Justin Kelley

5/9/2017

COSC/MATH 314

Final Project Description

My program consists of several modular functions, since I was not allowed to use the BigInteger library I had to construct functions to work around extremely large integers. I will go through the functions in the order that they are listed in the program.

Main: Main begins by taking in a text file(rsa.txt) which has all the cyphertext integers on it. It then creates an ArrayList ‘cypherText’ and iterates through the file adding every integer to the ArrayList. It then obtains the public key information(n,e) from the user and then iterates through every cyphertext integer in the ArrayList, decrypting and converting each integer into text form.

Factor: This function took in a number ‘n’ and returned an ArrayList of prime factors (aka: p and q)

mulInverse: This function took in an integer a(the number of which the inverse will be found),b(the modulo number), and n(also the modulo number). This function pushes a and b into the extEuclid function to return an s such that as = 1, then pushes s and n to the modCalc function return the answer to the s(mod n) and giving you the answer to a-1 (mod b).

extEuclid: This function takes in an integer a and a modulo b and using the extended Euclidean algorithm to solve for x in the equation ax + by = gcd(a,b). X is then used in the mulInverse function as ‘s’ to solve for a-1(mod b).

modCalc: This function takes in an integer and a modulo b and returns the answer to a(mod b).

modExp: This function takes an integer a, an exponent b, and a modulo n and returns the answer to ab(mod n). It does this by using the repeated squaring method and breaking down the large exponent into smaller chunks. This is method is necessary for computing large number since the primitive type “int” can only hold numbers up to 231-1.

decrypt: This function takes in a cyphertext integer ‘c’, a integer n(from the public key), and an integer e(also from the public key). It first pushes ‘n’ into the factor function and returns an ArrayList of all prime numbers of n. These prime numbers are now assigned to p and q and ‘mod’ is calculated through the equation mod = (p-1)(q-1). ‘d’ is then calculated by pushing (e,mod,mod) into the mulInverse function since d = e-1(mod(p-1)(q-1)). The function then decrypts c by pushing (c,d,n) into the modExp function and returning the answer to m = cd(mod n).

convertToText: takes in an integer m(the decrypted message) and inverting the equation n1(262) + n2(26) + n3, finds the numbers that correspond to the letters of the alphabet. It then uses the convertToLetter function to print out the letters.

convertToLetter: takes in an integer n(a number 0-25 to be converted to a letter) and matches it with an array of characters in alphabetical order, returning the matching letter.