Secure Instant Message Tool Design *(Modified)*

Ruiyang Xu and B.H.Priyanka

1. SETUP

1.1. Assumptions

We assumed that all the user's name and PDM information has already been stored on the server's database. So that no user need to do account registration to the server (though they still have to do the login registration). Also the client will know the public key of the server.

1.2. Architecture

The basic architecture of our design has one server with multiple clients (Figure 1). The server is used for login, which will do authentication of the user and then register it's information which can later be used by other users. The server will maintain two lists: one is permanent in some database and the other is ephemeral in local memory. As mentioned in the requirement, nothing will be stored in a client machine. All the key pairs are generated dynamically during run-time and all the info are fetched from server when necessary. The client basically only knows about the server's IP and port number. Every client have at least two ports (Figure 2) for communication: one for server and the rest for other clients.

2. PROTOCOLS

2.1. Authentication Protocol in Login

A PDM protocol is used during login authentication. As indicated in Figure 3, after successfully authenticated with each other, the DH key will be used in all client-server communication. The client will also generate a RSA key pair for peer authentication purpose later. The public key will then be stored in the server. *Currently those global tables might not be thread-safe, we didn't have enough time to add locks for them. So this might cause some race condition issue on the server*

2.2. Key Establishment Protocol And Peer Authentication

After the user type in the 'send' command, the source client will first ask the server for target user's information (i.e. IP, port number and stored public key, it might need to open a new port for later communication) and initiate a mutual public key authentication with the target using it's public key. The target client received this authentication request will also fetch the server to find out the source's information (which include its public key). And then using that to finish the mutual public key authentication. After the peer authentication, source and target will generate new RSA key pairs and exchange new public keys.

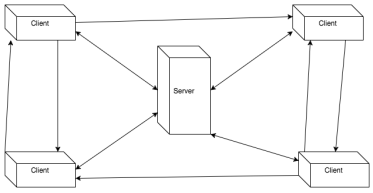

Fig. 1. The basic architecture.

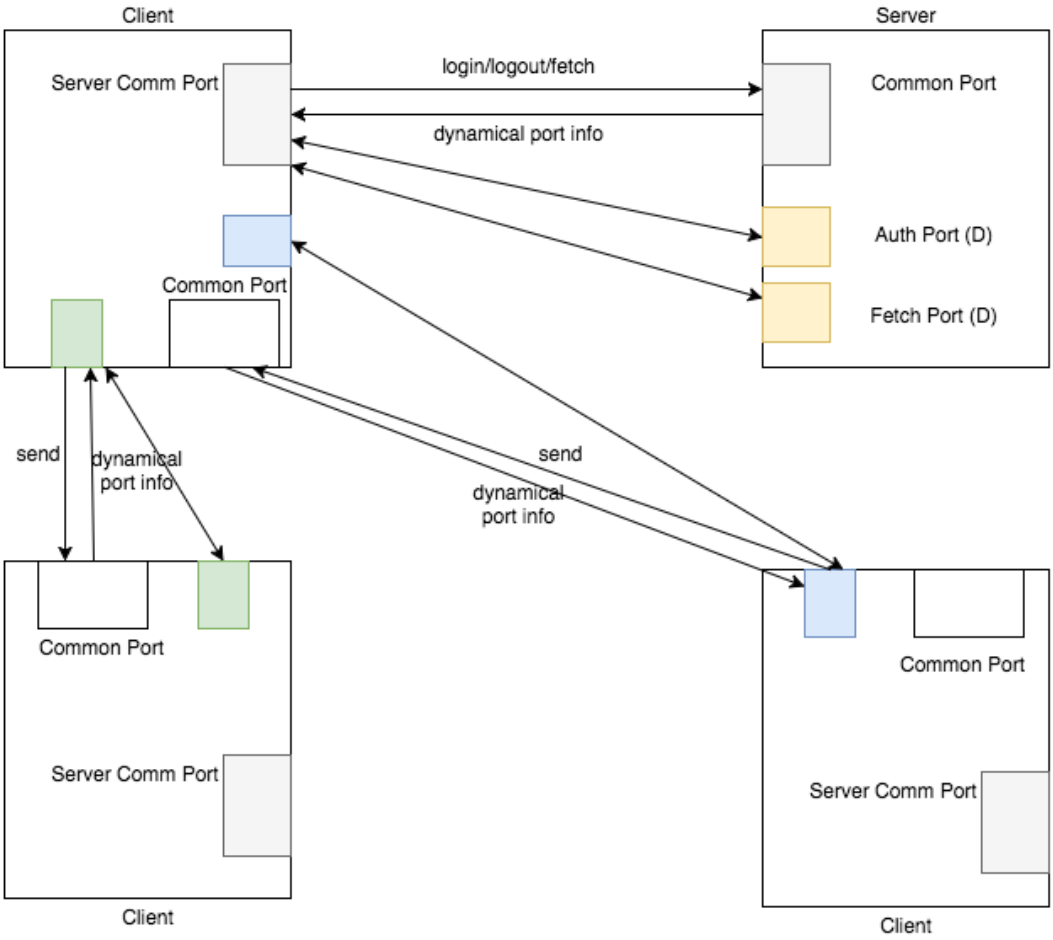

Fig. 2. Communication port design.

Table I. Permanent user info list

User	Moduli
Alice	$2^{W_1} mod\ p$
Bob	$2^{W_2} mod\ p$

Table II. Ephemeral user info list

User	IP	Port	RSA-Auth-Public
Alice	192.168.1.100	5505	*****
Bob	192.168.1.105	4567	*****

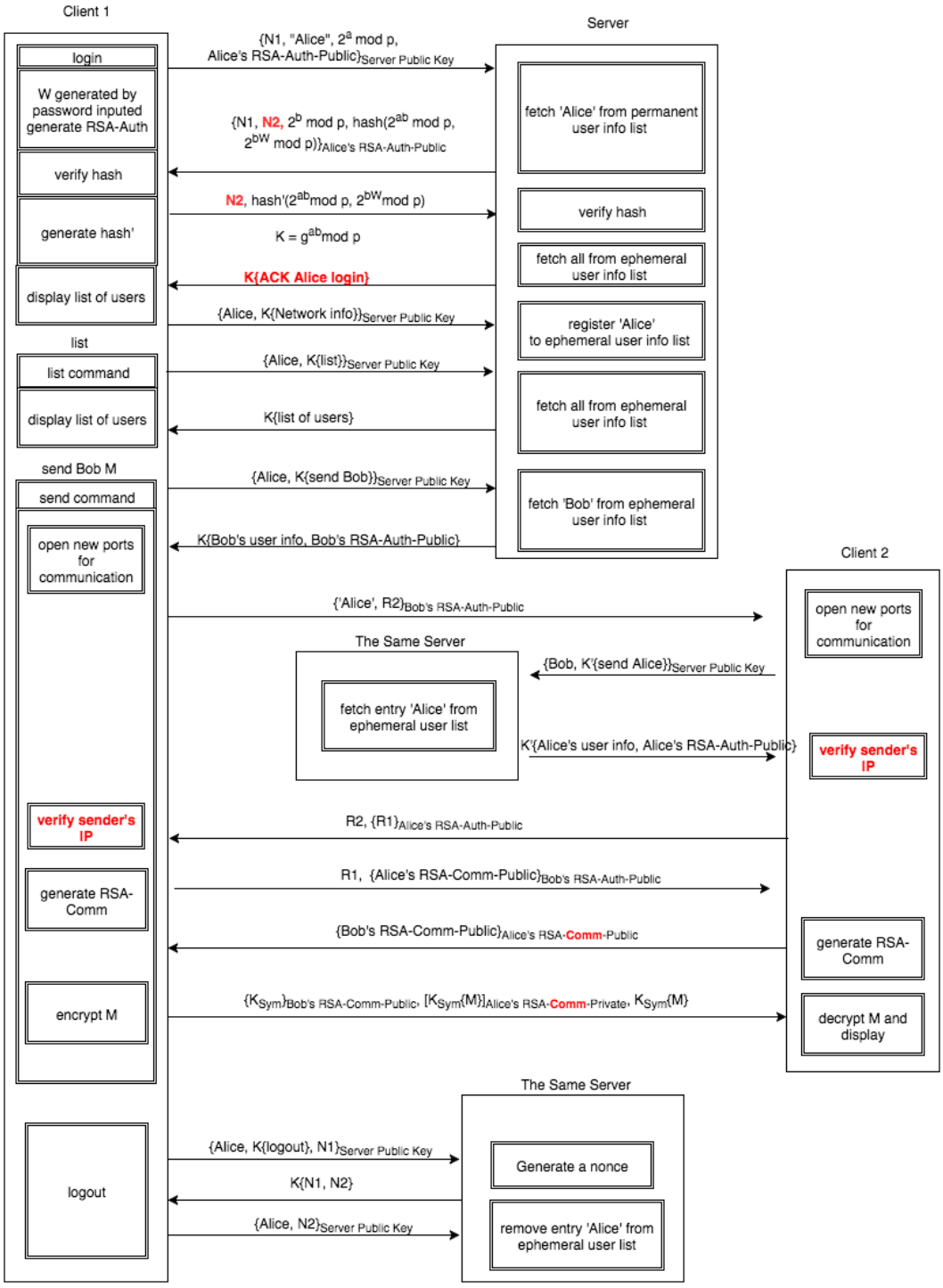

Fig. 3. The overall protocols. Modified parts are marked by red

2.3. Messaging Protocol

Source will generate a symmetric key to encrypt message, and encrypt the symmetric key with RSA public key. and also include a signature on the encrypted message for integrity verification.

2.4. Logout Protocol

After the user logout (that means terminate the client program), a 'logout' command will be sent to the server. The server then simply remove the user's entry from ephemeral user info list. To prevent a replay attack, nonces were used here. *Since using UDP, server will have no idea on logout if the client exit anbnormally. If have more time, we might be able to implement a 'heartbeat' message mechanism*

3. SECURITY FEATURES AND FLAWS

3.1. Use of Weak Passwords

Since this protocol is a strong password protocol, use of weak password will be susceptible only if the attacker has broken into the server database and has successfully done the dictionary attack on the stored moduli.

3.2. Denial of Service Attack

Only if the attacker can somehow know a user name, then he can initiate a large amount of login request. The server will then have to compute multiple times of $hash(2^{ab} mod\ p, 2^{bW} mod\ p)$. To foil this attempt, the server can use some delays for the same user's login request.

3.3. End-points Hiding And Perfect Forward Secrecy

This protocol do have end-points hiding. Every message is totally encrypted with a proper secrete key. And it also provides perfect forward secrecy, because RSA key are generated dynamically for each time message communication.

3.4. Untrustable/Compromised Server

If a server is untrustable, then we assume it has no idea of users' PDM moduli information. Then the mutual authentication will be foiled. A compromised server will only leak PDM moduli, and the attacker can hardly derive the original password from that. However, since we are not using a real random process to generate prime number from the password, this might be a flaw in current implementation and also leaves a chance for the attacker to crack the password. If we have enough time, a real PDM protocol can be implemented.

3.5. Replay Attack, Reflection Attack and Impersonation

The design avoids replay attack since it uses a randomly generated session tokens-NONCE on both the client and the server sides to establish authentication. It also avoids the reflection attack since every message has been encrypted with either rsa public key or the Shared DH key. It's also impossible to impersonate any other user.

3.6. Man-in-the-Middle Attack

Design is not vulnerable to a passive Man-in-the-Middle attack since there is no information that the attacker can obtain by eavesdropping or by any other means. Each message is secure. However, because using of UDP, an active MITM attack might be harmful: since dynamic port info has been sent around with plain text, the attacker can get this info and send some garbage to the peer so that foil the communication. If we have more time, then we'd rather not to use UDP and encrypt port info also.