Dillon Van Buskirk & Sandy Becerra

Applied Cryptography

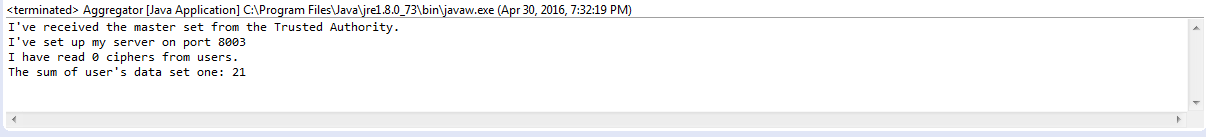
Dr. Qinghua Li

01/05/2016

For this assignment students were asked to implement a privacy-preserving sum aggregation protocol. This protocol was based on research done by Dr. Li.

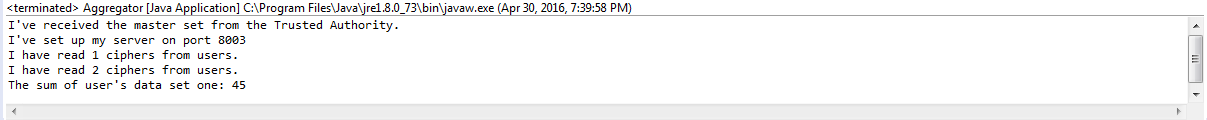
To start we picked our values for n, c, and user random data. 'n' is the number of users and we chose 'c' to be 4, user data to be the number of children (made delta for this data to be 5) and the age(delta for age we picked to be 100). We began by implementing the trusted authority, user1, and the aggregator. We tested one set of data and sent it to the aggregator. Because we were only using one user, the summation of the data would be equal to what the user random data was.

***For testing the one user, we used the user's age as the random data(in our case the user's age was hardcoded to be 21). Below are the results of what the aggregator outputs when ran.***



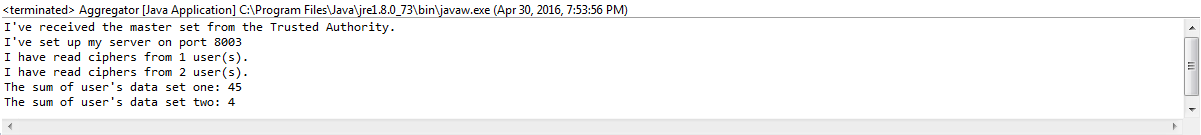
Once we had one user working and the aggregator functioning as we expected, we proceeded to add a second user and again ran the tests only with an iteration of single data for each user. Like when testing user1 at the beginning, we used the user's age as the random data. User1 maintained the age of 21 and for user2 we hardcoded the age of 24. Because we had the code from user1 working, we copied the code from user1 to user2 and changed data and port numbers to make sure the connection to the trusted authority and aggregator worked.

***Below are the results of the test that was done when user1 and user2 sent their encrypted data (age) to the aggregator. Keep in mind that user1\_age=21 and user2\_age=24. Sum=45.***



After having successfully sent encrypted data from two users to the aggregator and summing them together correctly, we continued by adding a second random data to each user (second random data will be the number of children). The second data was encrypted and sent to the aggregator by each user, and the aggregator summed the data received and printed out the result.

***below are the results of our final code for this assignment.***



Main Issues:

When we initially started this assignment, we didn't expect to be using classes such as KeyGenerator or SecureRandom. Initially we figured the secrets generated by the trusted authority could just be random generated numbers. This however didn't work when trying to use int arrays as the keys for HMAC because that requires a parameter of type Key.

Also, several port numbers had to be used in order to get users and aggregator to receive the correct data from the trusted authority. Users also had to have different ports to connect to the aggregator. This was something that was not accounted for during initial implementation, but having developed code incrementally helped avoid debugging incorrect code.