**COP 5615 - PROJECT 4 (PART -2)**

**PROJECT REPORT**

**FACEBOOK SIMULATOR**

**(Encryption and Authentication)**

In addition to the functionalities implemented in part-1of the project (also mentioned in the report for the same), part-2 of the project has additional security and authentication built on top of part-1.

**Initialization**

* When the Server starts, it generates it’s own pair of Public-Private Keys and also a secure random number (to be used for authentication).
* When the Client starts, it generates it’s own
  + Pair of Public-Private Keys
  + An initialization vector (to be used to encrypt the 1st block of its posts.
* The client sends its Public key to the server.

**Authentication**

* When a client actor is started, it sends a initClient message to the server.
* The server replies back with a secure random number.
* The client encrypts the secure random number (received from the server) with its own private key, and sends it back to the server.
* The server decrypts the encrypted secure random number (received from the client) with the public key of the client and matches it with the number it had sent.
* If it’s a match, it sends back an ‘Authenticated’ message.
* Only when a client receives an ‘Authenticated’ message from the server, it proceeds to run all the functionality modules.

**Encryption and Decryption of Posts**

Posting on Own Wall

A client selects 10 random friends from its friend-list, who can view the message it is about to post.

The Post object sent to the server comprises of :

* The encrypted message to be posted.
* The IV used for the post.
* The List of 11 Public-Key (of the friends) encrypted AES key (10 friends + self) to be used for decryption by the friends who have access to the post.
* The friends’ user id is appended in front of the corresponding encrypted AES (with ‘!’ as delimiter) to enable efficient searching while retrieval (explained below). Same is the done for the AES key encrypted with own Public Key.
* This post object is sent to the server to be added in the List of Post objects for the user’s own profile.

Posting on friend’s Wall

The client randomly selects a friend, on whose wall the message has to be posted.

A similar Post object as above is created containing:

* The encrypted message to be posted.
* The IV used for the post.
* The List of 2 Public-Key (friend + self) encrypted AES key to be used for decryption by the users who have access to the post (own or friend).
* The friend’s user id is appended in front of the corresponding encrypted AES (with ‘!’ as delimiter) to enable efficient searching while retrieval (explained below). Same is the done for the AES key encrypted with own Public Key.
* This post object is sent to the server to be added in the List of Post objects for the friend’s profile.

Wall Posts Retrieval

When a user(say A) tries to retrieve the posts of another user(say B), the flow of commands is as follows:

* The command is sent to the server with both the ids of A and B as parameters.
* If the ids are same, it means the user wants to view his own wall. If they are different, it means the user wants to view another users’ wall. A check is done to ensure the user is friends with the other user, before fetching the posts.
* The list of Post objects is pulled up for the user whose wall has to be viewed.
* For each Post object :
  + A substring of the AES encrypted keys is done and checked.
  + The id in the substring has to match the user id of the user who wants to view the posts.
  + When one of the ids match, the corresponding encrypted AES key, as well as the message and the IV is saved to a different list of Posts which have to be returned.
* This list of posts is then sent back to the client and each post is decrypted by the AES key obtained by decrypting the Public-Key-encrypted AES key obtained from server.

For demo, a random user (user Id 10) has been selected and it’s wall has been periodically fetched to illustrate the encrypted and the corresponding decrypted message and the wall’s updating process.