A Multilayer IP Security Protocol for TCP Performance Enhancement in Wireless Networks

Yongguang Zhang, *Member, IEEE*

***Abstract*—Transmission control protocol (TCP) performance enhancement proxy (PEP) mechanisms have been proposed, and in some cases widely deployed, to improve TCP performance in all-Internet protocol (IP) wireless networks. However, this technique is conflicted with IP-security (IPsec)—a standard IP security protocol that will make inroad into wireless networks. This paper analyzes the fundamental problem behind this conflict and develops a solution called multilayer IP-security (ML-IPsec). The basic principle is to use a multilayer protection model and a fine grain access control to make IP security protocols compatible with TCP PEP. It allows wireless network operators or service providers to grant base stations or wireless routers limited and controllable access to the TCP headers for performance enhancement purposes. Through careful design, implementation, and evaluation, we show that we can easily add ML-IPsec to existing IPsec software and the overhead is low. We conclude that ML-IPsec can help wireless networks provide both security and performance.**

***Index Terms*—IP-security (IPsec) protocol, network security, transmission control protocol (TCP) performance enhancement proxy (PEP), wireless networks.**

I. Introduction

TRANSMISSION control protocol (TCP) is the standard transport protocol for all-IP wireless networks, including third-generation/fourth-generation (3G/4G) cellular networks, satellite networks, wireless PAN, and mobile ad-hoc networks. However, it is well understood that standard TCP (i.e., TCP-Reno [1]) does not achieve optimal performance when operated over all these wireless networks [2]–[5]. This is because many such networks possess certain characteristics that are "unfriendly" to TCP, such as noncongestion losses [2], [6]–[8], long delays [3], [9], variable bandwidth [4], [10], [11], and dynamic changing topology [5]. Transport-aware link layer mechanisms are often necessary to correct these problems. For example, TCP snooping can drastically improve TCP performance over a lossy wireless link if the base station can inspect every TCP packets and deliberately delay or drop certain ones [7], [8]. Other similar techniques like indirect connections and explict notifications also provide improvements in wireless networks [6], [8], [12]. Furthermore, many satellite networks have deployed TCP spoofing and booster mecahnisms to reduce the impact of latency [9], [13]–[15]. Finally, the TCP performance over a wireless ad-hoc network can be enhanced if intermediate nodes can send explicit link failure notifications to the TCP sender [5], [16].

Collectively, these performance enhancement mechanisms are called TCP performance enhancing proxy (PEP) [17]. It refers to the category of techniques that intermediate nodes in the network interact with the TCP layer and influence its end-to-end behavior.

Take TCP snooping [7], [8], for example, the wireless base station inspects each TCP packet in transit and matches the TCP data packet in one direction with the TCP acknowledgments in the other direction. If packet losses are detected, the base station will retransmit the lost segments and suppress the "loss signals" (such as three-duplicated-acks) from reaching back to the TCP sender. Studies have shown that this improves the performance significantly [8].

Likewise in satellite networks, a PEP agent is installed at the satellite uplink gateway between the Internet and the satellite network. The agent inspects every TCP packets between these two networks. For data packets, the PEP agent generates and sends back "premature" acknowledgments to the TCP senders, without waiting for the data segments to be actually delivered and acknowledged by their intended receivers. These premature acknowledgments are specially formatted to be indistinguishable from the real acknowledgments—except that they come sooner—so as to shorten the perceived round-trip delay. Studies and practices have shown that this technique plays a critical role in the performance of satellite networks [9], [14], [18]–[20]. Because it is nonintrusive and immediately deployable, TCP PEP has been widely deployed in today's satellite networks and considered as the industry's best practice.

However, TCP PEP is conflicted with IPsec, an upcoming standard for secure communications in the Internet. Since IPsec applications and IPsec-enabled hosts are increasingly popular in the Internet, it is inevitable that IPsec will be used in all-IP wireless networks. As we will explain in Section II, if IPsec is used, TCP PEP will not function and as a result the performance of the wireless networks will degrade.

In this research, we first analyze the relationship between TCP PEP and IPsec and point out that IPsec's end-to-end protection model is unsuitable for many advanced networking paradigms that have been developed lately. We further develop a multilayer protection model for IPsec that can be integrated with TCP PEP. Through implementation and performance evaluation, we show that this is a viable solution for providing both security and performance in wireless networks.

Manuscript received February 10, 2003; revised September 26, 2003.

The author is with HRL Laboratories, LLC., Malibu, CA 90265 USA (e-mail: ygz@hrl.com).

Digital Object Identifier 10.1109/JSAC.2004.825993

II. Analysis of the Implication of IPsec in Wireless Networks

A. IPsec and End-to-End Security Protection Model

IPsec is a standard mechanism for providing secure communications over the public Internet [21]. The fundamental concept of IPsec is as follows. Before an IP datagram is sent to the untrustworthy Internet, it is encrypted and/or signed using an IPsec protocol. When it reaches the destination side, the datagram is decrypted and/or verified. There are two IPsec protocols: authentication header (AH) [22] for integrity but without confidentiality, and encapsulating security payload (ESP) [23] for confidentiality with optional integrity and authentication. There are also two modes of use: transport mode for protecting upper layer protocols and tunnel mode for protecting the entire datagram.

The granularity of security protection in IPsec is at the datagram level. IPsec treats everything in an IP datagram after the IP header as one integrity unit. Usually, an IP datagram has three consecutive parts: the IP header (for routing purpose only), and the upper layer protocol headers (e.g., the TCP header), and the user data (e.g., TCP data). In transport mode, an IPsec header (AH or ESP) is inserted after the IP header and before the upper layer protocol header to protect the upper layer protocols and user data. In tunnel mode, the entire IP datagram is encapsulated in a new IPsec packet (a new IP header followed by an AH or ESP header). In either mode, the upper layer protocol headers and data in an IP datagram are protected as one indivisible unit.

The keys used in IPsec encryption and authentication are shared only by the sender-side and receiver-side security gateways. All other nodes in the public Internet, whether they are legitimate routers or malicious eavesdroppers, see only the IP header and will not be able to decrypt the content, nor can they tamper it without being detected. Traditionally, the intermediate routers do only one thing–forwarding packets based on the IP header (mainly the destination address field); IPsec's "end-to-end" protection model suits well in this layering paradigm.

B. Conflicts Between IPsec and TCP PEP

Unfortunately, IPsec is conflicted with TCP PEP. TCP PEP operates on two pieces of state information stored in the headers of a TCP packet. They are TCP flow identification and sequence numbers within the flow. TCP flow identification is used to segregate TCP sessions for each TCP packet. It consists of source and destination IP addresses (both stored in IP header), as well as source and destination port numbers (both stored in TCP header). The sequence numbers are used to match acknowledgments with the data segments. This number is stored in TCP header. Without these two pieces of information, TCP PEP will not function.

When a TCP session is transported by an IPsec ESP protocol, the TCP header is encrypted inside the ESP header. It is, thus, impossible for an intermediate gateway outside sender or receiver's security enclaves to analyze an IPsec header to extract TCP flow identification and sequence number. This will, inevitably, break the TCP PEP mechanisms. The PEP agent cannot obtain the information needed to generate acknowledgments or to retransmit data segments.

IPsec AH protocol also presents a problem for TCP PEP, which often modifies the TCP acknowledgment (ACK) stream in order to influence the TCP behavior. The authentication protection in AH does not allow this mode of operation.

In a wireless network, if IPsec is deployed end-to-end between the Internet server and the user host and if the TCP sessions are protected in this IPsec, the TCP PEP mechanism will not be able to provide performance enhancement. The performance of the wireless network will suffer.

C. Fundamental Limitations of End-to-End Protection

The implication of IPsec is more than TCP PEP and more than wireless networks. Fundamentally, IPsec's "end-to-end" protection model and its strict layering principle are unsuitable for an emerging class of new networking services and applications. Unlike in the traditional minimalistic Internet, intermediate routers begin to play more and more active roles. They often rely on some information about the IP datagram payload, such as certain upper layer protocol header fields, to make intelligent routing decisions. In other words, routers can participate in a layer above IP. In addition to the "transport-aware link layer mechanisms" we discussed earlier, other examples of such services and applications are as follows.

- *Traffic Engineering.* The flow information in IPv4 is encoded in both the IP header and the upper-layer protocol headers, such as TCP or UDP port numbers. Therefore, any mechanism that discriminates between flows inside the network (generally called *flow classification*) will need to access the upper-layer protocol headers. If this is done inside the network (as opposite to classification at end-points), it may potentially conflict with IPsec. Flow classification is essential in providing rich classes of services and quality-of-service (QoS) support. These include flow-based and class-based queueing [24], random early detection (RED) [25], router-based congestion control and policing [26], integrated services [resource reservation protocol (RSVP)] and differentiated services (diffserv) [multiprotocol label switching (MPLS)], etc.
- *Traffic Analysis.* Legitimate network engineers and administrators often need the ability to monitor and analyze traffic from a network to perform variety of tasks like load/traffic control, capacity planning, diagnosis, intrusion detections, and firewalls. These tasks often require the ability to access upper-layer protocol headers within packets, while the advocated proliferation of IPsec encryption can prevent such analysis [27], or limit its granularity (for example, individual nodes or flows are inseparable in a IPsec tunnel between two gateways). The fact that such analysis is both necessary and essential means that we should find a way of accommodating both needs.
- *Application-Layer Proxies/Agents.* Some Internet routers can provide application-layer services for performance gains. For example, an intermediate router can become a transparent web proxy when it snoops through the TCP and then HTTP header of an IP datagram to determine the web page request, and serves it with the web page from the local cache. It is transparent to end-users but boosts

the responsiveness because the delivery paths for web request and data between the intermediate router and the web site server are eliminated.

- *Active Networks.* One step further, the active network architecture is a new networking paradigm in which the routers perform customized computation on the data flowing through them. Conceptually speaking, a single IP datagram can carry not only upper-layer protocol headers and user data, but also "code"—a set of executable instructions to be interpreted by the intermediate routers, for describing, provisioning, or tailoring network resources and services, and to achieve the delivery and management requirements. Obviously, the "code" portion of the IP datagram cannot be protected "end-to-end" under IPsec.

All these mechanisms rely on intermediate network nodes to perform "intelligent operations" based on the information encoded in the IP datagram payload. Although many view them as violations of the layering principle, many do have practical values in real world despite such concerns. However, IPsec advocates end-to-end security that prevents such access. This fundamental conflict makes it a very difficult problem to provide both security and extensibility, in one unified platform.

D. Approaches

The goal of this research is to develop a solution to resolve the conflicts between IPsec and TCP PEP. We first study several other approaches that address this problem.

1) *Replacing IPsec with a transport-layer security mechanism.* This approach is to use a transport-layer security mechanism as an alternative to IPsec to provide security services. The transport-layer mechanism, such as secure sockets layer (SSL) or transport layer security (TLS) [28], operates above TCP and works well with TCP PEP: it encrypts the TCP data while leaving the TCP header in unencrypted and unauthenticated form so that intermediate nodes can make use of the TCP state information encoded in the TCP header. In fact, many Internet applications already implement security with SSL or TLS, such as most web browsers (using HTTPS protocol) and mail programs (using SIMAP and SPOP). However, letting the entire TCP header appear in clear text exposes several vulnerabilities of the TCP session to a variety of TCP protocol attacks (in particular traffic analysis), because the identity of sender and receiver are now visible without confidentiality protection. Further, SSL/TLS works only on TCP, and not on user datagram protocol (UDP), thus, the range of applications is smaller than IPsec.
2) *Tunneling one security protocol within another.* Alternatively, it is possible to tunnel SSL/TLS inside an IPsec ESP–letting SSL/TLS protect the TCP data and IPsec protect the TCP header. However, there is a problem here too because IPsec encrypts both TCP header and TCP payload (SSL/TLS-protected data) as a whole. Thus, the encryption/authentication/decryption has to be done twice on the TCP data part. The intermediate router, for example, must decrypt the entire packet to access just the TCP header information. This is obviously an unnecessary waste of resources.
3) *Using a transport-friendly ESP format.* The transport-friendly ESP (TF-ESP) protocol format proposed by Bellovin of AT&T Research [29] modifies the original ESP protocol to include limited TCP state information, such as flow identifications and sequence numbers, in a disclosure header outside the encryption scope (but authenticated). This approach will work well for some TCP PEP mechanisms such as TCP snooping for wireless networks [8], but it does not suite for TCP spoofing because a write access is needed. To be able to support premature ACK, the TCP state information needs to be placed outside the authentication scope as well. Without proper integrity protection, this can be dangerous. Further, the unencrypted TCP state information is made available universally, including to untrustworthy nodes, which creates vulnerability for possible attacks. In addition, TF-ESP is not flexible enough to support all upper-layer protocols.
4) *Splitting IPsec into Two Segments.* Consider a typical terrestrial wireless network and a typical satellite network in Fig. 1. Each network has three entities: the user (end host), the gateway, and the server (end host), and two segments in the communication path: the wireless network between the user and the gateway, and the public Internet between gateway and the server. The wireless network segment is operated and managed by a network operator. If the users can trust this entity to provide proper security, it can put the PEP agent (at the Gateway) within its realm of trust. For example, the user can use the link-layer security mechanisms (such as cellular or satellite channel encryption) to send datagram over the wireless link, and use IPsec between the gateway and the server. In addition, it can use another IPsec between user and the gateway for additional protection over the wireless link. Obviously, this approach requires a total trust to the network operator.

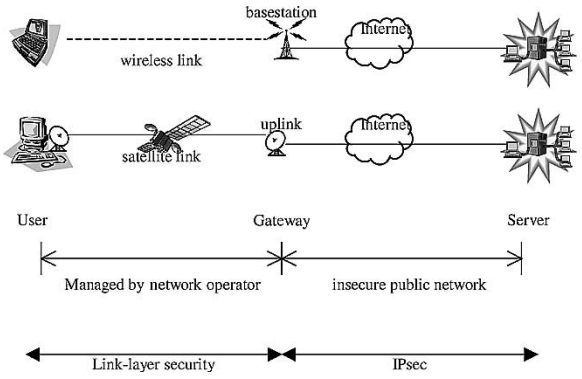

Fig. 1. Realm of trust in a wireless network.

Since all the above approaches have their limitations, we thus propose a fifth approach–to develop a *multilayer security protection scheme* for IPsec. The idea is to divide the IP datagram into several parts and apply different forms of protection to different parts. For example, the TCP payload part can be protected

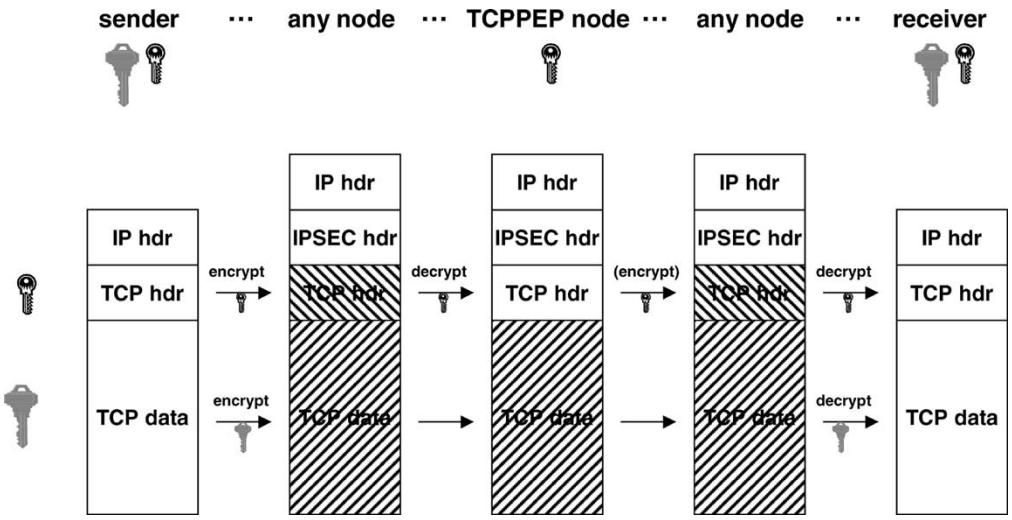

Fig. 2. Multilayer protection model for TCP.

between two end points while the TCP/IP header part can be protected but accessible to two end points plus certain routers in the network. It allows TCP PEP to coexist with IPsec, and provides both performance improvement and security protection to wireless networks. The rest of this paper will describe the principle, the design, and an evaluation of this approach.

III. Principle of Multilayer Security Protection

Our approach is called multilayer IP-security (ML-IPsec). It uses a multilayer protection model to replace the single end-to-end model. Unlike IPsec where the scope of encryption and authentication apply to the entire IP datagram payload (sometimes IP header as well), our scheme divides the IP datagram into *zone*s. It applies different protection schemes to different zones. Each zone has its own sets of security associations, its own set of private keys (secrets) that are not shared with other zones, and its own sets of access control rules (defining which nodes in the network have access to the zone).

When ML-IPsec protects a traffic stream from its source to its destination, it will first rearrange the IP datagram into zones and apply cryptographic protections. When the ML-IPsec protected datagram flows through an authorized intermediate gateway, a certain part of the datagram may be decrypted and/or modified and reencrypted, but the other parts will not be compromised. When the packet reaches its destination, ML-IPsec will be able to reconstruct the entire datagram. ML-IPsec defines a complex security relationship that involves both the sender and the receiver of a security service, but also selected intermediate nodes along the delivery path.

For example, a TCP flow that desires link-layer support from the network can divide the IP datagram payload into two zones: TCP header and TCP data. The TCP data part can use an end-to-end protection with keys shared only between the source and the destination (hosts or security gateways). The TCP header part can use a separate protection scheme with keys shared among the source, the destination, and certain trusted intermediate node (see Fig. 2). This way, no one in the public Internet other than the source, the destination and the trusted intermediate nodes has access to TCP header, and no one other than source and destination (not even the trusted intermediate node) has access to TCP data.

This scheme in effect provides a finer-grain access control to the IP datagram. Since ML-IPsec allows network operators and service providers to grant intermediate nodes limited access to IP datagram contents (such as TCP header), such access must be granted in a secure and controllable way. The identity of the intermediate nodes must be authenticated (using an out-of-band mechanism such as a public-key infrastructure) to prevent any man-in-the-middle attack. After authentication, keys or shared secrets corresponding to the authorized IP datagram zones must be distributed to the intermediate nodes, also using out-of-band mechanisms like Internet key exchange (IKE) [30].

IV. ML-IPsec Design Details

The architecture of ML-IPsec embraces the notion of zones, a new type of security association, the new AH and ESP header formats, and the inbound/outbound processing of ML-IPsec protocol packets. Their designs follow these two objectives: First, ML-IPsec should be fully compatible with the original IPsec in both protocol formats and processing software. Second, ML-IPsec should be based on the data structures and building blocks of IPsec, so that it can be easily added to an existing IPsec implementation. The purpose of these requirements is to reduce the deployment barrier: wherever IPsec can be used, ML-IPsec is also applicable.

A. Zones

A zone is any portion of IP datagram under the same security protection scheme. The granularity of a zone is 1 octet. The entire IP datagram is covered by zones except the IP header in the transport modes. Zones cannot overlap. Using the same TCP example, the portion of the IP datagram that contains TCP header (21st to 40th octet) is zone 1, and the TCP data portion (41st and above octet) is zone 2 (assuming transport mode and no TCP options).

A zone need not be a continuous block in an IP datagram, but each continuous block is called a *subzone*. A *zone map* is a

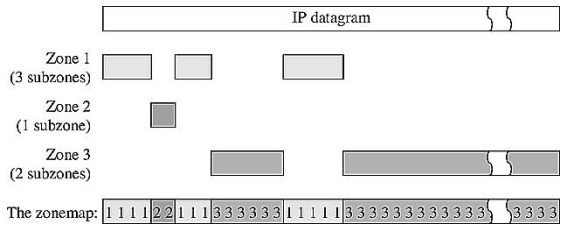

Fig. 3. Zone map sample.

mapping relationship from octets of the IP datagram to the associated zones for each octet. Fig. 3 shows a zone map example.

The zone map is a constant in a security relationship. That is, the zone boundaries in each IP datagram must remain fixed in the lifetime of the security association; otherwise, it will be extremely difficult to do zone-by-zone decryption and authentication. Since IP datagrams are variable in length, the zone that covers the last part of the datagram, usually the user data, should also be variable in size. Zone 3 in Fig. 3 is an example. It is also possible, theoretically, to define a phantom zone that does not correspond to any byte in an IP datagram.

B. Composite Security Association (SA)

1) Security Association (SA) and Composite SA: SA [21] is a key IPsec concept to define a simple security relationship from the sender to the receiver that affords the protection service. ML-IPsec, however, requires a much more complex security relationship to include sender and receiver, as well as the selected intermediate nodes. Since the security service is zone-by-zone, conceptually, we can use an individual security relationship to cover each zone, and then build a composite relationship to cover the entire IP datagram. Mapping this idea to the basic SA concept, ML-IPsec defines a new type of SA called composite SA (CSA). CSA is a collection of SAs that collectively afford a multilayer security protection for the traffic stream.

A CSA has two elements. The first element is a zone map to specify the coverage of each zone in an IP datagram. The zone map must be consistent in all nodes involved in the same ML-IPsec relationship. The second element in a CSA is a *zone list*–a list of SAs, one for every zone. In IPsec, SA contains a set of IPsec operational parameters including cryptographic algorithm, keys, lifetime, etc. However, some of the fields are used differently in ML-IPsec. The following fields, for example, are applicable only on the corresponding zone of the SA:

- lifetime of this SA;
- AH authentication algorithm, keys, etc;
- ESP encryption algorithm, keys, IV mode, IV, etc;
- ESP authentication algorithm, keys, etc.

The other fields have no meanings on the zone level. Therefore, with the exception of a *designated SA* in the zone list, the following fields are not used in other zonal SAs:

- sequence number counter;
- sequence counter overflow;
- anti-replay window;
- IPsec protocol mode;
- path MTU.

The designated SA is a special SA in the zone list, usually the first SA in the list. It takes responsibility to maintain parameters for the IP datagram layer and to "represent" the CSA in security processing. Thus, the designated SA does operate on the above fields as defined in the original IPsec.

2) Access Control in a CSA: A CSA involves the sender, receiver, and all the authorized intermediate nodes that collectively provide a multilayer security protection for a traffic stream. Therefore, an instance of CSA must be created in each of these nodes before the ML-IPsec service can commence. It will have these features.

- The zone map must be distributed and remain the same for all nodes.
- Each CSA instance must have a designated SA and the choice of designated SA must be consistent across all nodes.
- Because the designated SA is responsible for the integrity of the IPsec header (with fields like SPI and sequence number), all nodes must be able to process this SA.

Nevertheless, the zone list need not be the same for all nodes. In principle, each zonal SA independently determines the access list for that zone and not all nodes will have access to all zones. If some node does not have access to a zone, the corresponding zonal SA in the zone list will be null. For a particular zonal SA, an instance must be created in each authorized node as a step in CSA creation. By determining which zonal SA is to be created in which node, CSA enforces a multilayer access control for an IP traffic stream.

Since the designated SA must be consistent across all nodes involved in a CSA, they should all have access to the corresponding zone. For convenience, we call this zone for which the designated SA is chosen the *designated zone*. The requirement that all nodes must have access to one common zone is very natural in most applications; the designated zone is usually the first zone in the list, containing the IP header plus certain upper protocol headers. In rare cases where the zones accessible by intermediate nodes are disjoint, we must define a phantom zone of zero size and make it the designated zone. We can make an SA for this zone and use it as the designated SA. This nonetheless introduces extra overhead because the protocol needs to accommodate one more SA.

3) TCP Example: Here, is an example to illustrate the concept of CSA. It is a traffic flow from sender (the source or the outbound IPsec gateway) to receiver (the destination or the inbound IPsec gateway), passing through gateway (an intermediate node, e.g., PEP agent). We further assume that the desired security service is ESP transport mode. The contents of the corresponding CSA in sender, receiver, and gateway are illustrated in Fig. 4.

C. Protocol Headers

The same security protocol formats, AH and ESP, are used in ML-IPsec. Both AH and ESP have transport mode or tunnel mode, as indicated by the "protocol mode" field of the designated SA. Fig. 5 describes the format for both headers used in ML-IPsec.

The protocol header format for AH in ML-IPsec is almost identical to the original IPsec AH [22], except that the authentication data section in AH is further subdivided into zones. The

Sender or Receiver	Gateway
Zone Map • zone 1 = byte 1-20 • zone 2 = byte 21-EOP (*end-of-packet*)	Zone Map • zone 1 = byte 1-20 • zone 2 = byte 21-EOP
Zone List: • SA1 (designated) – sequence number counter – sequence counter overflow – anti-replay window – protocol mode = TRANSPORT – path mtu – lifetime – ... – encryption algorithm = DES-CBC – encryption key = *key1* – authentication algorithm = HMAC-MD5-32[a] – authentication key = *key2* – ... • SA2 – lifetime – ... – encryption algorithm = 3DES-CBC – encryption key = *key3* – authentication algorithm = HMAC-MD5-96 – authentication key = *key4* – ...	Zone List: • SA1 (designated) – sequence number counter – sequence counter overflow – anti-replay window – protocol mode = TRANSPORT – path mtu – lifetime – ... – encryption algorithm = DES-CBC – encryption key = *key1* – authentication algorithm = HMAC-MD5-32 – authentication key = *key2* – ... • SA2 = NULL

[a]Here HMAC-MD5-32 is a hypothetical keyed hash algorithm that produces a smaller 4-octet signature. Using smaller size authentication data on certain zones (usually the protocol headers) has the advantage of lower overhead. Otherwise, the standard HMAC-MD5-96 can be used.

Fig. 4. Elements of each CSA in sender, receiver, and gateway in a TCP example.

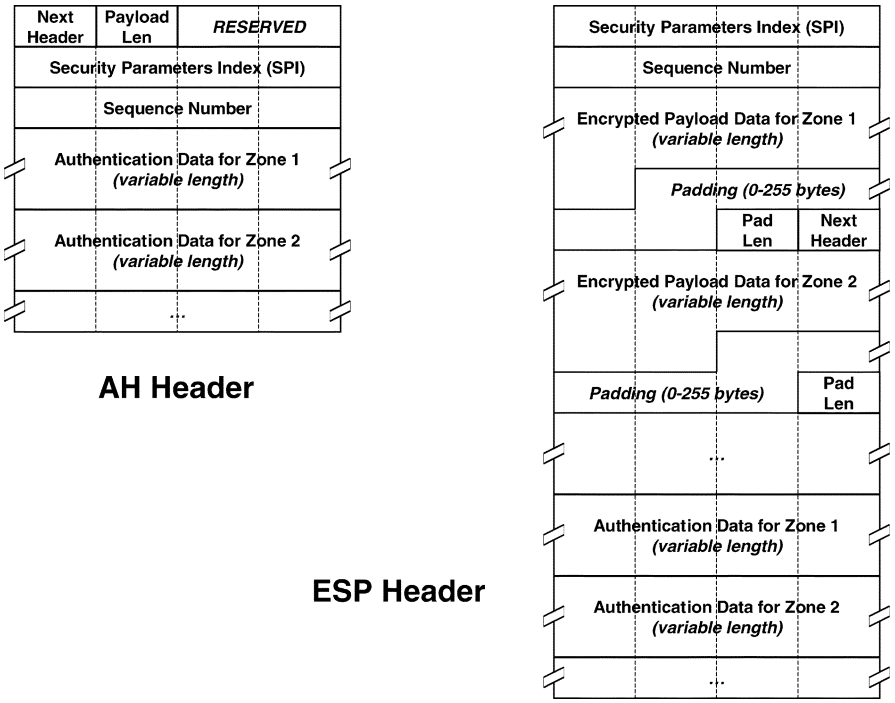

Fig. 5. ML-IPsec protocol header format.

authentication data field is a variable-length field that contains several integrity check values (ICVs) for this packet. The total length of this field is controlled by payload len. The size of each ICV is determined by the authentication algorithm used in each zonal SA, but must be an integral multiple of 32 bits. The boundaries of these zonal authentication data sections can be derived from the CSA.

ML-IPsec is perhaps more useful in ESP, where the IP datagram can be encrypted using different keys in different SAs. The ML-IPsec ESP header format follows the principle in IPsec ESP, but unlike IPsec ESP, the payload data field in ML-IPsec ESP is broken into pieces, one for each zone. The payload data for each zone, together with padding, padding length, and next header field (only in the designated zone), are collectively referred to as the ciphertext block for the zone. The size of each ciphertext block can be determined by the CSA, since all zones except the last one are fixed in size.

Similar to ML-IPsec AH, the optional authentication data field in ESP is also variable in length and contains several ICVs for this packet. The size of each ICV is determined by the

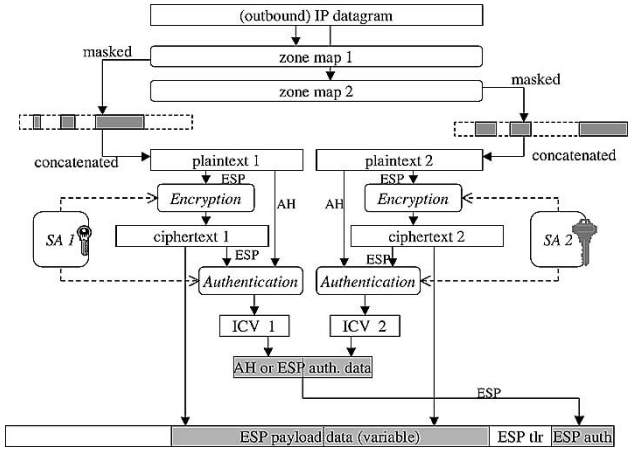

Fig. 6. Examples of outbound ML-IPsec processing.

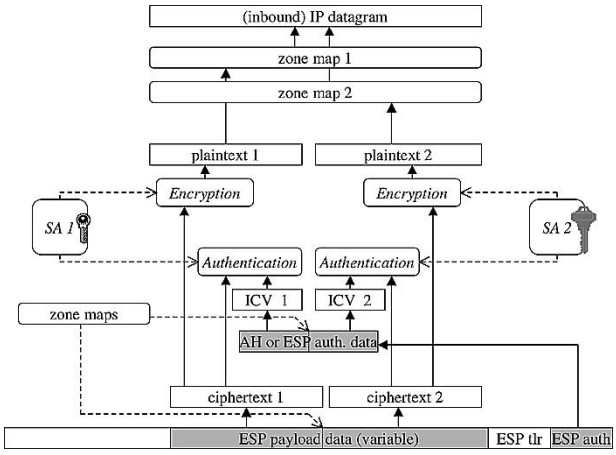

Fig. 7. Examples of inbound ML-IPsec processing.

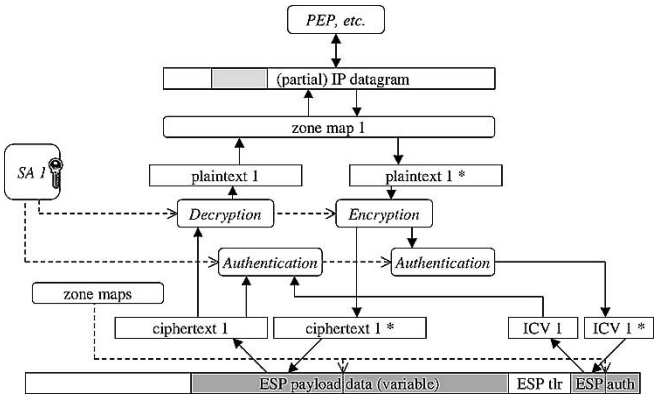

Fig. 8. Examples of partial in–out ML-IPsec processing.

authentication algorithm used in each zonal SA, but must be an integral multiple of 32 bits. The boundaries of these zonal authentication data sections can be derived from the CSA.

D. Inbound and Outbound Processing in ML-IPsec

The inbound and outbound processing of an IP datagram in an IPsec gateway is illustrated through a two-zone example in Figs. 6–8

1) ICV Calculation and Verification: The AH ICV calculation in ML-IPsec is rather different from the original IPsec. For the designated zone, the ICV is computed over the following:

- IP header fields that are immutable in transit;
- the AH header, including next header, payload len, reserved, spi, sequence number, and the authentication data (which is set to zero for this computation), and the optional explicit padding bytes if any;
- all octets in the designated zone.

For other nondesignated zones, the ICV is computed only over the octets of the zone.

The ICV verification during the inbound processing of an ML-IPsec datagram is also done zone-by-zone. A zone is authenticated only if the corresponding zonal SA is nonnull. The ICVs are calculated in the same way as described above, and the values are then matched against the ICVs stored in the Authentication Data.

2) Zone-by-Zone Encryption: On outbound processing, the sender takes the following steps in packet encryption.

- *Zone-Wise Encapsulation*: For each zone, all octets of all subzones are concatenated (in the order they appear in a datagram) and then encapsulated into the ESP payload data field for the corresponding zone.
- *Padding*: The sender adds any necessary padding to each zone's payload data field, to meet the encryption algorithm's block size requirement if any, and to align it on a 4-B boundary according to the ML-IPsec ESP format.
- *Encryption*: The sender then encrypts the resulting plaintext (payload data, padding, pad length, and next header) using the key, the encryption algorithm, and the algorithm mode indicated by the zonal SA and cryptographic synchronization data (if any).

The inbound processing (Fig. 7) is almost a simple reverse of the outbound processing—plaintext bytes will be restored to their original positions by the zone map.

3) Partial In–Out Processing at Intermediate Routers: In an intermediate node, a packet will go through partial inbound processing and then outbound processing (Fig. 8). If the router function (e.g., TCP PEP) operates on the partially constructed IP datagram and modifies the packet, the outbound processing must redo authentication and/or encryption for that zone, and replace the corresponding ICV and/or payload data field before forwarding to the next hop.

V. Performance Evaluation

We have implemented ML-IPsec and evaluated its performance. We first analyze the bandwidth overhead incurred in ML-IPsec and compare it with the original IPsec protocol. Then, we further study the system complexity through an actual implementation in Linux. Finally, we measure the performance of this implementation.

A. Bandwidth Overhead Analysis

The extra overhead introduced by the multilayer protection model includes IPsec datagram size and processing load. The datagram size in a two-zone ML-IPsec is likely to increase when

TABLE I
PACKET LENGTH COMPARISON (BYTES)

	Original IP	IPsec	ML-IPsec
AH Transport mode	$40+n$	$64+n$	$76+n$
AH Tunnel mode	$40+n$	$84+n$	$96+n$
ESP Transport mode	$40+n$	$64+\lceil(6+n)/8\rceil*8$	$92+\lceil(1+n)/8\rceil*8$
ESP Tunnel mode	$40+n$	$88+\lceil(2+n)/8\rceil*8$	$116+\lceil(1+n)/8\rceil*8$

TABLE II
PACKET LENGTH OVERHEAD (BYTES)

	IP → IPsec	IP → ML-IPsec	IPsec → ML-IPsec
AH Transport mode	24	36	12
AH Tunnel mode	44	56	12
ESP Transport mode	[30,37]	[46,53]	12 or 20
ESP Tunnel mode	[50,57]	[70,77]	20

we do authentication or encryption as two separate plaintext blocks instead of one in the original IPsec. For example, the concatenated ciphertext from two individually encrypted plaintexts might be larger than the single ciphertext of the concatenated plaintext, due to the synchronization data (such as an initialization vector in some encryption algorithm) and separate padding. The authentication data field is also bigger. For example, if HMAC-MD5–96 is used, the ICV is a fixed size of 12 B. If the CSA has two zones, the new IPsec datagram will increase by 12 B compared with the original IPsec one. To understand the increase in packet size caused by ML-IPsec, we conduct protocol analysis on TCP applications.

The datagram used in the analysis is a TCP datagram, with 20-B IP header, 20-B TCP header, no IP options, and no TCP options. For ML-IPsec, we assume the TCP datagram is divided into two zones, one for TCP/IP headers, and the other for the TCP payload. We calculate the overhead for both AH and ESP protocols (with authentication) and for both transport and tunnel mode. We analyze both ML-IPsec and IPsec for comparison purposes. In all the cases, we assume use of the HMAC-MD5–96 algorithm for authentication and the 3DES-CBC algorithm for encryption.

We have analyzed the overhead for all eight cases (AH versus ESP, transport mode versus tunnel mode, and IPsec versus ML-IPsec). Due to space limitations, we will not enumerate the calculation in detail. We only summarize the results in Tables I and II. The variable n denotes the length of the TCP payload in the original IP datagram.

While the use of IPsec to protect IP datagrams adds an overhead ranging from 24 to 57 B, the new ML-IPsec scheme adds only an additional 12 B to the AH protocol and a maximum 20 B to the ESP protocol. Assuming an average IP datagram of 536 B, that is only a 2%–3% increase. One way to further reduce the overhead increase is to use a "weaker" authentication algorithm for the "less important" field. For example, a 4-B HMAC-MD5–32 ICV may be sufficient for the TCP header zone, instead of the 12-B HMAC-MD5–96 ICV in the original IPsec. This saves 8 B for each ML-IPsec packet and brings the overhead down to the 1%–2% range.

TABLE III
ML-IPSEC SOURCE CODE SIZE

Source tree	Code size (KB)		Lines of code	
	Original (IPsec)	ML-IPsec increase	Original (IPsec)	ML-IPsec increase
lib	136	+1 (0.7%)	4514	+24 (0.5%)
utils	145	+35 (24.%)	5147	+393 (7.6%)
klips/utils	129	+3 (2.3%)	4583	+142 (3.0%)
klips/utils/ipsec	949	+50 (5.3%)	14947	+1592 (10.%)
Total	1359	+89 (6.5%)	29191	+2151 (7.4%)

B. Implementation Complexity

One major concern for ML-IPsec is the complexity it adds on top of IPsec. In particular, the CSA design is very complex in contrast to the existing SA management model. To fully understand the complexity issue and to validate our design, we have implemented the system and studied its performance.

Our first prototype implementation of ML-IPsec is done in year 2000. It is based on Linux FreeS/WAN version 1.1 and Linux kernel version 2.2.12. FreeS/WAN (www.freeswan.org) is a publicly available implementation of IPsec. Our implementation modifies the FreeS/WAN code to add ML-IPsec capabilities. The detail of this implementation can be found in [31].

We have hypothesized that ML-IPsec should be easy to implement on top of an existing IPsec system (provided that source code and documentation are available). This is because we have designed ML-IPsec to closely follow IPsec. Many of the data structures in ML-IPsec reuse the same from IPsec, such as the SA objects. Majority of the protocol operations in ML-IPsec, including all cryptographic operations, are pointers to the procedures in IPsec.

Indeed, our experience has verified this hypothesis. Our first prototype implementation took about 6 man months. A significant portion of the time was spent on understanding the source code of FreeS/WAN, which was a fairly complex system (30 000 lines of code).

Table III lists the statistics about the source code size in this implementation. The entire FreeS/WAN source code tree is divided into four parts (subtrees). In each part, we count the size (in terms of kilobyte) and the lines of code in the original FreeS/WAN and after our modification. The major change is at klips (*K*ernel *IP*sec *S*upport), the Linux kernel code that implements IPsec functions. In total, we have added approximately 7% to the source code (both in bytes and in lines of code).

C. Experimental Measurements

We further conduct an experimental measurement to study the performance of ML-IPsec. While we have given an analytical calculation on overhead earlier, this experiment is to measure the actual implementation in a running system.

We design the experiments to compare the performance of the following network configurations.

- *IP*: running standard Linux with standard IP, without FreeS/WAN. This is to provide a baseline case where no security is provided.

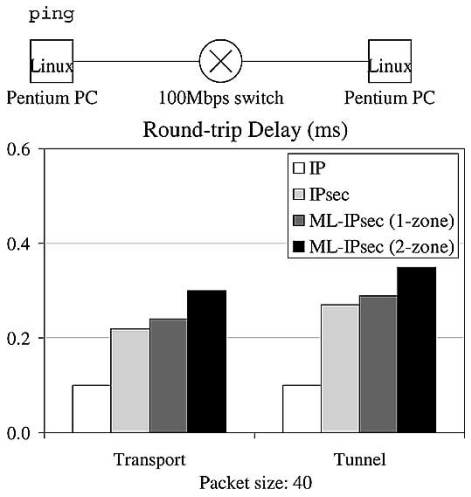

Fig. 9. Experiment 1: Processing delay.

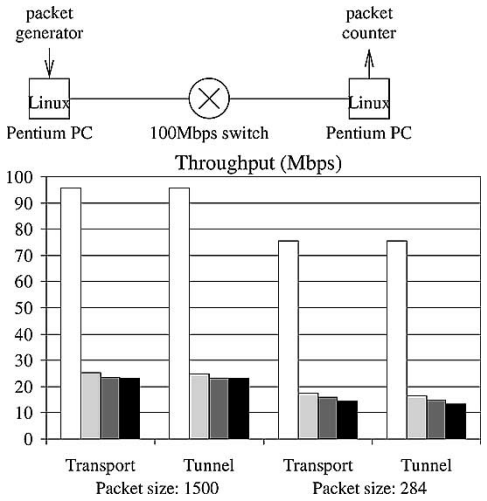

Fig. 10. Experiment 2: Comparing CPU overhead.

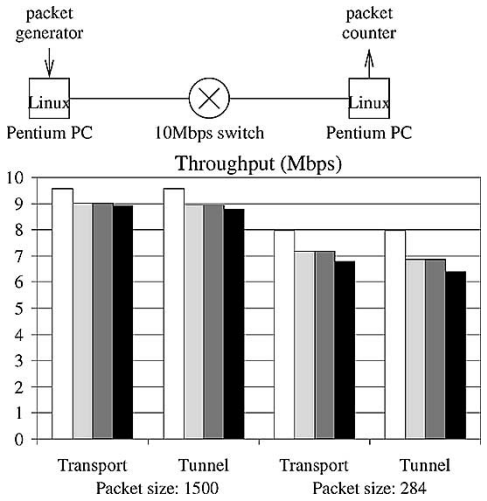

Fig. 11. Experiment 3: Bandwidth overhead.

- *IPsec*: running Linux with original FreeS/WAN, using ESP with authentication mode. This is to provide a baseline case where security is enabled.
- *ML-IPsec (one-zone)*: Running the modified FreeS/WAN and configuring ML-IPsec to have one single zone for the entire IP packet. Since the one-zone configuration has the same protection model as IPsec, this can reveal the overhead contributed to the ML-IPsec protocol format and processing.
- *ML-IPsec (two-zone)*: running the modified FreeS/WAN with two-zone configuration for TCP PEP. This is our target configuration that can support TCP PEP.

We conduct three sets of experiments to compare the following: the processing delay (to send/receive a packet), the CPU load, and the protocol format overhead. Figs. 9–11 illustrate the measurement results and the corresponding experiment setups (top subfigures). Each data point is the average of a hundred repeat runs. We compare under both transport and tunnel modes. For experiments 2 and 3, we also vary the packet size between two values: 1500 and 284 B.

Experiment 1 (Fig. 9) studies the per-packet processing delay, measured by the round-trip time of a 40-B ping packet. First, IPsec incurs more than twice the processing delay compared with IP, due to the security operations. On top of that, ML-IPsec (1-zone) incurs approximate 10% processing time. This is the overhead of ML-IPsec to process the extra protocol fields and data structures (like CSA). The processing time in the two-zone TCP PEP case is approximate 30% more, due to the fact that it requires two passes of encryption/authentication operations for each packet.

Experiment 2 (Fig. 10) studies the CPU load, measured by the network throughput. Here, the application (packet generator) sends data packets as fast as it can, and the host CPU is much slower (600 MHz Pentium) than the network (100 Mb/s). As a result, the network is never loaded and the CPU is the bottleneck. Hence, the network throughput is a direct measurement of the CPU load in sending/receiving the data packets: the lower the throughput, the higher the CPU load is. The result shows that ML-IPsec incurs between 8%–18% extra load over IPsec.

Experiment 3 (Fig. 11) studies the protocol format overhead, again measured by the network throughput. This time, we use a slower network (10 Mb/s) so that the bandwidth is now the bottleneck. A lower throughput implies higher protocol overhead. The result shows that ML-IPsec incurs a 2%–7% overhead, which is consistent with the calculation earlier (Section V-A).

VI. Conclusion

The end-to-end network security mechanisms such as IPsec and the rich network services such as TCP PEP for wireless networks are two fundamentally conflicting mechanisms. On the one hand, end-to-end security advocates the use of cryptography at the network layer to protect the payload over the untrustworthy Internet. On the other hand, certain network services rely on intermediate nodes to perform "intelligent operations" based on the packet data type–the information encoded in a higher protocol layer. It is the need to find the right balance between the two mechanisms and to achieve the goals of both, that makes for a difficult engineering problem.

Our attempt to solve the problem is based on the layering architecture for network security protocols. The approach presented in this paper may already have the right mix to provide both security and extensibility in one unified platform. Certainly, we have shown that through protocol design and system implementation ML-IPsec can easily be added to an

existing IPsec system and that its overhead is low. ML-IPsec has achieved the goal of granting trusted intermediate routers a secure, controlled, and limited access to selected portions of IP datagrams, while preserving the end-to-end security protection to user data. Currently, ML-IPsec approach is being adopted in several all-IP satellite networks [32], [33].

Our plan for future work includes an extension of IKE to support ML-IPsec. IKE is the key distribution protocol for IPsec, but we did not use it here because our current implementation uses manual keying only. It will be very important for ML-IPsec to be able to utilize automatic keying because it uses more keys, involves intermediate nodes, and requires a more complicated configuration than the original IPsec. The technical challenge will be finding the efficient mechanism needed for multiparty key distributions.

ACKNOWLEDGMENT

The author would like to thank the staff and intern students at HRL who have contributed to this project. In particular, he would like to acknowledge B. Singh who finished the first prototype implementation in 1999–2000. The author would also like to thank the following students who have interned with us and helped the project: I. Zohar, Stanford University (1998), P. Kumar Gonugunta, UIUC (1999), A. Shah, Stanford University (1999), S. Shah, University of Illinois at Urbana–Champaign (UIUC) (2001), and Venkatakrishnan VN, Stony Brook University of New York (SUNY)-Stony Brook (2001). In addition, we have all benefited from the e-mail discussions with participants in the IETF IPSEC working group and TF-ESP BOF.

REFERENCES

[1] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," IETF, RFC 2001, 1997.

[2] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 850–857, June 1994.

[3] C. Partridge and T. Shepard, "TCP performance over satellite links," *IEEE Network*, vol. 11, pp. 44–49, Sept. 1997.

[4] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc. 7th Annu. Int. Conf. Mobile Computing Networking (MobiCom'01)*, 2001, pp. 287–297.

[5] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *Proc. 5th ACM Int. Conf. Mobile Computing Networking (MobiCom'99)*, Aug. 1999, pp. 219–230.

[6] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. 15th Int. Conf. Distributed Computing Systems (ICDCS)*, May 1995, pp. 136–143.

[7] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. 1st Annu. Int. Conf. Mobile Computing Networking (MobiCom'95)*, 1995, pp. 2–11.

[8] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanism for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, pp. 756–769, Dec. 1997.

[9] T. Henderson and R. Katz, "Transport protocols for Internet-compatible satellite networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 326–344, Feb. 1999.

[10] D. Dutta and Y. Zhang, "An active proxy based architecture for TCP in heterogeneous variable bandwidth networks," in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM'01)*, Nov. 2001, pp. 2316–2320.

[11] ——, "An early bandwidth notification (EBN) architecture for dynamic bandwidth environments," in *Proc. IEEE Int. Conf. Communications (ICC'02)*, Apr. 2002, pp. 2602–2606.

[12] B. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Improving performance of TCP over wireless networks," in *Proc. 17th Int. Conf. Distributed Computing Systems (ICDCS-17)*, May 1997, pp. 365–373.

[13] A. D. Falk, "System design for a hybrid network data communications terminal using asymmetric TCP/IP to support internet applications," Master's thesis, Univ. Maryland, College Park, MD, 1994.

[14] V. Arora, N. Suphasindhu, J. Baras, and D. Dillon, "Effective extensions of Internet in hybrid satellite-terrestrial networks," Univ. Maryland, College Park, MD, Tech. Rep. CSHCN 96–2, 1996.

[15] D. C. Feldmeier, A. J. McAuley, J. M. Smith, D. Bakin, W. S. Marcus, and T. Raleigh, "Protocol boosters," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 437–444, Apr. 1998.

[16] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wir eless networks," *IEEE Pers. Commun. Mag.*, vol. 8, pp. 34–39, Feb. 2001.

[17] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," IETF, RFC 3135, 2001.

[18] V. Bharadwaj, "Improving TCP performance over high-bandwidth geostationary satellite links," Univ. Maryland, College Park, MD, Tech. Rep. ISR-MS-99–12, 1999.

[19] J. Ishac and M. Allman, "On the performance of TCP spoofing in satellite networks," in *Proc. IEEE MILCOM'01 Conf.*, Oct. 2001, pp. 700–704.

[20] N. Ehsan, M. Liu, and R. Ragland, "Evaluation of performance enhancing proxies in Internet over satellite," in *Wiley Int. J. Commun. Syst.*, vol. 16, Aug. 2003, pp. 513–534.

[21] S. Kent and R. Atkinson, "Security architecture for the Internet protocol," IETF, RFC 2401, 1998.

[22] ——, "IP authentication header," IETF, RFC 2402, 1998.

[23] ——, "IP encapsulating security payload (ESP)," IETF, RFC 2406, 1998.

[24] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance," IETF, RFC 2309, 1998.

[25] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.

[26] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Networking*, vol. 7, pp. 458–472, Aug. 1999.

[27] S. McCreary and K. C. Claffy, "Trends in wide area IP traffic patterns: A view from ames Internet exchange," presented at the ITC Specialist Seminar, Monterey, CA, Sept. 2000, [Online]. Available: http://www.caida.org/outreach/papers/2000/AIX0005/.

[28] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.

[29] S. Bellovin, "Transport-friendly ESP (or layer violations for fun and profit)," presented at the Panel Talk 1999 Network Distributed System Security Symp. (NDSS'99), San Diego, CA, Feb. 1999. [Online]. Available: http://www.research.att.com/~smb/talks/tfesp-ndss/index.htm.

[30] D. Harkins and D. Carrel, "The Internet key exchange (IKE)," IETF, RFC 2409, 1998.

[31] Y. Zhang and B. Singh, "A multi-layer IPsec protocol," in *Proc. Usenix Security Symp.*, Aug. 2000, pp. 213–228.

[32] M. Annoni, G. Boiero, N. Salis, H. S. Cruickshank, M. P. Howarth, and Z. Sun, "Interworking between multi-layer IPSEC and Secure multicast services over GEO satellites," Eur. Cooperation in the Field of Sci. Tech. Res., Tech. Rep. COST 272 TD-02–016, 2002.

[33] M. Annoni, G. Boiero, and N. Salis, "Security issues in the BRAHMS system," in *Proc. Ist Mobile Wireless Telecommunications Summit 2002*, June 2002.

Yongguang Zhang (M'94) received the Ph.D. degree in computer sciences from Purdue University, West Lafayette, IN.

He is currently a Senior Research Scientist with HRL Laboratories, LLC, Malibu, CA. His main research interests include mobile networking, security, and systems. He was also an Adjunct Assistant Professor in the Department of Computer Sciences at the University of Texas, Austin, from 2001 to 2003.