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 maintained by the Group Server, which will generate keys by leveraging a combination of an asymmetric key encryption and a hash function against unique properties of a file and group version control.

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.

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, k_e for encryption and k_a for authentication, we can create an authentication mechanism that addresses Threat 5. The sequence number and message are encrypted with k_e, and then the HMAC is done using 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

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).

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

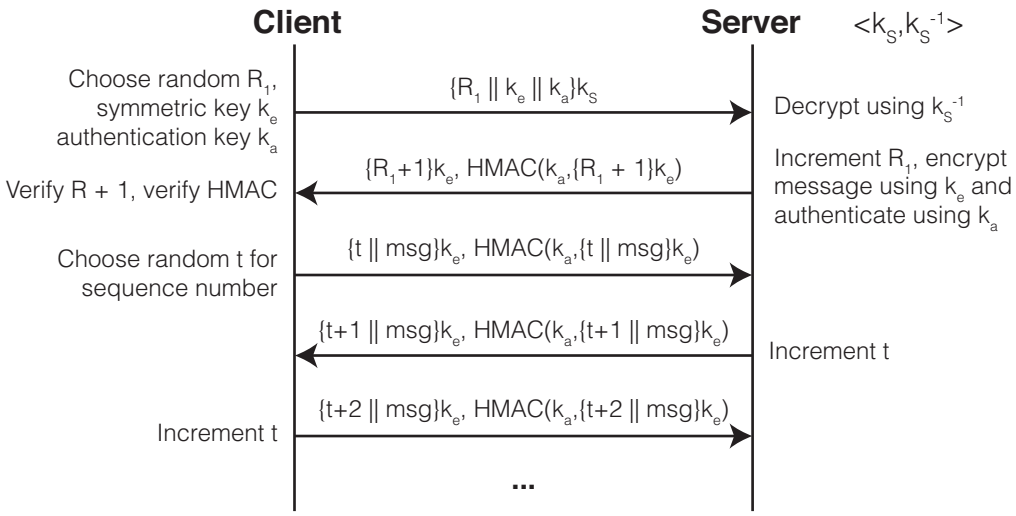

Figure 1: Threat 5 Mechanism

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 group members have access to the files and security from former group members is maintained.

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.

We start with a master version integer V_M, which represents the current iteration of a group. The value of V_M is kept in a hash table (see Figure 2) on the Group Server and its key is the name of the group. As new or modified files are uploaded to a File Server, they are assigned a version number V which is equal to whatever V_M is at the time of upload. The unique properties of the file (e.g. its group, owner, and file path) are used as the key in the hash table to place the value of V. Whenever a file needs to be downloaded, its unique properties are used to retrieve the file's version number, which in turn is used to generate its secret symmetric key for encryption and decryption. The Group Server handles the creation of the symmetric key as well. When the key is needed, the Group Server creates an array of bytes consisting of the unique file properties, concatenates the version number of the file onto that array (either V_M for a new or modified file, V for an existing file), and

Hashtable <String, Integer>

(group || owner || path)	version number
group	master version number
(group || owner || path)	version number
(group || owner || path)	version number
(group || owner || path)	version number
group	master version number
(group || owner || path)	version number
(group || owner || path)	version number
...	...

User request: Group Server Hashtable action
Group created:
    put (group name, 1)
Group adds member:
    no action
Group deletes member:
    put (group, get (group) + 1)
Group deleted:
    remove (group || * || *),
    remove (group)
File upload (new or modified file):
    put ((group || owner || path), get (group)),
    use to generate password for file encryption
File download:
    get (group || owner || path),
    use to generate password for file decryption

Figure 2: Threat 6 Mechanism: Hashtable and Operations

then encrypts that array with its private key k_G^{-1} using RSA. To prevent possible excessive exposure of private key encryptions, the encrypted data is then subjected to a 256-bit SHA hash to provide the final key that will be used for final encryption. The use of 256-bit SHA instead of the 160-bit SHA1 hash is twofold: the extra bits provide increased confidentiality and a 256-bit value is easily usable for 256-bit AES encryption.

Once the key is created, the Group Server can send it over a secure channel to the Client, whose software need not expose it to him/her but simply use it to encrypt the data that is being sent to the File Server (see Figure 3) or decrypt data being received from the File Server (see Figure 4). Thus, the client application and Group Server are the only parties who have any access to the key, leaving a malicious or leaky File Server with encrypted files that are of little use to anyone but authenticated group members.

Correctness and Security of Mechanism

To illustrate the security of the mechanism, we need to fully explain the handling of version numbers and how it affects forward and backward secrecy as group membership changes. As shown in Figure 2, the primary event that changes the master version number V_M is when a member is removed from a group. When that occurs, V_M is incremented by 1, and all subsequent file uploads are encrypted with keys that are generated with version numbers higher than the last version number used by the expelled group member, which we'll refer to as V'.

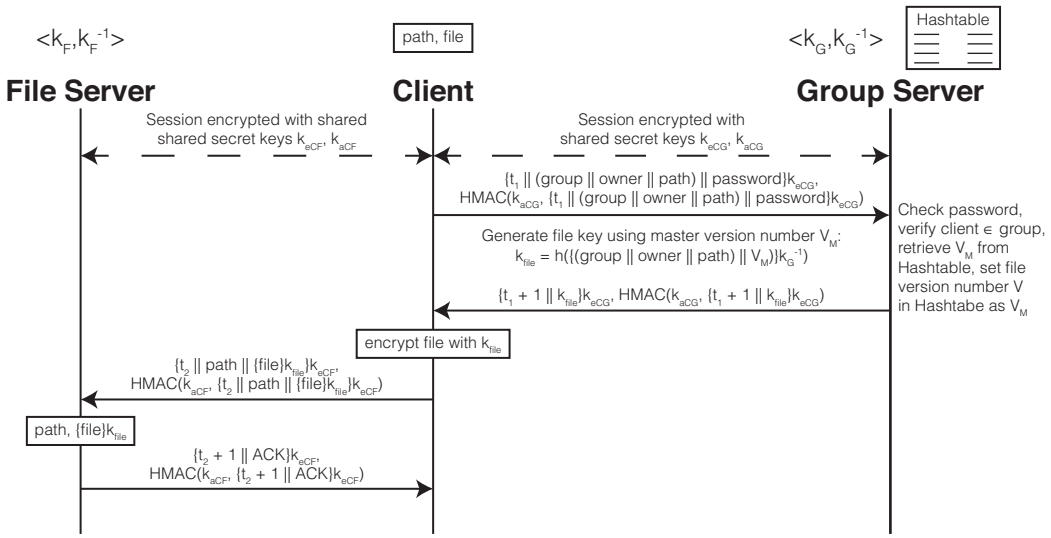

Figure 3: Threat 6 Mechanism: Upload Diagram

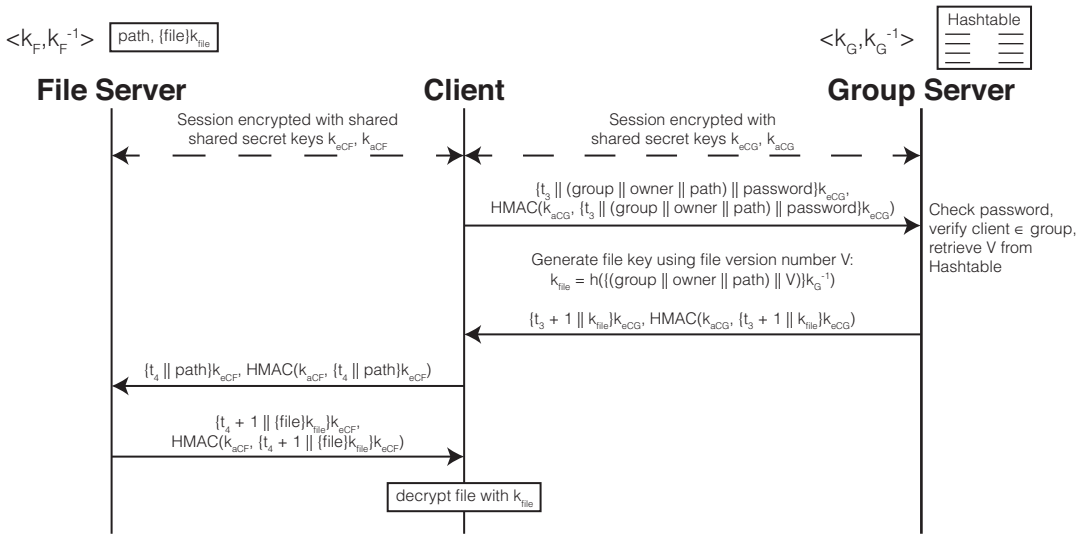

Figure 4: Threat 6 Mechanism: Download Diagram

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 file's key. 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, intercept the key sent from the Group Server to the client application. With this key, he or she could decrypt leaked encrypted files from the File Server, but *only* if the key's version number was the same as the leaked file.

This property illustrates the backward secrecy of the mechanism – without group privilege, the user is no longer able to intercept keys created after version V'. Thus, any file uploaded after the user was expelled from the group is undecipherable by key version V' or any keys of a previous version number.

Forward secrecy is not guaranteed by this mechanism, but can be eventually built if any of the files encrypted with key versions V' or earlier are modified, as any upload is encrypted with V_M.

Threat 7: Token Theft

Threat Description

stub

Mechanism Description

(see Figure 5)

Correctness and Security of Mechanism

stub

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!

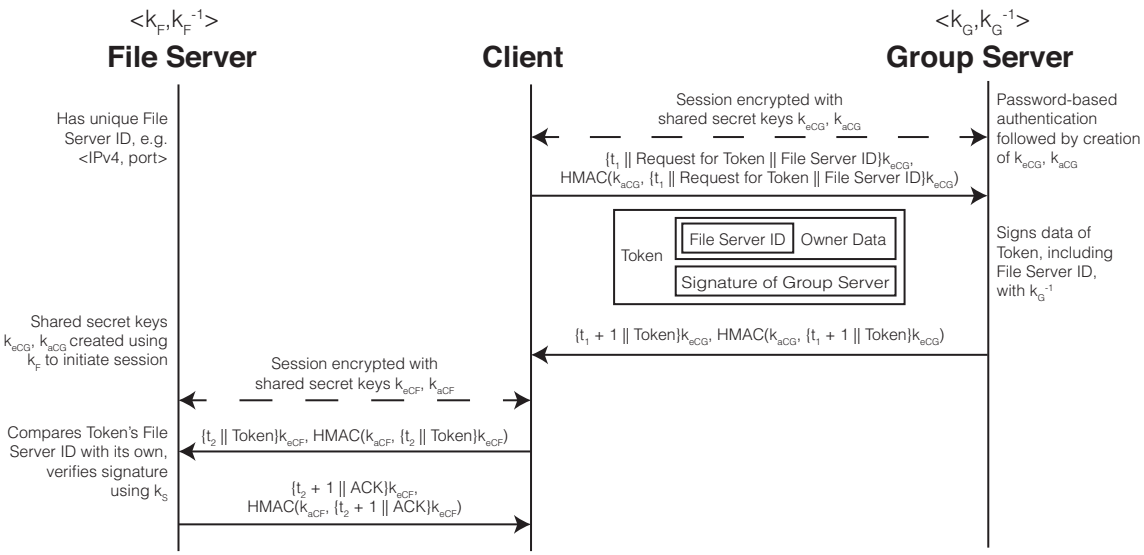

Figure 5: Threat 7 Mechanism

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.