Asian Journal of Information Technology 5(8): 834-847, 2006

Transparent Security System for Lan Communications

Sufyan T. Faraj and Firas R. Barjas
Department of Computer Engineering, College of Engineering,
University of Baghdad, Iraq

Abstract: Many attacks may be carried out against communications in Local Area Networks (LANs). However, these attacks can be prevented, or detected, by providing confidentiality, authentication and data integrity services to the exchanged data. This study introduces a transparent security system fOr LAN communications (TSSOLAC) that protects the communications in a LAN of Microsoft Windows machines from possible security attacks. On each host in the protected LAN, TSSOLAC transparently intercepts each outbound IP (Internet Protocol) packet and inserts a crypto header between the packet IP header and payload. This header is used to detect any modification to the content of the packet in transit and to detect replayed packets. Then, the system encrypts the IP packet payload and some fields of the inserted crypto header. On the other hand, the system transparently intercepts each inbound IP packet, decrypts its encrypted portions and then uses its crypto header to authenticate the packet. If the packet is properly authenticated, the system indicates it to upper protocols. To be transparent to applications, the TSSOLAC part that processes inbound and outbound IP packets is implemented as a NDIS (Network Driver Interface Specification) intermediate driver that resides between the Logical Link Control (LLC) and Medium Access Control (MAC) data link sublayers. The study presents the design, implementation and operation of TSSOLAC.

Key words: LAN, security, encryption, authentication, NDIS intermediate driver

INTRODUCTION

Networks have become a very important part in life. The services networks introduce to governments, businesses, companies and academic organizations cannot be dispensed. Transferring information without the use of networks can be a nightmare for everybody. Despite these facts, transferring information by computer networks allows hackers to carry out many attacks against communications in these networks. Some examples of these attacks are eavesdropping, masquerading, replay attacks, Denial of Service (DOS) attacks and active change to the exchanged information. Apparently, these attacks may result in loss of money, secret information and privacy.

Here comes the importance of networks security to protect communications in networks from the different types of attacks by providing different security services. The most important types of these services are confidentiality, authentication and data integrity.

Generally, security services are implemented by different types of security mechanisms and techniques, which are used to detect or prevent security attacks. Encryption is the security mechanism that can be used to provide confidentiality, whereas hash functions or Message Authentication Codes (MACs) can be used in different ways to provide authentication and data integrity.

The purpose of this study is to introduce the design, implementation and operation of the transparent security system for LAN Communications (TSSOLAC), which can be used to provide confidentiality, authentication and integrity to the data exchanged within a Local Area Network (LAN) of Microsoft Windows machines, so that attacks against the exchanged data are prevented (or detected). TSSOLAC provides the security services transparently, i.e. there is no need to change software on a user computer that benefits from the provided security services. Users can continue to use their usual network applications (without any change) while TSSOLAC provides security services to the exchanged data.

Corresponding Author: Sufyan T. Faraj, Department of Computer Engineering, College of Engineering, University of Baghdad, Iraq

SYSTEM SPECIFICATIONS AND DESIGN APPROACH

The following points shed the light on the specifications of the proposed TSSOLAC:

- TSSOLAC provides confidentiality to the exchanged data.
- TSSOLAC provides authentication and integrity to the exchanged data.
- TSSOLAC provides access control.

If received data is not properly authenticated or if it is replayed data, TSSOLAC collects audit information about it and produces an alert to the network security administrator indicating the reception of such invalid data along with the collected audit information. In turn, the network security administrator could monitor the audit information and take an appropriate action to prevent a possible attack that might cause the reception of the unauthenticated or replayed data.

- TSSOLAC is able to prevent DoS attacks launched by hosts unknown to it.
- TSSOLAC is transparent to applications.
- TSSOLAC does not affect the work of routers that may exist in the protected network.
- TSSOLAC is able to protect itself from attacks.
- Due to the rapid increase of networks, many people who are not so experienced in network security have become network administrators. For this reason, TSSOLAC provides a friendly and easy to use interface for them.

To achieve the features and specifications stated above, the following choices were adopted:

- Microsoft Windows 2000 was chosen as the platform under which TSSOLAC works for its reliability, high performance and security.
- TCP/IP (Transport Control Protocol/ Internet Protocol) was chosen as the transport protocol for the protected LAN.
- The LAN protected by the proposed TSSOLAC was chosen to be Fast Ethernet. TSSOLAC assumes that Ethernet frames encapsulate IP packets according to the Ethernet II (DIX Ethernet) encoding; because by default, Microsoft Windows 2000 TCP/IP stack transmits Ethernet frames using this encoding [1].

The Network Driver Interface Specification (NDIS) Intermediate Miniport (IM) driver was chosen to apply the required transparent security processing to inbound and outbound network traffic. NDIS IM driver was chosen for the following reasons:

1. It is well documented.
2. It is a kernel mode driver and all incoming or outgoing packets should pass through this driver. This feature prevents network packets from bypassing the security policy enforced by TSSOLAC.
3. The NDIS IM driver lies below the network layer, i.e. between the Logical Link Control (LLC) and Medium Access Control (MAC) data link sublayers [2]. This feature gives NDIS IM drivers a lot of control over network packets, without affecting other network protocol stack components. As an example, the position of NDIS IM drivers make them appropriate to be used to authenticate the source IP address of received IP packets.

The transparent security processing performed by the TSSOLAC NDIS IM driver mostly follows the way in which the IPSec protocol processes network packets, but it avoids some problems that exist in the IPSec design. These problems are (items 1-3 are quoted from [3]. Item 4 is quoted from [4]):

1. "IPSec is too complex to be secure. The design obviously tries to support many different situations with different options. The number of major modes of operation can be drastically reduced without significant loss of functionality. IPSec is well beyond the level of complexity that can be analyzed or properly implemented with current methodologies. Thus, no IPSec system will achieve the goal of providing a high level of security" [3].
2. "The ESP (Encapsulating Security Payload) protocol allows the payload to be encrypted without being authenticated. In virtually all cases, encryption without authentication is not useful. ESP should be modified to always provide authentication; only encryption should be optional" [3].
3. "When both encryption and authentication are provided, IPsec performs the encryption first and then authenticates the ciphertext. This is the wrong because going by the "Horton principle", the protocol should authenticate what was meant, not what was said. The meaning of the ciphertext still depends on the decryption key used. Authentication should thus be applied to the plaintext and not to the ciphertext" [3].
4. "The specifications of IPSec allow predictable -but random- Initialization Vectors (IVs) to be used in IPsec ESP encryption and explicitly allow the

common practice of using the last ciphertext block of encrypted data from an encryption process as the IV for the next encryption process. Predictable initialization vectors compromise IPsec confidentiality. By using an adaptive chosen plaintext attack, an attacker can break low entropy plaintext blocks using brute force and confirm guesses of the contents of arbitrary plaintext blocks. However, the preconditions of this attack are restrictive and the vulnerability is thus difficult, but probably not impossible, to exploit in practice" [4].

IPSec is used to secure IP packets only. Packets of transport protocols other than TCP/IP (IPX/SPX-Internet Packet Exchange/ Sequenced Packet Exchange- for instance) are not secured by IPSec. Hence, if IPSec is used to provide control over access to the hosts of a network, there will be no access control at all if transport protocols other than TCP/IP are installed on the hosts of that network because packets of these protocols are not affected by IPSec.

The developed TSSOLAC solves all of the problems mentioned above. It avoids the complexity and many modes of operation inherent in the IPSec protocol, it always provides authentication and encryption for IP packets, it authenticates the plaintext instead of the ciphertext and it uses a random IV for each IP packet.

TSSOLAC can be used to secure the packets of transport protocols other than TCP/IP protocol as it works below all transport protocols. However, the current version of TSSOLAC prevents communications using transport protocols other than TCP/IP protocol, but it provides the administrator an option for turning this feature off. The current version of TSSOLAC does not provide automatic key exchange. Future versions must implement robust key exchange protocols.

TSSOLAC ARCHITECTURE AND COMPONENTS

TSSOLAC consists mainly of the following three components:

- The Security Descriptors List (SDL), which stores security descriptors corresponding to hosts in the protected network.
- The Cryptographic IM Driver (CIMD) that applies the transparent security processing to inbound and outbound IP packets.
- The Security System Manager (SSM), which is a Win32 application that allows the network security administrator to control the operation of the CIMD and alerts him (or her) if a suspicious packet is received.

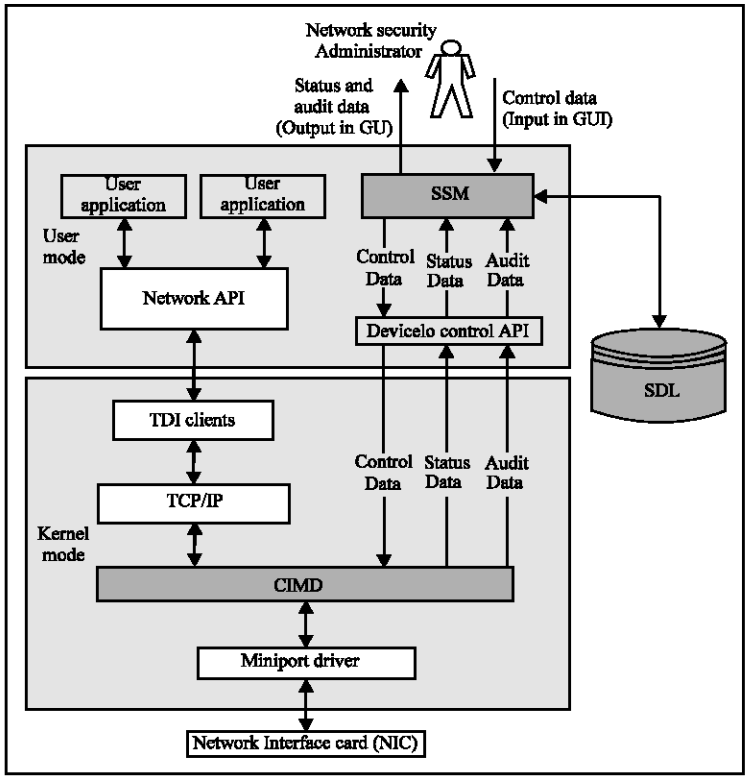

Fig. 1: TSSOLAC Architecture

Figure. 1 illustrates the TSSOLAC architecture and the interaction among its components. The following subsections introduce these components in more details.

The security descriptors list (SDL): The SDL stores security descriptors corresponding to hosts in the protected network. In other words, the SDL is essentially a list of the IP addresses of other hosts in the protected network and their security parameters. If the network security administrator wants to enable communication with a certain host, he (or she) must specify a security descriptor corresponding to that host. This security descriptor is used by TSSOLAC to process packets exchanged with that host. The security policy enforced by TSSOLAC blocks all inbound and outbound packets exchanged with a host, if the SDL does not contain a security descriptor corresponding to that host. TSSOLAC stores the SDL in the Windows 2000 Registry encrypted with a master key calculated using the hard disk serial number and a predefined value, so that it is inaccessible to unauthorized users. Moreover, choosing the Registry to store the SDL prevents non-administrator users from modifying the contents of the SDL.

Each security descriptor consists of the following security parameters:

IP address: This is the IP address of the host represented by the security descriptor.

Secure packets: This parameter states whether packets exchanged with the host represented by the security descriptor must be secured or not. This parameter is a Boolean flag. If its value is False, all other parameters except the IP Address parameter are ignored by TSSOLAC.

Hash algorithm: This parameter specifies the hash algorithm used in the authentication of packets exchanged with the host represented by the security descriptor. TSSOLAC implements two hash algorithms: SHA-1 (Secure Hash Algorithm-1) and MD5 (Message Digest 5).

Encryption algorithm: This parameter specifies the encryption algorithm used to encrypt or decrypt packets exchanged with the host represented by the security descriptor. TSSOLAC implements two encryption algorithms: Rijndael and Twofish.

Key Length: This parameter specifies the length of the encryption key used to encrypt or decrypt packets exchanged with the host represented by the security descriptor. The length could be either 128 bits or 256 bits.

Encryption key: This parameter stores the encryption key used to encrypt or decrypt packets exchanged with the host.

Sequence number counter: A monotonically increasing 32-bit value, placed in the crypto header, which is inserted into each packet that will be sent to the host represented by the security descriptor. This number is used with the anti-replay window to help in the detection of replayed packets. It is also used as the number of packets, which have been secured with the security descriptor currently used key and algorithms and which have been sent to the host represented by the security descriptor. The mechanism TSSOLAC uses to thwart replay attacks follows that used by IPSec, i.e. by using sequence numbers and anti-replay windows (see [5] for an explanation of how IPSec thwarts replay attacks). When a new security descriptor is created, or when the key or algorithms of an existing security descriptor are changed, TSSOLAC initializes the descriptor Sequence Number Counter to 0. When a packet is sent to the host represented by the security descriptor, the sender increments the counter and places the value in the Sequence Number field of the packet crypto header. The network security administrator should not allow the Sequence Number Counter of a security descriptor to be incremented from 2^{32}-1 back to zero because there would be more than one authentic packet having the same Sequence Number. Therefore, before the Sequence Number Counter of a security descriptor reaches 2^{32}-1, the security administrator should change the security descriptor encryption key and/or algorithms.

Packets received: This parameter holds the number of packets that have been received from the host represented by the security descriptor and processed with the security descriptor currently used key and algorithms. This parameter is reset to zero if the security descriptor key or algorithms are changed.

Anti-Replay window: A window that is used to detect replayed packets previously received from the host represented by the security descriptor. This parameter is reinitialized if the security descriptor key or algorithms are changed.

Bytes sent: This parameter holds the number of bytes that have been encrypted with the currently used key and encryption algorithm and sent to the host represented by the security descriptor.

Bytes received: This parameter stores the number of bytes received from the host represented by the security descriptor and that have been decrypted with the currently used key and algorithm.

The TSSOLAC manager (SSM): The SSM presents a Graphical User Interface (GUI) to the network security administrator, so that he (or she) can interact with the TSSOLAC and control its operation. The SSM provides the following functions:

- At initialization time, the SSM loads the encrypted SDL from the Registry, decrypts it and passes its security descriptors to the CIMD. The CIMD stores the passed security descriptors in a linked list in its own memory context for fast access. The CIMD uses the security parameters of each security descriptor to process inbound and outbound packets exchanged with the host represented by that security descriptor and it updates some of these parameters during processing.
- The network security administrator can use the SSM to add, remove, or change the parameters of a security descriptor corresponding to a host in the protected network. The SSM updates the SDL accordingly and uses the DeviceIoControl API (Application Programming Interface) ("The DeviceIoControl API sends a control code directly to a specified device driver, causing the corresponding device to perform the corresponding operation" [6]) to communicate with the CIMD and inform it of the changes.
- The SSM periodically communicates with the CIMD using the DeviceIoControl API to get status data reported by the CIMD and display it to the security administrator. The status data encompasses the mutable security parameters (that are updated by the CIMD) of all the security descriptors. These parameters are: Sequence Number Counter, Packets Received, Anti-Replay Window, Bytes Sent and Bytes Received.
- The SSM periodically communicates with the CIMD using the DeviceIoControl API to get the audit data associated with unauthenticated or replayed packets that were received by the CIMD, if any. The audit data associated with each suspicious received packet is stored in an entry in a linked list maintained by the CIMD. The audit data of each suspicious packet includes: the packet source IP address, the packet source MAC address, whether the suspicious packet is unauthenticated or replayed, the upper protocol data encapsulated by the IP packet, the packet total length and the time when the suspicious packet was received. If there is audit data, the SSM alerts the network security administrator and indicates the audit data to him (or her).

The cryptographic NDIS IM driver (CIMD): The CIMD consists of two parts (like any other NDIS IM driver): the miniport part and the protocol part. The miniport part exposes miniport entry points (MiniportXxx functions), which NDIS calls to communicate the requests of one or more overlying protocol drivers. The miniport part in turn forwards these requests to the underlying miniport NIC (Network Interface Card) driver after performing the required security processing, if any. The protocol part exposes protocol entry points (ProtocolXxx functions), which NDIS calls to communicate requests from underlying miniports. The protocol part in turn forwards these requests to overlying protocols after performing the required security processing, if any. The CIMD inserts into each outbound IP packet that should be secured a 40-byte crypto header between its IP header and payload. This header consists of three fields:

1 The IV field (16 bytes) that holds the random initialization vector used in the encryption.
2 The Hash field (20 bytes) that holds a hash code calculated over the immutable fields of the IP header, the IP payload and the Sequence Number field of the inserted crypto header. This hash code provides data integrity and authentication of the IP packet. The data integrity service detects any modification to the content of the packet in transit. The authentication service allows the receiving host to authenticate the sending host and thus preventing masquerade attacks. TSSOLAC uses a hash code protected by encryption rather than a Message Authentication Code (MAC). If MAC was used instead of hash code, it would be necessary to use two keys for each host that can be communicated with, one for the MAC and the other for encryption and decryption.
3 The Sequence Number field (4 bytes), which is used to detect replayed packets. Each time that a packet is sent to a host, the sender increments the Sequence Number Counter of the security descriptor corresponding to that host and places the resulting value in the Sequence Number field of the packet crypto header.

Then the CIMD encrypts the payload of the IP packet along with the Hash and Sequence Number fields of the inserted crypto header using the encryption key of security descriptor representing the packet destination

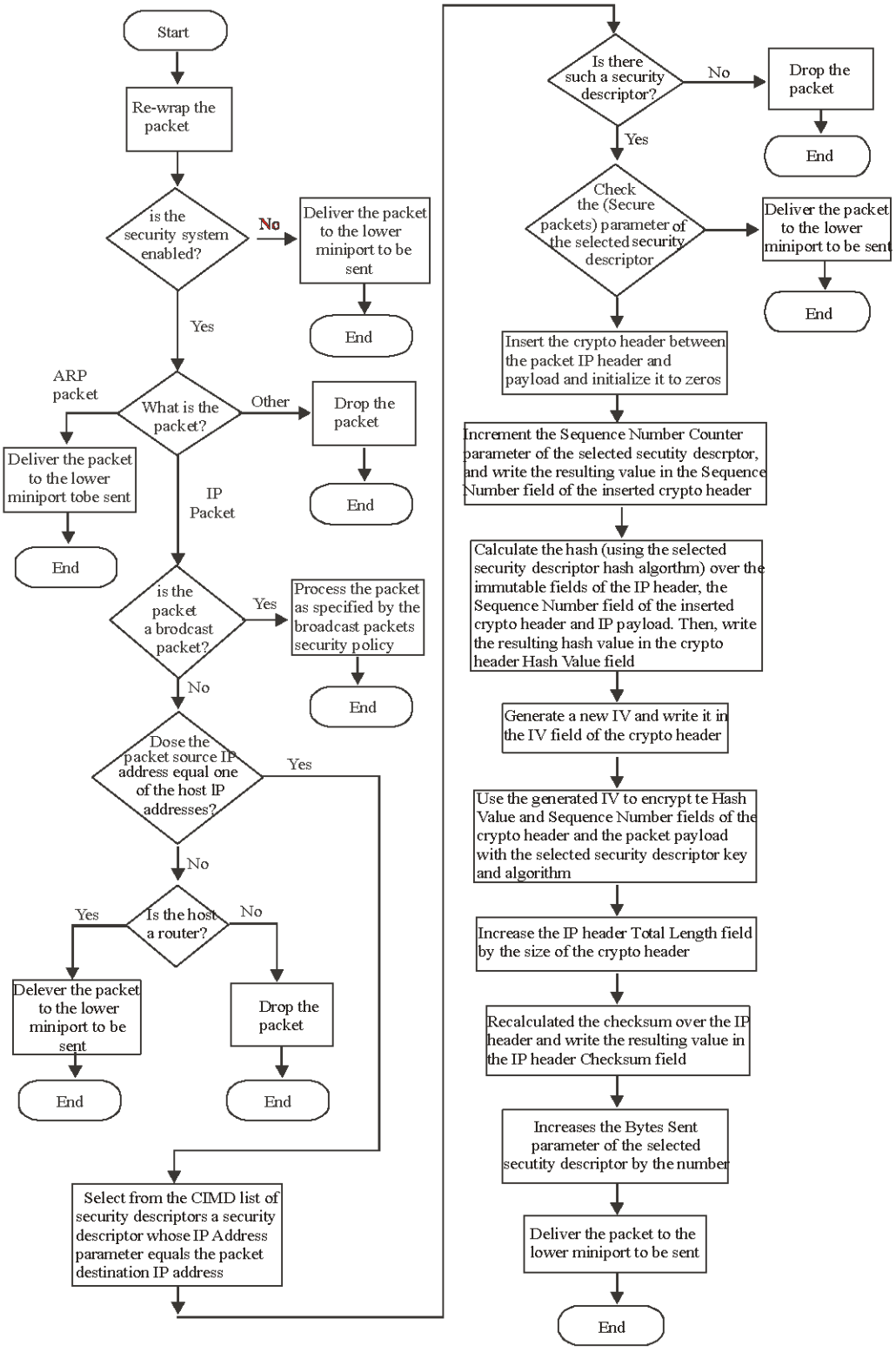

Fig. 2: Flowchart of the Miniport Part Processing of Outbound Packets

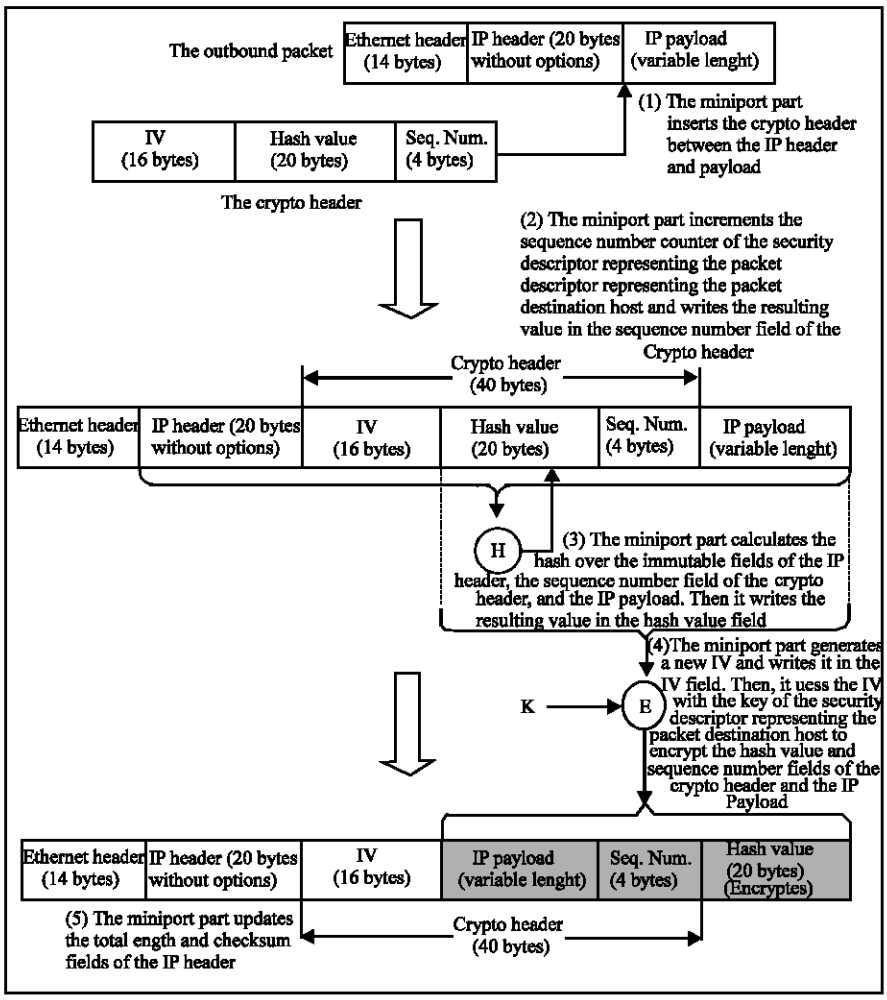

Fig. 3: The Processing of a Typical Outbound IP Packet

host. The encryption provides the required confidentiality to the exchanged data and it protects the hash code held in the crypto header. Figure. 2 shows a detailed flowchart clarifying the steps involved in the processing of each outbound packet. Figure. 3 shows the CIMD miniport processing of a typical outbound IP packet.

When the CIMD processes an outbound packet, it inserts the crypto header between the packet IP header and IP payload, so that routers in the protected LAN (if any) need not be changed and they can route the packet properly. Leaving routers unaffected requires also that the CIMD does not encrypt the IP headers of outbound IP packets. However, the CIMD changes the value of the Total Length field of the IP header of each outbound secured packet to reflect the new packet length resulting after the insertion of the crypto header. The Checksum field of the IP header is also changed to a new value calculated over the new IP header.

On reception, the CIMD decrypts the encrypted portions of each received IP packet and then it inspects its crypto header. If the packet is properly authenticated and it is not a replayed packet, the CIMD strips off the crypto header and then it delivers the packet to upper drivers. Otherwise, if the received packet is not properly authenticated or if it is a replayed packet, the CIMD collects audit information from this packet to be indicated to the network security administrator. Figure. 4 shows a detailed flowchart clarifying the steps involved in the processing of each inbound packet. Figure. 5 shows the CIMD protocol processing of a typical secured inbound packet.

TSSOLAC processes packets in layer 2 (data link layer) of the OSI (Open Systems Interconnection) protocol stack (between the LLC and MAC sublayers). Nevertheless, it does not provide link encryption and it is considered to be an end-to-end cryptographic system.

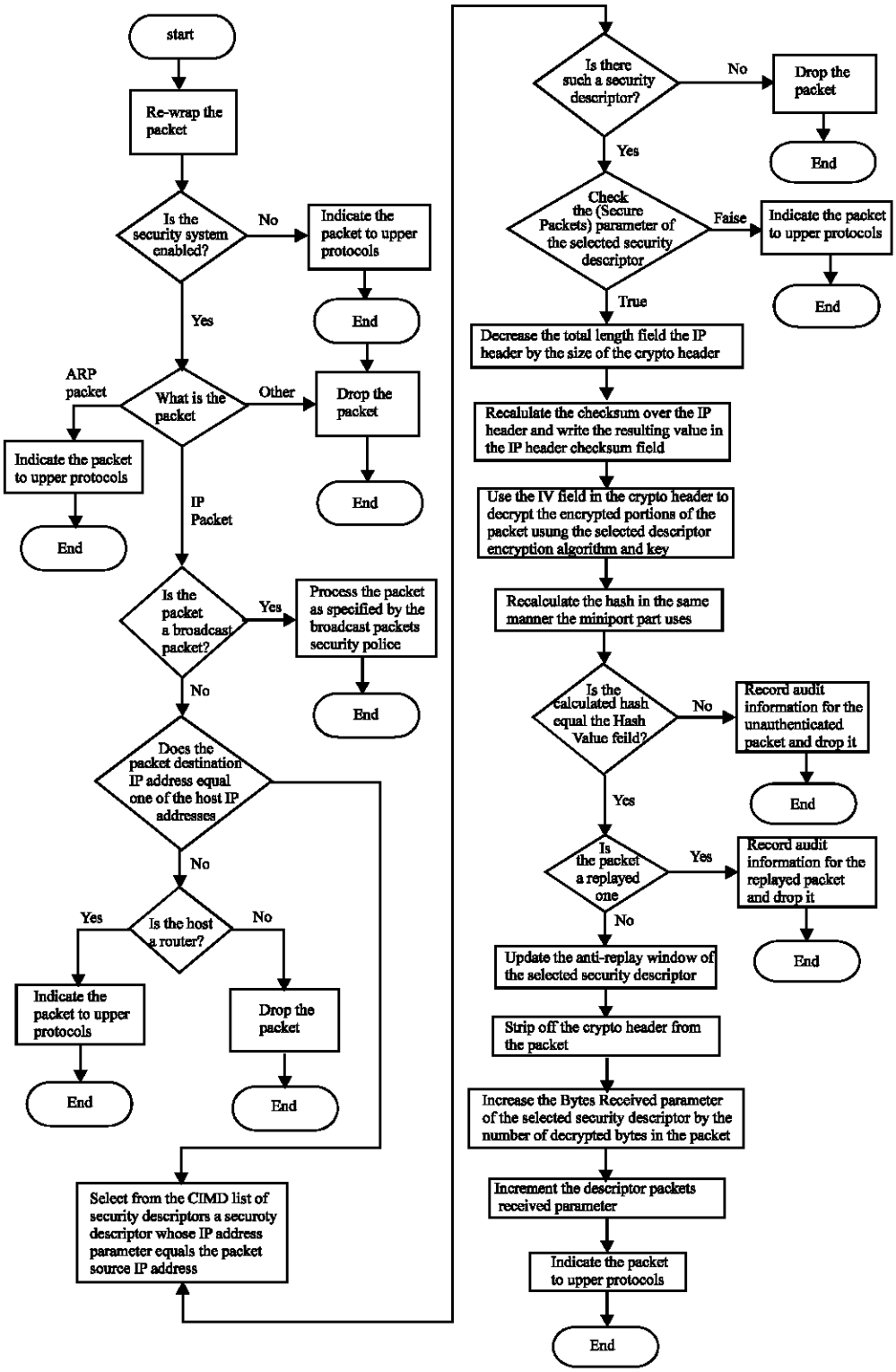

Fig. 4: Flowchart of the Protocol Part Processing of Inbound Packets

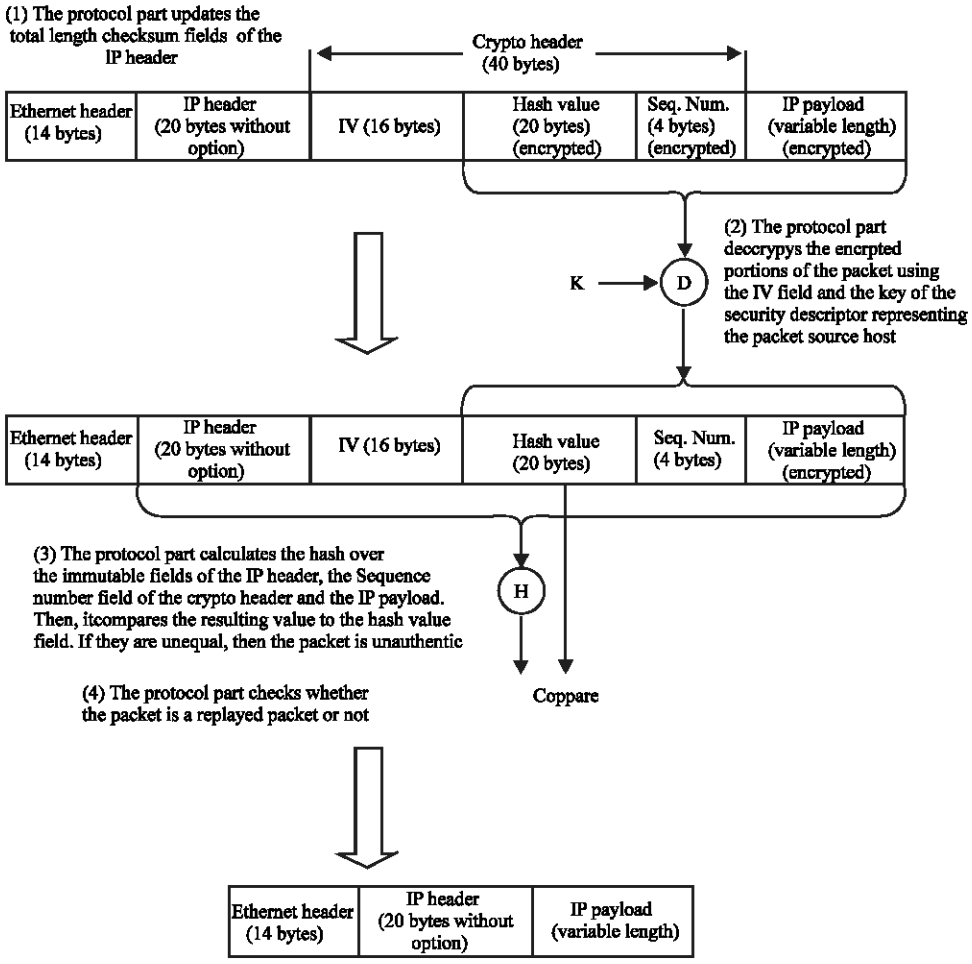

Fig. 5: The Processing of a Typical Secured Inbound Packet

Other design issues: This section analyzes a number of issues in a detailed manner and specifies the rules that TSSOLAC must follow to handle them successfully.

Large packets restrictions: Since the CIMD increases the length of outbound IP packets by the size of the inserted crypto header (40 bytes), the processing of large IP packets may impose the following two problems:

A-The first problem: If the CIMD inserts the crypto header into a large IP packet, whose size is larger than 1460 bytes, the size of the packet will become more than 1500 bytes after the insertion of the crypto header. This will cause the sending NIC to drop the packet, because maximum size of IP packets over Ethernet II is 1500 bytes [7].

Hence, to resolve this problem, TSSOLAC explicitly sets the MTU (Maximum Transmission Unit) of each network interface to 1460 bytes instead of the default value (i.e. 1500 bytes). This must be done so that the TCP/IP protocol will not send to the CIMD IP packets larger than 1460 bytes. In Windows 2000, the MTU of an *interface* can be changed by modifying the following Registry value [1]:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces*interface*\MTU.

The second problem: Routers are usually used to route packets between networks. However, routers that use Microsoft Windows Routing and Remote Access can be used as workstations at the same time. Hence, TSSOLAC can be installed on such routers to secure packets exchanged between them and other hosts. Of course, packets that are routed by a certain router that uses TSSOLAC are not processed by TSSOLAC installed on that router. Only packets that are originated from or destined to that router are processed by TSSLOLAC.

When TSSOLAC is installed on a router in the protected network, another problem arises. The reason behind this problem is that the MTU of that router will be set to 1460 bytes and this will disrupt the function of Path MTU (PMTU) Discovery [8] (which is used by default by Windows for packets destined to a non-local host [1]). This is because the IP layer on the sending host is unaware of the existence of the CIMD that increases the length of outbound packets. Even if the IP layer adheres to the Next-Hop MTU value in the ICMP (Internet Control Message Protocol) Destination Unreachable messages reported by some router in the packet path, the CIMD still increases the length of outbound packets by the size of the inserted crypto header after it receives them from the IP layer. Thus, resulting packets will have a length 40 bytes more than the Next-Hop MTU reported by the router and thereby causing the router to reject these packets.

TSSOLAC solves this problem by disabling PMTU Discovery. In Windows, when PMTU Discovery is disabled, an MTU of 576 bytes is used for all non-local destination addresses [1]. Hence, after the CIMD inserts the crypto header into an outbound packet destined to a non-local host, the packet length will not exceed 616 bytes (576 bytes, the MTU + 40 bytes, the size of the crypto header). This packet length is well below the MTU of intermediate routers that use TSSOLAC. Thus, packets having this length will not be fragmented by the intermediate routers. The PMTU Discovery can be disabled by setting the following Windows 2000 Registry value to 0 [1]:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\EnablePMTUDiscovery.

As another option, TSSOLAC allows the administrator to choose that no router in the protected network uses TSSOLAC, a case in which the 2nd problem does not exist and there is no need to disable PMTU Discovery on hosts that use TSSOLAC.

Broadcast Packets Processing: A broadcast packet is a packet going from one host to all hosts on the same subnet. A broadcast packet has its destination IP address Host ID all ones. As an example, a subnet having a Network ID of 200.50.45.0 and a subnet mask of 255.255.255.0 will have a broadcast address of 200.50.45.255. Windows uses broadcast packets for many functions, including NetBIOS (Network Basic Input Output System) name queries and announcements.

IPSec in Windows 2000 and Windows XP does not protect broadcast packets. Broadcast packets represent one of the IPSec default exemptions, which are not secured by IPSec in Windows 2000 and Windows XP [9].

The security system provides three options for the processing of broadcast packets, allowing the network security administrator to choose one of them:

- Broadcast packets are blocked.
- Broadcast packets are secured and processed in the same way normal packets are processed. However, the key used for encryption and decryption is shared among a number of hosts that are allowed to exchange broadcast packets by the network security administrator. Moreover, the security system does not provide anti-replay service for broadcast packets because of its infeasibility.
- Broadcast packets are permitted to bypass without securing them.

Tssolac conflicts: TSSOLAC was designed to thwart attacks that may attempt to change the contents of an IP packet, or use IP spoofing. Thus, like any other end-to-end cryptographic system that secures IP packets against such attacks, TSSOLAC conflicts with protocols and services that, during the transit of an IP packet, may change the contents of its IP payload or IP header fields that are protected by the system. TSSOLAC is known to have conflicts with the following:

1 Fragmentation. Apparently, fragmentation of a secured packet conflicts with the work of TSSOLAC. This is because the packet fragments other than the first one will not have crypto headers and so will be considered unauthentic by TSSOLAC on the target host. The first fragment will not be authentic as well, because its crypto header originally authenticates the total packet, not only the first fragment.

2 Network Address Translation (NAT). NAT is a method, by which an IP address of a packet is translated into another IP address, thus providing a mechanism for networks with private IP addresses to connect to the Internet using one or a small number of public IP addresses. NAT is commonly used because of the shortage of IP address space in IPv4.

Due to the fact that NAT may change the source or destination IP addresses of packets in transit, which TSSOLAC protects, TSSOLAC at the destination of the packets will consider them unauthentic. However, since NAT is not normally used inside a LAN, which is the network type that TSSOLAC was designed to protect (it is normally installed on a router that connects a LAN to an external network), the issue of incompatibility between the TSSOLAC and NAT is not important.

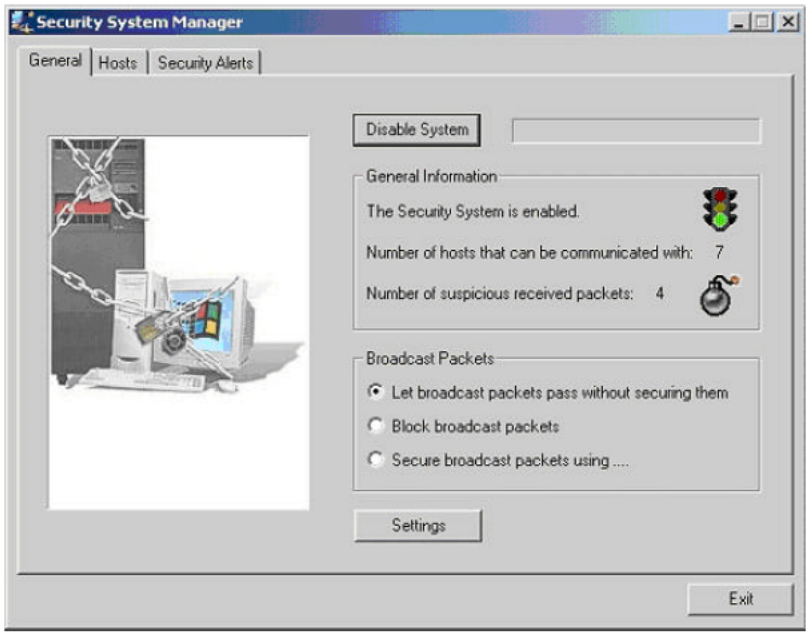

Fig. 6: The "General" Page

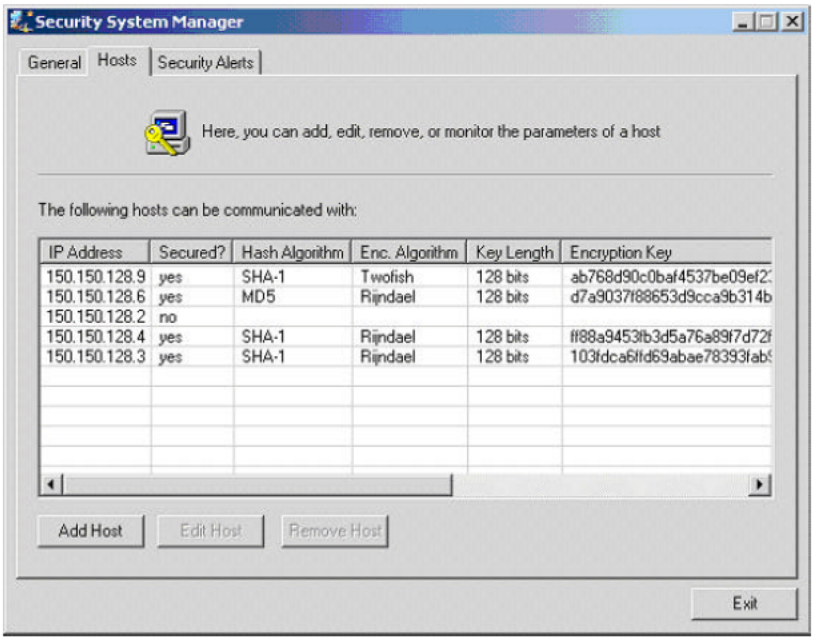

Fig. 7: The "Hosts" Page

Tssolac implementation: The CIMD was written in C language and using NDIS library support routines, then it was compiled and built using the Windows 2000 DDK (Driver Development Kit). The SSM was developed using Microsoft Visual C++ language. The GUI of the SSM is implemented as a property sheet that consists of three property pages: the "General" page, the "Hosts" page and the "Security Alerts" page.

The "General" page shown in Fig. 6 presents general information and it can be used by the network security administrator to turn on or off TSSOLAC. It can also be used to manage other settings. The "Hosts" page whose

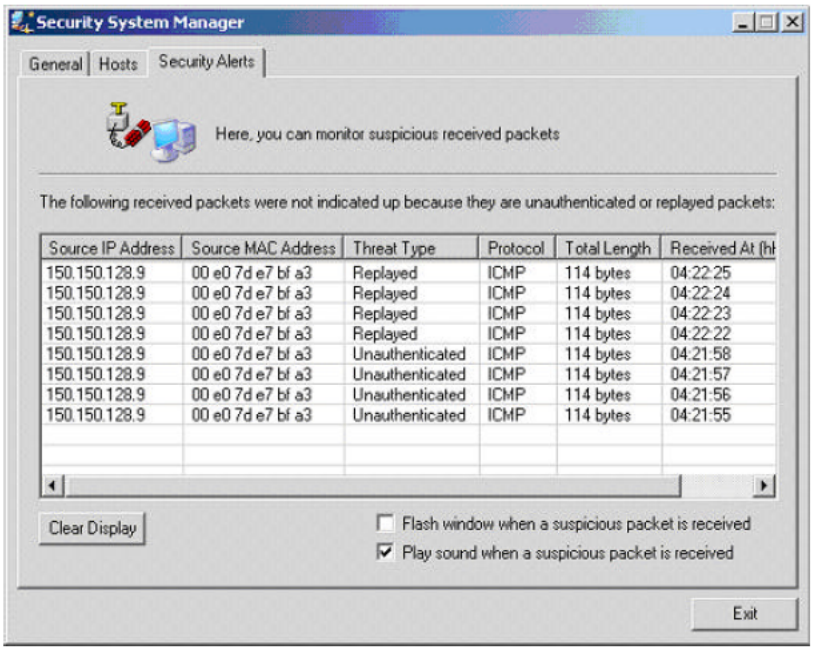

Fig. 8: The "Security Alerts" Page

snapshot is shown in Fig. 7 includes a list view control that displays the parameters of security descriptors representing the hosts that can be communicated with in the protected network. Moreover, the "Hosts" page can be used to add a new host, edit the security parameters of an existing host and remove an existing host. The "Security Alerts" page whose snapshot is shown in Fig. 8 page provides audit information associated with unauthenticated or replayed packets that were received by the CIMD, if any.

PERFORMANCE RESULTS

To measure the performance results, TSSOLAC was installed on 5 machines, which were connected as 100BaseTX Fast Ethernet by using Linksys 10/100 dual speed 16-port stackable Hub. The specifications of each machine were as follows:

- Processor: Intel Pentium III, 866 MHz
- Physical Memory: 128 Mbytes
- Hard Disk: Western Digital Caviar, 20 Gbytes
- LAN Card: Realtek RTL8139(A)
- Operating System: Windows 2000 Advanced Server

Four tests were performed on two machines (from now on, they will be called A and B) during the transfer of a 710-Mbyte file between them: Security test, network performance test, memory usage test and processor usage test. The file was located in A, while B was used to get the file from A. Network Monitor tool of Windows 2000 was used in the security test, whereas the Performance tool of Windows 2000 was used in the remaining tests, which were repeated on the two machines when they used TSSOLAC in each of the following cases:

1 The encryption algorithm is Rijndael and the hash algorithm is SHA-1.
2 The encryption algorithm is Rijndael and the hash algorithm is MD5.
3 The encryption algorithm is Twofish and the hash algorithm is SHA-1.
4 The encryption algorithm is Twofish and the hash algorithm is MD5.

For comparison purposes, the 2nd, 3rd and 4th tests were also repeated in the following cases:

1 When the two machines deployed Windows 2000 IPSec with a pre-shared key as the authentication method, ESP as the used IPSec protocol, 3DES as the encryption algorithm and SHA-1 as the hash algorithm used by the IPSec HMAC.
2 When nothing was used to secure network traffic. This situation will be referred to as: "Normal Case".

The following subsections present and discuss the results of all the performed tests.

Security test: Before transferring the file, TSSOLACs on A and B were configured to use Rijndael encryption algorithm and SHA-1 hash algorithm with a pre-shared 128-bit key. During the transfer of the file from A to B, the network interface of another machine (C) on the same network was configured to work in promiscuous mode (A promiscuous mode is a mode in which a network interface card receives all the frames in its network, even if they are not destined to it) and a number of frames going from A to B were captured.

To ensure that TSSOLAC provides a good protection against eavesdropping, a brute force attack was launched against the encrypted portion of one captured frame. The attack was performed by a tool that tries to decrypt the encrypted portion with a random key, calculates the hash code over the resulting packet as done by TSSOLAC and then compares the resulting hash code with the hash code field of the crypto header. If they do not match, it selects another key and repeats the same attempt again. The attack lasted for 48 hours with no success.

To ensure that TSSOLAC can thwart replay attacks, a number of captured frames were injected back to the network using a packet injector tool. All the injected frames were received by B, but were dropped by TSSOLAC on B because it considered them as replayed packets.

To ensure that TSSOLAC can thwart address spoofing attacks and it does provide access control, the IP address of C was changed to that of B and TSSOLAC on C was configured to use Rijndael encryption algorithm and SHA-1 hash algorithm with a random key. Then, C was used to ping A with a number of ICMP echo request packets. All the ICMP packets were received by A, but were dropped by TSSOLAC on A because it considered them as unauthentic packets.

Network performance test: The average packets/sec Performance tool counter was used to measure the network performance. Table 1 shows the results of the average packets/sec counter in all the cases.

When TSSOLAC is deployed, the degradation in the average packets/sec rate with respect to the Normal Case was about: 31.7% for (Rijndael,MD5), 35.6% for (Rijndael,SHA-1), 44.4% for (Twofish,MD5) and 45.5% for (Twofish,SHA-1). On the other hand, the use of the IPSec service with a pre-shared key as the authentication method, ESP as the used IPSec protocol, 3DES as the encryption algorithm and SHA-1 as the hash algorithm caused a degradation of about 30.9%. The degradation in the average packets/sec rate is a normal result because of the delay introduced by the security related processing on the two machines.

Table 1: The Results of the Network Performance Test

Case	Packets/sec at A	Packets/sec at B
Normal Case	10574.911	10572.754
IPSec	7307.058	7306.679
TSSOLAC (Rijndael,SHA-1)	6807.960	6809.903
TSSOLAC (Rijndael,MD5)	7218.649	7217.341
TSSOLAC (Twofish,SHA-1)	5761.566	5759.622
TSSOLAC (Twofish,MD5)	5879.617	5879.498

Table 2: The Results of the Processor Usage Test

Case	%Processor Time at A	%Processor Time at B
Normal Case	41.757 %	47.903 %
IPSec	84.787 %	85.212 %
TSSOLAC (Rijndael,SHA-1)	84.522 %	89.804 %
TSSOLAC (Rijndael,MD5)	83.674 %	88.590 %
TSSOLAC (Twofish,SHA-1)	85.060 %	90.825 %
TSSOLAC (Twofish,MD5)	86.638 %	88.600 %

Processor usage test: Processor usage was measured using the %Processor Time Performance tool counter, which is the "percentage of time the processor is executing a non-Idle thread. This counter was designed as a primary indicator of processor activity. It is calculated by measuring the time that the processor spends executing the thread of the Idle process in each sample interval and subtracting that value from 100% (Each processor has an Idle thread, which consumes cycles when no other threads are ready to run) "[10].

All services and applications on A and B not necessary for Windows to work properly were turned off before test started so that the obtained results reflect the effect of TSSOLAC on processor usage as accurately as possible. The results of the test are shown in Table 2. Before the transfer operation started, the %Processor Time was measured at A and B and its average was nearly 0.2% at each one of them. When either TSSOLAC or IPSec is used to secure network traffic, %Processor Time increases apparently due to the required security related processing. However, the increase in processor usage does not differ much for all the cases that have security processing.

Table 2 shows that the processor usage at B is larger than its usage at A in all the cases. This is due to the fact that B had to perform more operations than A such as reassembling and checking the transferred file.

Memory usage test: Memory usage was investigated using the Available Mbytes counter of the Performance tool. Before the file transfer operation started, the memory available at A was: 25.5 Mbytes, while that available at B was: 42 Mbytes. This difference is due to the fact that different services and applications were running on the two machines. Table 3 shows the results of the Available Mbytes counter at A and B when the file was transferred between them in all the cases. The memory used for file

Table 3: The Results of the Memory Usage Test

Case	Available Mbytes at A	Used Mbytes at A	Available Mbytes at B	Used Mbytes at B
Normal Case	22.500	3	36.383	5.617
IPSec	22.680	2.8	37.030	4.97
TSSOLAC (Rijndael,SHA-1)	21.000	4.5	35.230	6.77
TSSOLAC (Rijndael,MD5)	21.000	4.5	36.770	5.23
TSSOLAC (Twofish,SHA-1)	20.090	5.41	34.000	8
TSSOLAC (Twofish,MD5)	21.170	4.33	33.570	8.43

transfer at each machine was calculated by subtracting the available memory during the file transfer from the available memory before the file transfer started. As shown in the table, the memory used at each machine does not differ much for all the cases. It may also be noticed that the memory used at B is more than that used at A in all the cases. This is due to the fact that B had to perform more operations than A such as reassembling and checking the transferred file.

CONCLUSION

The most important points concluded throughout the design and implementation of TSSOLAC are listed below:

- To provide security to the data exchanged within a network, encryption and message authentication mechanisms must be used. However, these mechanisms impact the average packets/sec rate in the network, which degrades by a percentage depending on the complexity of the used algorithms. Moreover, these security mechanisms increase processor usage on the hosts that perform them approximately to the double.
- The development of an end-to-end cryptographic system that works in layer 2 (data link layer) of the OSI reference model can be made much easier, if the protected protocol that works in layer 3 (network layer) is aware of the existence of the cryptographic system. This may require that the protected protocol and the cryptographic system be developed by the same party, or by cooperative parties.
- Windows network drivers represent a powerful and efficient method to transparently intercept and process inbound and outbound network data. Among the well-documented network drivers, the NDIS intermediate driver is the more powerful and more efficient one.
- If a network driver below the IP layer (such as a NDIS intermediate driver) inserts a number of bytes into outbound packets that will be sent on a certain network interface, the MTU of that interface must be decreased by the number of bytes that will be inserted, so that large packets are not dropped by the network interface.
- If a NIC has its NDIS task offload capabilities enabled, these capabilities may improperly affect the work of installed packet-modifying NDIS intermediate drivers. Hence, these capabilities should be turned off for the modifying intermediate drivers to work properly.
- For future work, TSSOLAC may be improved to provide a secure automatic key exchange protocol. Indeed, the case of heterogeneous LANs including, for example, several kinds of Unix-based machines may also be considered.

ACKNOWLEDGMENT

Authors would like to thank Mr. Thomas F. Divine for his help and support.

Vitae:

REFERENCES

1. MacDonald, D. and W. Barkley, 2000. Microsoft Windows2000 TCP/IP Implementation Details, Microsoft Corporation.
2. Microsoft Corporation, Microsoft Windows2000 Driver Development Kit - Network Drivers.
3. Schneier B. and N. Ferguson, 1999. A cryptographic evaluation of IPSec, Counterpane Internet Security, Inc.
4. Nuopponen, A. and S. Vaarala, 2002. Attacking predictable IPsec ESP initialization vectors, Helsinki University of Technology, Finland.
5. Stallings, W., 1999. Cryptography and Network Security: Principles and Practice, second ed., Prentice Hall,
6. Microsoft Corporation, 2001. Microsoft Development Network, Platform SDK Documentation, April.
7. Hornig, C.A., 1984 standard for the transmission of IP datagrams over ethernet networks, RFC 894, April.
8. Mogul, J. and S. Deering, 1990. Path MTU discovery, RFC 1191, November.
9. Microsoft Corporation, IPSec default exemptions can be used to bypass IPSec protection in some scenarios, Microsoft Knowledge Base Article 811832, available at: http:// support. microsoft. com/ default. aspx?scid=kb; EN-US; 811832.
10. Microsoft Corporation, 2000. Microsoft Development Network, Windows 2000 Resource Kit Reference-Windows Performance Counters Reference, April 2001.