Jackson State University
Department of Computer Science
CSC 439/ 539 Advanced Information Security
Spring 2013
Instructor: Dr. Natarajan Meghanathan

Project # 1: Secure Client-Server Communication using RSA Algorithm

Max. Points: 100

Objective: In this programming project, you will encrypt (using RSA) a sequence of characters, given by your full name – first name followed by last name, separated by a blank space (for example: John Smith), at the client side and send the ciphertext (corresponding to each character) to the server. At the server side, you will decrypt each ciphertext character as they arrive, aggregate the characters as they arrive and then print the full name.

Seed Code: You are provided the Java code for the following three programs that you can use for reference to implement this project. You can download the Java code from the course website.

(1) RSA Key generation

(2) A program that inputs a public key-private key pair, the ascii value of the character to encrypt and prints the cipher text as well as the decrypted plaintext

(3) Socket programs to send a sequence of 10 Big Integers from the client to the server; the server receives and prints them

The BigInteger API in Java would be very useful. It is located at
http://download.oracle.com/javase/1.4.2/docs/api/java/math/BigInteger.html

Pre-Cursor to your Client-Server Implementation: Run the RSA Key generation program twice, each time passing a different seed value for the public key. Note down the public key, private key and the 'n' value generated for each case. One of these public-private-key pair and the corresponding 'n' value will be considered to represent the client and the other public-private-key pair and the corresponding 'n' value will be considered to represent the server.

Client-Server Implementation: Develop client and server socket programs that do the following:

Client Program

(1) Input the full name of the user; find the ascii integer value of each character in the name and store it in an array. You can read them as strings and use the String-based constructor of the Big Integer class to transform these values to Big Integers.

(2) Input the 'n' value for the client, the private key for the client, the 'n' value of the server and the public key of the server

(3) Run a for loop that goes through the ascii array, integer by integer, transforms them to a Big Integer, encrypting them as follows and sending each ciphertext big integer over the socket to the server

(i) First encrypt the ascii Big Integer with the private key of the client and its 'n' value

(ii) Encrypt the result of (i) with the public key of the server and its 'n' value

(4) After sending the encrypted versions of each ascii integer in the array, exit from the for loop, send a "-1" Big Integer (as illustrated in the sample client version of program # 3) to the server to indicate to the latter that all characters have been sent.

Server Program

(1) Input the 'n' value for the server, the private key for the server, the 'n' value of the client and the public key of the client. You can read them as strings and use the String-based constructor of the Big Integer class to transform these values to Big Integers.

(2) Run an infinite while loop (as shown in the sample server version of program # 3) and do the following

(i) Read every Big Integer received - if it is equal to -1, break from the loop and terminate the program

(ii) If the Big Integer received is not equal to -1, then do as follows:

(a) Decrypt the received Big Integer first using the private key of the server and the corresponding 'n' value

(b) Further decrypt the value obtained in (a), now using the public key of the client and the corresponding 'n' value

(c) Transform the Big Integer to a regular integer (the ascii integer value) and print the corresponding plaintext character

Before exiting from the loop, you would have printed, character by character, the full name sent by the client.

What to Submit:

1. ***An abstract*** (7.5 points) – briefly describe the actual project, in about 150-200 words
2. ***Introduction*** (15 points, about a page)
    a. Explain the working of the RSA encryption algorithm
    b. Explain the working of client-server programs using connection-oriented sockets
3. ***Client Code and its Description*** (25 points) – Explain the sequence of steps in your client code and also include the actual client code. You could refer to the segments of this code in your description and explain the segment.
4. ***Server Code and its Description*** (25 points) – Explain the sequence of steps in your server code and also include the actual server code. You could refer to the segments of this code in your description and explain the segment.
5. ***Execution at the Server Side*** (7.5 points): Explain the sequence of executions that you conducted at the server side and include screenshots for the same. You should include screenshots at least for the following:
    a. The public key and private key of the server (including the ‘n’ value) generated by running the RSA Key generation Java code
    b. The Big Integer values of the ciphertext characters received, their corresponding decrypted plaintext Big Integer values and the actual plaintext characters
6. ***Execution at the Client Side*** (7.5 points): Explain the sequence of executions that you conducted at the client side and include screenshots for the same. You should include screenshots at least for the following:
    a. The public key and private key of the client (including the ‘n’ value) generated by running the RSA Key generation Java code
    b. The ASCII values and the corresponding Big Integer values of the plaintext characters input by the user, and their ciphertext Big Integer values after undergoing the encryption as described above.
7. ***Conclusions*** (12.5 points, 300 to 400 words): Explain how the confidentiality and integrity of the data is maintained in your communication. Also, summarize your experiences with client-server programming, programming with Big Integer class and RSA encryption.