COP5615: Fall 2015
Project 4 - Part II

Team Members:

Aakriti V Agrawal	14331642	aakritiv.agrawal@ufl.edu
Divya Awatramani	46599609	dawatram@ufl.edu

The zipped folder includes Facebook Simulation

Folder Structure:

```
| + RestAPI
  | + src
    | +main
      | + scala
        | + demo
          | +SampleServiceActor.scala
          | +Stuff.scala
      | + resources
        | + application.conf
  | + build.sbt

| + Client
  | + src
    | +main
      | + scala
        | + demo
          | +Main.scala
      | + resources
        | + application.conf
  | + build.sbt
```

To run:

Command Prompt 1:

```
> cd Aakriti_Divya_Project4/FacebookSimulation/RestAPI
> sbt
> compile
> run
```

Command Prompt 2:

```
> cd Aakriti_Divya_Project4/FacebookSimulation/Client
> sbt
> compile
> run numUsers
```

numUsers -> number of users using facebook.

Security Structure:

Algorithms used:

- RSA1024 Public Key Cryptography – for symmetric key encryption and digital signature
- AES128 Symmetric Key Cryptography – for data encryption

Symmetric Key Exchange:

- RSA 1024 is used to encrypt the symmetric key.
- If User1 and User2 wants to communicate with each other, they require symmetric keys
- User1 encrypts his symmetric key with User2's public key and sends him the key.
- User2 then decrypt the key using his private key.

Data Communication:

- AES is used for data encryption and decryption.
- To access User2's profile, User1 fetches the encrypted data from the server.
- User1 uses his own private key to decrypt the symmetric key of User2, and then uses the symmetric key to decrypt the data.

User Authentication:

- RSA 1024 is used to digitally sign a token and authenticate the user to the server
- Server sends a Secure Random token to every client during initialization
- To access its data from the server client digitally signs the received token using its private key and sends it along with the request.
- Server verifies the received signature and authorizes the user.

Privacy:

- 3 levels: self, friends and public
- Keys are shared based on the privacy settings.
- For e.g. if User1 requests User2's profile and User2's privacy settings is "self" then the server doesn't allow the access to User1.

Implementation:

- During initialization of a user, server sends a token to the user for digital signature.
- Whenever the user wants to interact with the server, he sends the digital signature with the request to verify himself.
- User encrypts the data with his symmetric key before sending it to the server.
- Server stores all the data in the encrypted form.
- Keys are exchanged directly between the users based on their privacy settings.

- Server allows a user to access other user’s data based on privacy.
- Data is exchanged in encrypted format and only allowed users has access to it.