Assignment 6

Aayush Sharma Acharya

April 2, 2020

6.1: eavesdropping on rsa

(a) We know:

$$e = 15852553, n = 44331583$$

We first perform prime factorization for our number n, We can do that by using the following function:

```
getFactors :: Integer -> [Integer]
getFactors n = [x | x <- 2:[3,5..n-1], n `mod` x == 0]
```

Since we know that n is going to decomposed into two prime factors p and q, we do not need to check prime for each of our factors in the above function. We get the result:

$$n = 44331583 = 5003 \cdot 8861$$

If we let

$$p = 5003, q = 8861$$

We can calculate $\Phi(n)$ by:

$$\Phi(n) = \Phi(p \cdot q) = \Phi(p) \cdot \Phi(q) = \Phi(5003) \cdot \Phi(8861) = 5002 \cdot 8860 = 44317720$$

All of the above steps can be performed by our function:

```
getPhi :: Integer -> Integer
getPhi n =  foldl (\acc x -> acc * x) 1  $ map ((-)1) (getFactors n)
```

The above function gets the factor of the n, decreases them by 1 and then multiply both the elements in the list to get a scalar. Now we have to solve

$$ed \ (mod \ n) \equiv 1$$

We can solve this by using our egcd function and taking the second element of that function:

```
egcd :: Integer -> Integer -> (Integer, Integer, Integer)
egcd a 0 = (abs a, signum a, 0)
egcd a b = (d, t, s - (a `div` b) * t)
    where (d, s, t) = egcd b (a `mod` b)

sec':: (a, a, a) -> a
sec' (_, x, _) = x
```

So, solving the modular inverse we have:

$$d = 1951097$$

We can do everything all at once using the following simple function:

```
getDecryption :: Integer -> Integer -> Integer
getDecryption n e = sec' $ egcd e $ getPhi n
```

So, from the calculations, we have $(e, d, n) = (15852553, 1951097, 44331583)$. We use this in the following haskell code to decrypt our list:

```
data Key = Key Integer Integer deriving (Show)

dec :: Integer -> Key -> Integer
dec c (Key n d) = c^d `mod` n

decList :: [Integer] -> Key -> [Integer]
decList ml k = map (\x -> dec x k) ml
```

Decryption result in following list:

```
71,111,44,32,103,111,32,
97,119,97,121,32,67,111,
114,111,110,97,32,118,105,
114,117,115,33
```

(b) We use the following haskell code to convert our decryption list to character list:

```
ordList :: [Int] -> [Char]
ordList l = map chr l
```

Upon conversion we get: `Go, go away Corona virus!`

All of this code is provided in the zip file: Run `runhaskell rsa.hs` to execute it

6.2: proof of work

(a) `QctAhSrD` gets converted to SHA256 checksum with first 12 bits as 0s.

(b) For autogeneration of the plain text which gets convert to our required checksum we use the following short python code: (included in the zip file, run `python3 proof-of-work.py`

```
import hashlib
import random
import string

def convert_to_sha256(bs):
    return hashlib.sha256(bs).hexdigest()

def generate_8_byte():
    e = ''
        .join(
            random.choice(
                string.ascii_uppercase + string.ascii_lowercase + string.digits
            ) for _ in range(8)
        )
    return e, str.encode(e)

def main():
    while 1:
        plain, as_bytes = generate_8_byte()
```

```
hs = convert_to_sha256(as_bytes)
if (hs[0:3] == "000"):
    print(plain, " : ", hs)
    return
```