Authentication

It's not secure to transmit the password and pattern in plaintext through the network. So we need to protect the password and pattern by encryption or message digest. In our implementation, we use one handshake protocal instead of multiple handshakes for better efficiency. The format of authentication message is Key{Hash(TS | Pattern) | TS | Pattern}. The Key is 128bit generated from MD5 mesage digest of password. When it comes to the encryption part, we encrypt the Hash(TS | Pattern) | TS | Pattern by AES algorithm. Moreover, we add a salt to for better diversity. With the timestamp and salt, it's invulnerable to rainbow attack.

But someone could eavesdrop the message and resend the message to the server. It's easy to prevent the replay attack. In our implementation, the server record the timestamp. If the server receives a message with the same timestamp which already exists in the time window, the server will refuse the message. Of course, the server will also drop the message with invalid timestamp.

The Hash(TS | Pattern) provides intergrity protection for the message. In the implementation, we use SHA-1 message digest. After the server receives the message, the server will decrypt Hash(TS | Pattern) | TS | Pattern from the message, and compute the SHA-1 message digest from TS and Pattern. Then compare the hash value to that in the message to check if they match. If the two message digest are the same, then the integrity is fine.

After that, if the validation of pattern is successful, then the authentication is fulfiled.

However, there is some drawback in the protocal. Since the client couldn't authenticate the identity of the server, it's vulnerable to in-the-middle attack. The attacked could impersonate the server, get the message from the client and resend it to the server. Besides, the protocal is not secure when it comes to server break-in attack. The attacker could modify the time in the server.