Security Enhanced MPEG Player

Yongcheng Li* Zhigang Chen See-Mong Tan Roy H. Campbell

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield
Urbana, IL 61801

{*ycli,zchen,stan,roy*}@cs.uiuc.edu

Abstract

Conventional cryptography deals with the encryption and decryption of traditional textual data. The advent of networked multimedia systems will make continuous media streams, such as real time audio and video, increasingly pervasive in future computing and communications environments. It is thus important to secure networked continuous media from potential eavesdroppers. In this paper, we consider the process of real-time encryption and decryption for video streams. We implement a software-only *security enhanced MPEG player. The security enhanced player implements a protection hierarchy by specializing the encryption scheme based on MPEG's coding sequences. Encryption may be performed on only I frames, I and P frames, on all I, P and B frames. Increased protection incurs more overhead as more encryption is done. Our security enhanced MPEG player incurs small average overheads in terms of achievable frame rate compared with the unmodified MPEG player, depending on the MPEG frame size, encoding format, and encryption method used, with speeds fast enough for most multimedia Internet applications. This is demonstrated by its integration with Vosaic, a real time multimedia World Wide Web browser. We also observe that increased compression actually results in* less *cryptographical overhead, due to the fact that more compression means less data, as well as longer dependencies between MPEG frames. Our work shows that video streams can also be encrypted and decrypted while satisfying the real time requirements in the present day Internet.*

1. Introduction

In recent years, distributed systems security has been the object of considerable research[11, 1, 3, 12, 10]. Many distributed applications require information exchange over insecure public channels. Private exchanges require protection from eavesdroppers. Secure information exchange is a necessity in distributed systems, especially with the budding of Internet commerce.

Encryption and decryption provide the basic technology for building secure systems. There are two basic encryption methods:

- **Secret Key Encryption**[4]:

  A single key is used for both encryption and decryption. Only authorized users possess the key.

- **Public Key Encryption**[6]:

  Both a public key and a private key are used. In the widely known RSA method, the keys are complementary: each key may unlock ciphertext created with the other key. The public key is published and widely disseminated, while the private key is kept secret.

Public key encryption alleviates the problem of key distribution. However, decryption in currently known public key schemes is slower than in secret key schemes. Assuming the plain text is much larger than an encryption key, a compromise method is to encrypt the plain text to be delivered with a secret key, and to encrypt the secret key using the receiver's public key. The receiver can then quickly decrypt the ciphertext after it decrypts the secret key with its private key.

Security systems that have been developed include Kerberos[10], Riordan's Internet Privacy Enhanced Mail (PEM)[5], and Zimmerman's Pretty Good Privacy (PGP)[15, 16, 14] . Kerberos is an authentication system designed by Miller and Neuman for open network computing environments in MIT's Project Athena. Riordan's Internet Privacy Enhanced Mail (RIPEM) is a implementation of Privacy Enhanced Mail (PEM) which allows electronic mail to have the four security facilities provided by PEM: disclosure protection, originator authenticity, message integrity measures, and non-repudiation of origin. PGP by Philip Zimmermann uses public-key encryption to protect

*This work was supported by NSA grant 1-5-20369 MDA 904-94C-6113.

electronic mail and data files. PGP is full featured, fast, with sophisticated key management, digital signatures, data compression, and boasts good ergonomic design. It has also been the subject of considerable legal controversy.

The systems surveyed above deal with the cryptography of conventional textual data. The security problem of video and audio streams has not been considered in their design. This does not mean that video and audio streams do not need security, on the contrary, many of them, such as personal video streams, video conferences, and collaborative sensitive video data, require protection. While performance is an issue with the encryption and decryption of conventional textual data, it is more so with continuous media streams. Since playing video is a real time task, encryption and decryption cannot take too much time, otherwise system performance will suffer. There is a dearth of existing studies on the impact of secure encryption and decryption methods on real time video streams.

In this paper, we present a secure video encoder and player based on Berkeley's *Mpeg Player*[7, 8] and Zimmerman's PGP. We were interested in seeing how far we could go with embedding security in a video player, with a *software-only* implementation. The measured performance with our implementation is 25.28 frames per second (fps) where the original MPEG player performs with 29.87 fps for a video frame size of 160×120, and 11.03 fps compared with 12.38 fps for a video frame size of 320×240. The results show that video streams can also be secured without excessive performance degradation.

The rest of this paper is organized as follows. In section 2, we provide background on MPEG coding and PGP's encryption and decryption. In section 3, the implementation of our security enhanced MPEG player is described. In section 4, we present our experimental results for decryption and playback. In section 5, we look at encryption at the server end. In section 6, some observations into the relationship between compression and cryptography are made and analyzed. In section 7, we present the application of our encryption and decryption scheme in Vosaic[13], a prototype World Wide Web browser that incorporates real time video and audio into Web hypertext pages. Finally, we summarize our paper in section 8.

2. MPEG and PGP

One problem associated with the encryption and decryption of video streams is the real time requirement of time-constrained continuous media. Video frames need to be displayed at a certain fixed rate, measured in the number of frames per second (fps). If decrypting a video stream takes too much time, video playback will suffer when the player cannot meet the frame rate requirements.

The goal of our approach was to investigate the best performance achievable with a software-only implementation utilizing readily available, standard software components. We chose *Mpeg Player* as the client display and MPEG decoding software as it is easily available. We based our encryption and decryption methods on PGP software. Experiments show that the display rates reached by our security enhanced MPEG player is comparable with the original MPEG player.

2.1. *Mpeg Player*

Mpeg Player[7, 8] implements MPEG-1, the ISO/IEC standard for medium quality and medium bit rate video and audio compression. *Mpeg Player* was developed at the University of California at Berkeley. The compression ratio of MPEG is in the range of 50 : 1 to 100 : 1, depending on the image sequence type and the desired quality. The *Mpeg Player* decoder can display a video stream at close to the full frame rate of 30 fps on the platforms which we based our experiments (Sun Sparcstation 10 and Silicon Graphics Indy workstations).

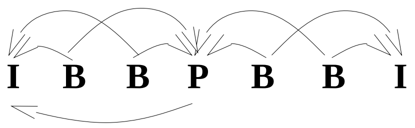

Figure 1. Inter-frame dependencies for the MPEG sequence IBBPBB.

MPEG encodes a video stream with three frame types:

- I-frames(intra-frames),
- P-frames (forward predicted frames), and
- B-frames (bi-directional predicted frames).

An I-frame is encoded as a single run-length encoded image independent of any past or future frames. A P-frame is encoded relative to the closest preceding reference frame (I-frame or P-frame). A B-frame is encoded relative to the closest preceding reference frame, the closest following reference frame (I or P), or both frames. If an I-frames is not decoded correctly, then all the following frames until the next I-frame will not be decoded correctly. Figure 2.1 depicts the dependencies in a common MPEG sequence IBBPBB.

A MPEG video stream is organized around a set of meta-data codes:

1. Sequence start code
2. Group start code

3. Picture start codes (one I-frame and some P or B-frames).

The group start code and the picture start code repeat until finally a sequence end code is met. The start code metadata is used for parsing the video stream. Figure 2 illustrates the structure of an MPEG file as indicated by its metadata codes.

2.2. PGP Software

Pretty Good Privacy combines the convenience of the RSA public key cryptosystem with the speed of conventional cryptography. It includes message digests for digital signatures, data compression before encryption, good ergonomic design, and sophisticated key management[15].

Each time a message is encrypted, a *session key* is randomly generated and used to encrypt the message. The session key is encrypted using the recipient's public key. The input and output parameters of PGP functions are file names or file descriptors: PGP reads a file, encrypts or decrypts it, or signs the file. It can also generate public keys, and add new public keys to a stored public-key file. PGP uses the RSA algorithm for its public key scheme and the IDEA algorithm for conventional secret key ciphering. Some extensions to PGP were added for video stream cryptography. The extensions are detailed below.

3. Security Enhanced MPEG Player

A simple way to encrypt a video stream is to apply some encryption scheme, such as that provided by PGP, on the whole video file. The MPEG file can be used as an input file to PGP, which then encrypts the MPEG file. The encrypted MPEG file is decrypted before it is displayed by the MPEG player. This works for some small video files where the whole video file can be transmitted on a network and stored at the receiver site before being displayed.

However, it is impractical to cache large video files at the receiver before it is displayed. The retrieval of large video files also imposes an unnecessary latency before playback is begun. Real time playback of networked videos require that videos be sent *on-demand*, and in *real-time.* In such a situation, the video file cannot be encrypted as a whole. Instead, it is *incrementally encrypted.*

3.1. Video Encryption with PGP

In our approach, we encrypt MPEG video files frame by frame, as frames are independently encapsulated units of MPEG files. Because PGP only deals with full files, we extended it to deal with buffered frames. The following functions were added to the original PGP distribution:

- *generate_session_key(key)*: a 24 byte random session (IDEA) key is generated and returned.

- *IDEA_buffer(key, CRYPTO_FLAG, inbuffer, outbuffer, bufferlen)*: This function does the IDEA encryption or decryption depending on the *CRYPTO_FLAG* using *key*, the input message is stored in *inbuffer* and encrypted or decrypted message is put in *outbuffer*. *Bufferlen* specifies the input buffer length.

- *encrypt_buffer(public_keys, inbuffer, outbuffer, bufferlen)*: Encrypt the message in *inbuffer* with length *bufferlen* using public key *public_keys* and output the result in *outbuffer*.

- *decrypt_buffer(inbuffer, outbuffer)* Decrypt an encrypted message in *inbuffer* using one's private key and store the output in *outbuffer*.

Functions *encrypt_buffer* and *decrypt_buffer* are used to encrypt and decrypt the random session key.

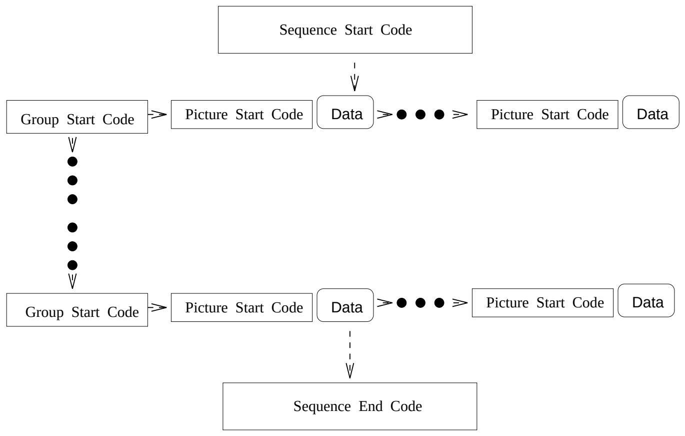

Figure 2. The structure of an MPEG file

In order to encrypt the MPEG file frame by frame, the file is parsed. Figure 2 illustrates the structure of a MPEG encoded file. The start codes are metadata that are used to help decoding. We use them to extract the frames. The sequence start code and sequence end code is used to indicate the beginning and the end of the file. The group start code indicates the start of a group of frames. The picture start code signals the beginning of a frame. The frame type is included in the picture start code. Since the start codes have no direct relation to the content of the video frames, they are left unencrypted. This allows the decryption process to parse the video stream in preparation for frame decryption.

3.2. Improving Performance Through a Protection Hierarchy

The properties of the I, P, and B frames can help further improve the encryption and decryption performance. Since B frames depend on I or P frames, and P frames depend on on the closest preceding I frame, we need only encrypt the I frames while leaving the P and B frames untouched. Without I frames, one cannot decode P and B frames.

However, the amount of information recoverable from a P or B frame without the associated I frame depends on the encoded video clip. The encoding pattern (i.e. IBBBP) specifying what type of frame is to be used to encode each source video frame is independent of the video's content. If the encoder is told to encode a source video frame as a B or P at a *scene transistion*, the encoder will not be able to detect any inter-frame dependencies, hence it will encode the frame's macroblocks (sub-parts of the frame) as I macroblocks, which would be fully decodable. In a typical MPEG video such as the Star Wars movie, scene transitions occur on the average of once every 7 seconds[2]. Encoding only the I frame would thus allow potential eavesdroppers to recover an average of several frames every 7 seconds.

This scenario illustrates a tradeoff in MPEG video stream cryptography between performance and security. A *protection hierarchy* is used where one may choose to encrypt

1. only I frames,
2. I and P frames, or
3. all I, P and B frames

in any video sequence. Stronger protection for securing video sequences is thus achievable by choosing extra frame types requiring encryption. Naturally, higher protection correspond to higher overheads in decryption.

4. Decryption and Playback

We compare the performance of our security enhanced MPEG player SE_MPEG against the original MPEG player. All experiments were performed on a Sun SPARCstation 10 machine, running Solaris version 2.4, with 32 MB of memory. We ensured that the comparisons were done under lightly loaded conditions. In this section, data is presented as MPEG files that are stored and read locally from disk. The local testing shows the maximum difference between SE_MPEG and original MPEG player. The MPEG files used in our experiments are listed in Table 1. We experimented with two different video frame sizes. The first is a small clip at a size of 160×120. The second is larger and longer, at 320×240. The MPEG coding pattern for both clips is IPBBIBB.

File Name	File Size (bytes) (bytes)	Frame Number	Frame Size
renat.mpg	555297	351	160×120
orincsa.mpg	1708105	850	320×240

Table 1. MPEG files used in our experiments.

File Name	Unen-crypted	Encrypt I,P,B	Encrypt I,P	Encrypt I
renat.mpg	29.87	23.10	24.10	25.28
orincsa.mpg	12.36	10.68	10.87	11.03

Table 2. Comparison of speed (fps) with the original MPEG player against the SE_MPEG player with encryption on I, I and P, or I, P and B frames.

Table 2 shows the experimental results for local files. The data in the table are the average of ten runs. For the small frame size video file, the original MPEG player has a display speed of 29.87 frames per second on our SPARCstation. When all the frames are encrypted, our SE_MPEG player has a display speed of 23.10 frames per second. When we encrypt I and P frames, we obtained a display speed of 24.10 frames per second. If only the I frame is encrypted, our SE_MPEG player reaches a display speed of 25.28 frames per second. Table 3 presents the experimental results in terms of performance degradation over the original insecure player.

The achieved frame rate with only I frames encrypted did not differ substantially from the baseline, and had only limited effect on the perceived video quality. Similar results were obtained for the large frame size video file. With decryption, the display speeds decreased from 12.34 fps to 11.06, or 10.89, or 10.71 depending on what types of frames were encrypted. The experimental data shows that that video stream decryption does not dominate processing time. Fast display speeds are achievable in our software-only implementation of encrypted video streams.

File Name	Performance Degradation (%)		
	I,P,B frames	I,P frames	I frames
renat.mpg	22.6	19.3	15.4
orincsa.mpg	13.6	12.1	10.7

Table 3. Percent performance degradation for the protection hierarchy.

25.28 frames per second is fast enough for multimedia Internet applications. In [9], Smith gives a series of Internet experiments with *Cyclic-UDP*, a media-specific protocol designed for current local and wide area Internet multimedia applications.[1] In Smith's experiments, a rate of 17 frames per second was needed in order to send a MPEG video sequence and with its associated audio from U.C. Berkeley to Cornell University. In such a scenario, our encryption and decryption operations will cause little performance degradation.

5. Encryption and Transmission

There are two ways to encrypt a video stream. The first is to encrypt the video stream in advance and then read the encrypted video file from disk and transmit it over the network to the client. The second way is to encrypt video frames on-the-fly, with no encryption done beforehand. The two methods trade off playback latency and CPU overhead for per frame processing against one another. Encryption in advance incurs no overhead in retrieving and transmitting the video stream. However, the client must wait while the whole video file is being encrypted. Doing the encryption on-the-fly can remove this latency, but extra CPU time is needed before each frame is sent out the network.

On-the-fly encryption also avoids the problem of transmission errors causing a larger failure than would occur if the error happened transmitting unencrypted frames. For example, if an error occurs in a packet containing an I frame, the frame is lost irrespective of whether it is encrypted or not. If the whole file is encrypted, then losing any part of the whole file requires retransmitting the piece lost, thus adding to the latency before playback can begin.

Some experiments were done to show the effect of each method. The machines used were Sun SPARCStation 10's running Solaris 2.4 as in the previous set of experiments.

5.1. Whole File Encryption

With our experiments, the time needed to encrypt a video file is approximately one quarter of the total local displaying time. That is, for an hour long video, 15 minutes are need to encrypt the whole file. The time is reasonable since the size of an hour long video stream is usually hundreds of megabytes. Reading the file from the disk and then writing it to the disk would have already taken a significant amount of time. This scheme is usable in situations where clients reserve a video and the server can use its idle time to encrypt the video file.

[1]In *Cyclic-UDP*, high priority packets are given a better chance of delivery by allowing more retransmissions of these packets if they are lost.

5.2. On-The-Fly Encryption

Frame Size	**B Frame** (ms)	**P Frame** (ms)	**I Frame** (ms)
160×120	2-4	12-15	18-22
320×240	2-4	12-15	36-44

Table 4. The encryption time of video frames.

In our study of the second method, we measured the time encryption imposes on the server. We obtained encryption times for the sample videos used in our experiments. This is illustrated in table 4. The encryption time is 2 to 4 milliseconds, 12 to 15 milliseconds, and 18 to 22 milliseconds, and 36 to 144 milliseconds for B, P, small I and large I frames respectively.

The experimental results showed smaller times than we expected. We conjecture that this is a consequence of the data being *encrypted as it is being copied for transmission*. Thus the memory to memory copy would have masked the overhead of encryption, and the processor cache would have been primed before transmission.

For a MPEG file with frame pattern IPBBIBB, the average per frame encryption time is approximately 9 milliseconds. Then, conservatively assuming that the encryption time is 10 milliseconds, a single server CPU can support at most 10 video streams at a frame rate of 10 fps with the encryption option on. If the server does not simultaneously transmit multiple encrypted video streams, then on-the-fly encryption is not a serious problem.

6. The Effect of MPEG Frame Pattern on Video Cryptography Performance

An important factor affecting the display speed is the I, P, B frame pattern in the MPEG file. The frame pattern for *renat.mpg* and *orincsa.mpg* is IPBBIBB. The display speed is increased when P or B frames are increased (corresponding to a better compressed file). An extreme case is to generate a MPEG file with all I frames. A separate experiment showed that with such an extreme file, and a frame size 160×120, a display speed of 17.09 fps can be reached without any decryption operation. With encryption and decryption, the video is displayed at a speed of 11.40 fps. Thus a 33 percent performance degradation occurs because of intensive computing.

We thus arrive at the following observation:

Observation
(Compression and Cryptography): *MPEG video compression and video cryptography go hand-in-hand: better*

compression also reduces cryptographical overhead.

Compression allows less data to be forwarded, thus a more highly compressed video stream will have less data to encrypt and decrypt than a video stream that is less compressed. This is intuitively obvious. In addition, because the MPEG standard makes use of forward and bi-directionally predicted frames, a more compressed video stream will have longer dependencies between frames. I frames act like decryption keys in the cryptographical sense. P and B frames in an MPEG picture group depend on the I frame, and possession of the I frame is necessary in order that the P and B frames be correctly decoded. Encrypting an I frame has *more effect* in MPEG groups that have more P and B frames (ie. those that are more compressed).

This means that protecting a more compressed video stream at any level of the protection hierarchy (ie. by encrypting I frames only, or I and P frames, or all I, P and B frames) incurs less overhead in both encryption and decryption than the same video stream that is less compressed.

7. Application in Vosaic

We integrated our video encryption and decryption mechanism in Vosaic[13], a World Wide Web real time video and audio browser. Vosaic, short for Video Mosaic, incorporates real time video and audio into standard hypertext pages which are displayed in place. Video and audio transfers occur in real time; there is no file retrieval latency. A real time protocol, called VDP, is used in Vosaic. VDP is specialized for handling real time video over the WWW. VDP attempts to reduce inter-frame jitter and dynamically adapts to the client CPU load and network congestion.

To transmit securely a video stream over the internet, the server must have the client's public key. A random session key is generated for each transmission. The random session key is encrypted with the client's public key and delivered to the client first. The client uses its corresponding private key to decrypt the random session key. The video stream is encrypted with the session key and the client decrypts the video with the session key. We performed a series of tests comparing the performance of encrypted and non-encrypted real time video streams in Vosaic. The VDP protocol adapts to network conditions between the server and client, as well as the client CPU load, by dropping frames at the server end based on a closed-loop feedback scheme from the client. The details are given in [13].

As VDP adapts to network and client load, the transmitted frame rate varies according to prevailing conditions. With the MPEG decoder built into Vosaic, a Vosaic browser client can achieve a maximum decoding speed of 13 frames/second. The test environment saw video streamed from a video server at the National Center for Supercomputing Applications (NSCA) to our laboratory. NCSA is on the campus of our institution and connected by Ethernet to the local campus LAN.

In our tests, unencrypted videos were displayed at rates from 5 to 10 fps. Under these conditions, our experiments showed no measurable difference between encrypted and non-encrypted video streams. Indeed, there was no perceivable difference in observed video quality between the encrypted and unencrypted versions, nor was there a measurable difference in displayed frame rates. Our software-only video cryptography scheme does not cause measurable frame rate degradation for an Internet multimedia application like Vosaic.

8. Conclusion

Conventional cryptography has traditionally dealt with textual data. We extended the PGP scheme and applied it toward the cryptography of continuous video streams. A security enhanced MPEG player was implemented based on Berkeley's *Mpeg Player* and the PGP software package.

MPEG video compression and video encryption are closely linked. The new encryption scheme is specialized for MPEG's coding sequences and orders the level of protection provided in a hierarchy. Taking advantage of the inter-frame dependencies in MPEG, one may choose to encrypt only I frames, I and P frames, or all I, P and B frames. Increased protection is traded off against more overhead as more encryption is done. We also observe that increased compression actually results in *less* cryptographical overhead, due to the fact that more compression means less data, as well as longer dependencies between MPEG frames.

The highest display speed in our *software-only* implementation was measured at 25.28 frames per second. This is fast enough for many Internet applications, such as Vosaic. Our experiments show that video streams can also be encrypted and decrypted while satisfying the real time playback requirement.

9. Acknowledgements

We thank our anonymous reviewers for providing insightful comments.

References

[1] M. Blaze. Key management in an encryption file system. In *Proceedings of 1994 Summer USENIX*, pages 27–35, Boston, MA, June 1994.

[2] Chuck Kalmenek. Personal communication, February 1994.

[3] B. Lampson, M. Abadi, M. Burrows, and T. Wobber. Authentication in distributed systems: theory and practice. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 165–182, Pacific Grove, CA, Oct. 1991.

[4] N. B. of Standards. Data encryption standard. In *Federal Information Processing Standards Publication 46*, Government Printing Office, Washington D.C., 1977.

[5] M. Riordan. RIPEM user guide: for RIPEM version 2.1. (Revised March 1995 for RIPEM 2.1 by J. Thomson) Available on the WWW via ftp://ripem.msu.edu/pub/crypt/ripem/ripemps, 1993.

[6] R. Rivest, A. Shamir, and L. Adleman. A method of obtaining digital signatures and public-key cryptosystems. *CACM*, Feb. 1978.

[7] L. Rowe, K. Patel, B. Smith, and K. Lin. MPEG video in software: representation, transmission, and playback. In *Proceedings of High Speed Networking and Multimedia Computing, IS&T/SPIE Symp. on Elec. Imaging Sci. & Tech.*, San Jose, CA, Feb. 1994.

[8] L. Rowe and B. Smith. A continuous media play. In *Proceedings of 3rd International Workshop on Network and OS Support for Digital Audio and Video*, San Diego, CA, Nov. 1992.

[9] B. Smith. Implementation techniques for continuour media systems and applications. PH.D Thesis, Dept. of Computer Science, University of California at Berkerley, 1993.

[10] J. Steiner, C. Neuman, and J. Schiller. Kerberos: an authentication service for open network systems. In *Proceedings of USENIX Winter Conference*, pages 191–202, Dallas, Texas, Feb. 1988.

[11] J. Stewart. SunOS, C2, and Kerberos: a comparative review. In *Proceedings of UNIX Security Symposium III*, pages 265–284, Sept. 1992.

[12] T. Woo, R. Bindignavle, S. Su, and S. Lam. SNP: an interface for secure network programming. In *Proceedings of 1994 Summer USENIX*, pages 45–58, Boston, MA, June 1994.

[13] Z. Chen, S. Tan, R. Campbell, and Y. Li. Real Time Video and Audio in the World Wide Web. In *Fourth International World Wide Web Conference*, Boston, MA, December 1995. World Wide Web Organization. Also available via http://choices.cs.uiuc.edu/research/Vosaic/vosaic2.html.

[14] P. Zimmermann. File formats used by PGP 2.6. Available on the WWW via ftp://ftp.pegasus.esprit.ec.org/pub/-arne/pgformat.ps.gz, May 1993.

[15] P. Zimmermann. PGP user's guide, Volume I: essential topics. Available on the WWW via ftp://ftp.pegasus.esprit.ec.org/-pub/arne/pgpdoc1.ps.gz, Oct. 1994.

[16] P. Zimmermann. PGP user's guide, Volume II: special topics. Available on the WWW via ftp://ftp.pegasus.esprit.ec.org/-pub/arne/pgpdoc2.ps.gz, Oct. 1994.