CSE 539, Applied Cryptography
Project 6: Due Dec 2 2019
RSA Operations

Projects are to be done in groups of two

You are provided with a C++ RSA library. You can extract the required files from the given Zip file.

The programs compile and run on Linux/Unix systems. Please use 32 bit systems, or 32 bit flags.

Instructions on how to compile and run are given in the myReadme file and samplerun.txt file inside the zip folder. It is preferred that you attempt to solve this problem in a Linux environment, as it is easier for the TA to use one machine to check your programs. (*Note: Please read the code contained, there are limitations on the size of numbers they handle*.)

You have to do the following:

1. Perform encryption and Decryption each using the RSA routines provided here.
    a. Create 10 instances of the RSA class without giving arguments, generate random message or assign messages, and perform encryption through each of the 10 classes.
    b. Create 5 instances of the RSA class by passing a large prime number [p](> 30,000), and perform encryption decryption
    c. Create 5 instances of the RSA class by passing 2 large prime numbers [p,q] (> 30,000) and perform encryption decryption
    d. Create 10 instances of the RSA class by passing 2 large non-prime numbers (> 30,000) and perform encryption decryption. In most of the cases the message should not get decrypted correctly.
    e. Show the results for encryption and decryption for each of the above cases. If you notice anything out of the ordinary, please record it. For example, if you are able to decrypt using non-prime numbers as the p,q values then it is out of ordinary.

2. Challenge Response: Scheme
    a. Create an RSA object. Call it RSA1
    b. Create a new RSA object, call it RSA2. Obtain the public key and modulus [n] of RSA1. Assign these two to the public key and N value in RSA2.
    c. Generate a random message [random BigInt number]. Encrypt it using the public key of RSA2 [You have stored the pub key of RSA1 in RSA2].
    d. Decrypt the value using the private key of RSA1.
    e. Match both the values (original message vs decrypted message), they should be the same. If so Challenge Response scheme is completed.

3. Blind Signature:

Blind signature is a kind of signature, where the signing authority does not know what is being signed. Blind Signatures are used to implement Anonymous money orders. You have to implement a blind signature scheme. The scheme you have to implement is described in wikipedia at the link: http://en.wikipedia.org/wiki/Blind_signature

The scheme can be briefly stated as:

a. Alice obtains the public key and Modulus N of the person (Bob) who is to sign the message
b. Obtain a random number and its inverse with respect to the Modulus [Not phi] of Bob
c. Alice obtains/generates a message to be signed.
d. Alice encrypts the random number with the public key.
e. Alice multiplies this value by the message
f. Alice then takes a modulus over N
g. Alice sends it to Bob
h. Bob simply decrypts the received value with the private key
i. Bob sends it back to Alice
j. Alice then multiplied the received value with the inverse and takes a modulus over N.
k. The value obtained above is the signed message. To obtain the original message from it, again encrypt it with Bob's Public Key.

To implement the sending and receiving, you may use whatever method you choose, you can use files, pipes or just pass it through functions, or through global memory. Ensure that the random number is around 16 bits long for the program to work correctly.

Some encryption examples are shown in rsaDriver.cpp and rsatry1.cpp in the zip file.

Submit: The code to do #3, blind signatures. The output must be self-explanatory. Code will be compiled and run.

Upload to Canvas.