# LES: Layered Encryption Security

Manish Karir John S. Baras
Center for Satellite and Hybrid Communication Networks
Department of Electrical and Computer Engineering & Institute for Systems Engineering
University of Maryland, College Park, MD 20742, USA
{karir, baras}@isr.umd.edu

***Abstract*— In this paper we discuss the concept of Layered Encryption Security(LES). We analyze the need for introducing layering in a security infrastructure. Layered encryption provides us the ability to selectively disclose different parts of data to different parties that might be interested in it without compromising the security of the other parts. We have implemented a simple prototype implementation to demonstrate the practical feasibility of the concept by modifying IPSEC. Measurements from our implementation indicate that the Layered Encryption approach does not add unacceptable delays and does not substantially increase processing requirements at either the end points or at intermediate nodes.**

## I. Introduction

Most encryption schemes that are in use today or are currently being proposed are generally monolithic in nature. The unstated underlying assumption among them is that all portions of the data to be encrypted are equally important, and therefore they use the same key for the entire data. While this is generally a safe assumption, there are scenarios where this subtle yet limiting assumption breaks-down and creates more problems than it solves.

Traditionally, it has been demonstrated that introducing layering into a system adds flexibility which allows for innovation and use in scenarios that the original system might not have supported. In fact the example of the layered TCP/IP stack is an excellent example of this concept. The layered structure has made the TCP/IP stack immensely flexible to the point that it continues to support applications and modifications which the original designers could not even have envisioned.

In stark comparison, security has always been promoted as an all or nothing feature. Security services(protocols and applications) are generally implemented as end-to-end systems. IPSEC is a good example of this approach. Only the end points in IPSEC have the ability to access data. This end-to-end security is considered a key feature and requirement of any secure system. However,the problem of how one can perform selective disclosure of data to different parties has not been addressed.

According to our layered approach, different parts of the data can be encrypted with different security schemes, infact, different parts of a single data packet could be encrypted with different encryption algorithms. This added flexibility leaves the decision of what to secure and what to leave unsecured up-to the users, who best know the structure of their data. This layering approach enables us to build systems that incorporate selective disclosure.

In this paper we illustrate the problems that are created by a monolithic approach to security, and then illustrate how a layered encryption scheme can resolve these problems. We also show, using IPSEC as a starting point, that implementing layering only adds marginal overhead to the base system, and that careful selection of what encryption algorithms for different parts of the data can easily offset any overhead. In addition, we argue that using layered encryption strengthens security as now two or more separate keys have to be broken in order to access the entire data.

## II. Problem Description

In this section we discuss four common applications that are broken by implementing a monolithic security scheme such as IPSEC as it stands today. In each case we argue for a more flexible solution, one that allows selective disclosure.

### A. Network Monitoring and Analysis

The ability to monitor and analyze traffic is essential for legitimate network designers, planners and administrators in order to monitor network usage and provide better service.

There are several applications currently in use for network analysis and monitoring. TCPdump, Snoop and Sniffer from Network Associates are some commonly used tools. These tools help network administrators and engineers to monitor and analyze the traffic on their network so that they can identify faults and bottlenecks. In the recent years significant progress has been made in the field of network analysis via the use of such network analysis tools which have forced us to question the very basic assumptions that had been made regarding the nature of network traffic. For example we now know that the simplistic Poisson arrival patterns for traffic which had been assumed for a long time, are no longer valid. Invariably, as new applications develop, traffic patterns in networks also change, sometimes drastically. Without the ability to use these tools we would not be able to identify such changes.

Network analysis and monitoring requires the ability to access headers within packets. However the use of IPSEC like security is in direct conflict with this requirement. A monolithic security structure prevents all access to packet headers even if the need is legitimate.

### B. Packet Classifiers and Firewalls

Packet classifiers rely on the ability to accurately read packet headers in order to make classification decisions. Most modern routers have the ability to perform such classifications for the purposes of providing some Quality of Service(not necessarily based on RSVP). The packet classification is usually done on the basis of fields in the packet headers (IP addresses, TCP ports, TCP flags, Application layer headers, etc.) and then packets are placed in queues on which some scheduling discipline is applied. However, if we use IPSEC like security, transport and even application layer headers can no longer be examined.

Firewalls are a simple extension of Packet Classifiers, where the algorithm that is performed on classified packets is not which queue they should be placed in but whether they should be dropped or accepted for forwarding. In this way, the operation of firewalls is equally hindered by IPSEC like security.

### C. NAT and Internet Service Providers

Network Address Translation (NAT) is a technique often used by Internet Service Providers to maximize their use of Internet addresses. NAT allows them to use private addresses internal to their network, and then use a fixed address pool when accessing data from the Internet [1]. The operation of NAT requires it to have both read and write access to packets traversing through it. NAT is also used by network administrators who wish to hide the IP addresses internal to their network from the outside Internet. This provides some security by means of obscurity.

In addition, NAT can be used in several applications other than IP address space conservation and security by obscurity. Some of these applications are:

- Load Balancing Servers: NAT can be used to create virtual servers which provide a single reference point to clients. Internally the virtual server routes the requests to one among several servers.
- Load Balancing Networks: NAT can be used for transparently using different ISP or routers when accessing a remote host. When an internal host wants to establish a new connection with a destination on the Internet, it just sends its packets to the NAT gateway. The NAT gateway , because it knows all connections, decides which provider will route this connection, replaces the source hosts (internal) address with one of the providers chosen and sends it out to this providers router.

Once again, as NAT requires access to information in packet headers, they cannot be used in conjunction with IPSEC like security protocols.

### D. Internet over Satellite and Wireless Links

Various solutions have been proposed to overcome the problem of running conventional Internet protocols over high delay(satellite) or error prone(wireless) network links. These include the use of proxies, protocol enhancements, as well as the use of new custom designed protocols. One solutions

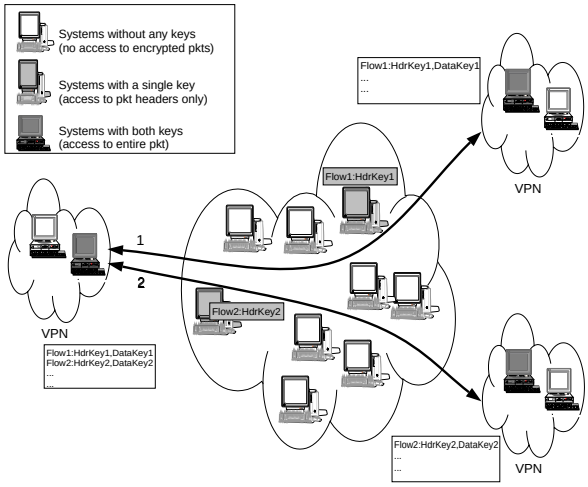


Fig. 1. Fine grained security control using LES

that has been shown to work quite well in practice is to hide the presence of such links from the protocols via the use of proxies or enhanced gateways [2] [3].

TCP connection splitting is performed at these proxies, thereby shielding TCP from the unique nature of a satellite or terrestrial wireless link. A protocol designed specially to work well for these links is then used to transport data over those links, and if necessary another TCP connection is setup on the other side to carry data over conventional terrestrial links.

The performance impact of the use of such techniques is enormous, which is why their use is quite common in such environments. However, in order to split TCP connections we need access to TCP headers, this is not available with conventional IPSEC like security protocols. Therefore, one is forced to make a tradeoff between security or improved performance.

## III. LES: Layered Encryption Security

The applications described in the previous section provide a good flavor for the types of problems a rigid security structure introduces. In this section we describe LES, our approach to providing security with flexibility. The concept of LES is essentially very simple. Learning from other flexible systems we propose to implement security in layers. While, LES can be generalized to any scheme where different parts of data are encrypted with different encryption keys and perhaps even with different algorithms, in this paper we focus on LES as applied to a single packet.

By using multiple keys, we provide the end-points of a flow with the ability to selectively distribute keys to different nodes, depending on the level of access they require to information within the packets of that flow. This concept is illustrated in Figure 1. The dark shaded systems have all keys to access all information in a packet, the lighter shaded intermediate systems only have keys to decrypt the header portion of the packet. The systems with no shading do not have any keys.

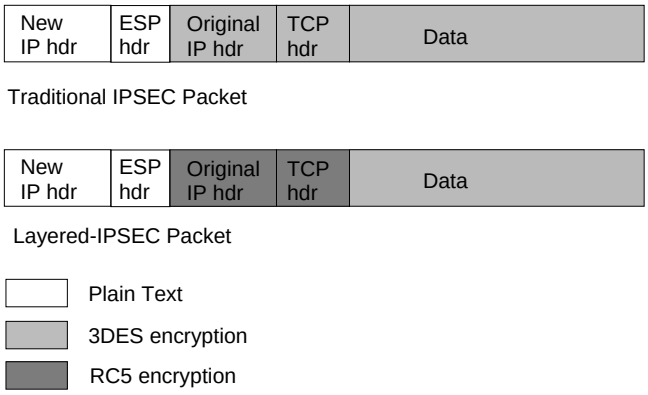


Fig. 2. Layered Packet Encryption in IPSEC

This concept as it applies to IPSEC was also developed independently by HRL Laboratories [4] [5]. In this form, the LES technique aims at adding flexibility to the existing IPSEC framework. In traditional IPSEC a single encryption key and scheme is used to encrypt the entire packet. The encryption scheme chosen can be RC5, DES or 3DES or any other algorithm. However, IPSEC does not make any provision for the use of multiple keys or multiple encryption algorithms. This makes the IPSEC framework quite inflexible. The concept of LES as it applies to IPSEC is illustrated in Figure 2.

The primary features of LES are listed below:

- Multiple key generation: Each end-point of the secure connection generates separate keys for each of the encryption algorithms it will use for that particular connection. Multiple keys can be used even if the same encryption algorithm is used to perform the encryption. In the original IPSEC specification, each end-point has two keys, one for sending and one for receiving data. In our scheme, if for example two encryption schemes or keys are being used, then there are a total of four keys at each end-point. One set is used for sending data and the other for receiving data.
- Multiple key distribution: Similar to the original IPSEC single key scenario, the keys are symmetric and therefore need to be exchanged between the connection endpoints. However, with LES, key distribution is much more complicated. Not only is the number of keys that need to be exchanged two or more times the number of keys in original IPSEC, but in addition, if other intermediate systems require a key the appropriate key needs to be distributed to them as well.
- Selective Decryption and Re-encryption: While the endpoints of the secure connection decrypt the entire packet, any intermediate systems that have been given keys to certain parts of a packet perform partial decryption on those portions. In some cases if the original packet is modified, re-encryption must be performed so that the end systems will still get what appear to be end-to-end encrypted packets. As the decryption and encryption keys are the same, this does not add much complexity to the intermediate system when compared with the benefits it provides.

By allowing layered encryption, LES aims at providing the much needed packet header visibility to intermediate systems, but at the same time does not completely reveal the contents of the entire packet to those systems as the data portion of the packet is still encrypted by a key that is not known to them.

We now briefly describe how the use of LES can solve the problems described in the previous without either increasing the complexity of the system unacceptably or significantly diminishing the overall security of the system.

- Network Monitoring and Analysis: The network monitoring system obtains keys for decrypting headers of all secure flows passing through it. For each LES encrypted packet, the header is decrypted and all relevant information is extracted. There is no need to re-encrypt as a captured packet is never retransmitted, network monitoring applications only receive a copy of the original packet.
- Packet Classifiers and Firewalls: The packet classifier/firewall obtains decryption/encryption keys(same for symmetric keys) for the packet headers of the LES flows passing through it. The packet header is decrypted, and examined to see if the classifier rules are met. The outgoing packet is then re-encrypted before retransmission.
- Network Address Translation: This functionality is similar to the functionality of the packet classifiers described previously. Packet headers are decrypted, address translation is performed and then packets are re-encrypted.
- Internet over Satellite and Wireless Links: The key used to encrypt the header is distributed to the TCP connection splitting gateways. When a packet from a LES encrypted flow reaches the connection splitting gateway, the header of the packet is decrypted and acknowledgments are spoofed. Depending on the topology, the decrypted packet headers may need to be re-encrypted before transmission.

The advantages of using LES are summarized below.

- No compromise on security: The end-to-end security for data payloads can be preserved. Even the packet headers are encrypted and are safe from attackers who do no have the appropriate keys.
- Privileged applications: Layered IPSEC gives us the ability to support privileged applications in the Internet, by allowing us to control up to a finer level who can read specific parts of a packet.
- Feasibility and ease of implementation: The intermediate systems in this scheme only have to decrypt a small portion of the entire packet. This takes much less computation than having to decrypt the entire packet making this scheme much more practical and feasible than using split IPSEC.
- Security: Probable plaintext cryptanalysis can be used to aid in cracking an encryption key. A probable plaintext attack works by looking at certain bit positions for which a likely value can be predicted [6]. A comparison

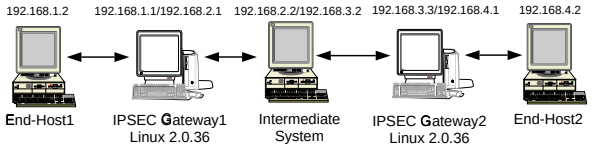


Fig. 3. LES Testbed Architecture

engine can count the number of such matches, and pick up certain packets for further analysis by a second stage cracking engine. Given that the IPSEC protocols when used in tunnel mode encrypt the IP header, a large number of bit fields have values that can be predicted. Therefore, the IPSEC protocols are vulnerable to probably plaintext attacks. Using the encrypted IP and TCP headers, attackers can obtain the encryption key, and then proceed to decrypt the data itself. However, the use of LES with IPSEC can protect against this type of an attack, as different keys are being used for the headers and the data. Therefore, there is very little probably plaintext information available to help in cracking the key used to encrypt the data.

While the advantages of using LES are significant there are some disadvantages that need to be considered as well.

- Key distribution: When we use LES not only are there multiple keys, but now different keys need to be distributed to different people depending on their authorization. While we do not describe any key distribution mechanism in this paper, we think existing key distribution mechanisms can be easily modified to handle this scenario.
- Overhead: The use of LES requires that multiple encryption decryption algorithms be used, which adds overhead not only to the end systems but also to intermediate systems that are authorized to have access to information within the packets. We studied this via our implementation, which is described in the next section.

## IV. Implementation Details

### A. Testbed Network Architecture

The testbed we setup for implementation of LES is shown in Figure 3. In our testbed, the IPSEC Gateways run the FreeS/WAN software. A secure tunnel is setup between these two gateways. End-to-end communication is tested by running applications between the two end hosts (ping, telnet, and ftp). The intermediate system between the two security gateways, functions as a general router in the base case when normal IPSEC is used, but is used to test and verify the correct operation LES in other scenarios. It basically represents a specific trusted and authorized host in the Internet that wants to access information contained in the packet headers of the encrypted flow between the two security gateways.

### B. FreeS/WAN

The LES testbed used version 1.0 of the FreeS/WAN written by developers at the Electronic Frontier Foundation (EFF) in Canada [7]. FreeS/WAN provides us with a base IPSEC implementation on top of which we can implement our modifications.

For our testbed, all changes to the FreeS/WAN software distribution were kept to a minimal. No modifications were made to packet formats, key distribution, etc. The only changes that were made were to add support for LES. This essentially involves two steps. In the first step, the starting point in a packet from which 3DES encryption takes place is moved forward past the transport layer header to the data payload part of the packet. In the second step, the transport header of a packet is encrypted using a second encryption algorithm or key.

Corresponding changes are also made in the decryption routines of the FreeS/WAN software, so that the right portions of the packet are decrypted with the corresponding algorithm.

### C. The RC5 Algorithm

We choose the RC5 Algorithm as our second encryption/decryption algorithm. RC5 is a fast symmetric block cipher suitable for hardware or software implementations[8].

In general, by choosing greater key sizes and more rounds in RC5, leads to a higher level of security. The performance of RC5 is directly proportional to the number of rounds, and is not affected by the key size. This flexibility of RC5 makes it ideal for our implementation of LES, as the tradeoff between speed(and hence throughput) and security can be balanced via appropriate parameter settings. A word size of 32 bits, a round number of 12, and a key length of 16 bytes are recommended as the nominal choice of parameters for RC5.

We implemented a RC5 encryption and decryption library for use in our LES testbed. The library was derived heavily from published reference code for the algorithm. Various modifications were added to make it more suitable for our purposes, however the core algorithm functionality was not modified in any way.

In order for the intermediate systems to be able to handle a large volume of traffic, its decryption and re-encryption has to be extremely efficient and fast. Therefore, rather than simply using 3DES with different keys on different parts of the packet, we use RC5 on the packet headers instead. RC5 is known for its speed in encryption/decryption. In order to do this we implemented an RC5 library for use both by the IPSEC gateways as well as intermediate systems.

### D. LES module for Intermediate Systems

We also implemented a dynamically loadable module for use in intermediate systems which might want to access the packet header information from a flow encrypted using LES enhanced IPSEC. This module has two components. The input component is responsible for removing the the ESP tunnel headers and decrypting the IP and TCP headers from the original packet. All applications that sit in between this input stage and the output stage see the actual packet header and can perform any required operations on them. The output

component of the module is the inverse operation of the input stage, the IP and TCP headers are re-encrypted using RC5 encryption, the IPSEC IP and ESP headers are replaced and the packet is sent out.

## V. Results and Measurements

### A. Applications

In order to demonstrate the feasibility of the LES approach, we implemented sample applications. We demonstrated LES for firewalls as well as for a network monitoring tool like TCPdump. In both cases only minimal changes were required to implement correct functionality. We were successfully able to demonstrate correct firewall functionality on an encrypted flow using LES.

We performed the following experiment. The intermediate system is as shown in Figure 3 is configured as a firewall. When IPSEC is not used the firewall works correctly. As an example we add a rule to block all telnet traffic. The firewall correctly blocks the attempt to telnet from one end-host to the other. However, now if IPSEC is enabled at the security gateways, the telnet session is permitted.

Next, we run our enhanced IPSEC code which implements LES on the security gateways, and our LES module for the intermediate system on the firewall. The LES module decapsulates the IPSEC ESP packet, decrypts the header information and passes the internal packet to the firewall. Therefore, in this case, the firewall rule is correctly able to determine that a telnet session has been requested and blocks the connection attempt.

We were also successfully able to demonstrate correct operation of a network monitoring and analysis tool such as TCPdump. LES was used at the IPSEC gateways, and we slightly modified TCPdump on the intermediate system to decrypt packets before attempting to analyze them. We were correctly able to examine and record information regarding not only the IP tunnel between the IPSEC gateway, but information from the IP and TCP headers of the tunneled packets as well.

### B. Timing Analysis

An important factor to consider when analyzing a new scheme is its practicality. In this section we present some measurement results from our implementation, and explain some design choices made during our implementation.

For the purposes of timing measurements we use the time stamp counter on the Pentium chip. The RDTSC (Read Time Stamp Counter) instruction is a two byte instruction, that returns the number of clock cycles since the CPU was powered up. Therefore, by reading this counter twice we can compute the number of cycles that have elapsed between the first and the second call. This provides us with a much more accurate and meaningful measurements of time, which are independent of the CPU clock speed.

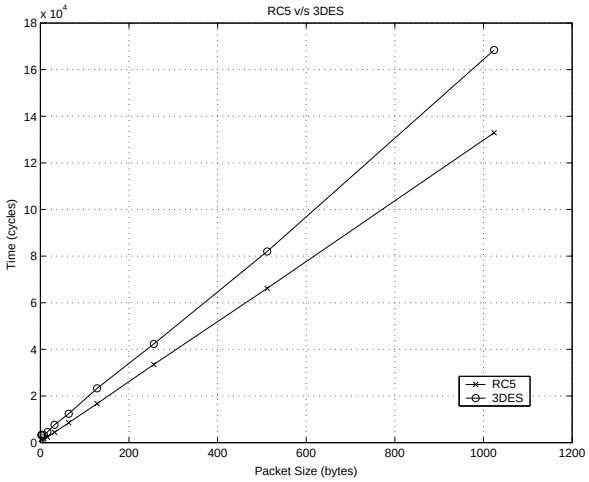


Fig. 4. Comparison of encryption times for RC5 versus 3DES

| Packet Size (bytes) | RC5 (cycles) | 3DES (cycles) |
|---|---|---|
| 2 | 1285.992 | 3217.761 |
| 4 | 1587.342 | 3347.378 |
| 8 | 1587.568 | 3207.518 |
| 16 | 2324.53 | 4516.135 |
| 32 | 4388.633 | 7625.232 |
| 64 | 8497.234 | 12351.9 |
| 128 | 16706.51 | 23253.11 |
| 256 | 33512.03 | 42314.46 |
| 512 | 66183.67 | 820002.76 |
| 1024 | 132944.6 | 168445.6 |

TABLE I
Average Number of Cycles to Encrypt a Packet using RC5 and 3DES

*1) 3DES v/s RC5:* An important consideration when implementing LES is what encryption algorithm to use for the different parts of the packet. The default FreeS/WAN implementation uses 3DES as its basic encryption mechanism. But RC5 is known for being quick.

Figure 4 shows a comparison of the time taken to encrypt a block of data using the 3DES algorithm and the RC5 algorithm. Measurements were taken for various block sizes, ranging from 2 to 1024 bytes. The measurements are in numbers of cycles and can be easily converted to seconds by dividing by the clock speed of the CPU, which was 166MHz in our case. As these encryption algorithms operate on small portions of data at a time, the time taken to encrypt increasingly larger blocks grows linearly. However, it is clear that the RC5 encryption(RC5 32/12/16 with 32 bit word size, 12 rounds, and 16 bytes of key length) takes much less time than 3DES(16 bytes of key length). The data from our measurements is summarized in Table 1.

However, the relative security offered by RC5 versus 3DES for same size key lengths have not been evaluated in the literature; though RC5 claims to be as secure as 3DES. Therefore, keeping in mind the faster speed of RC5, and weighing it against its unproven strength, our LES implementation, we decided to use RC5 for encryption of the packet headers and maintain the default use of 3DES for

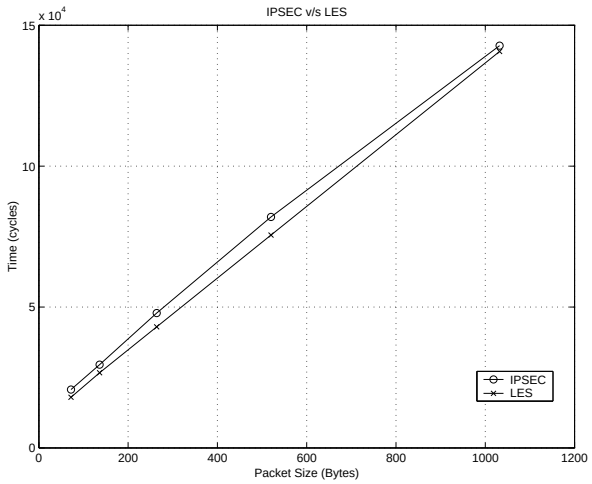


Fig. 5. Comparison of time required for normal IPSEC encryption versus LES

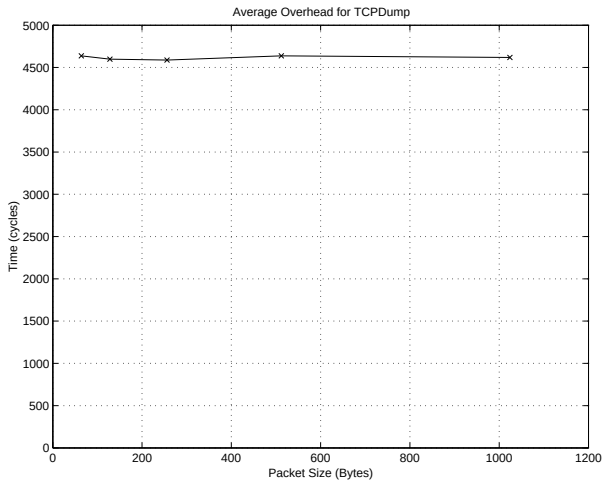


Fig. 6. Average Number of Cycles added to TCPdump by header decryption

| Packet Size (bytes) | Normal IPSEC (cycles) | Layered IPSEC (cycles) |
|---|---|---|
| 72 | 20713.26 | 17963.3 |
| 136 | 29555.53 | 26677.53 |
| 264 | 47864.7 | 42948.4 |
| 520 | 81971.09 | 75522.94 |
| 1032 | 142753.5 | 140684.2 |

TABLE II
AVERAGE NUMBER OF CYCLES TO ENCRYPT A PACKET IPSEC V/S LES-IPSEC

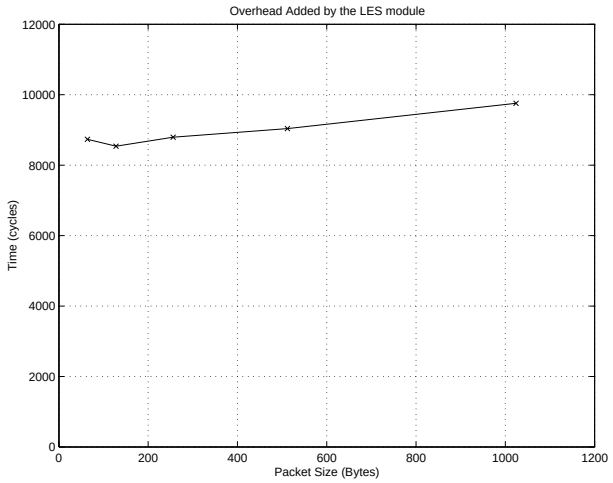


Fig. 7. Average Number of Cycles added by IPSEC module for intermediate nodes

encryption of the data portion of the packet.

This is a favorable design choice, as the faster speed of RC5 means that it is possible to decrypt and re-encrypt the packet header in the middle of the network, much more efficiently. This makes the LES much more practical.

*2) Overhead at End Points:* In order to justify our approach as a viable modification, we also have to show that we do not introduce a substantial delay when compared with an unmodified IPSEC Gateway. For this purpose we constructed an experiment where we first measured execution time for encrypting a packet entirely with 3DES as normal IPSEC does and then measure the time required to use LES(using both RC5 and 3DES) on a packet. We repeated this experiment for ping packet sizes of 64, 128, 256, 512 and 1024 bytes. An 8 byte header is added to our payload size specified. Our results are shown in Figure 5. As can be seen from the data in Table 2. The time measured in cycles to perform LES on a packet(with RC5 and 3DES) is slightly less than the time required to perform 3DES on the entire packet. RC5 is a much faster algorithm than 3DES as shown by our previous experiment. However, since it is used on such a small portion of the packet(only 20 bytes), the gains from its faster speed are not very large. Nevertheless, the difference is sufficient to offset the overhead of 2 function calls that LES must make for encryption, versus 1 function call for normal IPSEC.

Therefore, we see that implementing LES enhanced IPSEC has not added significant overhead to an IPSEC gateway. It should be noted that in this case the overhead has been offset by the use of much faster RC5 algorithm. If 3DES were to be used on both parts of a packet in LES, this would not be the case.

*3) Overhead at Intermediate Gateways:* There can be two types of intermediate nodes. The first kind are passive and do not need to allow packets to pass through them. The second kind are those that do allow packets to pass through them, with or without modification. For the first kind, the overhead comes simply from the need to decrypt the packet headers. There is no need to re-encrypt as these nodes are only passive observers of traffic.

The data from our measurements for the two different cases are summarized in Table 3. Figure 6 shows the overhead incurred by a traffic monitoring node that is running our enhanced TCPdump. The additional decryption overhead added is constant as irrespective of packet size; only the same amount of packet header bytes need to be decrypted.

Figure 7 shows the overhead of using LES on an intermediate node that needs to modify packet headers. In this case the overhead is dependent on the packet size as the

| Packet Size (bytes) | TCPdump (cycles) | LES Module (cycles) |
|---|---|---|
| 64 | 4637.133 | 8732.88 |
| 128 | 4598.633 | 8537.08 |
| 256 | 4587.333 | 8793.82 |
| 512 | 4637.133 | 9038.70 |
| 1024 | 4617.933 | 9755.92 |

TABLE III
OVERHEAD AT INTERMEDIATE NODES

additional processing required for handling LES includes a memory allocation for creating a new packet(packet headers need to be added). This operation is dependent on the size of the packet. However, even for packet sizes ranging from 64 bytes to 1K, the number of cycles only increases from 8500 to 10,000. This provides a good indication of how much overhead is added by the LES module.

## VI. CONCLUSION

In this paper we have presented the concept of Layered Encryption Security (LES). We argue that layering adds much needed flexibility to any security framework. We have implemented the concept of LES on our experimental testbed. Measurements from our implementation indicate not only the feasibility of this approach, but also serve to demonstrate its capability. We have demonstrated how the use of LES addresses some of the important problems currently being faced by network designers; the need to balance security and efficiency in a network. In the future we hope to examine modifications that might be needed to key distribution schemes to implement LES.

## ACKNOWLEDGMENT

The material presented in this paper is based upon work supported by the National Aeronautics and Space Administration(NASA) under award Number NCC8-235. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA.

## REFERENCES

[1] L. Vepstas. Linux network address translation. http://linas.org/linux/load.html, November 1998.
[2] V. Bharadwaj, J. Baras, and N. Butts. Internet service via broadband satellite networks. *Proceedings of the SPIE, vol. 3528, 169-180.*, pages 169–180, February 1999.
[3] V. Bharadwaj. Improving TCP performance over high-bandwidth geostationary satellite links. Master's thesis, Center for Satellite and Hybrid Communication Networks(CSHCN), University of Maryland, College Park, Maryland, 1999.
[4] Y Zhang. Multi-layer protection scheme for IPSEC. http://www.wins.hrl.com/people/ygz/ml-ipsec/draft-zhang-ipsec-mlipsec-00.txt, October 1999.
[5] Y. Zhang and B. Sing. A multi-layer ipsec protocol. *9th USENIX Security Symposium*, pages 113–128, Aug 2000.
[6] S. Bellovin. Proable plaintext cryptanalysis of the ip security protocols. *Proceedings of the 1997 Symposium on Network and Distributed Systems Security*, 1997.
[7] Linux frees/wan overview. http://www.xs4all.nl/ freeswan.
[8] R. Rivest. The RC5 encryption algorithm. *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption*, pages 86–96, 1994.