Security Broker for Multimedia Wireless LANs: Design, Implementation and Testbed

Aura Ganz*, Se Hyun Park*, and Zvi Ganz**
* Multimedia Networks Laboratory
ECE Department, University of Massachusetts
Amherst, MA 01003
ganz@tikva.ecs.umass.edu
** AIM Engineering Inc.
zvi@aime.com

Abstract— To secure interactive multimedia applications in wireless LANs (WLANs) it is pertinent to implement a number of security services such as authentication, key exchange and real-time encryption/decryption. WLANs, though, present a complex and challenging environment for implementing such security services since these services may deplete the limited network resources and increase the burden of supporting quality of service for multimedia applications. Consequently, a broker is needed to mediate proper security considering inputs such as user security requirements, user security literacy, available network resources, and security routines performance. In this paper we introduce a Security Broker that we have designed to fulfill these complex mediation needs. This broker is implemented in software and tested in a wireless LAN testbed. The reported security broker design and implementation considers the wireless LAN environment as well as the multimedia applications' quality of service requirements such as delay and throughput. We also introduce an Inline encryption/decryption software that encrypts/decrypts traffic on the fly. Our experiments show that in cases where relatively modern computers are used with properly chosen standard encryptors, then such computers can sustain the throughput and security requirements of interactive multimedia applications for WLANs such as telephone-quality audio, video conferencing, and MPEG video.

I. INTRODUCTION

Wireless local area networks (WLANs) are becoming a viable complementary solution to wired networks. WLANs provide support for wireless residential and business needs where cables are not cost-effective or prohibitive. Such WLANs are expected to carry legacy data traffic as well as video, audio and other multimedia applications. Transmissions over the open air carry a significant security risk for users since unauthorized users can intercept such transmissions or malicious attackers can penetrate the WLAN by impersonating legitimate users.

WLANs designers face a number of challenges dictated by users' requirements to transfer multimedia applications that need predetermined Quality of Service support as well as requirements for secured information delivery. These challenges are intensified by the WLAN poor channel quality, dynamic network behavior, limited available bandwidth and relatively low computational power stations.

In this paper we introduce the concept of a Security Broker that mediates proper security services in a WLAN environment. This broker provides security services such as authentication, re-authentication during the session, real-time encryption that encrypts packets arriving from the application and heading to the network and real-time decryption of traffic entering from the network. The real-time encryption/decryption service, denoted as the INLINE SECURITY LAYER, is implemented in the Service Layer Provider and uses Microsoft's CryptoAPI [4]. To ease the mediation process between the user and the network, we introduce the security services classification criteria and three security classes. Different designers may reflect their users needs with different classifications. We tested the security services on an experimental testbed comprised of PCs with Windows operating system. From our measurements we conclude that commercially-accepted standard software encryption routines are suitable for multimedia applications in WLANs. Moreover, it is cheaper than hardware implementation and more flexible for upgrades.

The paper is organized as follows. In the next section we introduce the concept of the Security Broker, its security services and security classes. In Section 3 we present the software implementation of the Security Broker and its graphical user interface. In Section 4 we describe the experimental testbed, the testing procedures and the end-to-end experimental results of the inline security layer. Section 5 concludes the paper.

II. SECURITY BROKER

The Security Broker design was guided by the following goals:

- easy to use by security-illiterate users as well as security-literate users.
- feasible to implement on a Windows operating system platform
- can be used by commercial off-the-shelf standard applications, (i.e., do not need to require security aware applications).
- possible to use standard encryption functions (e.g. as provided by CryptoAPI)
- provides the necessary security functions such as authentication and inline encryption
- easy to expand and upgrade
- low cost of ownership
- independent of the wireless network interface cards
A number of security mechanisms are required to effectively manage resources and provide a suitable security procedure for each connection. We have developed and implemented the following security features (see Fig. 1):

- **Authentication**: assures that the identity is not false.
- **Re-Authentication**: provides enhanced robustness during an established session.
- **Encryption/Decryption (Confidentiality)**: secures that the information is accessible only to authorized parties.
- **Integrity**: ensures that an intruder should not be able to substitute a false message.
- **Graphic User Interface (GUI)**: provides ease of use.
- **Security Classes**: defines the level of security according to users' requirements.

We differentiate between two main concepts: pre-establishing security and post-establishing security.

The pre-establishing security mechanisms are the **Authentication Protocols** we have published in [1]. These protocols were designed for the WLAN characteristics such as limited bandwidth and limited computational power. Our authentication protocols provide a unique session key and, therefore, reduce the total number of expensive computations such as public key encryption.

The post-establishing security mechanisms we have developed are the **Re-Authentication Protocol** [3] and the **Inline Security Layer** [2]. Upon successful conclusion of the authentication protocol, these mechanisms are used to perform real-time encryption and re-authentication during the session. The goal of **Re-Authentication Protocol** is to obtain a low computational complexity authentication and key exchange procedure which provides enhanced robustness in face of active and passive security threats. The re-authentication which is executed during the session, is implemented by an one-way handshake protocol with the agreed-on certificate parameters.

**Criteria for Classification**

The main objective of the **Security Broker** is to support continuous security service for MWLANs. This purpose should be considered with respect to user types, application properties, and terms of attacks. Therefore, we introduce a number of criteria for classifying the security levels:

The first criteria is based on three main goals of cryptography (confidentiality, integrity and authenticity) to prevent security attacks. We consider the following categories of attack:

- An unauthorized party gains access to an asset (interception). This is an attack on confidentiality.
- An unauthorized party not only gains access to but tampers with an asset (modification). This is an attack on integrity.
- An unauthorized party inserts counterfeit objects into the system (fabrication). This is an attack on authenticity.

Second, more advanced security aspects might be encountered to prevent passive and active attacks from a different angle by combining new security mechanisms of the security broker. These aspects might be engaged in the security levels by looking at the generic types of attack. These attack types involve replay which implies that intruders in the same service area actively capture a data unit and subsequently retransmit to make an unauthorized effect. Another general type of replay attacks is the modification of message to convert the meaning of original messages.

Third, the classification of security needs to provide a wide scope for various users. In the viewpoint of security matters, casual and security illiterate users are usually not aware of security risks. Since these types of users do not properly understand the security services that ensure protection of their information, they may be frustrated if required to set up security parameters. This aspect is taken into account in the security levels.

**Proposed Security Classes**

In this design we propose three security classes which determine security services and which can be easily selected by users. Other designs may consider different classes. Table 1 provides a summary of the security classes.

Most of security aspects such as confidentiality, integrity and authenticity have been considered in **Class 3**. This is the highest security. This class is geared for security literate users that have to make two decisions. First, the user must determine if the security parameters are suitable for the application in particular circumstances. Second, even if the remote side is only capable of accepting agreed-on parameters, the sending user must decide whether these parameters can be used for the requested session. For example, in our implementation, the security service specifications of **Class 3** (Table 1) recommend the 160-bit SHA, 128-bit MD5 with **authentication protocol**, **re-authentication protocol** and **inline security layer (bulk encryption with RC4)**.
For Class 2, the advanced security service such as “Re-
Authentication Protocol” is optionally supported pending
users’ request. This option might reduce the complexity
of the security procedure for users, but may be vulnera-
ble to the time-based information analysis. We have im-
plemented in Class 2 two types of hash functions MD2
and MD4. This class takes into account the most impor-
tant security aspects including the service against Replay
with the pre-establishing and the post-establishing mech-
anisms.

Class 1 provides instant access to security illiterate
users that do not request any encryption services. This
class may establish a wireless connection using weak se-
curity services, since both post-establishing mechanisms
(re-authentication protocol and inline security layer) are
optional. Therefore, if the quick connection is the main
reason, the security broker allows the user to perform any
applications just based on the initial authenticity.

III. SOFTWARE IMPLEMENTATION

The Security Broker is implemented in software and
runs on the Microsoft’s Windows operating system. The
functions of Security Broker are implemented in both
the application layer and the service layer provider above
TCP/IP. Fig. 2 shows the entire architecture of the soft-
ware security broker. The security broker invokes the
security services through the Graphic User Interfaces.

The authentication and re-authentication protocols are
implemented at the application layer with shared mem-
ory between these applications and the inline encryption
module.

The software implementation of the inline security layer
was described in details in [2]. We implemented our soft-
ware encryption algorithms using CryptoAPI [4] as crypto
service functions in WinSock 2 Layered Service Provider
that resides between the principal WinSock 2 DLL and
the lower-level base transport provider (see Fig. 2). At
the source, each packet that is generated by the applica-
tion is intercepted at the layer service provider, is en-
crypted using CryptoAPIs and is sent down to the base
transport protocol. At the destination the packets that
belong to this application are intercepted by the layer ser-
vice provider, are decrypted using CryptoAPIs and are
forwarded to the application.

The proposed architecture has the following advantages:

- we can use any off-the-shelf applications, i.e., the ap-
  plications do not need to be modified and be made
  aware of the fact that we use encryption techniques.
- we can use any off-the-shelf network interface card,
  i.e., we do not need to choose a network interface
card that incorporates encryption capability either in
  hardware or software.
- easy to upgrade the encryption algorithms.

A. Graphic User Interfaces (GUIs)

In this subsection, we present the GUIs of the security
mechanisms (Authentication Protocol, Re-Authentication
Protocol and Inline Security Layer). The main objective
of employing a graphic user interface is to resolve conflicts
between complex security procedures and ease of use. All
GUIs are programmed in C++ for Windows 95 environ-
ments. Each GUI follows the security services that we
described.

The security broker GUI shown in Fig. 3 supports three
main functions: set the client server role, set the class
level and security parameters, and open the gate of the
authentication protocol. We describe next the GUI for
each security service.

<table>
<thead>
<tr>
<th>Security Classes</th>
<th>CLASS 1</th>
<th>CLASS 2</th>
<th>CLASS 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Aspects</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption (Confidentiality)</td>
<td>possible or difficult</td>
<td>difficult</td>
<td>very difficult</td>
</tr>
<tr>
<td>Modification (Integrity)</td>
<td>possible</td>
<td>difficult</td>
<td>very difficult</td>
</tr>
<tr>
<td>Authentication (Availability)</td>
<td>possible</td>
<td>difficult</td>
<td>very difficult</td>
</tr>
<tr>
<td>Adv. Security Aspects</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-provision</td>
<td>possible</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Secure Authentication</td>
<td>very</td>
<td>moderate</td>
<td>low</td>
</tr>
<tr>
<td>User Type (Security level)</td>
<td>very</td>
<td>moderate</td>
<td>low</td>
</tr>
<tr>
<td>Performance</td>
<td>higher</td>
<td>moderate</td>
<td>moderate</td>
</tr>
</tbody>
</table>

Fig. 2. Security Broker Software Architecture
A.1 Authentication Protocol

The Authentication Protocol GUI is shown in Fig. 4. The GUI consists of graphical and textual inputs as well as an output message box. The user needs to choose receive mode or send mode before starting the authentication protocol. If Send Mode was chosen, the IP address of the remote side must be specified. The agreed-on initial Session Key should also be provided. The Message Box displays the progress of the authentication protocol.

The Authentication protocol uses a three way handshake procedure to validate the remote side. This protocol is performed with shared information such as a session key, an encryption type, a hash function type and a class level. The client side will accept and use the security parameters provided by the server side. If this obligation is satisfied and the proper authentication protocol is executed by both sides, the Message Box of the authentication protocol GUI displays ‘AUTHENTICATION: OK’. If the authentication protocol can not verify the remote side, the Message Box of the authentication protocol GUI displays both the failed step and the message ‘AUTHENTICATION FAILED!!!’.

A.2 Re-authentication Protocol

Upon successful conclusion of the authentication protocol, the Re-Authentication Protocol, we have published in [3], can be executed. For the ‘Client Side’, the GUI of the Re-Authentication Protocol (Fig. 5) is invoked. This GUI contains three main textual inputs to define the re-authentication parameters, e.g., the number of times we want to execute the re-authentication procedure and time intervals between consecutive re-authentication executions.

The main types of message boxes for the re-authentication protocol are ‘Re-Authentication SUCCESS!!’ and ‘WARNING!!’. In case of successful re-authentication, the message box shows the number of times we wish to perform the re-authentication procedure.

IV. EXPERIMENTAL TESTBED

In this section, we introduce the test platform of the Security Broker, testing tools and experimental results of the inline security layer.

Experimental Setup: The test platform involves two Windows 95 computers (one server and one client) equipped with wireless LAN adapters (AT&T’s WaveLAN ISA Card). The server side uses Pentium 166MHz MMX system, and the client side uses Pentium 90MHz system. The Client runs a number of applications like WSFTP Pro from Ipswich Inc. and Microsoft’s Media Player. The Server runs Microsoft’s FTP and HTTP server software. We set up the test platform in a semi-open office environment with approximately three meters distance between the computers.

Test Tools: The first tool is the Windows Sockets (WinSock) Configuration Tool. This tool is included in the WinSock 2 development kits and called ‘Sporder’. The main function of this tool is to display the installation of WinSock 2 Service Provider. We use this tool to show the status of our Inline Security Layer (or Security Layered
Service Provider (SLSP)). The second tool we use is EtherPeek which is developed by the AG Group, Inc. EtherPeek monitors (or snoops) the network data and helps us determine the correctness and performance of the protocols we have developed. This network tool can be viewed as a virtual intruder.

**End-to-End Experimental Results:** We introduce experimental results of a number of applications such as FTPs and Microsoft's Media Player. The available CPU resources are measured by the System Monitor program provided with Windows 95.

**Experiment 1:** Fig. 6 provides FTP throughput measurements versus different values of available CPU resources at the beginning of the experiment using WaveLAN. We use WSFTP Pro from Ipswitch, Inc.. We notice that in case both computers M1 and M2 have available CPU resources (Case 1) we obtain the same average end-to-end throughput of 1.65 Mbps which is dictated by the wireless LAN bandwidth. In Cases 2-4 M1 or M2 do not have enough available CPU resources and therefore we observe lower end-to-end average throughput in both cases, with and without the security layer. In Case 2 with the secure layer we obtain a lower throughput (401 Kbps) than without the security layer (920 Kbps) since encryption requires M1 CPU cycles which are already very highly utilized. Now the system bottleneck is at the encryption throughput which is dictated by available CPU resources at M1.

**Experiment 2:** In this experiment we use Microsoft's Media Player. The throughput measurements are obtained by DU Meter software by Hagel Technologies Inc. Media Player application displays at the client (M2) movie files retrieved from the server (M1). As opposed to FTP, this application requires significant CPU resources at the client to decompress the file. This fact is reflected in throughput measurements in Fig. 7. As can be seen from Case 1 in Fig. 7 which has the same initial conditions as Case 1 in Fig. 6 when we use encryption, we observe a significant decrease in performance (520 Kbps). This is due to the fact that there is competition for resources (0 available CPU resources during the experiment) at the client M2 which slows down the decryption process. In Case 2 we observe that for both cases in which we invoke or do not invoke the security layer we obtain very low throughput of 60 Kbps. In this case the bottleneck is created at the server (M1) which performs very slow processing of the file.

**V. CONCLUSIONS**

We have developed and implemented a Security Broker in the Windows platform that mediates and simplifies the daunting task of coordinating network resources, security needs and executed applications. The inline encryption was implemented in software in the Layer Service Provider and integrates CryptoAPI's crypto service functions. Using off-the-shelf FTP, and Microsoft Media Player applications, we have measured the suitability of software inline security services in wireless LANs. We draw a number of conclusions: 1) software inline encryption throughput is a function of the available computer resources such as available CPU resources and CPU speed, 2) inline encryption can limit the application throughput, 3) an application can decrease the encryption throughput in case it requires significant computer resources such as decompression, and 4) in case sufficient computer resources are available, software inline encryption is a suitable solution for wireless LANs that support multimedia applications.

**REFERENCES**


