Information Security Course Project Design Document

Gan Yue 11300180158

June 4, 2014

1 Design

1.1 Registration

The process is as follows:

1. After establishing connection, the client sends its identity (e.g. email address) to server.
2. Then we exchange a pair of symmetric keys among server and client using Diffle-Hellman algorithm.
3. After that, the server generate corresponding public key and private key using RSA algorithm, and sends back the encrypted (with symmetric key) private key and its hMAC to the client. In the meantime, server also sends the public key of itself to client.
4. Client checks the hMAC and saves its private key to file.
5. Server only stores client's identity and public key, and it recognize the client next time by its identity and private key.

1.2 Login

Login is a process of authentication:

1. Client sends a login request to server with a random number.
2. After receiving the request, server signs the random number with its private key and chooses a new random number, and then sends them back to client.
3. Then client checks server's sign with server's public key. In the meantime, client signs the new random number with its private key and sends it back to server.
4. Finally server checks client's sign and sends back a result (success or fail).

1.3 Friending

The process is as follows:

1. Client A sends a friend request to client B via server.
2. Client B responses the request to client A via server.
3. Now server sends their public key to each other.

1.4 Messaging

The process is as follows:

1. Client A generates a symmetric key, and encrypts it with B's public key.
2. Receiving the message, B decrypts it with its private key.
3. After that, A and B encrypt their message using the symmetric key and send it along with hMAC to each other
4. The receiver decrypts received message and checks integrity by calculate hMAC again.

1.5 Group Messaging

It's similar to messaging and the differences are as follows:

1. Group owner generates a symmetric key, encrypts it with group members' public key and sends it to every group member.
2. Group members decrypt received message with their private key and save the symmetric key.
3. When a team member sends a message to group, he or she encrypts the message with shared symmetric key and calculates hMAC. And server helps to send the message and hMAC to every member in the group.
4. Receiving a group message, group members decrypt it with symmetric key and check hMAC.

2 Analysis

2.1 Reliable key distribution

When server distribute client's private key, it encrypts the key using AES algorithm. In the meantime, there is hMAC with the encrypted private key to ensure integrity.

2.2 Reliable authentication

We use signature and verification part of RSA algorithm to achieve reliable authentication.

2.3 Secure key exchange

The key exchange between server and client uses Diffle-Hellman algorithm.

The key exchange between clients uses RSA algorithm.

2.4 Secure message sending

Messages sended is encrypted and decrypted using AES algorithm.

2.5 Integrity of message

When sending encrypted messages, we also send corresponding hMAC.

2.6 Efficiency

Applying RSA algorithm is not cheap, so we only use it for key exchange and authentication, and we use AES for secure message sending.

3 How to use

3.1 Requisition

Before running the program, please make sure you have Node.js and MongoDB installed.

3.2 Usage

1. Run a MongoDB server
2. Run server by "node ./server/server.js"
3. Make a copy of directory "client". Say it "client2". (Since we store keys in file, there will be conflict in one directory)
4. Run two client by running "node ./client/client.js" and "node ./client2/client.js"
5. Register by running "register $< username >$" (only have to register before first use)
6. Login by running "login $< username >$"
7. Add friend by running "friend $< friendName >$"

8. Approve or deny other's friend request by "approve $< other'sName >$" or "deny $< other'sName >$"

9. Show friend list by running "friendlist"

10. Get friend's public key by running "require $< friendName >$" (only have to require it the first time you talk to your friend)

11. Share a symmetric key with a friend to begin a talk by running "sendKey $< friendName >$" (send key every time you reconnect through socket)

12. After that, talk to the friend by running "message $< friendName >< message >$"

Any problem during use of the program, please do not hesitate to contact 11300180158@fudan.edu.cn.