# FinTech Assignment 2

*Nicole Erich and Lindsay Tober*

*February 22, 2017*

**Part 1: Create Keys**

$$p = 7, \quad q = 23, \quad e = 13$$

**Part 1-a) Creating the Public Key:**

Calculate $N$ using $p$ and $q$.

$$N = p * q$$
$$N = 7 * 23$$
$$N = 161$$

Choose $e$ (already set to 13) as a relative prime number to:

$$(p-1) * (q-1) = 132$$

The public key is composed of $N$ and $e$, so it is {161,13}.

**Part 1-b) Creating the Private Key:**

Find a value $d$ such that:

$$e * d = 1 (mod (p-1) * (q-1))$$

```
p = 7
q = 23
e = 13

N = p*q
d = 1 # initialize, will increment in loop

while(TRUE){
  left = e*d
  inside = (p-1)*(q-1)
  right = 1%%inside # will be 1

  check_key = left%%inside
  if(check_key == 1){
    if(check_key%%1==0){
      break
    }
  }
  d = d+1
}
```

The private key that meets the above critera is d = 61. This is within the provided range [55,65].

### Part 2: Sign the Message

We are signing the message, "12".

```
# Define message
M = 12

# Sign message
C = exp.module(M,N,d)
message_check = exp.module(C,N,e) # test message
```

The encrypted message is 124. Just to check, it is decryped using $C^3(modN)$, which returns the original message, 12.

### Part 3: Check if the Message is Authentic

To check if a message is authentic, we can run it through a function that validates the message against the signed message.

```
test.authenticity <- function(M,N,e){
  # Test each for M = C^e(modN)
  for(i in 1:dim(M)[1]){
    C_test = M[i,2] # signature
    M_test = M[i,1] # message

    check = exp.module(C_test,N,e)
    print(paste("Does ",M_test,"(message) = ",check,"?"))
    if(M_test==check){
      print("    Yes! Message authentic.")
    }
    else{
      print(paste("    No, cannot confirm authenticity."))
    }
  }
}
```

We are using the public key from Part 1, {161,13}, where $N = 161$ and $e = 13$.

Test the following messages for authenticity:

Message 1: [10,24], signature is [210, 453] Message 2: [11,30], signature is [519,370] Message 3: [12,16], signature is [12,594]

Each of these messages is actually two messages that have been encoded, so we will connect message 10 with signature 210, message 24 with signature 453, and so on. Using these pairings, we will use the public key to authenticate the validity of each message/signature.

```
# Public Key Already Defined as:
## N = 161
## e = 13

# Define a 6x2 matrix of zeros for message and signature
M=matrix(0,6,2)
M[1,] = c(10,210)
```

```
M[2,] = c(24,453)
M[3,] = c(11,519)
M[4,] = c(30,370)
M[5,] = c(12,12)
M[6,] = c(16,594)

test.authenticity(M,N,e)
```

```
## [1] "Does  10 (message) =  147 ?"
## [1] "    No, cannot confirm authenticity."
## [1] "Does  24 (message) =  26 ?"
## [1] "    No, cannot confirm authenticity."
## [1] "Does  11 (message) =  8 ?"
## [1] "    No, cannot confirm authenticity."
## [1] "Does  30 (message) =  27 ?"
## [1] "    No, cannot confirm authenticity."
## [1] "Does  12 (message) =  75 ?"
## [1] "    No, cannot confirm authenticity."
## [1] "Does  16 (message) =  76 ?"
## [1] "    No, cannot confirm authenticity."
```

If a message has been signed using the authentic private key, we should be able to authenticate the message using the components of the public key, such that the message $= C^e(mod(N))$. By decoding the signature using the public key, the only way to obtain the message back is if it was first encoded using the authentic key.

The output above shows that none of the 6 messages were authentic, as none are able to retrieve the message back after decoding.

### Part 4: Mining BitCoin

Generating proof-of-work for the message "2017" working with Hash function *md5*:

```
str.msg <- c("2017")
x <- 1000

while(TRUE){
  str.transformed <- paste(c(str.msg, as.character(x)),collapse="")
  str.Hashed <- digest(str.transformed,algo = "md5") # Use Hash algorithm to calculate
  check_hash <- substring(str.Hashed,1,3)
  if(check_hash == c("000")){ # Check whether the first 3 elements are 000
      break
    }
  # Make sure no infinite loop
  if(x >= 5000){
    break
  }

  x = x+1
}
```

The number $x$ attached to "2017" such that after the transformation with the Hash function the output starts with at least three 0's (zeros) is $x = 1267$. The hashed string is 000cf8f28c169080085032ca3669f0cc, which starts with three 0's (zeros).