Middleware Support for Tunable Encryption

Stefan Lindskog, Reine Lundin and Anna Brunstrom

Department of Computer Science
Karlstad University
SE-651 88 Karlstad, Sweden

Abstract. To achieve an appropriate tradeoff between security and performance for wireless applications, a tunable and differential treatment of security is required. In this paper, we present a tunable encryption service designed as a middleware that is based on a selective encryption paradigm. The core component of the middleware provides block-based selective encryption. Although the selection of which data to encrypt is made by the sending application and is typically content-dependent, the representation used by the core component is application and content-independent. This frees the selective decryption module at the receiver from the need for application or content-specific knowledge. The sending application specifies the data to encrypt either directly or through a set of high-level application interfaces. A prototype implementation of the middleware is described along with an initial performance evaluation. The experimental results demonstrate that the generic middleware service offers a high degree of security adaptiveness at a low cost.

1 Introduction

For wireless applications the protection of data has become an important requirement. Although encryption can provide adequate data protection, it may lead to severely degraded network performance in terms of latency and throughput or reduce the battery life time of mobile devices. As a result, various selective encryption schemes that produce less overhead compared to ordinary encryption have been proposed. Such schemes can provide a tradeoff between security and performance. However, most previous work on selective encryption have either been focused on a specific content [2, 24] or application area [3], or even been directly integrated into an application [13]. This implies that the encryption at the sender and the decryption at the receiver are tightly coupled.

In this paper, we present a tunable encryption (TE) service that is designed as a middleware. The general idea with TE is to provide various protection levels according to the principle of adequate security [14]. As pointed-out in [9], protection levels can be selected at run-time and specified in fundamentally two different ways: algorithm selection and selective encryption. Algorithm selection is when a protection level is specified through the selection of a particular encryption algorithm together with its related parameters. For example, it might be possible to specify a particular algorithm (DES, 3DES, AES, etc.), mode (Electronic Code Book (EBC), Cipher Block Chaining (CBC), etc.), key length, block length, and number of encryption rounds. Selective encryption, on

Lindskog S., Lundin R. and Brunstrom A. (2006).
Middleware Support for Tunable Encryption.
In *Proceedings of the 5th International Workshop on Wireless Information Systems*, pages 36-46

the other hand, is when a protection level is specified through only encrypting selective parts of the data. A TE service in which the protection level could be specified through both algorithm selection and selective encryption is referred to as a combined service with respect to the protection level. Different types of TE services can also be distinguished based on adaptiveness. Essentially, two main categories of adaptiveness exist: per-session and in-session. In a per-session adaptive service, the protection level is specified at the inception of a communication session and after that remains fixed during the lifetime of the session. In an in-session adaptive service, on the other hand, the protection level can vary during the lifetime of the session.

The TE middleware service described in this paper is based on selective encryption and provides an in-session adaptive service. The middleware service can be used by various applications and on different contents. This is due to the block-based selective encryption module that constitutes the core component of the middleware. The selection of which data to encrypt is made by the sending application and is typically content-dependent. However, the representation used by the core component is application and content-independent. Although the sender typically selects the blocks to encrypt based on content to achieve maximum security for a given encryption level (EL), the receiver only needs to know which blocks are encrypted to recreate the data. This allows a sending server to change its encryption procedure at will without the need to modify any of the receiving terminals. By providing a generic middleware service that can be used by application developers, the design of secure wireless applications will be simplified, thereby contributing to increased security.

The remainder of the paper is organized as follows. In Sect. 2, related work is presented. The proposed middleware service is discussed in more detail in Sect. 3. Section 4 presents a prototype implementation of the middleware as well as experimental results from the prototype. Finally, Sect. 5 discusses future work and Sect. 6 concludes the paper.

2 Related Work

The concept of selective encryption was independently introduced by Spanos and Maples [24], Li et al. [7], and Meyer and Gadegast [12] in 1995 and 1996 in order to reduce the amount of encrypted MPEG data in a video sequence, while at the same time providing an acceptable security level. Spanos and Maples proposed that only the I-frames in an MPEG video stream need to be encrypted. Li et al. proposed a protection hierarchy in which one may choose to encrypt (1) only I-frames, (2) I- and P-frames, or (3) all I-, B-, and P-frames in any video sequence. Meyer and Gadegast proposed four levels of encryption—from header only encryption to complete encryption. Further selective encryption methods for MPEG video are presented and discussed in [1, 6, 22, 28]. In addition, Sony [23] have recently announced that they use a scalable approach based on selective encryption in their Passage technology aimed for digital CATV networks.

Selective encryption has also been used to protect image data. In [16], a selective bit plane encryption is proposed for JPEG images. The authors showed that encrypting the most significant bit plane only, was not secure enough. However, they showed that a sufficient confidentiality level could in many cases be achieved by only encrypting two

bit planes, whereas encrypting four bit planes provides a high degree of confidentiality. Van Droogenbroeck and Benedett [2] suggested two different methods to selectively encrypt compressed and uncompressed images.

In [21], Servetti and De Martin present a selective encryption scheme for speech compressed with the ITU-T G.729 8 kb/s speech encoding standard. The authors claim that the proposed scheme offers an effective content protection and can easily be adapted to other speech coding standards as well. Goodman and Chandrakasan [3] propose a scalable encryption scheme that is aimed to maximize the battery lifetime of a wireless video camera. Their scheme is based on a stream cipher that allows varying levels of encryption for data streams with varying priorities. In addition, support for TE has also been integrated directly into several multimedia applications, e.g., Nautilus [13] and Speak Freely [25]. Various application areas for selective encryption are further discussed in [10].

In contrast to the work above, we have previously introduced the concept of block-based selective encryption [8] as a content-independent representation of the selectively encrypted data. In this paper, we use this concept as a core component in designing a generic TE middleware service that is aimed to support a variety of applications with different demands. A content-independent design, but for a different context, was also used by Griwodz et al. in their work on a selective data corruption scheme to protect video on demand (VoD) data [4].

A recent example of the use of algorithm selection, the other method to provide TE, is presented by Yogender and Ali in [29]. In the paper, they investigate the impacts of run-time security parameter changes when using the IPSec protocol in a Virtual Private Network (VPN) setting. Seven different security levels are defined and an adaptive model, which is used to switch between the various security levels, is described. Another example of the use of algorithm selection can be found in [26].

More dynamic security solutions have also been proposed for other security attributes. Authenticast, proposed by Schneck and Schwan in [19], is a user-level protocol that provides run-time security adaptation by offering variable levels of security through selective authentication (or rather selective verification). A security level is defined as the percentage of data that are authenticated. Through a user interface, referred to as the security thermostat, a user can specify a security level range. During operation Authenticast will chose an appropriate security level within this range based on CPU load. Other run-time adaptive (lightweight) authentication protocols are proposed and discussed by Johnson [5].

3 Middleware Design

The proposed middleware service is further described in this section. The description starts with an architectural overview followed by an in-depth discussion of the block-based selective encryption (and decryption) procedure. Different types of application interfaces provided by the middleware and transport service support are also discussed.

3.1 Architectural Overview

An architectural overview of the proposed tunable encryption middleware is illustrated in Fig. 1. In the figure, both a sender and a receiver are shown. The middleware provides a set of different high-level application interfaces on the sender side. The selection of which interface to use is dependent on the application as well as on the content to transfer. In the figure, three types of high-level application interfaces—content-specific, entropy-adaptive, and load-balanced EL—are exemplified. Additional high-level interfaces can be specified and added on demand. The three proposed interface types presented in the figure are further described in subsections below. An application is, however, not restricted to use one of these. Instead, a sender application can directly access the block-based selective encryption component. This allows an application to have complete control over the blocks that are encrypted. Yet another option is to use the proposed middleware service off-line to pre-encrypt data that will be sent later. In this case, data will be passed directly to the transport service when transferred.

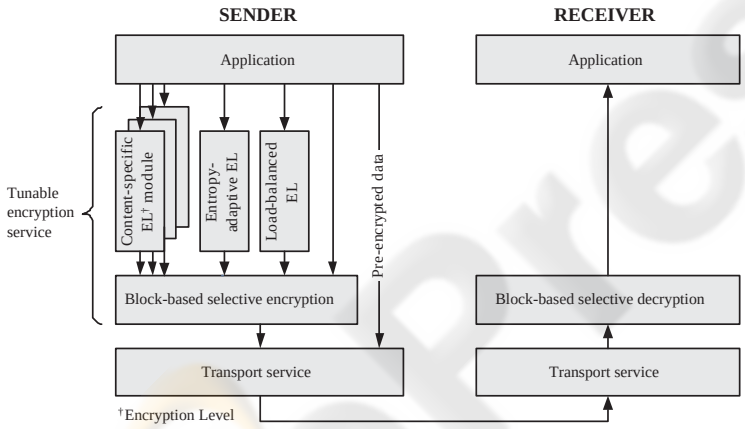

Fig. 1. An architectural overview of the middleware service.

As discussed above, the sender has many options when transferring data using the middleware. The receiver, on the other hand, always uses the same interface when receiving data and it is independent of how the sender has produced the output. All received data will pass the block-based selective decryption component before being delivered (in plain) to the application. By using a uniform interface at the receiver side, the design and implementation of the TE service is simplified and much more straightforward. The uniform interface allows a sending server to change its encryption procedure at will without the need to modify any of the receiving terminals.

3.2 Block-based Selective Encryption

The core component of the TE middleware architecture is the block-based selective encryption module. This module translates the application and content-specific selec-

tion of which data to encrypt, made by the sending application, into an application and content-independent representation. Its task is to encrypt chosen blocks of a data sequence, before sending it to the transport service. The remaining not chosen blocks are unencrypted, compressed or encrypted with a weaker encryption algorithm. We assume, in this paper, that the blocks are equally sized and defined by the used encryption algorithm. For example, if AES is used then the block size would be 128 bits and this is what is used in the prototype implementation described in Sect. 4.

The encryption procedure for block-based selective encryption within the TE middleware has three basic entities: a data sequence DS, a bit vector B, which is also referred to as the encryption mask, and an encrypted data sequence EDS. The DS, which is to be selectively encrypted, is divided into n equally sized blocks ds_i, $0 \leq i \leq n-1$. Hence, it can be written in the form:

$$DS = \mathop{||}_{i=0}^{n-1} ds_i \tag{1}$$

where $||$ denotes the binary concatenate operator.

The encryption mask controls the blocks that are to be encrypted. A block ds_i in DS will be encrypted if $B_{i \bmod |B|} = 1$ and left unencrypted if $B_{i \bmod |B|} = 0$, where $|B|$ is the length of the bit vector. The EDS, which also contains n equally sized blocks, is constructed by the following operation:

$$EDS = \mathop{||}_{i=0}^{n-1} \begin{cases} ds_i & \text{if } B_{i \bmod |B|} = 0 \\ E(ds_i) & \text{if } B_{i \bmod |B|} = 1 \end{cases} \tag{2}$$

where $E(ds_i)$ denotes encryption of ds_i. Note how B determines the EL of the EDS. The encryption procedure for block-based selective encryption is graphically illustrated in Fig. 2.

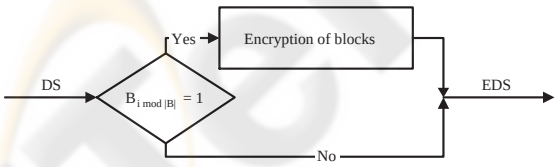

Fig. 2. Encryption of data using block-based selective encryption.

Before a receiver of an EDS can start the decryption process, it must know the correct mask used in the encryption process. Therefore, encryption control units, ECUs, are created, which consist of the size of the mask and the mask itself. The ECUs are always fully encrypted, using the same encryption algorithm as is used for data encryption. Updated ECUs can be sent during operation, which allows the EL to be changed dynamically.

In Fig. 3 we show an example of a data transfer using block-based selective encryption. Note how the number of encrypted blocks changes with a new ECU. The mask

will for example change from (1011) to (01011) when the second ECU is received. The mask implicitly divides the EDS into a sequence of encryption data units (EDUs) where the number of data blocks in an EDU equals the size of the mask.

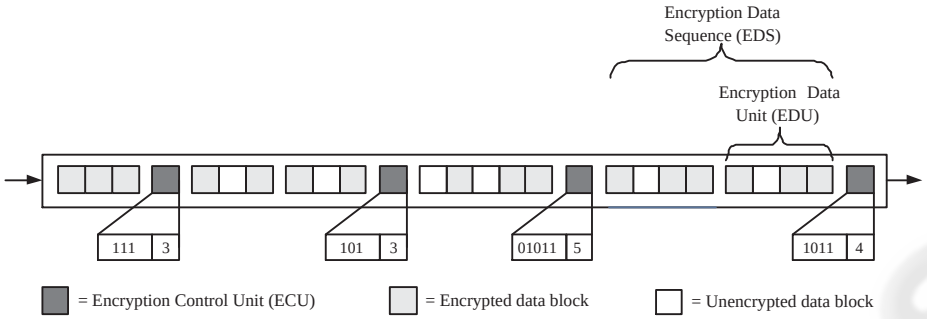

Fig. 3. Selectively encrypted data sequence.

3.3 High-level Application Interfaces

Although the block-based selective encryption component described above can be accessed directly by an application, different high-level application interfaces can also be built on top of it. The three types of high-level application interfaces illustrated in Fig. 1 are further described below.

Content-specific Encryption Level Modules. The first high-level application interface provides a content-specific service, which is a service consisting of several modules where each module is connected to a specific content. This means, that if we want to use the TE middleware content-specific service when building an application, we must have (or implement) a module that is tailored for the specific content used by the application. The major tasks of the modules are to create masks that protect the content to a desired degree. For example, if the application is MPEG and we only want to encrypt the I-frames created by the application, then the corresponding MPEG-module must be able to create masks that encrypt all I-frames during the data transfer.

Entropy-adaptive Encryption Level. The second high-level application interface provides an entropy-adaptive service. When this service is used a particular EL is automatically selected based on the entropy of the message. The assumption is that data with higher entropy will require a lower EL and vice versa. A message with low entropy, such as a plain text message, will use a higher EL than a message with high entropy, such as a message containing compressed data. The entropy could either be specified

by the application as a parameter passed to the interface module or determined by the module itself based on the type of data to encrypt. An extension of this idea is to adaptively change the EL during operation based on the current entropy on a subset (i.e., a window) of the message.

Load-balanced Encryption Level. The third interface provided by the middleware is a load-balancing service, which should be regarded as a complement to the other two interface types described above. The primary idea with this interface is to provide run-time EL adaptation based on system resource usage. In this case, the application specifies a number of acceptable EL settings where the highest specified EL is used as long as system resources are sufficient. By using this interface, an overloaded sender can, for example, decrease the EL in order to fulfill all its assigned tasks in a proper way. Later, if or when the load decreases, the EL can be raised again. The module implementing this interface needs to interact with the operating system to monitor load and/or battery characteristics on the host.

3.4 Transport Service

The TE service requires a transport service that can reliably transfer messages of various types. Such a service can in turn be implemented on top of a variety of transport protocols, including the Transmission Control Protocol (TCP) [18], Real-time Transport Protocol (RTP) [20] on top of User Datagram Protocol (UDP) [17], and Stream Control Transfer Protocol (SCTP) [27]. Depending on the choice of transport protocol, different functionality must be implemented to support a reliable message abstraction. Among the transport protocols mentioned above, minimal functionality on top of the transport layer needs to be implemented when using the SCTP protocol. Probably the most attractive features in SCTP, when implementing the middleware service, are that the protocol is at the same time connection-oriented and based on messages. Messages in SCTP simplify the coding and decoding of ordinary data and the encryption mask. The connection-oriented feature simplifies reassembly of packets on the receiver side. RTP/UDP also provides a suitable message abstraction but lacks built in reliability support. TCP, on the other hand supports reliable sequenced delivery but lacks a message abstraction.

4 Implementation and Performance

The key components of the proposed middleware have been implemented in C/C++ on top of SCTP. In order to evaluate the performance of our TE middleware a number of experiments were carried out. Since the overhead introduced by using one of the high-level application interfaces will vary, we chose to not include this part in the performance evaluation. Instead the data was passed directly to the block-based selective encryption module using different ELs.

In the experiments, two PCs connected by an Ethernet crossover cable were acting as sender and receiver. As previously mentioned, the AES algorithm with a 128 bit key was used. In all test cases, a randomly generated source file with the size of 10 MB was

transferred. The TE service is, however, not dependent on the size of the data source. The length of the encryption mask was 64 bits in all test cases and thus each EDU contained 64 data blocks. Each data block was 128 bits in size resulting in an EDU size of 1024 bytes. Each test run was repeated 40 times and the time for encryption and transmission was measured at the server side using the `gettimeofday()` library function. Since the variations between measures within the same test run were very small, only mean values are plotted in the figures.

4.1 Measured Computational Gain and Overhead

The purpose with the first test case is to verify that our TE service offers a linear scalability with respect to computational time for encryption at different ELs, and that the overhead introduced by the block selection mechanism is small. Computational time for encryption and transmission together were measured. As a reference, pure AES encryption without using the TE service was also measured. The calculated mean values are shown in Fig. 4.

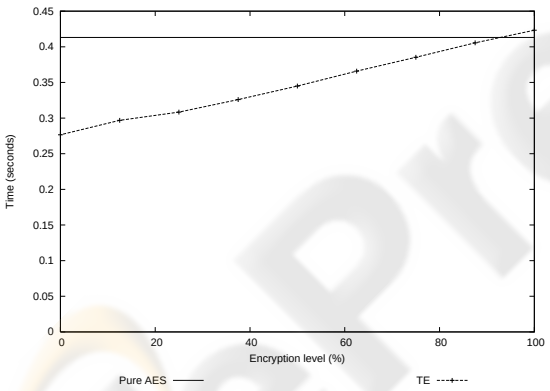

Fig. 4. Measured time for encryption and transmission at different ELs.

The figure indicates that the computational overhead scales linearly with respect to the EL. The overhead produced by the block selection mechanism can be found by comparing the measured time for pure AES encryption with the measured time for TE encryption. As long as the amount of encrypted blocks are less than or equal to 93%, our TE service produces less overhead than encrypting everything using pure AES. Hence, the cost of adding TE as a generic middleware is quite low.

4.2 Impacts of Dynamic Changes

The next test case investigates the impacts of changes of the encryption mask at runtime. Six different test runs were conducted with different numbers of encryption mask changes. In all runs, an EL of 50% was selected. In the first run, an initial encryption

mask was transferred and after that no changes at all were made. In the other five test runs, an encryption mask was initially transferred and after that the encryption mask was changed 1999, 2499, 3332, 4999, and 9999 times, respectively. These values correspond to changes after every fifth, fourth, third, second, and each 1024 byte EDU. Figure 5 shows the result of this test case.

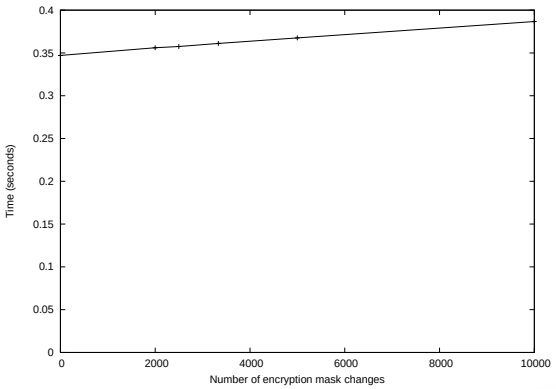

Fig. 5. Impacts of dynamic changes of the encryption mask when transmitting data with a 50% EL.

As can be seen in Fig. 5, the computational overhead of encryption mask changes at run-time is quite small. For example, the extra overhead produced when the encryption mask is changed 1999 times, i.e., after every fifth EDU, is on average less than 2.6% of the total computational time. Note that this is an extremely frequent change of the mask. Thus, the overhead of the mechanism for security adaptation is almost negligible.

5 Future Work

Our current implementation of the TE middleware includes the block-based selective encryption component and a transport layer service based on SCTP. In the future we plan to evaluate both the performance and security of all its components and also build a transport service based on RTP/UDP. An investigation of how the TE service performs at different ELs on a heavily loaded server is already under way. Our current approach is to use the TE middleware together with a video server that serves multiple simultaneously connected clients. Experimental evaluations on the client side when using the TE service are also needed. Additionally, a detailed analysis of the achieved security at different ELs is needed. Hence, the strength of block-based selective encryption and the various high-level interfaces, when different amounts of the content are encrypted, must be investigated. We are currently working on an idea that is primarily based on guesswork [15]. Some initial results have been reported in [11].

6 Concluding Remarks

This paper presents a TE middleware service that can be used by different applications and by different contents. The design of the different components that constitute the middleware is described. In addition, a prototype implementation of the middleware on top of SCTP as the transport protocol and with AES as the encryption algorithm is presented and evaluated. From the evaluation we conclude that the proposed TE middleware is promising with respect to the potential reduction of computational overhead for data encryption and that the added cost of providing TE over a generic middleware is low.

Acknowledgments

This research is supported in part by grants from the Knowledge Foundation of Sweden and is carried out as part of the work on the development of dynamic security services in heterogeneous wireless networks within NEWCOM.

References

1. H. Cheng and X. Li. Partial encryption of compressed images and videos. *IEEE Transactions in Signal Processing*, 48(8):2439–2451, August 2000.
2. M. Van Droogenbroeck and R. Benedett. Techniques for a selective encryption of uncompressed and compressed images. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'02)*, pages 90–97, Ghent, Belgium, September 9–11, 2002.
3. J. Goodman and A. P. Chandrakasan. Low power scalable encryption for wireless systems. *Wireless Networks*, 4(1):55–70, 1998.
4. C. Griwodz, O. Merkel, J. Dittmann, and R. Steimetz. Protecting VoD the easier way. In *Proceedings of the ACM Multimedia*, pages 21–28, Bristol, United Kingdom, September 13–16, 1998.
5. H. Johnson. *Toward Adjustable Lightweight Authentication for Network Acess Control*. PhD thesis, Blekinge Institute of Technology, Karlskrona, Sweden, December 2005.
6. T. Kunkelmann and U. Horn. Video encryption based on data partitioning and scalable coding: A comparison. In T. Plagemann and V. Goebel, editors, *Proceedings of the 5th Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'98)*, volume 1483 of *Lecture Notes in Computer Science*, pages 95–106, Oslo, Norway, September 8–11, 1998. Springer-Verlag.
7. Y. Li, Z. Chen, S. M. Tan, and R. H. Campbell. Security enhanced MPEG player. In *Proceedings of the 1996 International Workshop on Multimedia Software Development (MMSD'96)*, pages 169–176, Berlin, Germany, March 25–26 1996.
8. S. Lindskog and A. Brunstrom. Design and implementation of a tunable encryption service for networked applications. In *Proceedings of the First IEEE/CREATE-NET Workshop on Security and QoS in Communications Networks (SecQoS 2005)*, September 9, 2005.
9. S. Lindskog, A. Brunstrom, and E. Jonsson. Dynamic data protection services for network transfers: Concepts and taxonomy. In *Proceedings of the 4th Annual Information Security South Africa Conference (ISSA 2004)*, June 30–July 2, 2004.
10. T. Lookabaugh and D. C. Sicker. Selective encryption for consumer applications. *IEEE Communications Magazine*, 42(5):124–129, May 2004.

11. R. Lundin, S. Lindskog, A. Brunstrom, and S. Fischer-Hübner. Using guesswork as a measure for confidentiality of selectively encrypted messages. In *Proceedings of the First Workshop on Quality of Protection (QoP 2005)*, September 15, 2005.
12. J. Meyer and F. Gadegast. Security mechanisms for multimedia data with the example MPEG-I video, 1995. http://www.gadegast.de/frank/doc/secmeng.pdf.
13. Nautilus secure phone homepage. http://nautilus.berlios.de/, June 9, 2004.
14. C. P. Pfleeger and S. L. Pfleeger. *Security in Computing*. Prentice Hall, 3rd edition, 2003.
15. J. O. Pliam. *Ciphers and their Products: Group Theory in Private Key Cryptography*. PhD thesis, University of Minnesota, Minnesota, USA, 1999.
16. M. Podesser, H. P. Schmidt, and A. Uhl. Selective bitplane encryption for secure transmission of image data in mobile environments. In *Proceedings of the 5th IEEE Nordic Signal Processing Symposium (NORSIG'02)*, Tromsø/Trondheim, Norway, October 4–6, 2002.
17. J. Postel. RFC 768: User datagram protocol, August 1980. Status: Standard.
18. J. Postel. RFC 793: Transmission control protocol, September 1981. Status: Standard.
19. P. A. Schneck and K. Schwan. Dynamic authentication for high-performance network applications. In *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service (IWQoS'98)*, Napa, California, USA, May 18–20, 1998.
20. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A transport protocol for real-time applications, January 1996. Status: Standard.
21. A. Servetti and J. C. De Martin. Perception-based selective encryption of G.729 speech. In *Proceedings of the 2002 IEEE Internatinal Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 621–624, Orlando, Florida, USA, May 13–17, 2002.
22. C. Shi and B. Bhargava. An efficient MPEG video encryption algorithm. In *Proceedings of the Workshop on Security in Large-Scale Distributed Systems*, pages 381–386, West Lafayette, Indiana, USA, October 20–22, 1998.
23. Sony Electronics. Passage: Freedom to choose, February 10 2003. http://www.sonypassage.com/features.htm.
24. G. A. Spanos and T. B. Maples. Performance study of a selective encryption scheme for security of networked, real-time video. In *Proceedings of the 4th International Conference on Computer Communications and Networks (ICCCN'95)*, pages 72–78, Las Vegas, Nevada, USA, September 1995.
25. Speak Freely homepage. http://www.speakfreely.org/, June 9, 2004.
26. E. Spyropoulou, C. Ager, T. E. Levin, and C. E. Irvine. IPSec modulation for quality of secuirty service. In *Proceedings of the Third Annual International Systems Security Engineering Association Conference (2002 ISSEA Conference)*, Orlando, Florida, USA, Mars 2002.
27. R. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. RFC 2960: Stream control transmission protocol, October 2000. Status: Standard.
28. L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In *Proceedings of the ACM Multimedia 1996*, pages 219–229, Boston, Massachusetts, USA, November 1996.
29. P. K. Yogender and H. H. Ali. Impacts of employing different security levels on qos parameters in virtual private networks. In *Proceedings of the 24th IASTED International Multi-Conferenceon on Parallel and Distributed Computing and Networks (PDCN)*, February 14–16, 2006.