

Mobile Multi-Layered IPsec

Heesook Choi, Hui Song, Guohong Cao, and Tom La Porta
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {hchoi,hsong,gcao,tlp}@cse.psu.edu

Abstract—To achieve high throughput in wireless networks, smart forwarding and processing of packets in access routers are critical for overcoming the effects of the wireless links. However, these services cannot be provided if data sessions are protected using end-to-end encryption as with IPsec, because the information needed by these algorithms resides inside the portion of the packet that is encrypted, and can therefore not be used by the access routers. A previously proposed protocol, called Multi-layered IPsec (ML-IPsec) modifies IPsec in a way so that certain portions of the datagram may be exposed to intermediate network elements, enabling these elements to provide performance enhancements. In this paper we extend ML-IPsec to deal with mobility and make it suitable for wireless networks. We define and present performance measurements of an efficient key distribution protocol to enable fast ML-IPsec session initialization, and two mobility protocols that are compatible with Mobile IP and maintain ML-IPsec sessions. Our measurements show that, depending on the mobility protocol chosen, integrated Mobile IP/ML-IPsec handoffs result in a pause of 56-105 milliseconds, of which only 31-85 milliseconds may be attributed to ML-IPsec. Further, we provide detailed discussion and performance measurements of our ML-IPsec implementation. We find the resulting protocol only marginally reduces throughput compared to scenarios in which IPsec is used (4%), and when coupled with SNOOP, greatly increases throughput over scenarios using standard TCP over IPsec (165% on average).

I. INTRODUCTION

Data confidentiality and integrity are two critical issues for wireless, mobile networks. These issues are of growing importance as wireless service providers attempt to increase wireless data traffic by providing mobile VPN services. The most widely accepted method for ensuring data confidentiality and integrity is to pass encrypted data end-to-end using a mechanism such as IPsec [12].

For wireless networks, smart forwarding and processing of packets are also critical for overcoming the effects of the wireless links, especially highly variable delay and error rates. Several studies have shown that techniques such as smart scheduling with respect to the type of data being sent and regulation of TCP acknowledgment information, can greatly improve end-to-end performance in a wireless network [1], [5]. However, these services cannot be provided if end-to-end encryption is used, such as in IPsec, because the information needed by these algorithms resides inside the portion of the packet that is encrypted, and can therefore not be used by mobile routers.

Previous work, called Multi-layered IPsec (ML-IPsec) [20]

applies a modified version of IPsec so that certain portions of the user information may be exposed to particular intermediate network elements in a route. In this way, portions of a datagram may be encrypted end-to-end, while portions may be read and operated upon by network elements providing performance enhancements. However, the ML-IPsec as defined in [20] is designed for static environments and does not examine mobility.

In this paper we extend ML-IPsec to deal with mobility and make it suitable for wireless networks. We call our resulting protocol *Mobile ML-IPsec (MML-IPsec)*. We make the following four contributions: 1) we define and present performance measurements of an efficient key distribution protocol to enable fast MML-IPsec session initialization; 2) we define and present performance measurements of two mobility protocols that maintain MML-IPsec sessions; 3) we provide detailed discussion and performance measurements of our MML-IPsec implementation to quantify its performance impact compared to non-secure communication and communication using IPsec; and 4) we provide a detailed discussion and performance measurements of our implementation of SNOOP, and SNOOP executing over MML-IPsec to quantify the benefits of using MML-IPsec to enable performance enhancing algorithms in a wireless environment.

Our measurements in a wireless environment show that, depending on the mobility protocol chosen, integrated Mobile IP/ML-IPsec handoffs result in a pause of 56-105 milliseconds, of which only 31-85 milliseconds may be attributed to MML-IPsec. We found MML-IPsec only marginally reduced throughput compared to scenarios in which no encryption is used (9%), or those in which IPsec is used (4%), and when coupled with SNOOP, greatly increased throughput over scenarios using standard TCP end-to-end (50% on average), or using TCP over IPsec (165% on average). Our conclusion, based on these results, is that MML-IPsec is a worthwhile protocol to pursue because it enables large performance improvements while providing end-to-end secure transfer of user data.

The rest of this paper is organized as follows. In Section II we present an overview of related and previous work, including a description of ML-IPsec. In Section III we discuss our design of MML-IPsec, our model for integrating Mobile IP with IPsec and MML-IPsec, the software platform on which we base our implementation, and the test bed used to evaluate the performance of the protocols. In Section IV we present

our key distribution protocol, our two mobility protocols, and characterize their performance. In Section V we present our implementations of MML-IPsec and SNOOP and their performance. Section VI concludes the paper.

II. BACKGROUND

Several studies have shown that the performance of classic data communication protocols can be quite poor when used over wireless links. In particular, the performance of TCP, the reliable Internet transport protocol, can be degraded by the loss and delay characteristics of a wireless link. Consequently, there have been several efforts aimed at improving the performance of TCP on wireless links. Two of the more promising works do not require any modifications to TCP, but instead perform smart processing (forwarding, filtering, scheduling) on TCP/IP packets based on information gleaned from observing packet flows. In [1], the authors show that by snooping on TCP/IP packets at the wireless edge, determining when packet loss has occurred by detecting duplicate acknowledgments, and performing fast local retransmissions, TCP performance can be greatly improved.

More recently in [5], it was uncovered that TCP performance is adversely affected by the highly variable delay experienced on the 3G wireless links. The effect is that TCP acknowledgments sent on the uplink tend to be compressed causing them to arrive at transmitters back-to-back. The compression of acknowledgments results in transmitters sending bursts of data. These bursts of data can overflow buffers at the wireless edge resulting in high packet loss. The solution proposed is to regulate the flow of acknowledgments to ensure that buffers do not overflow.

In both of these proposals, the node at the wireless edge must observe information in the TCP header to execute their algorithms. The need to have nodes inside the network examine packet payloads to perform smart packet processing or perform packet classification is in direct conflict with the current Internet model of security implemented by the IPsec protocol suite [12], [10], [11]. IPsec supports a variety of operational modes including packet authentication, packet encryption, or both. In the most secure mode, tunnel mode, the entire IP packet is encrypted and encapsulated with a new IP packet header. Therefore, intermediate nodes in the network do not have access to the original IP header information, nor the information contained in any of the transport layer or application layer protocols. This precludes the network from performing smart packet processing and packet classification to improve end-to-end performance.

There are several possible solutions to this problem. Protocols such as TLS [6] and SSL [7] provide security above the transport layer. With their use, user payloads are encrypted, but the TCP and IP headers are in the clear. Therefore, intermediate nodes may access required information to perform many of the performance enhancements discussed above. The main drawback is that the TCP and IP header information is in the clear throughout the entire network allowing for possible eavesdropping to determine communication patterns

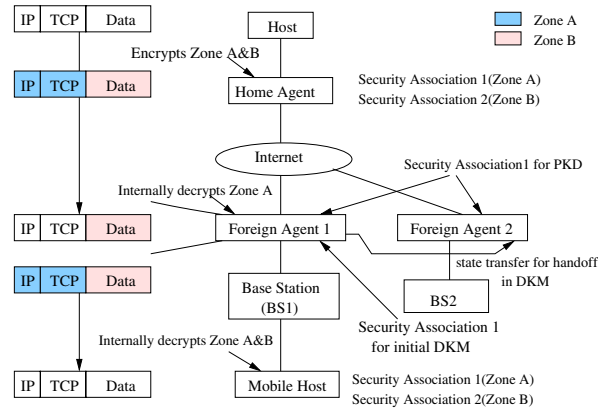


Fig. 1. Mobile Multi-Layered IPsec Example.

and traffic characterizations. Also, these protocols do not enable application layer packet classification for protocols such as RTP.

A more flexible solution is defined in the ML-IPsec protocol [20]. This protocol allows a user to define zones within an IP packet. Each zone is encrypted and authenticated with its own security association (SA). Each zone may be accessed (decrypted) by different network elements. This requires SAs to be established between a client and several nodes in a network, each of which can decrypt a certain portion of the IP packet while being unable to view the entire packet.

For example, consider the wireless network of Figure 1. In this example, the corporate firewall acts as a Mobile IP Home Agent (HA). Foreign Agent (FA) 1 requires access to TCP/IP header information to perform smart packet processing. Using IPsec, secure communication would entail running an IPsec tunnel between the HA and Mobile Host (MH), in which case FA1 would not have access to the TCP/IP header information. Using ML-IPsec, this header information would be included in Zone A which is accessible to FA1. However, the user payload would be placed in Zone B which is not accessible to FA1. In this way, the user information is protected end-to-end and the TCP/IP header information is protected from all nodes except FA1 which may perform smart packet processing.

While ML-IPsec is a promising start, it has limitations and several unknowns. First, it requires that SAs (secret keys, algorithms, parameters, etc.) be established between multiple elements for a single data session. This requires an efficient key distribution algorithm which has yet to be defined. Second, mobility is not supported. The mobility requires that new SAs be established as a mobile host moves during a data session. For example, in Figure 1, if the mobile host moves from base station 1 (BS1) to BS2, SA1 must move from FA1 to FA2. These modifications must be performed quickly so that sessions are not disrupted during a handoff. This is more complex than mobility in basic Mobile IP [16] because in this MML-IPsec system, multiple SAs exist that operate on the user data.

Third, there is no data available on the performance trade-offs between the overhead of supporting multiple zones versus the benefits of packet classification or smart packet process-

ing. Specifically, mobile access routers, e.g., FAs, will have hundreds of flows passing through them, so the overhead of the key distribution and initialization, handoffs, and per packet processing, must be kept low to achieve high performance.

IKE [8] supports key distribution and mutual authentication between two nodes but requires extensions to support the multiple SAs used in ML-IPsec and is not suitable for mobility. Support for key distribution in mobile networks is the focus of [18]. An efficient method of key distribution and authentication between a home network, security server in a foreign network, and a mobile host is presented. However, this work does not address the distribution of multiple keys required for a ML-IPsec, does not account for mobility, and does not provide any implementation or performance insights.

III. MOBILE MULTI-LAYERED IPSEC - SERVICES AND SOFTWARE PLATFORM

In the following subsections we present background on ML-IPsec and how to make it suitable for wireless networks; then we propose solutions to integrate IPsec and mobile IP; the last two subsections describe the software platform on which we base our implementation, and our test bed.

A. Protocol Services

The original ML-IPsec [20] is defined to allow network layer packets to be segmented into zones, each of which is protected, i.e., encrypted, authenticated, or both, independently. Corresponding hosts have access to all zones and can therefore authenticate and decrypt the entire packets. Selected intermediate nodes are given access to one or more selected zones, and may therefore decrypt and authenticate only these portions of the packet. Before communication can commence, a set of SAs, called a composite security association (CSA), must be established, one for each zone in each node for which access to the zone is permitted. As defined in [20], the number of zones and the number of intermediate nodes with access to at least one zone are not limited. Also, there is no limit on the number of zones in a packet, and zones are not required to cover contiguous bits in a packet.

We have somewhat restricted the definition of ML-IPsec to meet the needs of the known methods of enhancing wireless system performance, while keeping the processing complexity low. First, we limit the number of allowed intermediate nodes to a single node, specifically the Mobile IP FA. We choose this node because the vast majority of wireless enhancements operate on a node close to, or supporting a wireless link, and do not require changes to any other portion of the network. Second, we limit the number of zones to two, one for the packet header and one for the payload. The rationale is that most algorithms require access to TCP/IP header information, and not packet payload. This restriction can be easily relaxed. Finally, we define zones as contiguous portions of the packet to ease processing.

In addition to these changes, we have also defined a key distribution protocol for MML-IPsec and two mobility protocols (Section IV).

B. Integrating Mobile IP, IPsec, and MML-IPsec

We assume that if the basic IPsec is used, the IPsec tunnel extends between the HA and the MH. If MML-IPsec is used, the MML-IPsec tunnel includes the HA and the MH, while the FA has access to the header part of the TCP/IP packet. In addition, we assume reverse tunneling [15] is used for data transmitted from the MH so that packets in both directions are consistently encrypted.

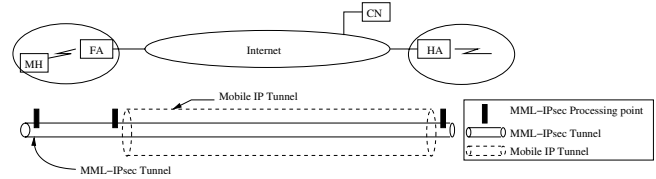


Fig. 2. Integrating Mobile IP and MML-IPsec/IPsec.

Several research projects [2], [3] have proposed solutions to integrate Mobile IP and IPsec. SecMIP [3] is based on the configuration in which the MH uses the DNS/DHCP service to get the COA (Care-of-address). Without using the FA, the MH gets a new collocated COA using DHCP. SecMIP uses the IPsec tunnel to protect the Mobile IP tunnel. While this is a simple scheme to provide security in Mobile IP, the handoff delay is high because the MH must re-establish the IPsec tunnel on every handoff.

SMN[2], by Portland University, establishes an IPsec tunnel between the HA and the MH. When the MH moves to the foreign network, it has to register its COA to the HA. However, the established IPsec tunnel prevents the Mobile IP registration message from reaching the FA because every packet, including the Mobile IP registration packet, is encrypted with IPsec. In order to solve this problem, the client software is changed so that the Mobile IP packet is not encrypted. The re-establishment of IPsec tunnel on every handoff will degrade the end-to-end performance because it incurs high handoff latencies.

We propose a different integration model for Mobile IP and IPsec which is then largely re-used for integrating Mobile IP and MML-IPsec. This model is similar to SMN, but does not require changes to the client software, and does not require IPsec tunnels to be re-established after each handoff. To enable Mobile IP registration messages to be received by the FA when an IPsec tunnel is in place between the MH and HA, we add an additional routing entry in the MH and leverage the fact that route selection chooses the route having the longest prefix match among multiple matched entries. When an agent advertisement is received by a MH, it adds a route entry for the FA. The new route entry specifies the FA address as the gateway for all packets destined to the FA. After adding this route, the Mobile IP registration message addressed to the FA will match the new route, and therefore be sent directly to the FA, instead of using the old entry through which packets are encrypted.

To eliminate the need to re-establish IPsec tunnels after each handoff, we leverage the fact that when using Mobile IP, while

the COA of the MH changes after each IP layer handoff, the IP addresses of the MH and HA remain constant. Therefore, the IPsec tunnel may remain intact. In our model, during Mobile IP registration, the HA simply updates the routing entry for IPsec packets destined to the MH to be forwarded through the new Mobile IP tunnel. This does not require any message beyond the standard Mobile IP registration.

Figure 2 depicts this integration model which can be applied to both IPsec and MML-IPsec. The MML-IPsec tunnel is established between the HA, FA and MH, within the Mobile IP tunnel. An advantage of our model is that it does not restrict the scheme of obtaining a COA as that in SecMIP, and provides a seamless integration between Mobile IP and MML-IPsec.

Let's first consider the the data traffic flow from the MH to the correspondent node (CN). The FA receives the encrypted traffic through the ML-IPsec tunnel or IPsec tunnel. The outer IP header has the source address as the MH IP address, and the destination address as the HA IP address. The FA routes this data traffic into the Mobile IP tunnel based on the reverse tunnel of the Mobile IP. Next, let's consider the traffic from the CN to the MH. This traffic is intercepted by the HA. The HA encrypts the traffic and encapsulates it within the Mobile IP tunnel, which has the outer header source address as the HA IP address, and the destination address as the FA IP address. If IPsec is used, when receiving the encapsulated traffic, the FA decapsulates the Mobile IP outer header and transmits the encrypted traffic to the MH. If ML-IPsec is used, the intermediate node decrypts the first zone (packet header), performs some processing, re-encrypts this zone, and forwards the data.

C. Software Platform

Our implementation is based on Linux FreeS/WAN version 1.99, an open source IPsec implementation available free on the web (under GNU license term), on Linux kernel version 2.4.21. The FreeS/WAN system has two major components: the Pluto Daemon and the Kernel IPsec Support (KLIPS). The Pluto Daemon implements the IKE protocol [8]. We modified portions of the Pluto Daemon to implement our key distribution protocol, called Multi-Layer IKE (ML-IKE) and to support the interface between ML-IKE and the MML-IPsec transport module. We made major modifications to KLIPS to integrate IPsec and SNOOP.

D. Test Bed

To evaluate the performance of our protocols, we set up a test bed as shown in Figure 3. The base stations are Dell Pentium desktops (P4 2.4Ghz), and the MHs are Dell Pentium laptops (Mobile P4 2.4Ghz). These are equipped Orinoco Prism 802.11b wireless cards configured in ad hoc mode so that the desktop machines act as base stations. All these machines are running RedHat 9.0 with Linux kernel version 2.4.21.

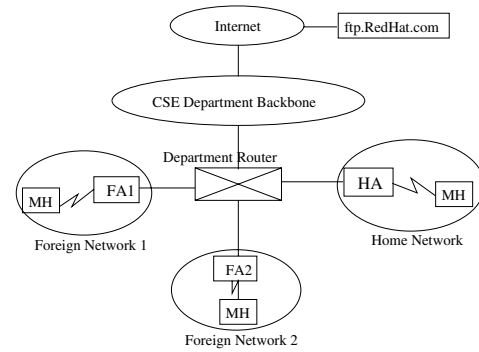


Fig. 3. Mobile Multi-Layered IPsec Test Bed.

IV. KEY DISTRIBUTION AND MOBILITY MANAGEMENT

In this section we present efficient automatic key management protocols for MML-IPsec integrated with Mobile IP. We include procedures for session initialization (ML-IKE) and mobility management. Our goal is to enable fast handoffs while maintaining MML-IPsec sessions. In our model, the nodes involved in the MML-IPsec session are the MH, HA, and FA. The MH and HA have access to both zones in the MML-IPsec packets; the FA serves as the intermediate node and has access to the first zone of the packet containing the TCP/IP header. The key management protocols are responsible for establishing the required SAs between these nodes, and for enabling mobility.

The key management protocols have two phases. In the first phase, a MML-IPsec session is initialized using ML-IKE. This includes determining if a FA will be involved in the secure session, and hence requires the use of MML-IPsec. The second phase of the protocols supports mobility. We propose two protocols for this purpose. The first, called *Proactive Key Distribution (PKD)*, pre-establishes SAs with not only the current FA, but its neighbors as well. Therefore, when a MH moves to a new FA, the SA already exists. The second, called *Dynamic Key Migration (DKM)*, requires SAs to migrate between FAs as a user moves.

In the following subsections we first discuss ML-IKE, PKD, and DKM, in detail, and then present our implementation and performance results.

A. Initialization

When a MH leaves its home network, it executes Mobile IP registration procedures. In addition, ML-IKE is invoked. Figure 4 shows the initialization flows between the HA, FA and MH, with the Mobile IP registration.

The initialization phase begins after the HA has sent the Mobile IP registration reply to the FA. First, the HA establishes an ISAKMP SA [13], [8] with the FA and MH so that session key information may be exchanged securely. We use the Main Mode procedures defined in IKE [8] for this purpose (flow (5) and (6) in Figure 4). Note that the establishment of the ISAKMP SA between the FA and HA occurs in parallel with sending the Mobile IP registration reply to the MH.

The second step of the initialization is to establish the CSAs in the MH, HA and FA. The CSA has two elements, a

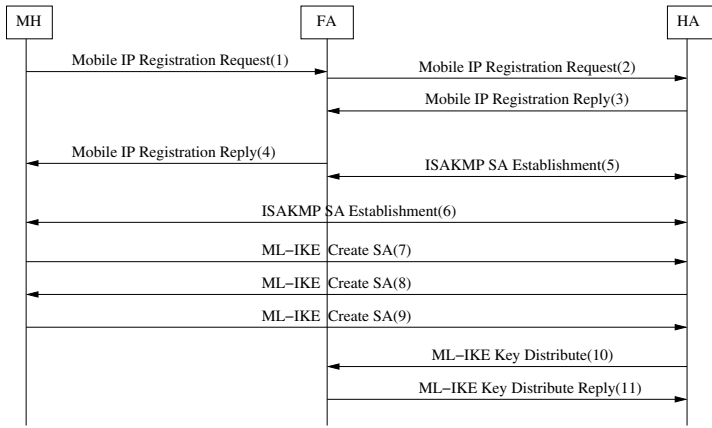


Fig. 4. Mobile IP Registration and MML-IPsec Key Initialization.

zone map and a zone list. The zone map states the start and stop positions of the zones in the IP datagram. The zone list contains the SAs for all the zones. The HA, FA, and MH create and keep an instance of the CSA. The source and destination (HA and MH) store a complete list of SAs. On the other hand, the FA has a non-null SA in the zone list for the zone that it supports, and a null SA for the zone that it does not support.

The HA and MH setup a MML-IPsec CSA using the ML-IKE Create SA Exchanges (flows (7)-(9) in Figure 4). During these exchanges, the MH and HA exchange the complete zone map and SAs to compose a CSA. We define a new payload type called “Zone Map” which delivers the zone map information. In addition, we modify the key exchange protocol to allow for multiple SAs to be included. The secret key values for all zones are decided in the MML-IPsec Create SA Exchanges.

Once the CSA is established between the MH and HA, the HA delivers the CSA to the FA, using the ISAKMP SA with the FA. For the zone to which the FA has access, i.e., the zone covering the TCP/IP header, the HA sends the corresponding non-null SAs to the FA (flow (10) - (11) in Figure 4) with the corresponding symmetric key values. A new payload, called “SECRET”, delivers the symmetric key values.

Upon completion of the ML-IKE procedure, data transmission using MML-IPsec may take place.

B. Proactive Key Distribution (PKD)

The goal of PKD is to enable a fast handoff by pre-distributing keys in FAs that are neighbors of the current FA, so that very little overhead is incurred during the real-time handoff. For example, in Figure 1, SA1 is placed in both FA1 and FA2 when the session is established. The distribution of the CSA information to these neighboring FAs is performed after the ML-IKE exchange is complete, so initialization overhead will not be increased. The disadvantage of this approach is that the active key information must be stored in more nodes than are actively being used, thus creating a higher chance of the session key being compromised.

Figure 5 shows the PKD protocol flow. The FA, when finished initialization, notifies the HA of its neighbor FAs (1) in Figure 5). The HA distributes the MML-IPsec CSA

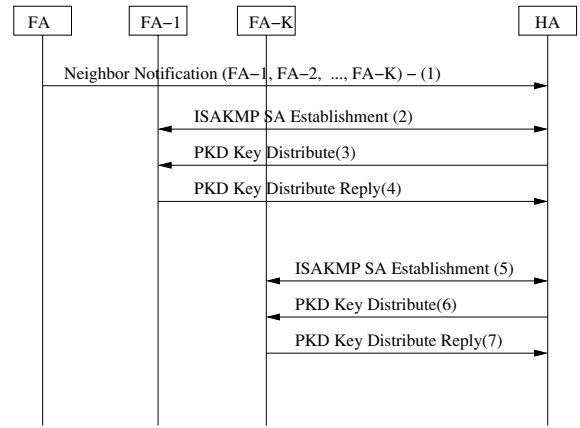


Fig. 5. Proactive Key Distribution Protocol Flow.

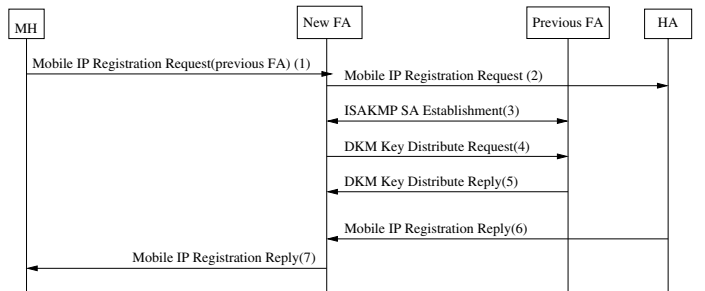


Fig. 6. Directed Key Migration Protocol Flow.

information established via ML-IKE to the neighbor FAs. The HA establishes an ISAKMP SA to each neighboring FA to transmit ML-IPsec CSA securely (flow (2) and (5) in Figure 5).

PKD can be performed in two ways: (a) point-to-point sequential key distribution; (b) multicast key distribution. In this paper, we use the former to simplify the implementation.

When the MH moves to a new FA, the handoff latency is low because the MML-IPsec CSA information is already loaded in the new FA. When the new FA receives a Mobile IP registration reply from the HA, the FA internally activates the MML-IPsec CSA. The HA only changes the internal binding of the MML-IPsec tunnel to the new Mobile IP tunnel with the neighbor FA.

C. Directed Key Migration (DKM)

Unlike PKD, in DKM the CSA information is only stored in the FA that is actively serving the MH. Therefore, when a MH changes FAs, the CSA must be migrated from the old FA to the new FA in a secure manner. For example, in Figure 1, SA1 must be moved from FA1 to FA2. This method only requires that the CSA be stored in a single intermediate node, but incurs a higher latency than PKD because more signaling is required during the handoff.

Figure 6 shows the DKM protocol flow. When a MH moves to a new FA, it detects the movement using standard Mobile IP techniques. We modify the Mobile IP Registration [16] by adding an extension to include the previous FA address, as is done in Mobile IP with Route Optimization [17].

First, the MH transmits a Mobile IP registration message with the previous FA information to the new FA (flow (1) in Figure 6). The new FA uses the previous FA information to decide where to retrieve the MML-IPsec CSA information. The new FA initiates the DKM protocol, and relays the Mobile IP registration message to the HA simultaneously.

In DKM, if there is no ISAKMP SA established between the previous FA and the new FA, the new FA establishes an ISAKMP SA with the previous FA so the key information is transferred securely. Once this ISAKMP SA is established, the new FA transmits the MML-IPsec CSA information request to the previous FA (flow (4) in Figure 6). The previous FA authenticates the new FA and sends the response to the new FA (flow (5) in Figure 6). The response message includes the ML-IPsec CSA including the secret key values.

Note that the DKM protocol is processed in parallel with the Mobile IP registration between the new FA, HA and MH.

D. Rekeying and Revocation

There are several reasons why rekeying or key revocation may take place when using MML-IPsec. For example, if a CSA lifetime expires, or a CSA is determined to be insecure, it may be revoked, or if secure communication is still desired, rekeying may take place during which a new CSA is established. Further, key revocation may take place when a Mobile IP tunnel is deleted, for example when a MH returns to its home network or powers off. Finally, rekeying may take place when using MML-IPsec if the number of FAs that share the CSA exceeds a threshold value due to mobility.

IKE [8] defines rekeying procedures so that peers can initiate the establishment of a new SA. We propose to reuse this approach for operation on MML-IPsec CSAs. In this approach new CSAs are established while the old CSA is active. Once the new CSA is established, the MH, FA and HA change to use the new CSA for data transfer, and the old CSA is deleted. While we did not implement the rekeying procedures, by establishing the new CSA while communication is ongoing, we expect the disruption due to rekeying to be small.

E. Implementation

We implemented the key exchange protocols (ML-IKE, PKD and DKM) on the test bed shown in Figure 3. The Dynamic Mobile IP Linux implementation developed by Helsinki University of Technology (HUT) [9] was used in the test bed. Each subnetwork has a different set of wireless configuration parameters such as *ssid* and *channel number*. The handoff occurs when the wireless configuration parameters are changed. We manually trigger handoffs through a shell script to run controlled experiments.

The implementation consists of several blocks: ML-IKE, PKD, DKM, Key Management, and the interface to MML-IPsec which will be described in the next section. The communication between these blocks is via the Unix Domain Socket in Linux.

ML-IKE manages, negotiates, and establishes the MML-IPsec CSA for initialization, while PKD and DKM support

mobility. ML-IKE also manages the state machine of the protocols. To implement these protocols, we extended the IKE of FreeS/WAN [19], which is implemented in the Pluto Daemon, to include the new exchange types described previously. For the ISAKMP SA establishment, Pluto was used without any modification.

The Key Management keeps the establishment state of both ISAKMP SA and MML-IPsec CSA based on the source and destination addresses. Using the connection status, it coordinates Mobile IP and ML-IKE. If there is already an established MML-IPsec CSA in the case of PKD, the key management activates MML-IPsec through adding the route entry and IPsec binding. Otherwise, it initiates the establishment of the ISAKMP SA if necessary.

To implement the interface between the key exchange protocols and the MML-IPsec module, we modified the Pluto Daemon by adding a new user interface to construct the CSA. The interface constructs a newly-defined zone message and sends the message to the PF_KEYv2 [14] socket. PF_KEYv2 is a new socket protocol family used by trusted privileged key management applications (e.g., ML-IKE, PKD and DKM) to communicate with the operating system’s key management internals (i.e., FreeS/WAN’s Security Association Database (SADB)). We have modified the PF_KEYv2 source code to handle the newly-defined zone message. The PF_KEYv2 socket will construct the CSA when it receives a zone message.

F. Performance

In this section, we discuss the performance of the key management protocols measured on our test-bed. We tested the performance of our integrated IPsec/Mobile IP solution as a baseline.

In Table I, we show the message processing time measured by the *gettimeofday* system call in Linux. The results in Table I measures the processing time on receiving a message or event, excluding the pre-processing, post-processing and transmission time over the media. The node initiating the message flow is the “Initiator,” and the recipient is called the “Responder”. For example, in DKM, the Initiator is the new FA and the Responder is the previous FA providing the CSA information. In ML-IKE, the MH is the Initiator and the HA is the Responder.

TABLE I
MESSAGE PROCESSING TIME

Message Flow	Processing Time		
	Initiator	Responder	Total
ISAKMP SA	46.0 ms	48.0 ms	94.0 ms
ML-IKE Create SA	48.2 ms	46.8 ms	95.0 ms
ML-IKE Key Distribution	3.3 ms	0.2 ms	3.5 ms
PKD Key Distribution	2.8 ms	0.2 ms	3.0 ms
DKM Key Distribution	9.0 ms	1.2 ms	10.2 ms
Neighbor Notification	0.6 ms	8.3 ms	8.9 ms

In order to measure the handoff latency, we evaluate the time delay from the point that the MH sends a Mobile IP solicitation message until it finishes establishing the MML-IPsec CSA or IPsec SA. Table II shows the handoff delay for the pure Mobile IP, Mobile IP integrated with IPsec, and

Mobile IP integrated with MML-IPsec. For the last two cases, it includes the overhead of establishing the ISAKMP SA.

The Mobile IP registration is necessary for all cases. The pure Mobile IP handoff comprises only the Mobile IP registration overhead and shows almost the same delay for the initialization and handoff phases. In the case of Mobile IP with IPsec, the initialization time includes the Mobile IP registration and IPsec secure connection establishment between the HA and MH.

Similarly, the initialization of MML-IPsec takes between 650 - 665 milliseconds depending on if PKD or DKM is used. This includes the Mobile IP registration and MML-IPsec establishment with the key distribution to the FA or FAs. The MML-IPsec initialization time is made up of three components. First, the message processing time, including ISAKMP SA establishments, ML-IKE Create SA Exchanges, ML-IKE Key Distribution, and Mobile IP registration procedures, is measured at approximately 311 milliseconds. Second, the FA runs four shell scripts to create local connections and bind the local connections into MML-IPsec interfaces when receiving the Mobile IP registration reply. The four shell scripts take 320 milliseconds to execute. Finally, there are transmission and internal communication latencies which are responsible for the remaining of the delay. These shell commands, and their additional overhead, may be eliminated if the source code for the various initialization procedures are modified to interact directly, a change planned for our next version of ML-IKE.

In Table II, we show the total latency for initialization, and the latency (in parenthesis) if the shell script overhead is eliminated.

TABLE II
HANDOFF DELAY

Phase	Pure MIP	MIP with IPsec	MIP with ML-IPsec	
			PKD	DKM
Initialization	26 ms	302 ms	650(330) ms	665(345) ms
Handoff	25 ms	54 ms	56 ms	105 ms

Note: MIP is an abbreviated form of Mobile IP.

The handoff delay measurements show that a handoff using PKD incurs an additional 31 milliseconds of delay, while a handoff using DKM incurs an extra 85 milliseconds of delay. These results are encouraging when we consider that these are well within the range of a TCP time-out value.

V. MML-IPSEC IMPLEMENTATION AND EVALUATION

In this section, we first present our detailed implementations of the MML-IPsec protocol and SNOOP [1], and their integration. Then, we present the experimental methodology and the performance evaluation results.

A. Implementation Details

1) *The Implementation of MML-IPSEC*: We follow the high-level design outlined in [20] to implement MML-IPsec. As discussed in Section III, we modified this design to make it suitable for wireless networks, and to account for the fact

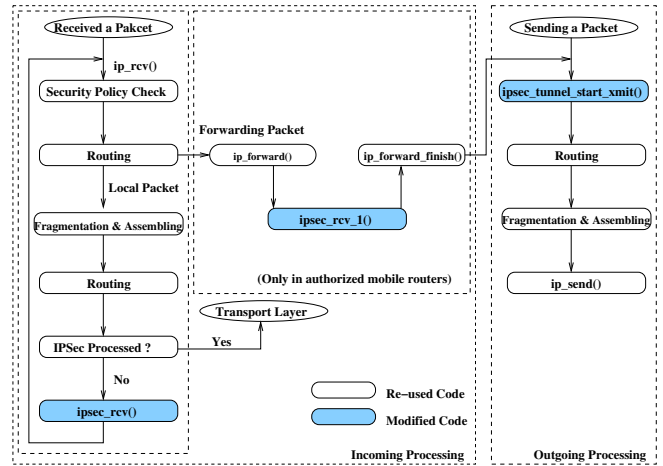


Fig. 7. Implementation of MML-IPsec

we used Linux FreeS/WAN version 1.99 while in [20], version 1.1 was used.

In KLIPS, we introduce two new concepts as discussed in Section III and [20]: a zone and a CSA. A CSA has two elements: a zone map and a list of SAs for all the zones.

One SA in the SA list is chosen to be the designated SA. The designated SA is responsible for maintaining parameters at the IP datagram level and representing the CSA in security processing [20]. The designated SA must be consistent across all nodes involved in a CSA, and all nodes must have access to the corresponding zone. In our case, all the authorized nodes, including end hosts and authorized FAs, have access to the first zone; i.e., the TCP/IP header portion of the packet. Thus, we always choose the first zone's SA as the designated SA.

We create two new data structures, called zone and subzone, to accommodate the zone concept. The data structure for SAs is modified to accommodate the concept of CSA. This is done by including two new fields in the SA data structure *ipsec_sa*: One is a pointer to the zonemap for the CSA, and the other is a flag showing whether the SA is a designated SA.

Figure 7 shows the modifications to the FreeS/WAN source code for realizing MML-IPsec. We modify three procedures: *ipsec_rcv()*, *ipsec_rcv_I()*, and *ipsec_tunnel_start_xmit()*. The first two is used when a packet is received, and the last one is used to send a packet. *ipsec_rcv()* is used for local packet processing, and *ipsec_rcv_I()* is used for forward packet processing. When the end host/gateway receives a MML-IPsec packet destined for itself, it uses *ipsec_rcv()* to decrypt the packet before forwarding it to the transport layer. This requires information about the two zones in the packet.

If an authorized FA receives a MML-IPsec packet for which it has a valid SA, it uses *ipsec_rcv_I()* to decrypt the first zone of the packet. At this point, the FA may perform any smart processing, which may need the information contained in the packet header. Once this is done, the first zone of the packet is re-encrypted and the packet is forwarded to the next hop.

For outgoing processing when sending a packet, *ipsec_tunnel_start_xmit()* is modified to perform encryption and authentication on two zones instead of the entire packet.

2) *The Implementation of SNOOP*: The SNOOP protocol was first defined in [1]. SNOOP executes in a base station and monitors the TCP header within the packet it is forwarding. The idea behind SNOOP is to detect errors incurred by the wireless link at the base station, and perform local retransmissions to recover from the errors locally. In this way, the TCP sender will not see the transmission error, resulting in a larger average window size and hence higher throughput. SNOOP detects errors by observing duplicate acknowledgments. If a duplicate acknowledgment is observed, SNOOP retransmits the subsequent TCP segment from its local buffer, and deletes the duplicate acknowledgment from the traffic stream. In this way, the lost segment is recovered without an end-to-end TCP retransmission. For details, please refer to [1].

The SNOOP protocol was first implemented in BSD/OS 2.0. Because our environment, as described in Section III, is very different from that used in the initial implementation, SNOOP needs to be re-implemented. First, the original version of SNOOP was implemented in user space. Because our goal is to integrate SNOOP with MML-IPsec, which is implemented in the kernel, we have to re-implement SNOOP in the Linux kernel.

Second, BSD/OS 2.0 uses a different timer management system, which is not supported in Linux. BSD provides some timer management functions, such as *timeout()*, which is a user space implementation and can be called directly, but it is not available to kernel programs. We designed a complete set of timer management functions in Linux, including functions for initializing, setting, starting, stopping, clearing, and resetting the timer, and functions that execute a timeout action when the timer expires.

Third, BSD uses the *mbuf* data structure and related functions to manage the memory buffers used by the kernel's networking subsystem. In Linux, a new data structure called *sk_buff* and a set of related functions are used for this purpose. We carefully re-designed SNOOP using the features of the *sk_buff* data structure. For example, in order to avoid copying the whole data packet when caching the packet at the base station, we use *pskb_copy()* to copy and save the data packet, which is similar to the reference counting mechanism in *mbuf*. In this way, we also get a private *sk_buff* header in which we can modify the TOS (Type of Service) field of the IP header to send a locally retransmitted packet with a higher priority.

Finally, in order to measure the performance in a controlled environment, we implemented a Poisson-distributed bit-error model to generate errors, similar to that in [1]. The MH generates errors based on the distribution. If an error is generated, the checksum of the data packet is changed, and this packet will later be dropped.

3) *Integration of MML-IPsec and SNOOP*: The integration of MML-IPsec and SNOOP occurs at the authorized intermediate routers. In our test bed, these are the routers acting as base stations and FAs.

Figure 8 shows how the Linux kernel is modified to integrate MML-IPsec and SNOOP. From the figure, we can see that before the packet is forwarded, *ipsec_rcv_1()* is called to

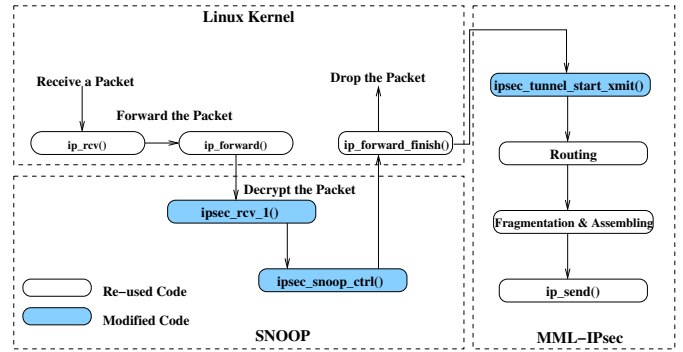


Fig. 8. Integration of ML-IPsec and Snoop

decrypt the first zone of the packet. The *ipsec_snoop_ctrl()* function is then called, which has access to the TCP/IP headers in plain text, and the SNOOP protocol is executed. If the packet should be forwarded, the first zone will be re-encrypted before it is transmitted to the next hop.

B. The Experimental Setup

We use the test bed shown in Figure 3 to evaluate the performance of MML-IPsec and SNOOP. We transferred a large file from a fixed host to a MH with SNOOP running on the base station. A 5MB file was transferred from ftp.redhat.com to the MH, with and without SNOOP running on the base station. Figure 9 shows the results for an average of five runs for different bit error rates, comparing the performance of TCP Reno and TCP Reno with SNOOP. These results match closely with that in [1].

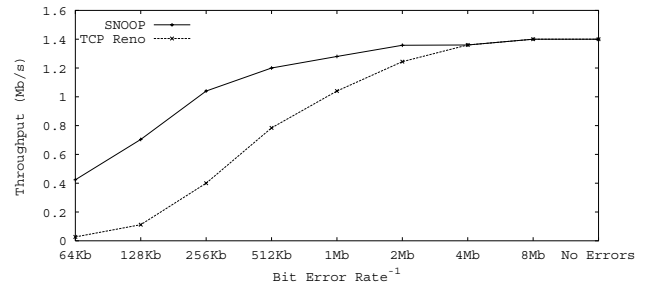


Fig. 9. Throughput Comparison of SNOOP and TCP Reno

Since we cannot implement MML-IPsec in ftp.redhat.com, and we want to run experiments in a controlled environment, we modify our test bed as shown Figure 10. The laptop, named TLP1, acts as a MH. One desktop, named NETS, is configured as the base station for the MH. Another desktop, named METS, is configured as a FTP server. NETS and METS are connected using a 100Mbps Ethernet. MML-IPsec is installed in TLP1, NETS and METS. We installed NIST Net [4], a network emulation package that runs on Linux, in KNICKS. NIST Net allows a Linux PC to be set up as a router to emulate a wide variety of network conditions. In our experiments, we use it to insert packet transmission delays between NETS and METS to emulate a wide area network. For example, to emulate downloading a file from a remote ftp site (e.g., ftp.redhat.com) with 50ms delay, we insert 50ms

transmission delay using NIST Net. As shown in Figure 10, all the traffic between TLPI (the ftp client) and METS (the ftp server) are configured to route through KNICKS, where packet transmission delays are inserted.

SNOOP is installed in NETS and is integrated with MML-IPsec. We limit the raw bandwidth of the wireless link to 2Mbps so that we can compare the results to that in [1]. The TCP data packet size in our experiments is 1,460 bytes when MML-IPsec is not running. When MML-IPsec is running, because of the additional IPsec headers (such as ESP header and IP over IP header if IPsec tunnel mode is in use), the TCP data packet size is smaller than 1,460 bytes. For example, in our experiments, where we use the tunnel-mode ESP protocol to encrypt the packet, the TCP data packet size is 1,391 bytes.

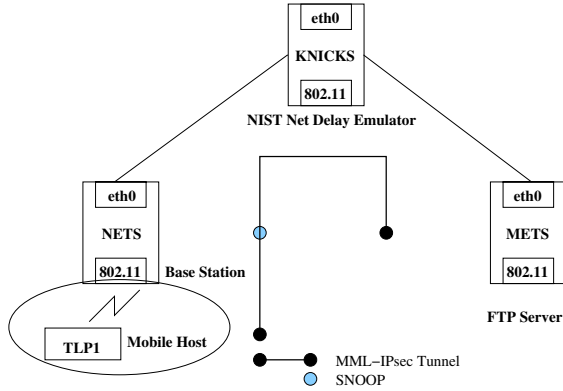


Fig. 10. Testbed

1) *System Parameters:* We list the test bed’s configuration parameters in Table III and IV. Table III shows the parameters we use to test SNOOP, and Table IV shows the parameters in evaluating MML-IPsec. In Table III, the SNOOP Maximum Window, which is the data buffer size in the SNOOP module at the base station, is set to 50. This is large enough to buffer all the data packets sent from the sender so that the SNOOP module is never overloaded.

In the SNOOP module, three timers are running: the local retransmission timer, the persist timer and the garbage timer. The SNOOP Initial RTO (Retransmission TimeOut) is used to set the initial timeout value for the local retransmission timer. The persist timer will expire if the SNOOP module does not receive any packets from either the receiver or the sender for a long period of time. When it expires, it will retransmit all the un-acknowledged data packets in its buffer. We set the persist timer to 1 second. The garbage timer is used to clear the buffer for a connection if there is no activity in that connection for a very long period of time. We set the garbage timer to 10 seconds in our implementation.

When testing MML-IPsec, we establish a MML-IPsec connection between TLPI (the MH) and METS (the ftp server). The packets transmitted in this connection are encrypted using the tunnel-mode ESP encryption protocol. The IP packet is divided into two zones each of which is encrypted separately. As shown in Table IV, the first zone is 40 bytes, which includes the 20 byte IP header and 20 byte TCP header (not including

TABLE III
SNOOP-RELATED TCP MODULE PARAMETERS

Maximum Segment Size (bytes)	1,460
Sender Buffer Size (bytes)	65,535
Snoop Maximum Window	50
Snoop Maximum Connections	64
Fast Transmit	Enabled
Fast Recovery	Enable
Selective ACK (SACK)	Disabled
Karn’s Algorithm	Enabled
Snoop Retransmission Threshold	4
Snoop Initial RTO (ms)	100
Snoop Minimum RTO (ms)	100
Snoop Persist Timeout (ms)	1,000
Snoop Garbage Timeout (ms)	10,000
NIST Net Transmission Delay (ms)	100

TABLE IV
MML-IPSEC PARAMETERS

IPsec Tunnel Mode	Enabled	
Encryption	Enabled	
Authentication	Disabled	
ML-IPsec Overhead (bytes)	69	
Maximum Segment Size (bytes)	1,391	
Number of Zones	2	
	First Zone	Second Zone
Zone Size (bytes)	40	Variable
Encryption Protocol	ESP	ESP
Encryption Algorithm	3DES	3DES

the TCP Options and Padding fields); and the second zone covers the remainder of the packet. Both zones are encrypted with the ESP 3DES encryption algorithm using different SAs, each of which has different encryption/decryption keys. NETS has the SA for the first zone and decrypts the TCP/IP header of the packets so that it can execute SNOOP if enabled.

Five configurations have been tested: pure TCP Reno, TCP Reno over SNOOP, TCP Reno over IPsec, TCP Reno over MML-IPsec, and TCP Reno over SNOOP integrated with MML-IPsec. To evaluate these configurations under different wireless link conditions, we use the Poisson-distributed bit-error model to generate a wide range of bit-error rates. We test these configurations by transferring a 10MB file from METS (the ftp server) to TLPI (the MH) with different bit-error rates. For each configuration, we execute the transfer five times. Our comparison is based on the average throughput of the five runs.

C. Performance Evaluation Results

We use throughput as the metric to evaluate the performance of different configurations. To further explain the reason behind these results, we also illustrate the TCP congestion window size and the TCP sequence number of different configurations.

1) *Throughput:* Figure 11 compares the throughput of different configurations under different bit-error rates. From the figure, we see that SNOOP greatly improves the performance of TCP Reno when the error rate is high. For example, when the bit-error rate is 1.53×10^{-5} (1-bit error in 64K bits), the throughput with SNOOP is three times higher than that without SNOOP. In fact, TCP Reno with SNOOP always achieves a

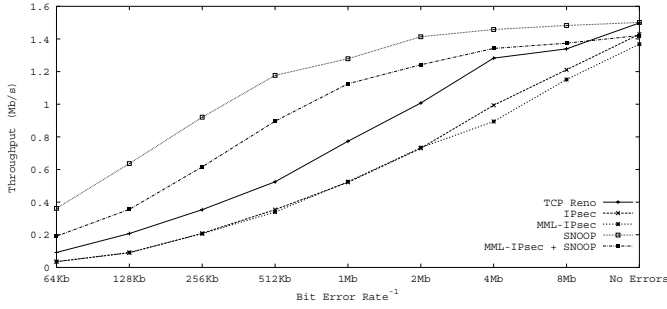


Fig. 11. Throughput Comparison of Different Configurations

higher throughput regardless of the bit-error rate. Even when the bit-error rate is very small, e.g., 1.19×10^{-7} (1-bit error in 8Mb), SNOOP achieves 10% higher throughput than regular TCP Reno.

From the curves of IPsec and MML-IPsec, we can see that when no error exists, the overhead of these two protocols are about 5% and 9%, respectively. As the bit-error rate increases, the throughput of both protocols drops dramatically. It is interesting to see that when the bit-error rate is higher than 4.77×10^{-7} (1-bit error in 2Mb), the throughput of IPsec and MML-IPsec is almost the same in most cases. For example, when the bit-error rate is 1-bit error per 2Mb, the throughput of both approaches drop by 27%. When the bit-error rate is 1-bit error per 64Kb, their throughput drops by 62%. This is because the overhead incurred by the error is more significant than the overhead incurred by encryption/decryption.

When SNOOP is integrated with MML-IPsec, the throughput is higher than either TCP Reno over MML-IPsec or IPsec. SNOOP integrated with MML-IPsec can deliver as much as five times higher throughput than that without SNOOP. Even when the error rate is low, SNOOP integrated with MML-IPsec can improve the performance by 19% and 14% compared to TCP over MML-IPsec and IPsec, respectively.

From the figure, we can also see that in an error-prone environment, SNOOP over MML-IPsec always achieves higher throughput than TCP Reno alone. The improvement increases as the bit-error rate increases. Figure 11 shows that SNOOP over MML-IPsec increases the throughput by 3% to 108% for different bit-error rates.

These results conclusively show that in a wireless error-prone environment, by integrating performance enhancing algorithms such as SNOOP with MML-IPsec we can achieve security and performance simultaneously.

2) *TCP Sequence Number*: Figure 12 shows the evolution of TCP sequence number versus time with different configurations when the bit-error rate is 1.9×10^{-6} (1-bit error in 512k bits). From the figure, we can see that if SNOOP is running, the sequence number progresses much faster than that without SNOOP. For instance, when SNOOP is integrated with MML-IPsec, the sequence numbers increase at twice the rate of TCP with IPsec.

3) *TCP Congestion Window Size*: Figure 13 shows the size of the congestion window versus time at the ftp server with a bit-error rate of 1.9×10^{-6} (1-bit error in 512K bits). This

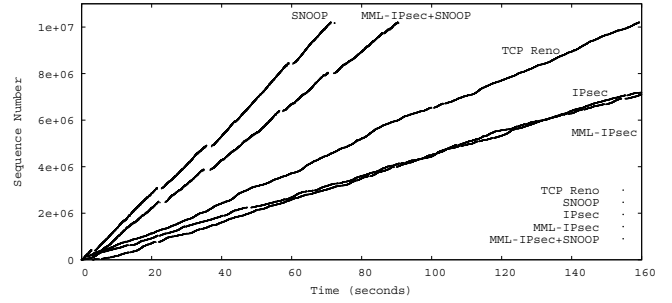


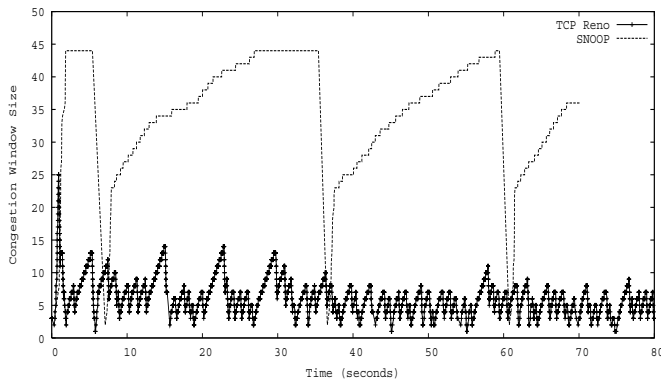
Fig. 12. Sequence Number Evolution of Different Configurations

figure compares three configurations: pure TCP Reno, TCP Reno over SNOOP, IPsec, MML-IPsec, and SNOOP integrated with MML-IPsec.

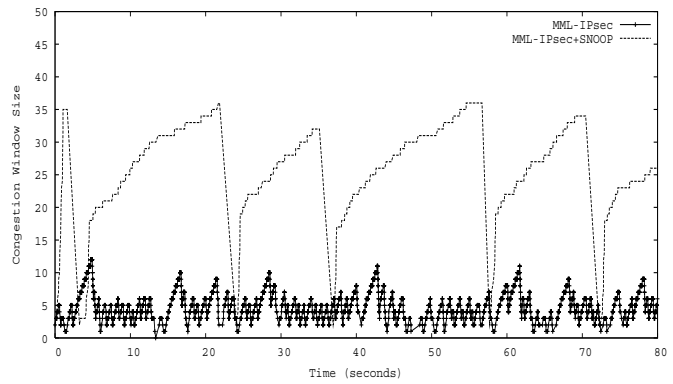
From the Figure 13 (a), we see that SNOOP maintains a much larger congestion window than pure TCP Reno. On average, the congestion window size of SNOOP is about 35 while TCP Reno's is about 6. This can be explained as follows. In SNOOP, the base station caches all the data packets before forwarding them to the MH over the wireless link. When the base station detects a packet loss, it will retransmit the packets to the MH from its local cache and suppress the duplicate ACKs. In this way, the sender's congestion control mechanisms, such as fast retransmit and fast recovery, will not be invoked. Thus, SNOOP prevents the congestion window from shrinking. In the case of pure TCP Reno, wireless loss will be treated as congestion in the network. Whenever a packet is lost or three duplicate ACKs are received, the transmitter will drop the congestion window size to half and then increase the window size gradually.

Figure 13 (a) also shows that the congestion window size of SNOOP drops to two on several occasions. This is because of timeouts at the sender. Sender timeout occurs when the congestion window is very large and there are several packet losses in the same sending window. When the congestion window size is large, for example 40, a packet loss will generate as many as 39 duplicate ACKs if all the packets after the lost packet are received correctly. Our tests shows that to process all these duplicate ACKs requires up to ten times of the normal packet processing time. Therefore, packets transmitted after the lost packet will experience a long round-trip time and it is possible that the timer for these packets will expire before getting acknowledged. In such cases, the sender will fall back to the slow start phase and reduce the congestion window to two.

From Figure 13 (b), we can also see that when integrated with MML-IPsec, SNOOP has a much larger congestion window size than pure MML-IPsec, which can be also explained as above. The figure shows that the integration of SNOOP increases the average congestion window size of the MML-IPsec protocol by a factor of five, from 5 to 26. The improvement in the congestion window size shown here also explains the dramatic performance improvement achieved in Figure 11, which shows that the throughput of MML-IPsec with SNOOP is increased by 170% compared to without



(a) TCP Reno vs. SNOOP



(b) MML-IPsec vs. MML-IPsec with SNOOP

Fig. 13. Congestion Window Size

SNOOP, when the bit-error rate is 1.9×10^{-6} (1-bit error in 512K bits).

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a simplified version of MML-IPsec, an efficient key distribution protocol for initializing secure wireless sessions, and two protocols for managing mobility for these secure sessions. We call this suite of protocols MML-IPsec. We showed through extensive performance testing of our implementations of these protocols that MML-IPsec successfully enables performance enhancing algorithms to be introduced into wireless networks. In particular we showed that when SNOOP is integrated with MML-IPsec, impressive throughput gains are achieved over pure TCP, or if a secure session is desired, TCP over IPsec. In addition, we showed that the mobility protocols add only a small amount of delay to the handoff time, significantly less than a typical TCP time-out value. The compromise incurred when using MML-IPsec is that a single intermediate node inside a network is able to access TCP/IP header information in plain text; note that no other node in the network can view this information and that the user payload is still encrypted end-to-end. We believe these results justify the use of protocols such as MML-IPsec and motivate further work to improve the security aspects of this protocol. Specifically, revoking keys in nodes that are no longer in an active route, and supporting rekeying without disrupting data transfer are two areas requiring more work.

REFERENCES

- [1] H. Balakrishnan, S. Seshan, E. Amir and R.H. Katz, "Improving TCP/IP Performance over Wireless Networks," *Proc. of ACM Mobicom*, 1995.
- [2] J. Binkley, "An Integrated IPSEC and Mobile-IP for FreeBSD," *Technical Report 01-10*, 2001.
- [3] T. Braun and M. Danzeisen, "Secure Mobile IP Communication," *Workshop on Wireless Local Networks at the 26th Annual IEEE Conference on Local Computer Networks*, 2001.
- [4] M. Carson and D. Santay, "NIST Net - A Linux-based Network Emulation Tool," *Computer Communication Review*, 2004.
- [5] M. Chan and R. Ramjee, "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," *Proc. of ACM Mobicom*, 2002.
- [6] T. Dierks and C. Allen, "The TLS Protocol," *IETF RFC 2246*, 1999.
- [7] A. Freier, P. Karlton and P.C. Kocher, "The SSL Protocol, Version 3.0," *Netscape Communications Corp.*, 1996.
- [8] D. Harkins and D. Carrel, "The Internet Key Exchange Protocol," *IETF RFC 2409*, 1998.
- [9] Helsinki University of Technology, "Dynamics Mobile IP Software," October 2001.
- [10] S. Kent and R. Atkinson, "IP Authentication Header," *IETF RFC 2402*, 1998.
- [11] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," *IETF RFC 2406*, 1998.
- [12] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," *IETF RFC 2401*, 1998.
- [13] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," *IETF RFC 2408*, 1998.
- [14] D. McDonald, C. Metz, and B. Phan, "PF.KEY Key Management API, Version 2," *IETF RFC 2367*, July 1998.
- [15] G. Montenegro, "Reverse Tunneling for Mobile IP," *IETF RFC2344*, 1998.
- [16] C. Perkins, "IP Mobility Support for IPv4," *IETF RFC 3220*, 2002.
- [17] C. Perkins and D. Johnson, "Route Optimization in Mobile IP," *Mobile IP Working Group Draft draft-ietf-mobileip-optim-09.txt*, 2000.
- [18] L. Salgarelli, M. Buddhikot, J. Garay, S. Patel, and S. Miller, "Efficient Authentication and Key Distribution in Wireless IP Networks," *IEEE Communications Magazine*, 2003.
- [19] www.freeswan.org, "Introduction to FreeS/WAN 1.99," October 2002.
- [20] Y. Zhang and B. Singh, "A Multi-Layer IPsec Protocol," *Proc. of 9th USENIX Security Symposium*, 2000.