**Introduction: CS 1653 Phase 4**

**T5: Message Reorder, Replay, or Modification**

**Threat:**

The threat of having messages tampered with by an active attacker is a high risk for server and client systems. Reorder, replaying, and modification can place mistrust in the users. An active attacker could manipulate the messages and destroy the structural integrity of the Secure File Server model.

**Mechanism:**

**A close up of a piece of paper

Description automatically generated**

Figure 1. T5 Mechanism

The mechanisms to combat threats 1-4 already allows the client and servers to secure the connection through a session key. Although this prevents eavesdropping it does not prevent an active attacker from manipulating sent messages. Therefore, we have to adopt new methods to encrypt the messages themselves. The methods that we propose are using sequence numbers and message authentication.

Message authentication is to verify the integrity of what is sent to the other party. We have chosen to use HMAC with SHA1 to authenticate these messages. In doing so we also need to incorporate a shared secret key.

Sequence numbers can be used to alert either parties that the message has been tampered with. This can be accomplished with either a parallel array that is sent alongside our messages.

**Defense:**

The security of the mechanism is ensured by the message authentication protocols and the sequence numbering. Any modification to a message will be evident when the receiver uses HMAC with a shared authentication key to verify the message. If the digest values do not agree, then tampering has been detected and the receiver should terminate the session. Sequence numbers provide security against replay and reordering. With both parties of a session keeping track of a sequence number, a replay of any previous message is easily detectable by the receiver, as the sequence number will be less than what is expected by the receiver. By requiring the Client to choose and the Server to choose, we can also detect replay of an entire session as each side of the exchange is choosing a random number at the outset of the session, and thus the probability of a replay successfully containing the correct responses to challenges or sequence initiation is insignificant. Reordering of messages is also easily detectable when each party is checking sequence numbers, as any message that does not contain the expected sequence number is suspect.

**T6: File Leakage**

**Threat:**

Previously, files saved to a File Server were encrypted over the network but not in storage. Therefore, an untrusted File Server could leak fully readable files that any user, malicious or not, could read. Obviously, we must encrypt the files to protect them against unauthorized access, but at the same time we must also enforce group privileges where only group members have access to the files and also maintain security by preventing former group members from accessing files created after their dismissal.

**Mechanism:**

**A picture containing text, door, standing, baseball

Description automatically generated**

Figure 2. T6 Mechanism

Due to the fluid nature of groups in this System, users can add/remove the personnel in the group, we need to incorporate and store symmetric secret keys for group files. An authenticated group member will then submit to the Group Server the group which they wish to upload or download to. The Group Server embeds the keys for that particular group in the group member's token and returns it to the group member. The group member may now use the File Client to encrypt and upload files or download and decrypt files.

Though the File Server should never see any group keys, it must be aware of which key was used to encrypt the file which it has stored - this will be done in a Share File class that will include the key version. This parallel array will store the index of the list that the group’s keys will be stored. Therefore when a file is uploaded to the file server it will be encrypted with the most recent key or the key with the highest index. The user uploading the file will send the key version (index number) to the File Server as part of the upload process. When the File Server receives a request to download a file, it will include the key version with the encrypted file so that the File Client can use the correct key from the user's token to decrypt the file.

**Defense:**

The correctness of the mechanism relies on proper distribution of keys and accurate record keeping of files and the key versions used to encrypt them. The onus of distribution falls on the Group Server, which must authenticate a user, check that the user is in fact a member of a particular group, and finally provide the user with a token that contains the keys the user will need to successfully interact with the File Server through the File Client. The record keeping of files and key versions is handled by the File Server through its Share file class. For the File Server to do its part, it must be given the key version by the user through the File Client. Without the key version number, the File Client would be unable to decrypt the file as it would not know which key to use.

**T7: Token Theft**

**Threat:**

Based on the description of T7, we need to ensure that the tokens stolen by the file server and assigned to another user are only usable on the server at which the theft takes place. In a realistic situation, the valid token holder Alice may log into the file server 1, which steals her token and passes it off to another user Bob.Without comprehensive and strict regulation, Bob can get access to Alice authority and do anything in the system with the name of Alice; in another word, Bob will be able to do anything that Alice’s security token allows Alice to do. Such a situation will not only exert a negative influence on Alice but also the users who share the same group with Alice. Bob may even cause more significant harm if Alice is in the ADMIN group because Bob can modify the user and group data of the entire system through the identity of the manager of Alice’s security token. ADMIN’s token gained by the file server 1 and re-assigned to another user will have a severe impact on the whole file-sharing system.

**Mechanism:**

**A close up of text on a white background

Description automatically generated**

Figure 3. T7 Mechanism

The use of the RSA algorithm can reduce the occurrence of the above risks as much as possible. Through the mechanism that the Group server repeatedly checks user’s signature and their request along with the submitted token, the group server can re-design the submitted token’s authority as soon as that the group server detects the unmatching between requester’s signature and the submitted token. This method reduces the possibility that the stolen token can be used beyond the place where theft happens.

**Defense:**

Through sign and verify, the group server effectively determines the requester’s real information and whether the uploaded token belongs to him or her. By securely storing and checking the matching information of each user with corresponding token, the group server reduces the possibility of a token stolen by the file server still gaining capability out of that file server.

**Conclusion**

**Defense for T1-T4:**

RSA indeed contributes to solving these threast. By generating key pairs at the group server side when there is no UserList.bin file and when ADMIN user creates other user, and set the keypairs at UserList.java toward the specific user under above situations, the system is able to check the request information and signature provided by user with the signature and information stored in group server; if user indeed request his or her own token, then the group server provide the token to him or her; otherwise, return error information.

While the user tries to utilize any feature on both file server and group server, the token they provided will be checked under two steps. First, whether the signature information stored in their token matches with the one stored on the group server. Second, whether the detailed information in their token, like issuer, subject, and each group, are matched with the one stored in the group server. If a user's provided token passes the two steps mentioned above, then the authentication to utilize the specific feature will be given. User’s token will be checked every time when a user requests for certain functionality corresponding to complete mediation.

Finally, the threats of unauthorized file servers and data leakage can be remedied by simply using the models described above. We would have the group server authenticate the connections between the client and file servers. Making sure that it is verified through the user list we can still use RSA to encrypt and decrypt the keys to authenticate. With the implementation of T5-T7, the functionality of T1-T4 would not be influenced.