CS 1653: Applied Cryptography and Network Security
Term Project, Phase 3

Lindsey Bieda leb35@pitt.edu Tucker Trainor tmt33@pitt.edu

March 2, 2012

Introduction: Cryptographic Techniques

(an introductory paragraph or two that broadly surveys the types of cryptographic techniques that your group has decided to use to address threats T1-T4)

Threat 1: Unauthorized Token Issuance

Threat Description

Begin by describing the threat treated in this section. This may include describing examples of the threat being exploited by an adversary, a short discussion of why this threat is problematic and needs to be addressed, and/or diagrams showing how the threat might manifest in your groups current (insecure) implementation.

Mechanism Description

Next, provide a short description of the mechanism that you chose to implement to protect against this threat. For interactive protocols, it would be helpful to provide a diagram explaining the messages exchanged between participating principals. (See the notes from Lecture 10 for example diagrams.) Be sure to explain any cryptographic choices that your group makes: What types of algorithms, modes of operation, and/or key lengths did you chose? Why? If shared keys are needed, how are they exchanged?

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 2: Token Modification/Forgery

Threat Description

By modifying or forging tokens issued by the Group Server, a user may be able to gain access to files that would otherwise be forbidden. By changing the group information embedded in a token, a malicious user can access groups that they are not members of, which would grant unauthorized access to files belonging to those groups. Furthermore, if a malicious user assigns him or herself as the owner of a specific group, he or she will have the ability to add or delete group members, as well as freely modify files belonging to that group. This threat is a direct attack on the integrity of a secure group server.

Mechanism Description

In order to maintain the integrity of token issuance, we should verify that a token is valid every time an operation involving a token is invoked. The best way to verify would be to match the token in question with the one on the Group Server. To efficiently perform this mechanism, we take a hash of both tokens, create an authenticated and encrypted channel between the user's application and the Group Server, and compare the hashes. If the hashes match, then the user is using the same token that is on the Group Server and is thus valid.

Correctness and Security of Mechanism

As the hashing process does not necessarily require explanation, the security of the mechanism lies with the verification channel between the Group Server and the application requesting validation. If we can maintain the security of the channel and introduce a degree of randomness to foil replay attacks, then we should be able to eliminate token fraud.

Threat 3: Unauthorized File Servers

Threat Description

The purpose of a secure file server is that, put simply, your files are secure. If a user can unknowingly be directed to an unauthorized file server, any other security safeguards are rendered moot, thus threatening the confidentiality of the user's files and the perceived integrity of the entire service. If a user can be convinced that they are connected to file server s while actually being connected to a malicious file server s', they are at risk of uploading confidential data to an untrusted source or downloading malicious content from an untrusted source.

Mechanism Description

In order for a file server to be trusted by a user, it must provide some authentication to the user that only the actual server can know. Though certificates would be an ideal solution, they are outside the scope of our project. Instead, we can rely on public and private keys as a way of validating a file server. Upon the first connection to a server s, the user will be asked if they wish to accept the public key s_{k}. If the user agrees, s_{k} is stored by the user in their client application along with other identifying details of the server (e.g. server name, IP address, etc.). When the user wishes to authenticate the server, he or she can use s_{k} to encrypt a challenge to s. If s is in possession of the private key s_{k}^{-1}, then s is able to return the challenge to the user and authentication is complete. An unauthorized server s' would be unable to correctly guess s_{k}^{-1} and therefore could not decrypt of the challenge and complete authentication.

Correctness and Security of Mechanism

The main threat to this mechanism is replay attacks, where a passive monitor might see a repeat of a challenge and be able to perform a man-in-the-middle attack. Therefore, a level of randomness in the challenge as well as encrypting the exchange between the user and s may be necessary to ensure correctness and security.

Threat 4: Information Leakage via Passive Monitoring

Threat Description

Suppose Eve can listen to an information exchange between Alice and Bob. Even without being able to interrupt or modify the exchange, Eve can still glean enough information to perform malicious acts. If insufficient security is in place, Eve may be able to gather enough data to

- know the contents of the exchanges;
- to impersonate Alice or Bob;
- use offline password guessing to discover passwords or other secret information.

Eve does not need to be an active participant in a conversation to illicitly benefit from it, and thus exchanges between Alice and Bob must be kept secure.

Mechanism Description

To maintain a secure channel between Alice and Bob during a continuous series of messages, the most efficient solution may be to create a session key between them. A session key avoids having to recreate a secure channel after each exchange, resulting in increased efficiency over

alternatives. To create a session key, we can implement public-key authentication protocols to not only provide authentication but also continued secrecy after the exchanges. By using a Diffie-Helman exchange to create a shared secret key between Alice and Bob, which can then be used to encrypt a session.

Correctness and Security of Mechanism

If the key exchange is performed securely, then each session is not only properly encrypted, but is also protected even if the Diffie-Hellman decryption keys are compromised, as the modulo arithmetic prevents direct decryption, which bolsters the security of the mechanism.

Summary and Errata

(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!)