M.S. Project Proposal

Securing Tuple Space:
Secure Ad Hoc Group Communication Using PKI

Kyle R. Morse
Rochester Institute of Technology
Department of Computer Science
krm4686@cs.rit.edu

Committee:
Chair: Dr. Alan Kaminsky
Reader: Dr. Hans-Peter Bischof
Observer: Dr. Stanislaw Radziszowski

Table of Contents

Overview……3

Project Summary…… 4

Part I: Secure TupleBoard API……... 5

Part II: Ad Hoc Music Distribution System…… 6

Deliverables……8

Project Timeline…… 9

Current Work……10

References…… 11

Overview

Secure group communication in an ad hoc network is a largely unexplored research area. While there exist group key exchange protocols, a recent M.S. project [4] showed that various forms of the Group Diffie-Hellman protocol become computationally infeasible quickly as group sizes increase. Moreover, currently available protocols were not designed to be implemented in an ad hoc network where nodes sporadically enter and leave the group. This project aims to explore establishing secure group communication in an ad hoc network through public key infrastructure.

The project will make extensive use of the tuple space distributed computing paradigm. A tuple space is a store of tuples, or lists of objects, from which consumers may read tuples matching filter criteria and to which producers may post new tuples [1]. While there exist several implementations of tuple spaces in Java (e.g. [2]), they use a client/server architecture which is incompatible with the ad hoc network setting for this project. Therefore I will be using the TupleBoard API [3] which is an implementation of tuple space designed for developing distributed applications in and ad hoc environment.

Public key infrastructure (PKI) provides a framework for establishing and authenticating secure communication between devices. A trusted certificate authority (CA) generates an identifying token, or certificate, for an authorized device. The certificate contains the device's public key and other identifying information and is digitally signed by the CA to prevent forging. This public key may be used to initiate secure communication with a device. In addition, any devices not possessing a valid certificate will be denied from communicating with the rest of the ad hoc group. Details on how this will be applied are discussed further in the following sections.

The market for digital music has exploded in recent years. Unfortunately, much of the media available from online music outlets (e.g. ITunes, Napster) is crippled by the plague of DRM. A major goal of this project will be to create a music distribution network which does not infringe on fair use rights but also provides resistance to piracy.

Project Summary

Currently the TupleBoard API includes only very limited security. The initial goal of this project will be to add support to the existing security framework for additional encryption algorithms, dynamic key exchange, authentication of devices and communication and role-based authorization of device participation in the system. These additions will be discussed in more detail in the next section.

Once the TupleBoard API is completed the next phase of the project will be to design a peer-to-peer (P2P) music sharing application which utilizes this new security architecture. P2P systems have a long and continuing history of litigation with major record labels and artists. The proposed application will provide a secure environment in which peers may only download or distribute media authorized by a certificate authority, in this case the record label. The application will attempt to address the concerns of music piracy through the novel use of public key infrastructure (PKI) in an ad hoc network. The application is described further in the section titled Part II: Ad Hoc Music Distribution System.

Finally the performance of the application and TupleBoard API will be evaluated in both the secure and insecure contexts. Metrics will be developed to compare messaging and computational overhead and their impact on the end user experience. A study will also be conducted comparing the proposed scheme with existing research in the area of secure ad hoc group communication.

The notion of applying PKI to an ad hoc network which lacks client/server infrastructure may seem like a strange idea. However I think it should be an interesting exercise. My goal in this project is to gain experience in developing a flexible and reusable middleware library and to design and implement a secure software system. I feel this project is a perfect blend of two of my major interests - cryptography and music.

Part I: Secure TupleBoard API

The main additions to the TupleBoard API will be support for authorization, authentication and encryption. Authorization will be implemented though the concept of roles. A device may only be permitted to post a tuple which is permitted by its role. Similarly a device may only read or take tuples which its role permits. To put this in my application context devices may only share or download media for which they are authorized. Currently there are no provisions to support this. The ability to authenticate tuples is also an important security feature. Authentication of a tuple will be performed by verifying its digital signature. In this manner it can be proven that a tuple's contents are intact and that it was posted by the expected device. Finally, the addition of encryption allows all network activity generated through tuple board interaction to be protected from capture by unauthorized devices.

The first task in applying public key infrastructure to the tuple board will be to create a certificate authority (CA). The CA is a trusted entity which has the task of generating certificates for devices. The certificate should contain information which identifies the device (e.g. serial number, username, etc…), expiration date, the device's authorizations and the necessary cryptographic keys to support the proposed security schemes. The certificate may also contain additional application data. This certificate is digitally signed by the CA before distribution to the device so its contents may be verified as authentic.

Once the CA is complete the next step will be to add additional *SecurityContext* objects. All security operations in the tuple board are implemented through these security contexts. When a tuple is posted it is processed by the active security context. Similarly when a tuple is read it is again processed by a matching security context object. Currently the TupleBoard API defines two of these. The first is a *DefaultContext* which performs no security functions. The second is a *FixedHelixSecurityContext* which provides symmetric encryption capabilities using a static, predefined key.

A *PublicKeyContext* will need to be defined in order to support encryption using a public key system such as RSA. This context will be used when exchanging session symmetric keys with new devices. In addition a *SymmetricKeyContext* will be defined to support encryption using a symmetric key system. This context will be used for all file transfer tuples and any additional device interaction within a file sharing session. Both of these new contexts will also rely on new tuple authentication and authorization modules. The authentication module will use digital signatures to verify a tuple being read has not been tampered with and was posted by the expected device. The authorization module will prevent reading or posting of tuples which a device's certificate does not allow. Further details on how and where these various new contexts will be used in the application can be found in the next section.

Part II: Ad Hoc Music Distribution System

Once the TupleBoard API is secured I will develop a P2P music sharing application based upon it. The system will be designed to provide authorized access to DRM-free digital music in an ad hoc network. Devices authorized as providers will advertise media available for download which in turn will be consumed by authorized subscribers.

In order to participate in an ad hoc music sharing session a device must first register with a record label. The record label will be an extension to the CA developed in the TupleBoard API and will have the ability to generate certificates which additionally specify the media a particular device will be allowed to transfer. In addition it will support two roles – provider and subscriber. Providers will be permitted to redistribute downloaded content while subscribers will only be able to download. These roles will be enforced through the authorization functionality added to the TupleBoard API.

Once a device has obtained a valid certificate it will be able to participate in the music sharing system. The first step necessary is to obtain the session communication key. Upon starting a sharing session a device will first check to see if any other devices are active. If no session is found the device will generate a random symmetric session key to be used by the active *SymmetricKeyContext* when reading and posting tuples. Otherwise the device will post a session start tuple using the default security context which will be interpreted as a request for the session key currently in use. As with all tuples this request will first be verified to have come from a device possessing an authentic certificate. The tuple will be rejected if it was generated by a malicious device. This authentication step prevents malicious devices from obtaining the session encryption key and therefore being able to capture file data transferred between devices.

If the request is authentic a device will post a session join tuple as a reply which contains the current session encryption key. This tuple will be encrypted using a *PublicKeyContext* before being posted. The new device may then decrypt this tuple, again using a *PublicKeyContext* object, to obtain the symmetric session key for communicating with other devices in the system. All tuples posted and read from this point on will be encrypted and decrypted using a *SymmetricKeyContext* object keyed with the newly obtained session key.

Now that the device has a fully initialized instance of a *SymmetricKeyContext* object it may participate in the system. If the device is a provider it will post an encrypted tuple containing a list of the media it is authorized to share. Otherwise it will read all subscriber media list tuples in order to determine what music is currently available from other devices in the network. The final step is to actually transfer digital music files securely between authorized devices.

In order to obtain a file from a subscriber a provider must first post a file request tuple. This tuple will then be read by a provider. The request will be rejected by the provider if the subscriber's certificate does not authorize download of the requested file. Otherwise the provider will post a media file tuple containing the requested file. If the provider is not authorized to share the requested file the tuple will be rejected. Otherwise the subscriber will read the tuple and thus download the file. Note also that any file or request tuples

which are unencrypted or encrypted using an incorrect session key will be rejected with the assumption they were posted by malicious devices.

The system should also have the ability to change symmetric keys periodically in order to thwart any brute force cracking attempts by malicious users. This key refresh could be performed by a device posting a key change tuple. All other devices currently in possession of the current session key could read and decrypt this tuple and update their session key. This new key would not be able to be read by devices lacking the current session key. In addition, any key change requests would be rejected unless their originator is an authentic device.

The described architecture and tuple flow provides a secure music distribution platform for an ad hoc network. All communication to and from malicious devices is rejected since they do not possess a valid certificate and are unable to obtain the session symmetric key. As with any system built upon public key infrastructure trust is the cornerstone of the security policy. It is therefore of utmost importance for a CA to only distribute certificates to devices known to be protocol abiding and trustworthy.

Deliverables

Upon completion of this project the following deliverables will be presented:

1. Secure TupleBoard API containing:
    a. Certificate authority
    b. Additional security contexts
    c. Message authentication
    d. Role-based authorization of actions in tuple space

2. Secure Ad Hoc Music Distribution Application

3. Final Report containing:
    a. Detailed API design
    b. Detailed application design
    c. Summary of implementations
    d. Performance study
    e. API developer's manual
    f. Application user's manual
    g. Comparison with related work
    h. Future work

Project Timeline

The following is a tentative schedule for the completion of major phases of this project:

Proposal Completed	02/23/07
Background Research	03/12/07
TupleBoard API Design	03/19/07
API Implementation & Testing	04/09/07
Certificate Authority Implementation	04/16/07
Ad Hoc Music Sharing Application Design	04/23/07
Application Implementation & Testing	05/08/07
Project Final Paper	05/18/07
Project Defense	05/25/07

Current Work

I have been meeting with Professor Kaminsky on a weekly basis in order to discuss background details of this project. I have spent some time getting familiar with the current capabilities and usage of the TupleBoard API and have started to plan what will need to be added to it to support secure group communication. By the end of this quarter I plan on having a firm idea of exactly how my application will utilize the TupleBoard and want to be ready to start full fledged design and implementation at the beginning of March.

I have posted a website to keep track of my progress as I continue work on this project: http://www.cs.rit.edu/~krm4686/msproj/index.html

References

[1] Gelernter, D. 1985. Generative communication in Linda. *ACM Trans. Program. Lang. Syst.* 7, 1 (Jan. 1985), 80-112. DOI= http://doi.acm.org/10.1145/2363.2433

[2] IBM's TSpaces: Technology Overview. (Aug. 2003). http://www.alphaworks.ibm.com/tech/tspaces

[3] Kaminsky, Alan. TupleBoard Library. http://www.cs.rit.edu/~ark/tb.shtml

[4] Kim, Jisoo. "Group Key Agreement Protocols with Implicit Key Authentication." RIT Computer Science M.S. project, July 2005. http://www.cs.rit.edu:8080/ms/static/spr/2004/4/jsk4445/index.html