Nathalie Ruano
Dr. James Cannady
CS 4325 - Intro to Security
12 July, 2018

Project 3: Crypto - Have fun with RSA

This paper will explain the reasoning behind choosing certain methods to solve RSA problems. In part II of this project, we will describe how to obtain factors (p and q) of a given N value, as well as, obtaining private key from p, q, and e. In part III, we will describe how to find whether to public keys share similar factors and then obtaining private key if so. In part IV, we will describe how to obtain original message from given C and N values.

**Part II**

In get_factors(n) function we will be attempting to obtain the factors of our public key n. The first step was to obtain the sqrt of our n, as our factors will not be greater than this value. Once this was completed, computation was performed in order to make sure that the sqrt obtain is of odd value in order to traverse until we find a value that when n is mod by it will give us zero (meaning that its divisible by it). This will be our P value. After finding this initial value we will just divide N by our P value which will give us the last factor Q.

After completing get_factors(n), we use the obtained P and Q factors along with exponential e value in order to derive the private key. The formula to derive our private key is as follows: $d = e^{-1} \bmod phi(n)$ where phi(n) is $(p -1) * (q - 1)$.

**Part III**

In this part of the project, it was assigned to find waldo which means to find the two public keys will need to share a common prime factor. This can be done by using gcd; if the gcd between two keys is 1 then it means they do not share a common prime number but if number is different then this means we found waldo. Once we find “waldo” , we use both public keys and exponent in order to find the private key value of d. This can be done by finding our common factor. As mention previously to find the common factor we use gcd on our public keys: $gdc(n_1, n_2)$. After our gcd is derived, we only need to continue to use one of the public keys to solve for our private key so we will us use $n_1$ in our example. In order to retrieve private key we will need

to perform the same steps done in part II. We will divide $n_1$ by the found common prime factor p, in order to retrieve our prime common factor $q_1$. After deriving our two common factors, we will find out what our phi($n_1$). As stated in part II, phi($n_1$) is (p - 1) * ($q_1$ -1). Once we figure out our phi($n_1$), we will using the extended euclidean algorithm to find out our private key, d. It is to be noted that d can result in a negative number which would have to be taken care by adding phi($n_1$) to it as -d mod phi($n_1$) == (-d + phi($n_1$)) mod by phi($n_1$).

**Part IV**

In the last part of the project, we will use Chinese remainder theorem in order to obtain C such that $c_1$,$c_2$,.,$c_n$ mod by $n_1$, $n_2$,…,$n_n$ will give us a congruent solution. Once we retrieve the value derived from Chinese remainder theorem then we find the cube root of the value. This is because C == $m_3$ mod $N_1N_2N_3$ which means that any message that we encrypt must be smaller than the modulus: $m_3 < N_1N_2N_3$ which can be reduced to C = $m_3$, therefore leaving us only to cube root the value C in order to decrypt our message. Due to data range limits when deriving the cube root of C, an implementation of binary search is done to handle the large range of numbers.

**Citations**


“Chinese Remainder Theorem.” Haversine Formula - Rosetta Code, rosettacode.org/wiki/Chinese_remainder_theorem.

“Chinese Remainder Theorem.” Wikipedia, Wikimedia Foundation, 11 July 2018, en.wikipedia.org/wiki/Chinese_remainder_theorem.

“Understanding Common Factor Attacks:An RSA-Cracking Puzzle.” Understanding Common Factor Attacks: An RSA-Cracking Puzzle, www.loyalty.org/~schoen/rsa/.