This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

Performance-aware Security of Unicast Communication in Hybrid Satellite Networks

Ayan Roy-Chowdhury
Electrical and Computer Engineering
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: ayan@umd.edu

John S. Baras
Electrical and Computer Engineering
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: baras@isr.umd.edu

***Abstract*—In this work, we address the performance problems that arise when unicast security protocols IPSEC and SSL are applied for securing the end-to-end communication in hybrid satellite networks. Satellite networks use TCP and HTTP performance-enhancing proxy servers to overcome the adverse effect of the large delay-bandwidth product of the satellite channel. However, the proxy servers cannot function when IPSEC and SSL are used for secure unicast communication in hybrid satellite networks. We therefore propose the use of the Layered IPSEC (LES) protocol as an alternative to IPSEC for network-layer security. We describe a modification to the Internet Key Exchange protocol if dynamic key establishment is needed for Layered IPSEC. For application-level security of web browsing with acceptable end-to-end delay, we propose the Dual-mode SSL protocol (DSSL) to be used instead of SSL. We describe how LES and DSSL protocols achieve the desired end-to-end communication security while allowing the TCP and HTTP proxy servers to function correctly. Through simulation studies, we quantify the improvement in performance that is achieved using our proposed protocols, compared to traditional IPSEC and SSL.**

I. Introduction

Satellite links suffer from longer propagation delays compared to terrestrial links. The delay can be as high as 500ms round-trip for a geostationary satellite link. Most Internet traffic uses the Transmission Control Protocol (TCP), which is highly susceptible to the delay-bandwidth product and exhibits very poor performance in satellite channels.

To mitigate the negative effects of the satellite propagation delay on Internet traffic, commercial satellite networks usually implement a split-connection TCP Performance Enhancing Proxy (PEP) [1]. A PEP agent is installed at the satellite gateway between the satellite network and the Internet. The PEP agent inspects every TCP packet that flows through the network and sends back premature acknowledgments to the TCP senders. Studies have shown that this technique leads to significant performance improvement in satellite networks [2].

Commercial satellite networks also employ HTTP proxy servers, at the central hub and each client location, to improve the speed of response to web browsing requests for Internet traffic. When the remote client makes a request for a webpage, the web server responds with the requested base HTML page. The hub HTTP proxy server intercepts and reads the web page and sends multiple GET requests to the destination web server to retrieve all the embedded objects in the base page. This exchange occurs over a high-speed terrestrial connection between the hub and the Internet, thereby saving the time each request would have needed for a round trip over the satellite link. As the objects are retrieved by the hub, they are immediately forwarded to the client proxy. The client browser GET requests are terminated at the local proxy server, which forwards the pre-fetched documents to the client browser immediately. The net result is that only a single GET request from the user browser traverses the satellite link, while a set of rapid responses quickly deliver the requested webpage and associated elements to the browser.

A. Proxy Performance Problem with Unicast Communication Security

Two protocols that are widely used for secure unicast communication are the Internet Security Protocol (IPSEC) [3] and the Secure Socket Layer (SSL) protocol [4].

IPSEC creates an end-to-end *secure channel* at the network layer for the secure transfer of traffic between two end users. The problem with using IPSEC in satellite networks is that it disables the functionality of the PEPs. The IP packet payload, which includes the TCP header, is encrypted with keys known only to the end points. Therefore a TCP PEP, which is an intermediate node in the communication path, cannot read or modify the TCP header, since the PEP does not know the keys. Consequently the PEP cannot function, leading to a degradation in the performance of the TCP protocol.

The Secure Socket Layer (SSL), on the other hand, operates above the transport layer in the protocol stack and establishes a secure HTTP (HTTPS) session on a need basis. SSL encrypts the TCP payload (the application layer HTTP data) between the client and the server, but the TCP header is transmitted in the clear. Therefore the TCP PEPs can function correctly with SSL. Howwever, the HTML webpage encrypted into SSL records are readable only by the client and the server who have the decryption keys. The keys are not available to the HTTP proxy, and therefore the HTTP proxy cannot read the HTML webpage. Consequently, HTML object pre-fetching by the hub proxy server cannot take place. The net result is that a web

978-1-4244-3435-0/09/$25.00 ©2009 IEEE

(a) Original IPSEC ESP tunnel mode encryption

(b) Layered IPSEC ESP tunnel mode encryption

Fig. 1. Packet format for IPSEC and Layered IPSEC. Key $K1$ is shared between end-points only. Key $K2$ is shared between end-points and TCP PEPs.

page with $n-1$ embedded objects takes $n*RTT$ to get loaded, an increase in delay by a factor of n.

Our objective is to propose solutions that allow IPSEC and SSL to work in conjunction with TCP and HTTP proxy servers in hybrid satellite networks, so that the unicast communication is secured without sacrificing the performance optimization algorithms. For this, we look at prospective candidate protocols and evaluate their performance through simulations.

The rest of this paper is organized as follows. In section II, we describe the Layered IPSEC protocol that we propose to use as an alternative to IPSEC. We also propose modifications to the Internet Key Exchange protocol that is used by IPSEC for dynamic session establishment. Section III describes the Dual-Mode SSL protocol, which we propose as an alternative to SSL. Performance analysis of the proposed protocols are given in sections IV and V. Section V also describes optimized versions of the Dual-mode SSL. We briefly review related work in section VI and conclude this paper in section VII.

II. Layered IPSEC for co-existence with TCP PEPs

For network layer encryption and integrity protection, we consider the Layered IPSEC Security protocol (LES), which is based on the concept of breaking the IPSEC encryption in multiple encryption regions or zones on a single packet basis. The method has been proposed independently in [5], [6]. Although the finer details in the two approaches are different, the basic idea is the same. Known as multilayer IPSEC or ML-IPSEC [5], and Layered IPSEC (LES) [6], the idea is to encrypt different regions of the IP packet using different keys (fig. 1b). The TCP payload is encrypted with key $K1$ which is shared only between the endpoints. The original IP header and the TCP header are encrypted with key $K2$ which is shared by the endpoints with intermediate authorized nodes like the TCP PEP. Therefore a TCP PEP can decrypt the header portion of the ESP packet with $K2$ and read the TCP header to do its performance optimizations. But the PEP cannot read the TCP payload and therefore cannot access the actual data since it does not posses the key $K1$.

In [6], the authors have demonstrated the correctness of operation of LES and have shown that its performance is comparable to IPSEC for the specific cryptographic algorithms used. [5] shows that the throughput overhead of ML-IPSEC in a simple test-bed is 2%-7% compared to IPSEC.

We believe that the LES approach would be an effective security solution in hybrid satellite networks that would allow TCP PEPs to function effectively. However, LES introduces higher complexity and higher communication overhead compared to IPSEC in the establishment of the secure channel that now requires co-ordination not only between the end points, but also with the proxy servers. We therefore investigate the performance of end-to-end traffic when LES is implemented in conjunction with TCP PEP optimizations in a hybrid satellite network, and compare to the case where IPSEC is implemented (and therefore the TCP PEPs cannot function).

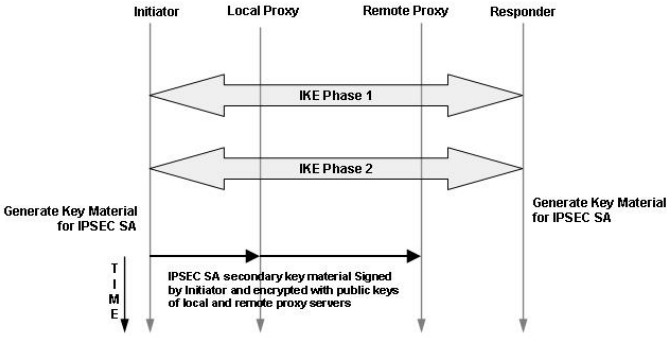

Fig. 2. Addition to IKE handshake mechanism for LES key management

A. Modification to Internet Key Exchange Protocol (IKE) for Layered IPSec

Both [5] and [6] assume pre-shared keys between the end points and the TCP PEP to establish the secure channel. However, in situations when the IPSEC encryption keys are dynamically determined at the time of establishing the secure channel, the Internet Key Exchange (IKE) [7] is used to generate the keys. IKE performs a series of handshakes between the end points to establish the two keys used by IPSEC for encryption/decryption and two keys for authentication (the usage of the keys in IPSEC is uni-directional).

IKE operates in two phases - *phase 1* and *phase 2*. The phase 1 exchange happens once and it creates a security association that allows multiple phase 2 connections to be set up between the client and the server. The phase 1 exchange can happen in either the *main* mode, in which there are 3 pairs of message exchanges between the end points, or in the *aggressive* mode, in which all the exchanges are condensed into a total of 3 messages.

In IKE phase 2 *quick* mode, there is a total of 3 exchanges between the initiator and the responder peers, during which the two parties verify the keying material that each will use for the session. The phase 2 exchange uses the session keys established in phase 1 to do mutual authentication and establish a phase 2 session key. Based on the phase 2 session key, the two end points agree on a set of four keys used by IPSEC.

We propose to modify the IKE protocol to incorporate the generation of additional keys needed for LES. In the modified protocol, in IKE phase 1 the initiator entity (which would be the remote client node in our scenario) includes the certificates of the remote and hub proxy nodes in the protection mechanism negotiation stage with the responder entity (the server node in our scenario). The keying data that is exchanged between the end points in the modified IKE phase 1 is subsequently used in IKE phase 2, so that the client and the server agree on a set of six keys - the four keys for forward and reverse encryption and authentication between the client and server, and two additional keys to be used by the sender to perform layered encryption on the IP header and also layered authentication. We add a fourth message dissemination to IKE phase 2, in which the client distributes the two additional keys to the local and remote proxy servers. The client encrypts the IP header encryption keys using the public keys of the proxy servers (we assume that the public keys and certificates of the proxy servers are available to the client), authenticates the message using a digital signature, and sends the authenticated message to the proxy servers. Figure 2 illustrates the step that we add to IKE phase 2 for key management for LES.

Additionally, we propose to use only the aggressive mode of IKE phase 1 exchange, and the quick mode of IKE phase 2 exchange. This is to contain the negative effect of the long round-trip delay on the overall performance - we have to ensure that the delay incurred due to the IKE message exchanges do not neutralize the advantages that might be gained due to the use of LES. The IKE phase 1 aggressive mode will reduce the delay by 50%, compared to the IKE phase 1 main mode. Whether that is sufficient savings, we evaluate through simulations in section IV.

III. Dual-Mode SSL: HTTP proxy-friendly Secure Web Browsing

When the HTTP traffic is secured using SSL and the security policy does not allow for trusted third parties, we propose the use of a modified SSL protocol, the *Dual-Mode SSL* (DSSL) protocol. As shown in fig. 3, the secure connection in DSSL has two modes - an end-to-end *main* mode connection between the client and the web server, and a *secondary* mode connection that has the hub HTTP proxy as an intermediate node. The security parameters for both modes are negotiated between the client and the server. Since DSSL extends the SSL protocol to include support for HTTP proxy servers, the message structures and many of the protocol steps are similar. DSSL introduces a series of additional message exchanges between the client and the HTTP proxy server, and between the proxy server and the web server. The various stages of the DSSL protocol are described below.

Stage 1: Client-proxy handshake phase 1. When a remote client wants to establish a secure session for the very first time, it might not be aware of the security parameters of the HTTP proxy. It therefore establishes a connection to the HTTP proxy and initiates the first stage of the DSSL protocol, the client-proxy handshake phase 1. In this phase the client

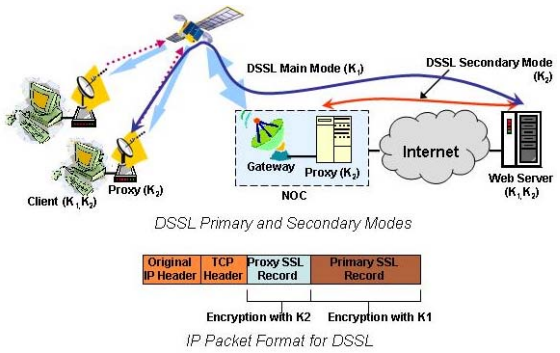

Fig. 3. Dual-Mode SSL for HTTP optimization

sends its security certificate to the HTTP proxy, which in turn responds with its own security certificate. The client thus obtains the public key of the HTTP proxy server from the proxy's certificate. The start and end of the communication in each phase is marked by a "Hello" and a "Done" message respectively, for each of the participating entities. This is in accordance with the original SSL protocol.

Stage 2: Client-server handshake. Once the client has obtained the security certificate of the HTTP proxy, it contacts the web server to exchange security credentials and to establish the session keys for the secure web session. This stage is similar to the SSL protocol, with two exceptions - (i) the client sends both its own certificate and the HTTP proxy certificate to the web server, and (ii) in the key exchange step, the client generates both primary and secondary keys and sends them to the web server.

Stage 3: Client-proxy handshake phase 2. After the client and the web server have established the session keys in the second stage, the client again contacts the HTTP proxy and instructs it to obtain the session keys from the web server.

Stage 4: Proxy-web server handshake. The HTTP proxy contacts the web server and sends its certificate to authenticate itself. Upon correct authentication, the web server sends the secondary session key to the HTTP proxy.

Stage 5: Client-proxy handshake phase 3. This final stage of the key establishment protocol is essentially a continuation of the client-proxy handshake phase 2. After the proxy obtains the secondary key from the web server, it contacts the client to confirm that it has received the key. The establishment of the primary and secondary keys between the client, the web server and the proxy is now complete.

All the message exchanges in DSSL are authenticated using digital certificates.

Let $K1$ be the encryption key for the main mode, and $K2$ be the encryption key for the secondary mode. When the client makes an HTTPS request, the client proxy sends local replies to the client browser. The web server, on receiving the request, parses the requested HTML page to obtain the embedded object links, which are collated into a new HTML page. The object links HTML page is then encrypted by DSSL using $K2$ to create the proxy SSL record. DSSL encrypts

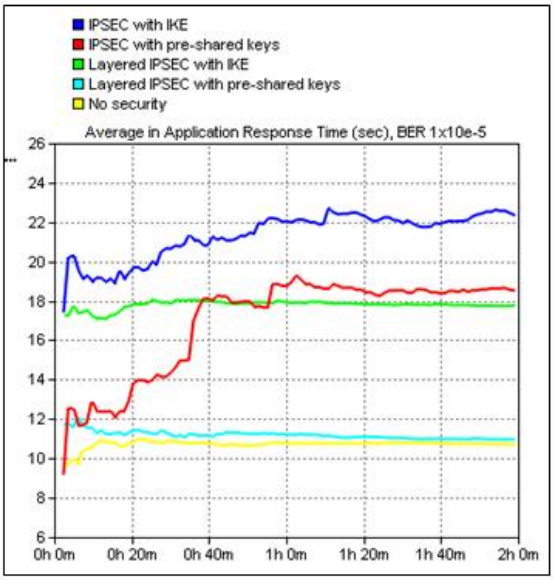

Fig. 4. Comparison of custom application response time (global average) for satellite link BER 1×10^{-5}.

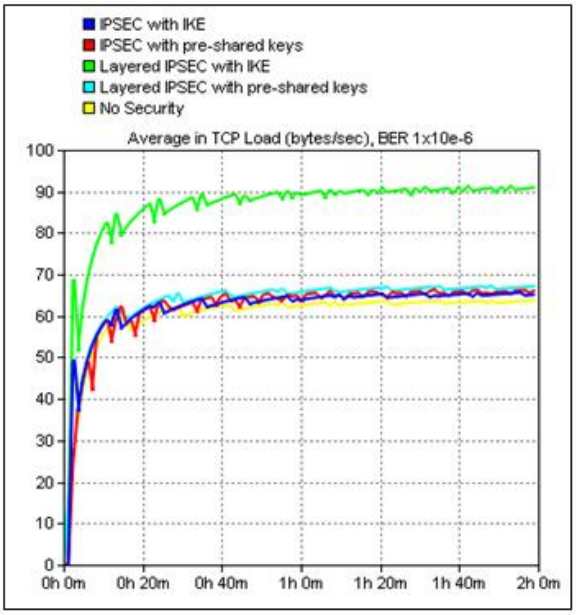

Fig. 5. Average TCP load at client for BER 1×10^{-6}.

the base HTML page using $K1$ to create the primary SSL record. The two records are appended together and sent to the client in an IP packet (fig. 3). The hub proxy intercepts the IP packet, extracts the object links from the proxy SSL record using $K2$, and pre-fetches the embedded objects. The web server always encrypts the actual objects using $K1$, so that the hub proxy cannot read the base HTML page data. The hub proxy transfers the embedded objects to the client together at one time. Thus the HTTP proxy functionality is preserved in DSSL while maintaining end-to-end security of the HTML page contents.

IV. Performance Evaluation of Layered IPSec with IKE modifications for Hybrid Satellite Networks

We have analyzed the performance of Layered IPSec with IKE modifications through simulations in Opnet Modeler [8]. The simulation setup consists of a remote client connected to a server via a satellite link. All communication between the client and the server pass through the remote and satellite hub TCP PEPs. The satellite link delay is 130 milli-seconds, uplink bandwidth is 256 kbps and the downlink bandwidth is 70 Mbps.

Figure 4 shows that the application response time for LES is significantly better than that of IPSEC, both when IKE and pre-shared keys are used[1]. This is because when IPSEC is used, the TCP optimizations are not working and therefore TCP considers the channel error to be signs of congestion and thus goes into recovery mode quicker. The graphs also indicate that using IKE adds significantly higher delay compared to using pre-shared keys. This is due to the multiple message exchanges between the client, the server and the PEPs that are needed by IKE to establish the secure channel. Each message exchange goes over the satellite links and adds to the overall delay. In fact, the delay for LES with pre-shared keys is nearly as low as that of unsecured data transmission with full TCP optimization, which has the lowest delay. The slightly higher delay for the former is primarily due to the IPSEC processing overhead at the nodes, and the slight transmission overhead due to the larger packet sizes due to LES headers and trailers. The effect of TCP optimizations is so pronounced that the lack of optimizations can have a greater effect on the overall delay than the IKE overhead. This is illustrated by the delay graph of IPSEC with pre-shared keys, which starts out much lower compared to LES with IKE (as can be expected), but it climbs higher when the un-optimized TCP in the former case runs into channel errors.

Figure 5 shows that the average TCP load due to LES with IKE is much higher than the other cases. At simulation time 1 hour 40 minutes, the LES TCP load is 89.938562bps, which is 42% higher than the TCP load for unsecured transmission (63.333333bps), and 37.7% higher that IPSEC with IKE (65.318627bps). However, this high overhead is mostly due to IKE. In the case of LES with pre-shared keys, the TCP load is 66.686275bps at simulation time 1 hour 40 minutes. This is only 1.1% higher than IPSEC with pre-shared keys (65.941176bps), and 5.3% higher than unsecured transmission.

The results indicate that Layered IPSEC can be a viable alternative to IPSEC for satellite networks, with comparable byte overhead while providing significant improvement in application performance. However, this holds true only if the secure channel is established apriori. While using IKE will still result in improved application response times for high channel error conditions, it might introduce unacceptably high

[1]In all the graphs, X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.

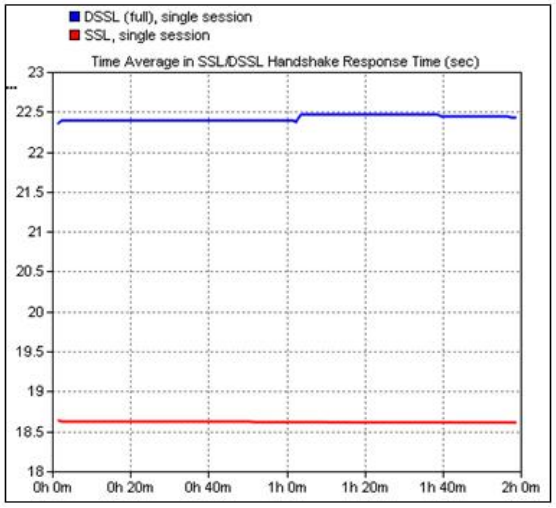

Fig. 6. DSSL and SSL handshake time comparison.

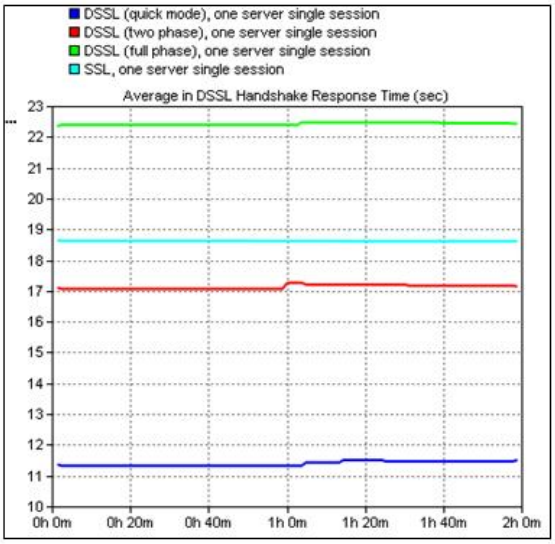

Fig. 7. Handshake time comparison for DSSL versions and SSL.

overhead even with the proposed modifications.

V. Performance Evaluation of Dual-Mode SSL

We have analyzed the performance of the DSSL protocol and associated web browsing by Opnet simulations. The setup is similar to that described in section IV, with the difference being that HTTP proxy servers are used in place of the TCP PEP servers. We conducted simulations for different scenarios - (i) unsecured web browsing with functional HTTP proxy, (ii) SSL security and hence non-operational HTTP proxy and (iii) DSSL security with HTTP proxy functional.

Figure 6 compares the response times for completing the DSSL handshake against the SSL handshake. The graph highlights the delay overhead of DSSL due to the additional steps of the client communicating with the proxy server in DSSL stages 1, 3 and 5, and the proxy server contacting the web server in stage 4. The DSSL overhead ranges between 20.3% and 20.74% over the SSL handshake delay.

To reduce the high delay involved in DSSL handshake, we propose a condensed version of DSSL, in which the handshake consists of only the first two stages. In the first stage, the client contacts the HTTP proxy and exchanges each other's digital certificate. Subsequently, the client contacts the web server. Once the client and the web server have established the security parameters in stage 2 of DSSL, the client makes the first HTTP "Get" request. This request goes through the HTTP proxy and triggers the proxy to send its certificate to the web server, and a request for the DSSL session secondary keys. The HTTP proxy request is piggy-backed on the client HTTP "Get" request. The server responds to the client request with the base webpage in a HTTP "Post" response. In addition, the server responds to the proxy key request with its own certificate and the DSSL secondary session keys, encrypted with the proxy's public key. This response to the proxy is piggy-backed on the HTTP response to the client. The proxy receives the combined response from the web server and is thus able to retrieve the secondary session keys from the encrypted message, using its private key. Consequently, it is able to decrypt the relevant portions of the HTTP "Post" response and therefore can perform the HTTP acceleration. We refer to this variant of DSSL as *DSSL two-phase*.

The first stage in DSSL two-phase is necessary in the situation that the client and the HTTP proxy server do not share any security association beforehand. However, if the two entities are apriori aware of each other's security information (via their digital certificates), then this phase is not needed. Every time the client contacts a web server to initiate a secure web session, it passes to the web server a locally cached copy of the HTTP proxy's certificate. The DSSL protocol can thus be further reduced to just one stage, that of stage 2. The DSSL secondary session key is transmitted to the HTTP proxy piggy-backed on the first response from the server, as described in the DSSL two-phase. We refer to this optimization of the DSSL protocol as *DSSL quick mode*.

Figure 7 compares the response times of DSSL handshake for its various versions. At simulation time 1hour and 20 minutes, DSSL quick mode handshake time is 33% less than DSSL two-phase (11.5 seconds and 17.2 seconds, respectively). The quick mode time is nearly half (48.8% less) than the handshake time for full version of DSSL (22.465 seconds), and is 38.8% less than that of SSL (18.61 seconds).

The optimizations in DSSL two-phase or DSSL quick mode are proposed to overcome the detrimental effect of the long propagation delay of the satellite channel on the DSSL handshake protocol. This optimization however requires further changes to the HTTP protocol, to allow piggy-backing the DSSL secondary keys on the initial HTTP exchanges between the client, the web server and the proxy server. Figure 8 compares the application response times for multiple secure browsing sessions with multiple servers. The times for all cases of DSSL are much lower than that for SSL, and are comparable to unsecured web browsing. The additional

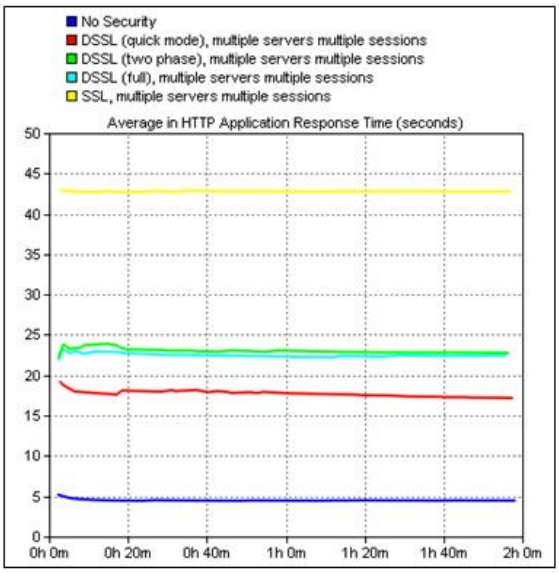

Fig. 8. HTTP response times for different versions of DSSL and comparison to SSL and unsecured web browsing.

delay is primarily due to the security overhead at the different nodes to maintain the secure session and to encrypt/decrypt the traffic.

VI. Related Work

Olechna et al [9] have suggested two solutions to the IPSEC problem. The first approach is to move the TCP PEP gateways to the endpoints. The TCP optimizations are done on the traffic in the clear, and then the traffic is encrypted using IPSEC. This approach improves the performance, but when a packet is lost or received in error, TCP goes into congestion avoidance phase and the transmission is reduced in half. The second approach is to split the secure connection into two at the satellite gateway. This allows the gateway to decrypt the IPSEC packet and read the headers and thereby do performance optimizations. However, this requires trust in the satellite gateway and might be unacceptable to users who require strong end-to-end security.

Several modified TCP protocols have been proposed that perform better compared to the original specification in the event of channel errors or delay, or when IPSEC is used. A discussion of these TCP enhancements can be found in [10].

The problem of HTTP proxy performance with SSL has been addressed by the industry by breaking up the end-to-end single SSL connection between client and server into multiple SSL connections [11]. In this solution, the client browser creates a secure HTTP connection with the Remote Page Accelerator (RPA) at the client satellite terminal, a second connection is created between the RPA and the Hub Page Accelerator (HPA), and a third connection is between the HPA and the server. The RPA performs all necessary handshaking with the client browser. The HPA can decrypt the SSL traffic from the server and perform the desired object pre-fetching. The major drawback of this scheme is that it requires a high level of trust in the intermediate nodes, which might be unacceptable when absolute end-to-end security is desired.

The DSSL concept is partly similar to the multiple-channel SSL concept proposed in [12]. However, there the authors do not differentiate encryption in primary and secondary SSL records - they suggest that HTTP traffic with lower security requirements be encrypted entirely with keys known to intermediate nodes. For our security requirements, that approach would be unacceptable.

VII. Conclusion

In this work, we have investigated the adverse effects of IPSEC and SSL on unicast communication in satellite networks. As a solution to the problem, we have proposed the use of layered IPSEC with modified IKE for securing unicast communication while allowing performance optimization algorithms to function simultaneously. We have also proposed the DSSL protocol with three variations, to replace SSL for secure HTTP in hybrid satellite networks. Through simulations we have shown that the performance of the proposed protocols compare favorably to standard IPSEC and SSL in hybrid networks.

Acknowledgment

The material is based upon work supported by the National Aeronautics and Space Administration under award No. NCC8235. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

[1] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, *Performance enhancing proxies intended to mitigate link-related degradations*, IETF RFC3135, June 2001.

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

[3] R. Atkinson and S. Kent, *Security Architecture for the Internet Protocol*, IETF RFC 2401, November 1998.

[4] *The SSL Protocol Version 3.0*, IETF Transport Layer Security Working Group, http://wp.netscape.com/eng/ssl3/draft302.txt, November 1996.

[5] Y. Zhang, "A multilayer IP security protocol for TCP performance enhancement in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 4, pp. 767–776, May 2004.

[6] M. Karir and J. Baras, "LES: Layered encryption security," in *Proceedings of the Third International Conference on Networking (ICN'04)*, Guadeloupe, French Caribbean, March 2004.

[7] D. Harkins and D. Carrel, *The Internet Key Exchange (IKE)*, IETF RFC 2409, November 1998.

[8] "Opnet Modeler," http://www.opnet.com/products/modeler/home.html.

[9] E. Olechna, P. Feighery, and S. Hryckiewicz, "Virtual private network issues using satellite based networks," in *Military Communications Conference (MILCOM) 2001*, vol. 2, 2001, pp. 785–789.

[10] P. Chitre, M. Karir, and M. Hadjitheodosiou, "TCP in the IPSEC environment," in *22nd AIAA International Communications Satellite Systems Conference and Exhibit (ICSSC) 2004*. Monterey, California: AIAA, May 2004.

[11] *SSL Accelerator*, Spacenet Inc., http://www.spacenet.com/technology/advantages/ssl.html.

[12] Y. Song, V. Leung, and K. Beznosov, "Supporting end-to-end security across proxies with multiple-channel SSL," in *Proceedings of the 19th IFIP Information Security Conference (SEC 2004)*. Toulouse, France: IFIP, August 23-26 2004, pp. 323–337.