Facebook API with Secure Data and Secure Access

The objective was to develop Facebook API using Spray Can library suite of Scala which is based on Actor model and has REST integration. JSON was used for serialization and deserialization.

We have generated the APIs for:

i) **Page** (home page)
ii) **Post** (posts by user)
iii) **Friends List** (user's friends)
iv) **Profile** (user's own profile)
v) Photo
vi) Album

SprayClient:

ClientObject.scala: This is the main program which instantiates all the user actors, each of which can emulate the activities of Facebook users. The main actor also keeps a lookout for the timeout and initiates a system shutdown after the given runTime.

Worker.scala: This program represents each Facebook user. The code is developed to generate all the desired GET and POST requests from respective users which are then sent to the server.

Each user also has a scheduler with time arguments to generate requests at regular intervals or just once.

SprayServer:

We have developed the servers(ServerDBs) in a distributed fashion so that load is shared among them.

Server.scala basically serves as the router which resolves the requests and directs the request to the respective serverDBs.

ServerDB.scala: This program stores all the POSTs by the user: status, photos, albums, and friendsList in respective data structures. For the GET requests it looks up for the values asked for and sends them to the client.

Functionalities Implemented:

- post
- getStatus
- postPhoto

- getPhoto
- getProfile
- getFriendsList
- getHomePage
- postAlbum
- getAlbum
- getFriendsProfile
- getFriendsPublicKey

Security:

Authentication: We have used digital signatures for authentication of each user using RSA asymmetric keys. The server authenticates each user before processing any of its requests.

Steps:

- Server generates a Secure Random token using SHA1PRNG and sends it to the user.
- The user digitally signs the token using its private key and sends it back to the server.
- The server decrypts it with the user's public key and thus verifies the user's identity.

Encryption: Data (status and photos) is always stored encrypted in the server. We have used AES symmetric keys for encryption.

Steps:

- User generates a private AES key for each post (status and photo) it makes.
- The data is encrypted using that key and stored in the server after encrypting it with its private key.
- For fetching data, the user is first authenticated and receives encrypted data from the server. Then we fetch the encrypted key corresponding to the data and decrypt it using the public key and then use the decrypted AES key to decrypt our data.
- For fetching friend's data, user gets their encrypted data and public key from the server. Gets the encrypted AES keys(encrypted with friend's private key) for the friend's data from the friend. Decrypts the keys using friend's public key. And then decrypts friend's data using the decrypted AES keys of the friend.