CS 1653: Applied Cryptography and Network Security
Term Project, Phase 4

Lindsey "Hellman" Bieda leb35@pitt.edu
Tucker "Diffie" Trainor tmt33@pitt.edu

April 5, 2012

Introduction: Cryptographic Techniques

The additional threats to our Secure File Server model present new challenges to ensuring the confidentiality and integrity of a user's data. To protect against these threats, we can introduce cryptographic techniques that either augment or supplement existing methods that addressed Threats 1-4.

We will continue to use AES for symmetric key encryption, and to ensure against tampering of AES encrypted messages we will employ HMAC with SHA1 to authenticate messages sent over an encrypted channel. Embedding sequence numbers into messages will also help in detecting replay or reordering by a malicious agent. To combat against file leakage and maintain group privileges over file access, files saved on any File Server will be encrypted with AES, the keys of which will be created and maintained by the Group Server, which will generate them with a reasonably secure pseudorandom number generator and strictly control how they are released to authorized group members.

Threat 5: Message Reorder, Replay, or Modification

Threat Description

Whereas Threats 1-4 encompassed the possible efforts of a *passive* attacker that could only eavesdrop on connections between clients and servers, Threat 5 now introduces the danger of an *active* attacker who is capable of inserting, reordering, replaying, or modifying messages. Though encryption of the communication channel was enough to thwart a passive attacker and maintain confidentiality, an active attacker is able to disrupt the integrity of communications by effectively nullifying messages by tampering with them.

Reordering and replaying of messages can convince either end of a communication channel that the other party wishes to perform an action that is not what is actually intended by that party despite the correctness of the mechanism. An active attacker can

also alter the message itself at the bit level, causing changes that could randomly alter the intention of the message or garble it completely. Modification is not only an affront to the integrity of the Secure File Server model but also can reduce availability if no message can be trusted as to its accuracy.

Mechanism Description

The mechanism builds upon the protocols used in Threats 1-4 that establish a secure session key between a client and a server. Though messages passed in the session are secure from eavesdropping, they are not secure from reorder, replay, or modification. We use two methods to eliminate these new threats: sequence numbers and message authentication.

Sequence numbers are a simple yet effective tool that would alert either end of channel that a message has ben reordered or replayed. We modify our message format so that it includes an integer field to store the sequence number. Then, after receiving a message, the receiving party increments the sequence number by 1 and uses that value in their response. Replays and reordering of messages are easily detectable by either party, as the sequence number will reveal an inconsistency. To prevent a replay of the entire session, we restrict the initiator of the session (i.e., the Client) to choosing the challenge R, and the responder (i.e., the Server) to choosing the random sequence number $\mathtt{t}$.

Message authentication is a cryptographic principal that is used to verify that a message has not been tampered with. We have chosen the HMAC with SHA1 protocol to authenticate messages. To use HMAC in the most secure manner, we need to create an additional shared secret key to use for the keyed hash of HMAC. We can do this at the same time as the creation of the shared secret key for the AES symmetric encryption. With these two keys, $\mathsf{k_e}$ for encryption and $\mathsf{k_a}$ for authentication, we can create an authentication mechanism that addresses Threat 5. The sequence number and message are encrypted with $\mathsf{k_e}$, and then the HMAC is done using $\mathsf{k_a}$ on the encrypted sequence number and message (see Figure 1). The receiver of the encrypted message and HMAC digest will then be able to verify against tampering.

Correctness and Security of Mechanism

The correctness of the mechanism is similar to that of Threat 4: Information Leakage via Passive Monitoring. The creation of the challenge and checking the response is crucial if the Client is to verify that the responding Server is who they claim to be, which is proven by successful decryption of the challenge using the Server's private key. The addition of a sequence number to the mechanism means that both parties must keep track of the incrementation of the number to ensure that replay has not occurred–if either party finds that the number has not been properly incremented (e.g., a $\mathtt{t}$ value of 57 is sent, but a $\mathtt{t}$ value of 59 is returned) then the party must terminate the session. Both parties must also be responsible for using the encryption and authentication keys correctly and consistently

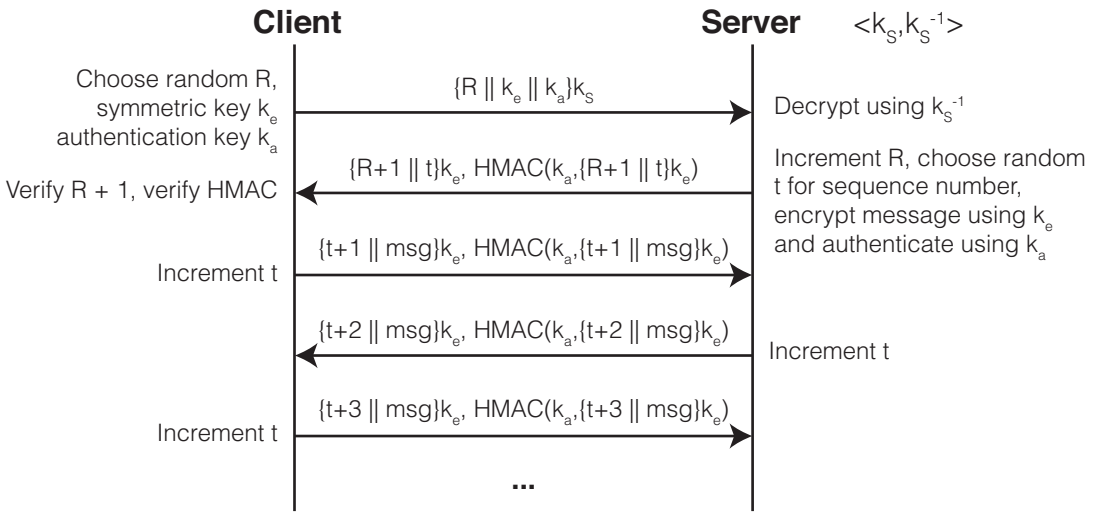

Figure 1: Threat 5 Mechanism

for every message exchange.

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 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 R and the Server to choose t, 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.

Threat 6: File Leakage

Threat Description

The threat of file leakage represents a major vulnerability of the previous set of threats to our model. 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 Description

If groups were limited to just their owners, the threat could easily be eliminated with a single secret key for all files that was only known by or accessible to the owner. However, the fluid nature of groups quickly complicates this ideal and requires a more elaborate mechanism.

Because of how group membership is tied to file access, we use the Group Server to generate and store symmetric secret keys for group files. The Group Server maintains a `Hashtable` object with `<K,V>` parameters `<String groupName, ArrayList<Key> keyList>`. Upon creation of a new group, the Group Server creates a random 128-bit AES key that is stored under the group's name in the hash table. As additional keys for a group are needed, they are randomly generated and added to that group's `ArrayList`.

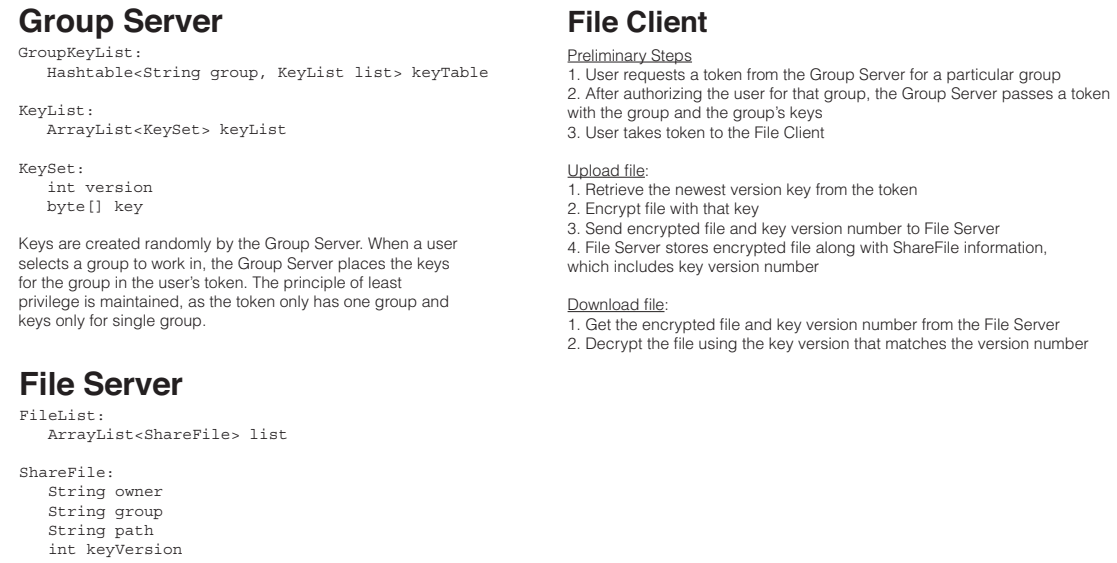

Figure 2: Threat 6 Mechanism

For group members to access a set of keys for use in the File Client, they must authenticate themselves to the Group Server via username/password authentication. 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 is has stored. We modify the ShareFile class to include the key version, which is simply the index of the key in the ArrayList where the group's keys are stored. When a file is uploaded to the File Server, the most recent key (i.e., the key with the highest index number) is used to encrypt the file. 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.

Correctness and Security of Mechanism

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 ShareFile 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 file as it would not know which key to use.

To illustrate the security of the mechanism, we need to fully explain the handling of key version numbers and how it affects forward and backward secrecy as group membership changes. The primary event that creates a new key is when a member is removed from a group. When that occurs, a new key is created by the Group Server and this new key will be used by existing group members to encrypt uploaded files. Let us call the set of keys before the group change V, and the set of keys after the group change V'.

If the expelled group member was generally unmalicious, then when he or she tries to access files from their former group, the Group Server would see that the user was no longer in the group and deny access to the group's key set. Any previous downloads of unencrypted group files cannot be prevented, but future attempts at downloads by a user playing by the rules are thwarted. If the user was more malicious, he or she may have been able to, while still a group member, extract the key set V from their token. With this key set, he or she could decrypt leaked encrypted files from the File Server, but *only* if the file's encryption key was a member of V.

This property illustrates the backward secrecy of the mechanism – without group privilege, the user is no longer able to access or intercept keys of the set $V' - V$. Thus, any file uploaded after the user was expelled from the group is undecipherable by the keys in V.

Forward secrecy is not guaranteed by this mechanism, but can be eventually built if

any of the files encrypted with keys from V are modified, as any upload is encrypted with keys from V' that are not in V.

Threat 7: Token Theft

Threat Description

stub

Mechanism Description

(see Figure 3)

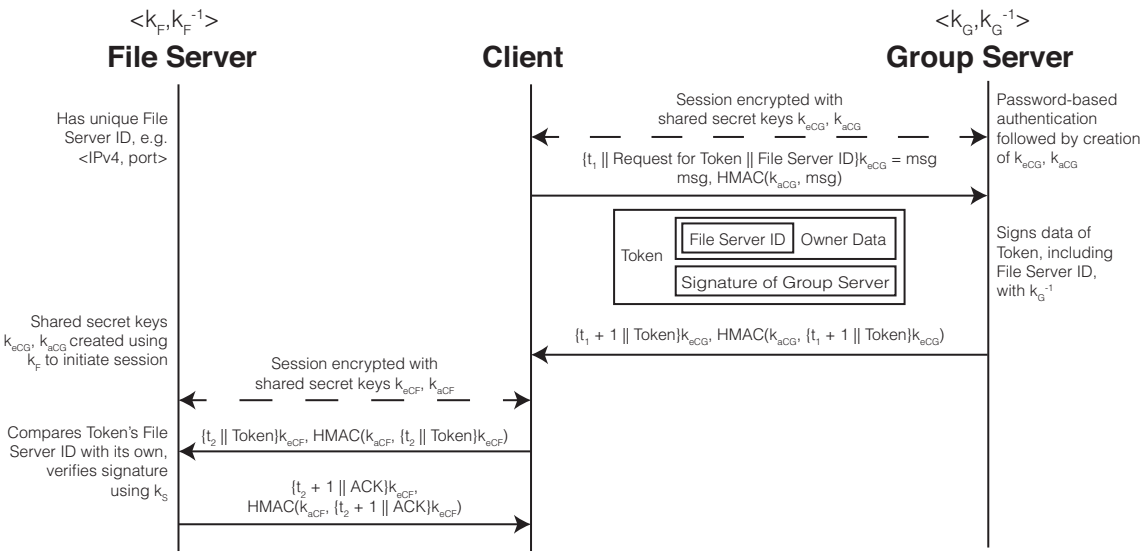

Figure 3: Threat 7 Mechanism

Correctness and Security of Mechanism

Lastly, provide a short argument addressing why your proposed mechanism sufficiently addresses this particular threat. This argument should address the correctness of your approach, as well as its overall security. For example, if your mechanism involves a key agreement or key exchange protocol, you should argue that both parties agree on the same key (correctness) and that no other party can figure out the key (security).

Discussion and Commentary

After completing one section for each threat, conclude with a paragraph or two discussing the interplay between your proposed mechanisms, and commenting on the design process that your group followed. Did you discuss other ideas that didnt pan out before settling on the above-documented approach? Did you end up designing a really interesting protocol suite that addresses multiple threats at once? Use this space to show off your hard work!

Threats 1 through 4 revisited

Finally, spend about one paragraph convincing me that your modified protocols still address the threats T1-T4 described in Phase 3 of the project. Full credit for Phase 4 requires that all Phase 3 threats are still protected against.