a message

Use the private key you generate above, sign a message: 12.

```r
# encrypting number 12 using the private key N = 161,d = 60. We essensially need to calcualte (12^60)mo
M <- 12
N <- 161
d <- 60

M.encrypted <- exp.module(M,N,d)
print(M.encrypted) # Show the signed number.
```

```
## [1] 64
```

The signed number is 64.

Part3: Check if a message is authentic

Message1: [10; 24]; signed message: [210; 453]. Message2: [11; 30]; signed message: [519; 370]. Message2: [12; 16]; signed message: [12; 594].

public key: N=161, e=13

```r
#exp.module(M,N,d) function calcualts (M^d)mod(N)
#message1:
c11 <- 210
e <- 13
N <- 161
M11 <- exp.module(c11,N,e)
print(M11)
```

```
## [1] 147
```

```r
c12 <- 453
M12 <- exp.module(c12,N,e)
print(M12)
```

```
## [1] 26
```

M11 is 147 not equal to 10, M12 is 26 not equal to 24, decrypting signed message using the public key, the result does not match with the text, message 1 is not authentic.

```r
#message2:
c21 <- 519
M21 <- exp.module(c21,N,e)
print(M21)
```

```
## [1] 8
```

```r
c22 <- 370
M22 <- exp.module(c22,N,e)
print(M22)
```

```
## [1] 27
```

M21 is 8 not equal to 11, M22 is 27 not equal to 30, decrypting signed message using the public key, the result does not match with the text, message 2 is not authentic.

```
#message3:
c31 <- 12
M31 <- exp.module(c31,N,e)
print(M31)
```

```
## [1] 75
```

```
c32 <- 594
M32 <- exp.module(c32,N,e)
print(M32)
```

```
## [1] 76
```

M31 is 75 not equal to 12, M32 is 76 not equal to 16, decrypting signed message using the public key, the result does not match with the text, message 3 is not authentic.

Part 4: Mining BitCoin

In this part we will generate proof-of-work, similar to what Bitcoin miners do. Speci???cally, suppose that you receive a message: "2017", and you work with Hash function md5. Find a number x attached to "2017" such that after the transformation with the Hash function your output starts with at least three 0's (zero). To reduce the amount of work, you can start with x = 1000.

```
str.msg <- c("2017")
k <- 1000
str.transformed <- paste(c(str.msg, as.character(k)),collapse="")
str.Hashed <- digest(str.transformed,algo = "md5") # Use Hash algorithm to calculate
while (substring(str.Hashed,1,3)!= c("000")) #check whether the first 3 elements are 000
  {
  k <- k+1 # add 1 to the k and start the next iteration if the first 3 elements are not 000
  str.transformed <- paste(c(str.msg, as.character(k)),collapse="")
  str.Hashed <- digest(str.transformed,algo = "md5") # Use Hash algorithm to calculate
  }

print(k) # Show the result
```

```
## [1] 1267
```