Project 3:  
CS6035

Andy Guo  
aguo43@gatech.edu

# TASK 2

To improve password security, you can add a salt value to the end of the password before it is hashed. As a result, if there are multiple of the same passwords, they’ll result in different hash values as the salt at the end of them will make them different.

# TASK 3

Your answer starts here…

# TASK 4

Given one of the public keys n, I had to factor it into p and q first in order to get the totient value. In order to that, I used Pollard’s p-1 algorithm, which is used for integer factorization. With that algorithm, I was able to find the value for p and then divided n by p to get q. After getting p and q, I was able to calculate the totient since they were both prime numbers. To get the private key, I had to get the modular multiplicative inverse of the public key e and the totient of n. With both values ready, I used the extended Euclidean algorithm to find the modular multiplicative inverse. The extended version of Euclidean algorithm finds the greatest common divisor between two numbers and Bezout’s coefficients. The reason why this works to find the modular inverse is due to Bezout’s lemma. Bezout’s Lemma states that for two nonzero integers *a* and *b* and *d = gcd(a,b)*, then there exists two integers *x* and *y*, such that *ax + by = d*. For our problem of finding the private key relies on finding the *d* value for de = 1 mod (p-1)(q-1). The public key e and the totient value should be coprime, meaning that they have no common factors between them besides 1. We can look at de = 1 mod (p-1)(q-1) as de + (p-1)(q-1) = 1 mod (p-1)(q-1), we trim it down further and we are looking at de + (p-1)(q-1) = 1, which fits in with Bezout’s Lemma. As a result, the factor of e will be the private key value. Once we get that value, we have to get the modulus of that against the public key n, to get the actual private key value.

# TASK 5

The public key was vulnerable in this task because though finding the two prime factors of a large integer, it is faster to find greatest common divisor between two large numbers. We had an entire list of public key values to try and find the greatest common divisor between the public key n and one of the public keys in the list. One the gcd was found, it could be called p as it is one of the two factors of n. Then simply dividing n by p, we are able to get q. With the p, q values along with the public key e, we are able to get the private key using the extended Euclidean algorithm again.

# TASK 6

The broadcast RSA attack works on focusing on the fact the same public exponent was used for each of the encrypted messages. With that common value, we are able to use the Chinese remainder theorem to solve for the common m value.

# References

1. Your references start here…