Trusted Security Devices for Bandwidth Conservation in IPSec Environments

C.D. Mano and A. Striegel

Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
{cmano, striegel}@nd.edu

Abstract. Information security a is constant concern of Internet data. One security solution is IPSec, which is a set of protocols that provides both data confidentiality and authenticity. Another concern is the last mile bandwidth limitation on many Internet connections. This problem can be mitigated by bandwidth conservation techniques such as Application Layer and Stealth Multicast (SMC). Combining IPSec and multicast techniques would be ideal, but is not possible due to the nature of encrypted data and the requirements of multicast messages. We present the concept of a Trusted Security Device (TSD) which provides efficient bandwidth usage while maintaining security levels offered by IPSec. A TSD cooperates with clients and servers while implementing SMC technology. Minor modifications to clients and servers are necessary to enable discovery, key exchange, and communication between clients, servers, and TSDs. TSD technology is applicable to streaming data where confidentiality, authentication, and bandwidth conservation are concerns.

1 Introduction

Information security is a constant concern of data transmitted through an electronic network. Security can be increased by using various data encryption techniques. Encryption techniques can be classified into two basic categories, asymmetric and symmetric. Asymmetric, or public key, encryption can be used to authenticate senders of data, while symmetric keys are typically used to provide confidentiality.

Secure Socket Layers (SSL) [1] and the IP Security Protocol Suite (IPSec) [2] are two popular methods of creating secure communication channels between two parties on the Internet. SSL is an application based technique that is commonly used in web-based communication such as web browsers. IPSec is implemented at the network layer and therefore does not rely on the support of higher level applications. Both technologies use a combination of asymmetric and symmetric encryption to provide both authentication and confidentiality to network data.

It is becoming increasingly common for home and small business Internet users to serve data to remote clients. Streaming multimedia such as audio and video, as well as online game hosting are a few examples of content that is

being served by these users. Subscription based multimedia streams require a technology such as SSL to prevent theft of service. IPSec can be used in gaming environments, as in Microsoft's XBox Live, to prevent players from cheating by receiving extra information or transmitting false information.

A typical home or small business office has a limited bandwidth connection to the Internet. In cases such as Asymmetric DSL, the upstream bandwidth is even more constrained than the downstream. Bandwidth conservation techniques such as Application Layer Multicast (ALM) and Stealth Multicast (SMC)[3] are able to reduce these bandwidth demands, but are not compatible with security protocols such as those used in SSL or IPSec. This leaves hosts to choose between security and bandwidth while the situation demands that both exist together.

The marriage of bandwidth conservation and information security creates new and interesting problems which must be solved. In [4], the authors analyze security issues as they apply to multicast. In particular the topics of authentication, confidentiality, and key management are discussed. In [5] and [6] efficient key management techniques are presented. While these studies are effective in a traditional multicast infrastructure they do not apply to ALM or SMC environments.

We present the concept of a Trusted Security Device (TSD) as a solution to this problem. A TSD works in conjunction with clients and servers to utilize bandwidth conservation methods while maintaining secure communications. Section 2 discusses the motivation for this work as well as background information about IPSec and SMC. Section 3 presents the concept of the TSD and discusses communication and encryption protocols which are needed for the unit to function properly. An online gaming scenario is used to explain the entire process. Section 4 briefly discusses other applications of TSD technology, and 5 presents its possible weaknesses. The final section is a summary of this work.

2 Motivation and Background

Consider a case where a server is utilizing SSL to send a subscriber-based live music stream. The purpose of SSL is to restrict the audience to those who have paid the appropriate fees. Remember, however, that SSL is really a point-to-point protocol, so a distinct unicast stream is be needed to broadcast to each subscriber. The protocol could be adjusted so that all clients, following authentication, would receive the same symmetric key so that encrypted data would look identical. This would enable SMC to provide bandwidth conservation services, as will be explained later in this section. However, while eavesdropping would still be prevented, the source of the data cannot be authenticated, which may still pose a problem.

This problem is evident in the case where the data is a stream of stock prices. Because a group key is being used for the encryption we know that the data is coming from a member of the group, but we cannot authenticate which member of the group is sending it. This is a risky proposition when your hard earned money is at stake! As stated before, SSL and IPSec are able to

provide authentication of Internet traffic. This, however, is not feasible because while in a unicast environment a shared secret symmetric key provides source authentication, in this multicast scenario a shared group key only provides group authentication, not source authentication.

The remainder of this section discusses IPSec and SMC, and provides insight into reasons for their inability to work together in the streaming data scenarios presented.

2.1 IPSec

IPSec is a set of protocols designed to provide security services at the IP Layer. With data encryption being handled at this layer it provides security to networked applications that do not have encryption capabilities such as SSL built in to them. Virtual Private Networks (VPN) are a popular implementation of IPSec which is commonly used to create secure connections between a user and an entity such as a company. A telecommuting employee may use a VPN to log into their company's network so all data that traverses the Internet is secure.

IPSec offers two protocols for handling network traffic security, IP Authentication Header (AH) [7] and Encapsulating Security Payload (ESP) [8]. The protocols are similar in that they are able to provide authentication, integrity, and replay protection. AH also provides integrity of the IP header and non-repudiation. ESP provides confidentiality via encryption. The two protocols can be used together to obtain all required security characteristics. In this study we are particularly concerned with the following security characteristics:

- Confidentiality
- Integrity
- Source Authentication
- Replay Protection

As mentioned previously, in order for SMC to work, a group symmetric key, rather than pairwise unique symmetric keys, must be used. This eliminates the source authentication characteristic which is required. Another option is the use of a ticket. A ticket is a smaller piece of data, such as a checksum of the message, which is encrypted with a private key and then added to the message. The receiver could verify the ticket which provides source authentication. The computational requirements of asymmetric key encryption is still a problem however, because this must be done for every packet being sent.

2.2 Stealth Multicast

SMC is a method of converting redundant unicast Internet traffic into efficient multicast packets. This is enabled by the use of a module called a Virtual Group Detection Manager (VGDM) [3]. A VGDM performs similarity analysis on packet payloads and, based on pre-defined heuristics, caches the packet or forwards it on to its destination. A cached packet will become a member of

a virtual group, or will simply be forwarded on as normal. Virtual groups are comprised of a single packet payload along with the destinations of all packets with the identical payload. A group is used to create a single multicast packet to be delivered via SMC. Multicast packets are received by a SMC enabled device on the client side where the packet is converted back to the original unicast packet.

Again, this method is dependent on the ability of the VGDM to identify similar packet payloads in order to create virtual groups. Encryption of data creates a problem because unless the same key is used for all transmissions identical plaintext creates different ciphertext. As was discussed before, a group symmetric key allows the similarity detection to take place, but we suffer because source authentication is not possible.

3 Trusted Security Device

We propose the concept of a Trusted Security Device to solve this problem. A TSD works in conjunction with clients and servers to provide security for data transmissions. This includes the four major characteristics of confidentiality, integrity, source authentication, and replay protection. It is even effective in a heterogeneous environment where not all clients are enabled with a TSD unit. The TSD works as a trusted party which is delegated the responsibility of providing security. Minor modifications are required of clients and servers in order to facilitate direct communication with the TSD. VGDM capabilities from SMC are incorporated in order to reduce the amount of redundant traffic sent to the Internet.

We illustrate the functionality and use of a TSD with a possible real life implementation of an XBox Live gaming system. XBox Live is a service which allows XBox owners to host networked games across the Internet. It is one of many scenarios where home users may host servers and distribute streaming or other content through the Internet.

3.1 Architecture

Figure 1a is a typical XBox Live architecture consisting of multiple XBox console machines, with one being designated as the game server (XGS), and the XBox Live Server (XLS). Using a custom version of IPSec, Microsoft's XBox communicates and authenticates with the XLS. Authentication is made possible by a 2048-bit asymmetric key which is hardwired into the XBox. Once an XGS is authenticated it is listed as an open server for other players to join. An XBox client (XC), authenticated in the same manner, is able to join a game on a listed game server. Following the coordination of the game server and clients, communication travels directly between the XBox consoles. While Microsoft does not release details of their protocols or implementation, our analysis of XBox Live traffic shows that communication is done with encrypted unicast UDP packets.

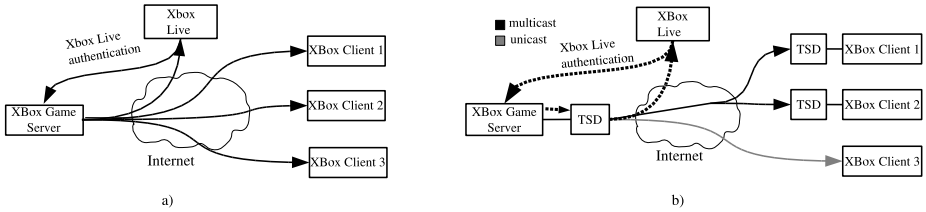

Fig. 1. Configuration and data flow of typical XBox Live setup

We will assume that each connection between a client and game server is established as a secure IPSec connection. This results in encryption using a different shared secret key between the server and each client. It is important to have this secure communication for game data to prevent cheating. In an unsecured environment unscrupulous players may collect information about the game or transmit false messages to other players in order to gain an unfair advantage.

Figure 1b shows a similar environment as previously described with the addition of TSDs on the game server and two of the three client machines. The TSD lies in-line between the XBox and the router. While the TSD will analyze and retransmit much of the XBox game data, for all other communication, including authentication from the XBox Live server, messages from clients, and dhcp requests, the TSD acts as a pass-through device. The following subsections describe the process of TSD discovery and the encryption protocols used to provide necessary encryption while allowing a VGDM to improve network efficiency.

3.2 Discovery

Each individual XBox, including the game server, must discover whether or not a TSD exists in their local environment. Because of the high level of trust which must exist between an XBox and TSD this requires a protocol that allows an XBox to discern between an authorized TSD and a spoofed, and possibly malicious, one.

First, authentication between an XBox and the XLS is established as normal. This is enabled by the hardwired 2048-bit asymmetric key, and results in a secure IPSec connection between the XBox and the XLS. Once this connection has been established an XBox can discover its local TSD.

Similar to a Public Key Infrastructure (PKI) [9], an XBox uses the XLS as a trusted third-party to establish the authenticity of the TSD. First, the XBox must broadcast a message to its LAN.

$$XBox \rightarrow BCast: \ (M, ID, N, ttl = 1)$$

where

$$N = E(Nonce, K_s)$$

This message is comprised of four items. First, a message M states that the message is requesting discovery of a TSD. Next is an identification number of the XBox followed by a nonce encrypted with a secret symmetric key generated by the XBox. This is for the purpose of authenticating a future message from the XLS. Finally the message has a TTL of one hop, which is sufficient because of the network topology requirements of the TSD.

If any device other than a TSD receives this broadcast message it will be discarded. A TSD will use the message as a signal to send a message to the XLS in order to authenticate itself with its local XBox.

In similar fashion to the XBox, the TSD authenticates itself to the XLS via a hardwired asymmetric key.

$$TSD \rightarrow XLS: \ E(ID + N + ST_s, TSD_{priv})$$

The XLS is able to verify that the message is coming from a valid TSD because it holds the corresponding asymmetric key of the secret hardwired key TSD_{priv}. The message includes the identification number and the encrypted nonce that the TSD received from the local XBox. The identification number is the same number as is held by the XLS already, so the XLS can identify the source of the original discovery request. The key ST_s is a symmetric key generated by the TSD that is established for communication between itself and the XBox. The XLS flags the XBox as being TSD enabled and then sends the following message to the XBox.

$$XLS \rightarrow XBox: \ E(''TSD/OK'' + N + ST_s, XS_s)$$

where XS_s is the secret symmetric key used for the IPSec connection between the XBox and the XLS. The XBox decrypts the message, authenticating the source, and obtains the message that a local TSD has been authenticated. The authentication of the TSD is strengthened by the decryption

$$XBox: \ D(N, K_s) = Nonce'$$

and a check to verify that $Nonce'$ equals $Nonce$, the original value the XBox calculated.

The XBox now holds the key ST_s which will be used for secure, authenticated communication with the TSD. The location of the TSD is unknown, so the XBox broadcasts a message similar to the initial authentication request.

$$XBox \rightarrow BCast: \ E(ConnectionRequest, ST_s)$$

Only the TSD will understand the message, as it generated ST_s, so it replies, establishing a secure communication channel.

This protocol allows an XBox to identify and authenticate a TSD in its local network without prior knowledge of its existence. A spoofed TSD and message replay are the two attacks which we are concerned with here. A spoofed XBox is not a concern because we assume it is not possible based on the XBox

authentication protocol. The hardwired asymmetric key in the TSD prevents spoofing in exactly the same way as the XBox. The nonce is unique to each authentication request made by the XBox. This prevents replay attacks because a nonce will be invalid for future authentication attempts. The secret symmetric key is also unique to each request, adding additional strength against replay attacks. The only unencrypted message is the initial authentication request from the XBox. A spoofed authentication would be defeated because an XBox cannot be spoofed, and the XBox would have to verify the original encrypted nonce in the authentication request.

Now each XBox knows whether or not it has a TSD and is ready to join a game. When an XC joins an XGS an IPSec connection is established just as in the standard architecture. This results in a unique symmetric key, SC_s, between each XGS-XC pair. If game play started at this point it would proceed just as in the standard configuration. The next step is the delegation of keys, which allows the TSD to take over security responsibilities.

3.3 Trust and Delegation

There are three types of key exchanges needed to create a secure environment. First is the key exchange between the XGS and its local TSD. Second is between the TSD of the XGS and the TSD enabled XCs. The final exchange is between the XCs and their corresponding local TSD. Table 1 summarizes these keys which will be used for communication during game play. In the notation: S = Server, C* = Client (* represents identification number), and T = local TSD. So T in ST refers to the server's local TSD and in C^1T represents C^1's TSD. This exchange takes place at the beginning of each game, but because players can join or leave mid-game, is can be done for a single player at any time.

First the XGS must transfer a list of XCs along with their symmetric keys SC_s^* to its local TSD.

$$XGS \rightarrow TSD_{XGS} : \ E(XC\ List + Key\ List, ST_s)$$

Next, TDS_{XGS} must determine which of the XCs are TSD enabled. This is accomplished by sending each XC an inquiry using the appropriate symmetric key obtained from the XGS.

$$TSD_{XGS} \rightarrow XC_* : \ E(TSD?, SC_s^*)$$

The TDS_{XGS} intercepts replies and can then divide the XCs into two group, TSD enabled and non-TSD.

TSD_{XGS} must now generate a shared group key for multicast communication as well as an asymmetric key pair to be used for authentication. This is not the same as the 2048-bit key used for authentication with the XLS, but is a smaller key used specifically for the current communication. It then distributes the keys to TSD group members. Assuming a scenario such as figure 1b.

$$TSD_{XGS} \rightarrow C_1 : \ E(SGs + S_{pub}^{tsd}, SC_s^1)$$

Table 1. Notation of encryption keys

Key	Notation
Symmetric Key for S and C*	SC_s^*
Symmetric Group Key	SG_s
Asymmetric Keys for S's TSD	S_{priv}^{tsd}, S_{pub}^{tsd}
Symmetric Key for S and local TSD	ST_s
Symmetric Key for C* and local TSD	C^*T_s

$$TSD_{XGS} \rightarrow C_2 : \ E(SGs + S_{pub}^{tsd}, SC_s^2)$$

At this point TSD_{XGS} cannot communicate directly with the TSDs of the XCs. The final step enables this communication by each XC transferring the group and private keys just received to their local TSD.

$$XC_* \rightarrow TSD_{C_*} : \ E(SGs + S_{pub}^{tsd}, C^*T_s)$$

Now all keys necessary for the communication protocol have been generated and shared. A summary of each unit and the keys they possess are listed in table 2. The communication protocol, in conjunction with these keys, will create a secure environment similar to the basic XBox scenario, while adding the capability of a VGDM to allow a more efficient use of bandwidth. This process will be shown as a step-by-step progression in the following subsection.

Table 2. Summary of Key Possession

Unit	Keys
XGS	SC_s^*, ST_s
TSD_{XGS}	SC_s^*, ST_s, SG_s, S_{priv}^{tsd}
Non-TSD C_*	SC_s^*
TSD Enabled C_*	SC_s^*, SG_s, C^*T_s, S_{pub}^{tsd}
TSD_{C_*}	SG_s, C^*T_s, S_{pub}^{tsd}

3.4 Message Processing

We are only concerned with the flow of a message from an XGS to an XC. Messages in the opposite direction proceed just as before using the established IPSec channel between the game server and client as no advantage can be gained in terms of bandwidth conservation in this direction.

Figure 2 shows the flow of messages sent from an XGS to its clients. Assume the server generates a message, M, which will be sent to clients C_1, C_2, and C_3, where C_1 and C_2 each have a TSD (Figure 1b). The game server encrypts and sends messages as it would in a standard, non-TSD, configuration.

$$XGS \rightarrow TSD : \ E(C_1 + M, SC_s^1) = M_1^{'}$$

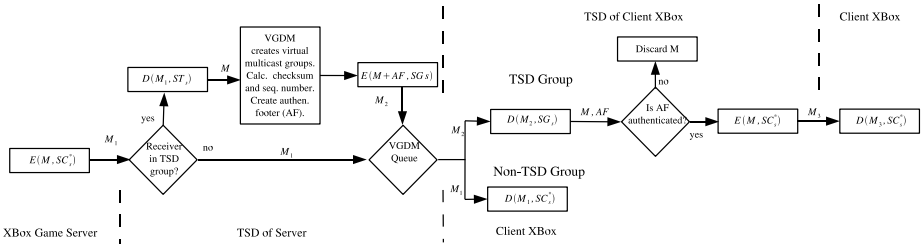

Fig. 2. Flow of messages sent from the XBox Game Server to XBox Clients with and without a TSD

$$XGS \rightarrow TSD: \ E(C_2 + M, SC_s^2) = M_1^{''}$$
$$XGS \rightarrow TSD: \ E(C_3 + M, SC_s^3) = M_1^{'''}$$

The messages are intercepted by the TSD are checked to see if the destination XC is TSD enabled. TSD group messages are decrypted using the appropriate symmetric key.

$$TSD: \ D(M'_1, SC_s^1) = M$$
$$TSD: \ D(M''_1, SC_s^2) = M$$

Client C_3 is not part of the TSD group, so its message is simply placed in the VGDM queue.

$$TSD \rightarrow C_1: \ M_1^{'''}$$

Non-group messages are queued before delivering in order to minimize delivery time differences between originally temporally close messages.

The messages which are destined for TSD group members are placed in the VGDM staging area for similarity detection to take place. The VGDM is the part of SMC technology which creates multicast packets for delivery on networks lacking traditional multicast infrastructure. The VGDM analyzes data payloads and creates a message group based on the similarity of the data payloads. The result is a single message that is to be delivered to multiple XCs.

Prior to encrypting the payload of the group message an authentication footer (AF) is created (Figure 3). The AF is comprised of an MD5 hash of the message and sequence number. The AF is encrypted with the private key of the TSD, appended to the message and then the entire message is encrypted with the group key.

$$TSD \rightarrow C_1, C_2: \ E(M + E(AF, S_{priv}^{tsd}), SG_s) = M_2$$

The AF is dependent on the level of security required for a particular application. In this application a 2-byte sequence number and 4-byte hash would be sufficient.

The symmetric group key offers confidentiality and group authentication. However, it does not authenticate the sender within the group. Public key encryption of the entire message would allow for this authentication, but computational requirements for public key encryption make this unrealistic. The AF

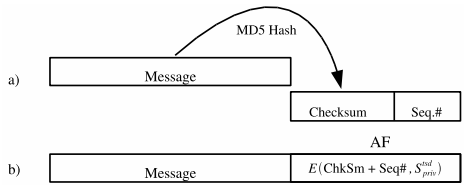

Fig. 3. An authentication footer is an encrypted 4-byte MD5 hash of the message and a 2-byte sequence number. This provides source authentication for XBox clients.

provides three important aspects of data security. First, the sequence number prevents replay attacks. More details will be given during the description of the receiver TSD. Second, the checksum verifies the integrity of the data, which ensures that the message has not been tampered with. Finally, encryption with the private key of the TSD provides for source authentication. The public key encryption done here is very minimal when compared to encrypting the entire message, so the added latency of delivering the message is minimal.

The VGDM then transmits the group messages via ALM or some other multicast method. The client side TSD receives the multicast packets and decrypts both the message and the resulting AF.

$$C_*TSD: \ D(M_2, SG_s) = M + AF$$

$$C_*TSD: \ D(AF, S^{tsd}_{pub}) = checksum + seq.number$$

The checksum is validated by calculating an MD5 hash of M and comparing the results. The sequence number is compared to the previous number and if it falls within a pre-determined range, such as plus or minus 10, the message is accepted. The message is then encrypted and delivered to the XC.

$$C_*TSD \rightarrow C_*: \ E(M, SC^*_s) = M_3$$

Notice that the XBoxs use the same keys for sending and receiving as they would in a non-TSD scenario. Also, the only additional information they hold concerns their own local TSD. The TSD of the XGS is the only entity that holds all knowledge of the TSD enabled XCs. So the only modification needed to the XBox is to enable it to discover a TSD and exchange keys.

In an online gaming environment backward and forward confidentiality are not much of a concern. In other environments it may be, but it can be easily handled with key updates. Sophisticated key update protocols, typically for large audiences, are discussed in [4]. Generally speaking, when a client joins or leaves, or at set intervals, the new keys must be distributed to the TSD group members. In a gaming scenario, with a limited number of participants, it would probably be sufficient to perform this operation at the start of each game, but can be done as needed. The SG_s keys and the S^{tsd}_{pub} and S^{tsd}_{priv} keys would need to be regenerated by the TSD of the XGS and appropriately distributed through the secure IPSec connections just as was done during the initial key exchanges.

The security attributes of data integrity and confidentiality, replay protection, and sender authentication have all been maintained with the addition of the TSD in the online gaming framework. At the same time the TSD has enabled a technology such as SMC to efficiently transmit data via a multicast protocol. This allows hosts with constrained or limited upstream bandwidth to provide streaming multimedia content while reducing QoS problems associated with bandwidth constraints. Bandwidth conservation is obviously an advantage to the content host, but it is also positive for the receivers. This enables them to take advantage of a provider's efficient data stream, improving the QoS of the presentation on their end.

4 Other Applications

We have presented the TSD concept within the framework of a popular online gaming architecture. However, the capability of the TSD is not limited to such a scenario. The true ability of the TSD is the offering of data confidentiality and source authenticity while reducing distribution bandwidth requirements.

The target application for this technology is anything that needs to provide information security while delivering content to multiple receivers. Previously we mentioned the streaming stock price example. This, as well as most information data streams, is an ideal place for TSD technology. General public subscription based audio and video streams may or may not be candidates depending on source authentication needs. However, for private groups, such as corporate, government, or military organizations, source authentication is critical and TSD services would be a great benefit.

We have begun to investigate how the TSD would fit into a distributed backup system. This would allow secure data to be efficiently transferred to multiple backup sites simultaneously. This scenario is more complex than streaming media because of the effects of dropped packets. We will be investigating a method of efficiently incorporating the ability to handle this in the future.

5 Weaknesses

The addition of a TSD obviously offers a new point of attack for malicious users. However, the authentication process for the TSD is similar to that of an XBox, so in terms of security strength they can be considered equal. In other scenarios the TSD would still be authenticated in a way that provides equal security strength to the existing infrastructure.

A system that does not require modification of existing systems would be ideal in terms of ease of deployment and scope of use. The TSD, however, does require modifications to be made in order for trust to be established and keys to be exchanged. This requirement, while necessary and unavoidable, prevents the TSD from becoming a general purpose network appliance, but does position it as a revenue generating add-on product.

6 Summary

The marriage of bandwidth conservation and information security can be performed by the implementation of a trusted security device. Bandwidth conservation techniques such as ALM or SMC can be used to reduce redundant network traffic and do not suffer the deployment problems of traditional multicast. SMC relies on the ability to detect identical data payloads to create virtual multicast groups for efficient delivery of data. IPSec is a protocol for secure data transfer, but does not work with a SMC because SMC is not able to detect similar data when encrypted with different keys.

The TSD is a trusted device to which clients and servers delegate the responsibility of providing authentication and confidentiality of data. TSD implementation requires minimal modification to servers and clients to be able to function together. Discovery, key exchange, and the encryption protocol are the three major steps to creating a secure infrastructure with TSD.

A TSD infrastructure allows secure streaming data to be efficiently broadcast across the Internet. It allows bandwidth limited hosts to increase their audience while minimizing bandwidth costs. End users will benefit by receiving more efficient TSD enabled streams which increase the QoS over the traditional unicast streams from the same server.

References

1. P. K. Alan Freier, Philip Kariton, "The SSL protocol: Version 3.0," Netscape Communications, Inc., Mountain View, CA, March 1996.
2. S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," Internet Engineering Task Force: RFC 2401, November 1998.
3. A. Striegel, "Stealth multicast: A catalyst for multiccast deployment," in *Proceedings of IFIP Networking*, Athens, Greece, May 2004, pp. 817–828.
4. T. Hardjono and G. Tsudik, "IP multicast security: Issues and directions," *Annales de Telecom*, 2000.
5. I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure internet multicast using boolean function minimization techniques," in *Proceedings IEEE Infocomm'99*, vol. 2, 1999, pp. 689–698.
6. G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 628–639, 2000.
7. S. Kent and R. Atkinson, "IP Authentication Header," Internet Engineering Task Force: RFC 2402, November 1998.
8. S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," Internet Engineering Task Force: RFC 2406, November 1998.
9. R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure," The Internet Society: RFC 3280, April 2002.