COMPSYS302 Python Project

Name: In Ha Ryu
UPI: iryu815

About

The client requested to model a digital communication model with high data security. We have been assigned to implement a functional prototype of the system that is social media network like communication with some peer-to-peer method. As developers, our class have discussed and developed protocol and requirements of the prototype and individuals have come up with functional prototype.

Specification

The requirement of the system includes:

1. Allows a user to log into the system.
2. The system can automatically find other users on other computers.
3. User can create and maintain a simple profile page.
4. Users can send messages, images, audio and PDF files to each other.

Therefore, the prototype has:

1. Login page
2. 2FA
3. Functionality of sending messages
4. Functionality of reading messages
5. Functionality of sending Files
6. Functionality of reading received Files (embedded viewers)
7. Functionality of viewing who is currently online
8. Functionality of getting and reading users profile page
9. Functionality of editing the users profile page
10. Request and acknowledge of Deletion of unwanted Message
11. Auto refresh

The main reason for this communication method is to keep the data safe from the unauthorised user, therefore requires high level of security while maintaining the fluency of the user experience. This could be done by having high level encryption with some hashing standards. The functional prototype of the system has some encryption and hashing standard at this stage.

The following encryption standards are included in the prototype:

1. XOR Encryption
2. AES-256 Encryption
3. RSA-1024 Encryption
4. Mix of RSA-1024 and AES-256 Encryption
5. Mix of RSA-2048 and AES-256 Encryption

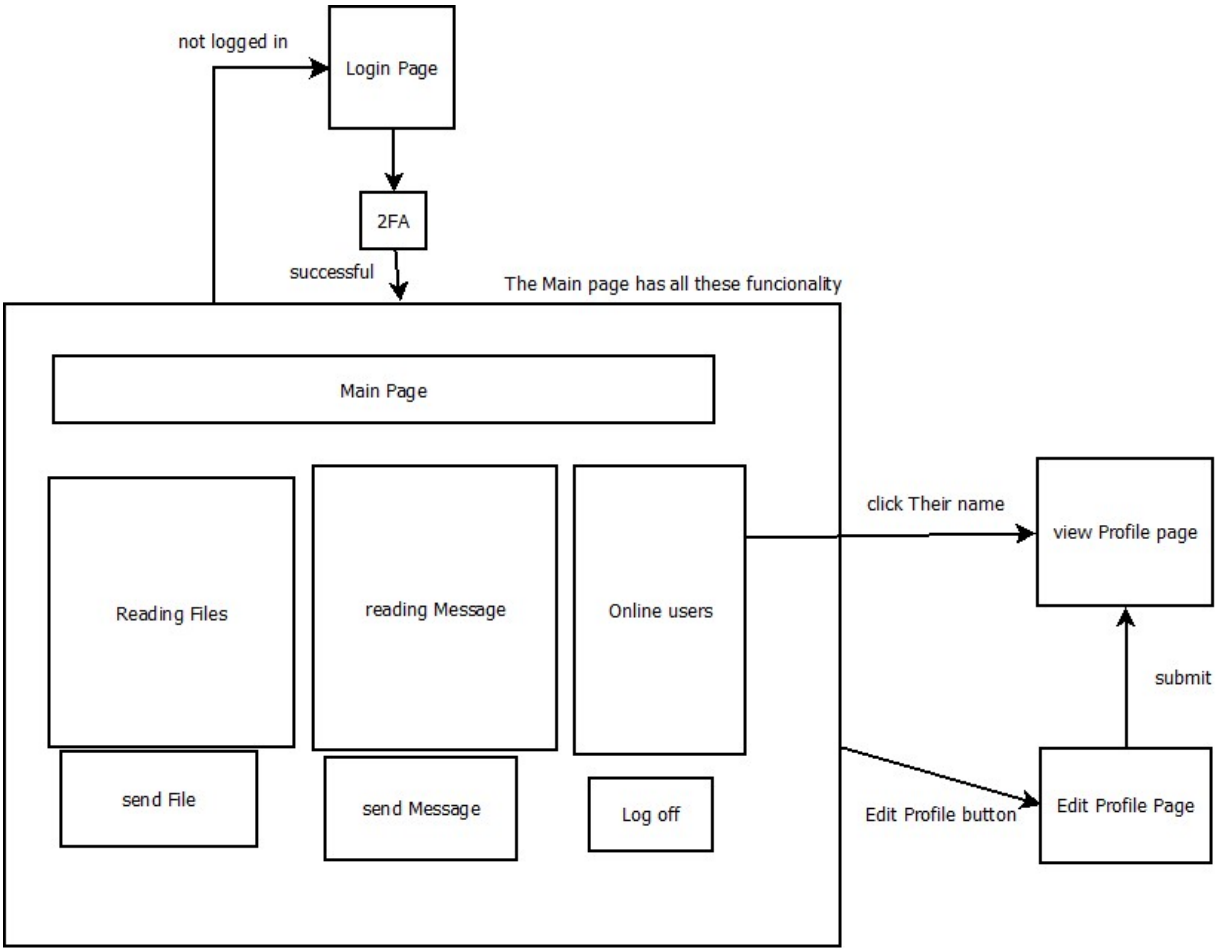
The following hashing standards are included in the prototype:

1. SHA-256(without salt)
2. SHA-256(with salt)
3. SHA-512(with salt)
4. bcrypt
5. scrypt

In addition to the security, 2FA Authentication method is implemented on protocol. The prototype uses google authenticator which give QR code to setup authenticator for first time use. Use would have to give the code form the authenticator and give the right username and password to login. This gives extra security to the system.

System Design

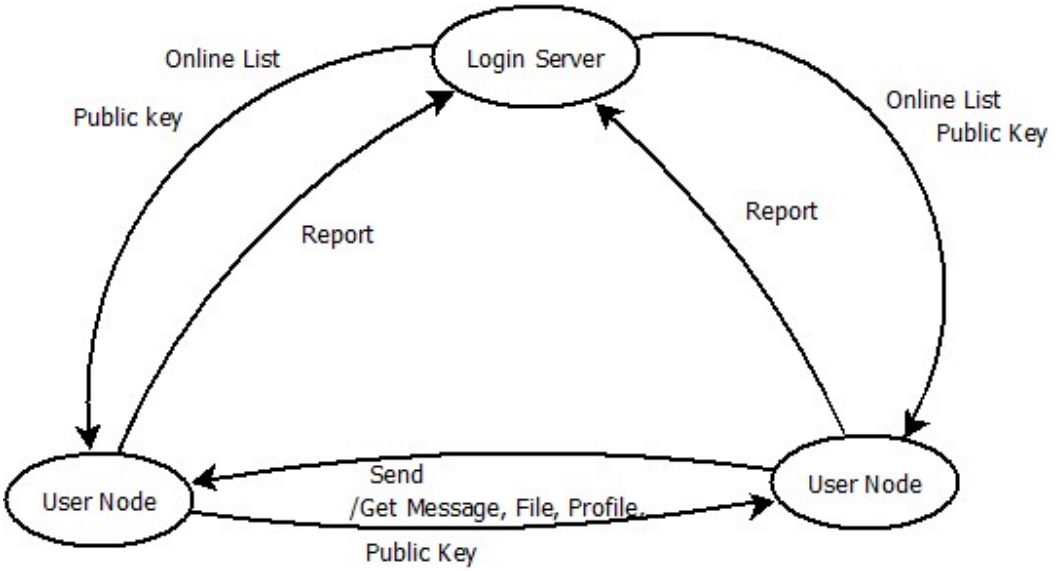
Page view of the prototype:



User will face the login page at the start, when it tries to login, 2FA authentication is required. If user successfully login, Main page that reads messages, files, list and online users appear. It also has several forms such as send file, send Message, edit profile and log off which functionality follows the name. Upon clicking the name of online users, the page moves to view their profile page. Also, by clicking edit

profile, page that enables you to edit profile appears. Here you can upload new profile picture, change names and description such as location and others.

Top -level view of the system:



Upon the login, the user node reports their existence and their location such as IP address and port indication of physical location to the Login server. Then the User Node can get their online list and the public key. Using the online list and public key the user node can acknowledge the existence of other user node, and therefore find, send and get the required information. The information includes the message, file, profile and public key which all necessary for the communication.

The system describes the hybrid system where both central sever and peer-to-peer (aka. p2p) method is used. After getting the location of the other node, the sending and getting the information is all based on p2p method. The weakness of the system is that the login server is required to know the existence of other node and therefore if we have system failure in login server, although it is simple and light, can fail all the system.

We cannot always have login server running as it requires maintenance and development which we can say that this is not so sustainable. The solution to this is to have back-up server running in case of maintenance or to implement p2p. Since back-up server is costly, it is not ideal. The solution with p2p requires further research.

Development

As developers, we have discussed, developed and proposed the protocols of the system. Every node has to follow system, so they have correct way when communication with other node. After summing up, we have come up with the protocol.

For p2p:

1. All inputs and outputs for p2p APIs must be in json format unless stated.
2. List of Encoding that we should support.

3. List of Encryption, method of encryption we should support.
4. Encryption Key for XOR and AES
5. List of Hashing we should support.
6. List of error code.
7. List of function and its name and parameters

For Login server to user:

1. All communication with the server is over straight HTTP unless specified.
2. List of function and required parameters that user node can call
3. List of error code that user node receives
4. Meaning of location parameter
5. Rate limiting of the login server
6. Login server encryption key for AES

We also have found some significant issues while developing the prototype. Upon the integration with another user, some format of the information was different to another. The major example was the error code. Some gave the error code with string while some gave it with integer. Due to these differences we have discussed and although most people agreed that it should return as string, the solution was however, to have the output of the format changed to string whenever it is received.

Future development

Python 2.7 support ends at 2020 which is two more years from now. If we move to the python 3, since some of the syntax are different, it can raise an error which we have to re-write and re-test again.

To lessen the dependency of the login server we need to implement look up algorithms. This can be done by using distributed hash table method (aka. DHT) with some manual initial link. We need some kind of initial link such as giving location of the user node to other so some still have the information of user node. This can be done through the login server. Once everyone has some information about others the request node can call other known nodes to retrieve information about target user which will be their id and location with some hash standard. If the request node does not have the information that matches, it looks up new user node that is unknown to the request node, request to that node to find the target. If the information matches, then the user node should be able to retrieve the existence of target node.

The protocol is Back-end focused, thus it needs more development on front end. Although the use of CSS is appealing to the user, front-end functionality such as implementation of JavaScript should be developed.

For purpose of the login server security, the encryption key can be more randomised/not fixed. Login server can have a fixed AES key of shared through the wanted user which can be used to request the time-based/randomised public encryption key for RSA. Then user node can use that public encryption key to send username and password to the login server. The benefit of this solution is that since the RSA encryption key is not fixed, even they hijack the encryption key and the encrypted message, they will not be able to use it since the RSA encryption is temporary. Hijackers will not be able to decrypt the message to get username and password and even if they do, they will require AES key to report and retrieve important information such as other nodes public key.