Ph 20: Assignment #7

Maya Fuller

Due: June 11, 2018

Problem 1

Congress discovered that an unnamed foreign government purchased ads on various social media sites. The ads targeted US voters in order to sway public opinion in favor of the candidate Joe Exotic. The founder of FaceMash (a social network affected by the data breach), Zuck Markerburg, wants to make sure the data of his site's users is protected.

Problem 2

RSA is asymmetric public-private key cryptosystem. It's based on the factorization of the product of two large prime numbers. The user creates and publishes a public key based on two large prime numbers and an auxiliary value. RSA is relatively slow, but this can be solved by passing encrypted shared keys for symmetric key cryptography. Another problem is that if the same message is send to e or more recipients in an encrypted way, and the receivers share the same public key, e, but different p and q, the message can easily be decrypted via the Chinese remainder theorem.

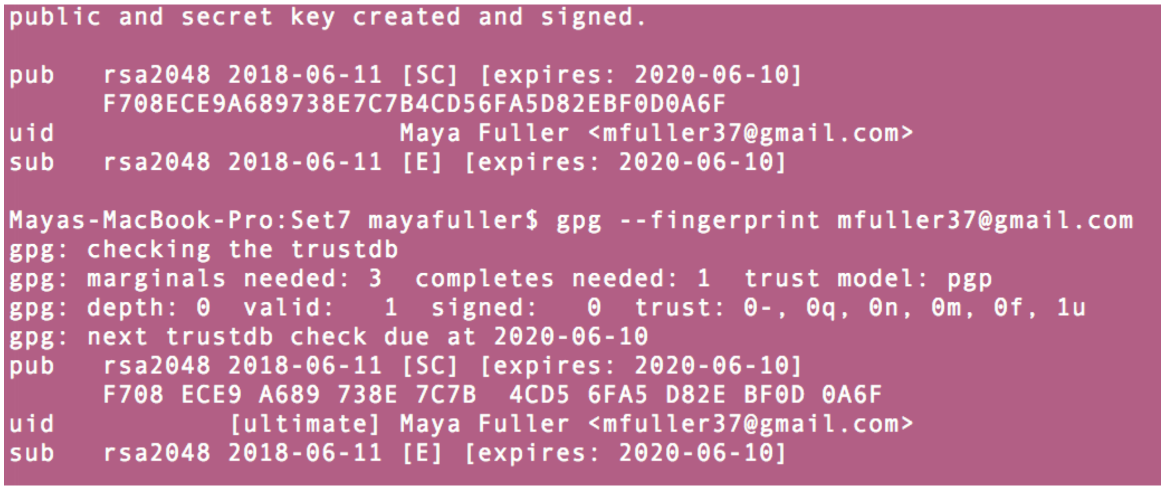
Problem 3

```
import math
import random

def isPrime(n):
    return (all(n%i!=0 for i in range(2,int(math.sqrt(n))+1)))

def lcm(x,y):
    return int((x*y)/(math.gcd(x,y)))

# Calculate the modular multiplicative inverse given e and lambda_n
def modMultInv(e,lam_n):
    for i in range(1,lam_n):
        if i*e % lam_n == 1:
            return i

def generateKey():
    # 64 to 256 -> 6 to 8 bits
    primes = [i for i in range(64,256) if isPrime(i)]
    p = random.choice(primes)
    q = random.choice(primes)
    n=p*q
    lam_n=lcm(p-1,q-1)
    # generate a list of possible values of e
    poss_e = [i for i in range(1,lam_n) if math.gcd(i,lam_n)==1]
    e = random.choice(poss_e)
    d = modMultInv(e,lam_n)
    return (n,d,e)

def encryption(m,pubKey):
    (n,e)=pubKey
    c = pow(m,e,n) #(m**e)%n
    return c

def decryption(c,privKey):
    (n,d)=privKey
    retreived_m=(c**d)%n
    return retreived_m

# This collision attack calls the encryption function for all possible values
# of m and checks what return values are equivalent to c.
def collision_attack(m,pubKey):
    (n,e)=pubKey
    poss_m=[]
    c = encryption(m,pubKey)
    # 65536 is the max value for n if p and q are 8 bits or less
    for i in range(65536):
        if encryption(i,pubKey) == c:
            poss_m.append(i)
    print('Possible values of m are: {}'.format(poss_m))

keyGen = generateKey()
privateKey=(keyGen[0],keyGen[2])
publicKey=(keyGen[0],keyGen[1])
m = random.randint(0, keyGen[0])

print('m = {}'.format(m))
print('Public key is: (n,e) = ({},{})'.format(publicKey[0],publicKey[1]))
print('Private key is: (n,d) = ({},{})'.format(privateKey[0],privateKey[1]))
print('c = {}'.format(encryption(m,privateKey)))
print('decrypted m = {}'.format(decryption(encryption(m,privateKey), publicKey)))

collision_attack(m,publicKey)
```

Problem 4

GPG uses a combination of symmetric-key cryptography and public-key cryptography. It encrypts messages using asymmetric key pairs generated by each user. Then public keys can then be exchanged with other users. Because GPG is command-line-based, it can't easily be incorporated into other software. This is often overcome using the API wrapper, GPGME. RSA keys generated with YubiKey were found to be affected by the ROCA vulnerability, weakness that allows the private key to be recovered from the public key in keys generated from affected devices.

Problem 5



```
public and secret key created and signed.

pub   rsa2048 2018-06-11 [SC] [expires: 2020-06-10]
      F708ECE9A689738E7C7B4CD56FA5D82EBF0D0A6F
uid                      Maya Fuller <mfuller37@gmail.com>
sub   rsa2048 2018-06-11 [E] [expires: 2020-06-10]

Mayas-MacBook-Pro:Set7 mayafuller$ gpg --fingerprint mfuller37@gmail.com
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2020-06-10
pub   rsa2048 2018-06-11 [SC] [expires: 2020-06-10]
      F708 ECE9 A689 738E 7C7B  4CD5 6FA5 D82E BF0D 0A6F
uid           [ultimate] Maya Fuller <mfuller37@gmail.com>
sub   rsa2048 2018-06-11 [E] [expires: 2020-06-10]
```

Problem 6

The essential processed for secure authentication are enrollement, authentication, and life-cycle maintenance. Enrollment involves applying to a credential service provide (CSP) to become a subscriber, authentication is the process of receiving credentials (e.g. token, credentials, etc), and life-cycle maintenance is the CSP's task of maintaing the user's credential. A strong password can be compromised through a key logging attack. Through a variety of techniques, the keys struck on a keyboard can be captured in order to obtain passwords or other confidential information. One-time passwords are an effective countermeasure for key logging attacks.

Problem 7

Though RSA and similar cryptosystems can't be broken with solely math, there are variety of the other vulnerabilities. Something as simple as not generating random enough numbers can completely compromise a public-private key pair, making a flawless cryptosystem quite difficult to acheive. In my threat model, personal information wasn't taken from users individually, but from an external data analysis database. The information in the database could be better secured by encrypting data from each person individually rather than encrypting many people's information in batches. To make sure that the information stays secure, the entire database shouldn't use the same public-private key pair, but several depending on the amount of data that is bing stored.