# An Evaluation of Current MPEG-1 Ciphers and their Applicability to Streaming Video

Jason But, Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{jbut,garmitage}@swin.edu.au

***Abstract* - Copyright protection is one of the many aspects of implementing a commercially successful video streaming service and encryption of content is one means through which Copyright can be protected. MPEG-1 is one of the many video compression techniques used in streaming video and a number of MPEG-1 ciphers have been proposed in the past. These ciphers were primarily designed with the idea of public storage of MPEG-1 content. This paper considers the suitability of these ciphers in a streaming video context, concluding that no existing cipher is suitable for use in streaming MPEG-1 video and that a new cipher is required for this purpose.**

***Keywords- MPEG Encryption, Streaming Video, Video-on-Demand, Copyright Protection***

## I. Introduction

Protection of digital content delivered by a streaming video service is one of numerous issues that must be addressed for such a service to be a commercial success. A suitable solution must take into account how streaming video is provided and incorporate into that framework.

A distributed server design involves the use of distributed streaming servers acting as a cache providing video streaming services to the local area. These services will include advanced functionality such as indexed and high-speed playback modes [1, 2]. Advanced playback modes of video caches must continue to be supported when streaming encrypted video. This leads to the development of a set of requirements that an MPEG-1 cipher must meet before it can be considered suitable for use in streaming video [1, 3].

In this paper we examine a number of existing MPEG-1 encryption techniques and evaluate their suitability for use in streaming video. We show that none of the evaluated ciphers meet all of the criteria for use in streaming video. While the MPEG-1 video compression algorithm could be considered out of date, the principles of MPEG-1 compression and its subsequent encryption can typically be applied to more recent video compression algorithms such as MPEG-2 and MPEG-4.

## II. Cipher Requirements

A typical solution for streaming video is Distributed Streaming Servers [2]. Video content is cached at a local streaming server from where it is streamed to a number of local viewers [2, 5]. Distributed servers increase the number of customers who can receive concurrent video streams for a given level of network resource consumption. Copyright protection and Digital Rights Management of streaming video must be considered in view of a Distributed Server implementation [4].

### *A. Distributed Streaming Servers*

Streaming servers ought to be platform-agnostic, offering greater platform choice to the system designer. The servers themselves also becomes targets for theft of content stored on that server [3]. Digital Rights Management must enable content to be installed on servers in encrypted form. As a streaming video solution should function with a range of streaming server products, so should the MPEG-1 cipher be independent of the streaming server implementation [3].

Fig. 1 shows the general relationship between Digital Rights Management and Copyright protection in a distributed server environment. The encrypted content is made freely available by the content owner and cached in encrypted form at the local streaming servers. When a

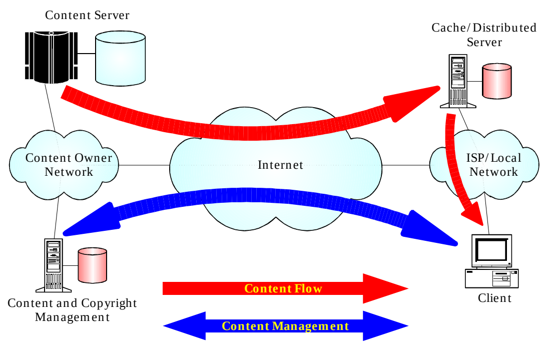

*Fig. 1. Content Protection in a Distributed Server Environment*

consumer wishes to view the video, they purchase a decryption key from the content owner and then stream the encrypted video from the local video server, decrypting it with their purchased key [6].

There are a number of advantages to decoupling the cipher and streaming server implementations. First, the server platform can be chosen based on functionality and cost rather than on the security features. Second, security is implemented by encryption experts rather than by streaming server developers. Third, existing streaming server products can be utilised as is, and future streaming video products will be supported [3].

## *B. Streaming Server Implementation*

A cipher must function within existing and future streaming server environments, so it is important to consider how streaming servers are typically implemented. Video is typically streamed over the network at the average bit rate of the encoded stream, where it is buffered and fed into the decoder at the encoded variable bitrate (Fig. 2). The mechanics of network delivery differs between server implementations, but there is a common factor – the original bitstream is reconstructed prior to decoding [3, 7].

There are three approaches to perform this function:

- Stream the MPEG-1 System Stream over the network, the client passes this data directly to the decoder [3, 6].
- Server extracts encoded Video and Audio Streams from a stored file and delivers them separately. The System Stream is reconstructed at the client prior to being passed to the decoder [3, 6].
- As above, but the client passes each stream to independent MPEG-1 Video and MPEG-1 Audio decoders [3, 6].

***Requirement:*** It must be possible to extract the Video and Audio Streams from the encrypted bitstream and to decrypt and decode these streams independently, without changes to existing streaming server implementations.

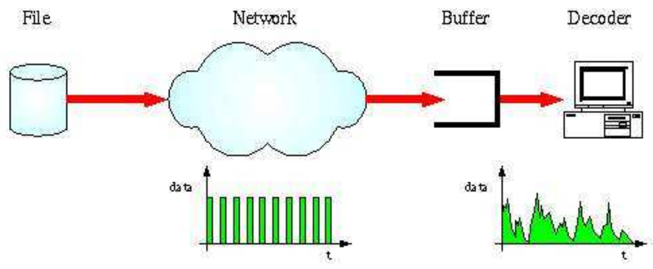

*Fig. 2. Constant Bitrate MPEG Streaming*

Many servers partially decode the bitstream upon installation to confirm that it will be possible to provide indexed and high-speed playback modes [3, 6].

***Requirement:*** The encrypted bitstream must have valid MPEG-1 bitstream syntax so that it can be successfully installed on the server.

Indexed playback is typically implemented by locating timestamps encoded in the stored video file and commencing streaming from that point. The decoder is reset and recommences decoding of the new bitstream [3, 8].

***Requirement:*** Index points into the video stream are typically the start of any Group of Pictures (GOP) in the original bitstream. The streaming server must be able to locate each GOP in the encrypted bitstream, ensuring support of indexed playback of an encrypted bitstream.

***Requirement:*** The cipher must be able to resynchronise itself such that decryption can correctly occur at allowable indexation points (start of GOP) within the video stream.

Not all streaming servers provide high-speed playback modes. Typically a new MPEG-1 Video Stream is constructed from the first I-Frames of each GOP in the original bitstream (Fig. 3). This new bitstream can be played in either direction, utilising less resources than simply playing the original bitstream at a faster rate [3, 9]. The audio stream is not delivered during high-speed playback [3].

***Requirement:*** The streaming server must be able to locate each I-Frame within the encrypted bitstream.

***Requirement:*** The cipher must be able to resynchronise itself such that decryption can correctly occur during high-speed playback modes, each I-Frame must be able to be decrypted independently.

## *C. Other Requirements*

Other desirable qualities of the cipher include scalability, security and support of existing decoder platforms [3].

***Requirement:*** Encrypting the bitstream must not affect the maximum number of concurrent streams that the server can support.

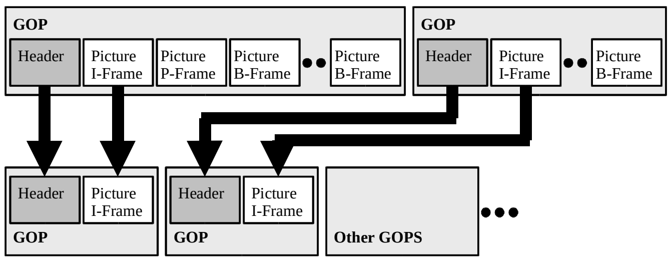

*Fig. 3. Generating a High-Speed Playback Bitstream*

***Requirement:*** The cipher module should not form an explicit part of the decoder, allowing the use of existing and future MPEG-1 decoder implementations.

## III. Existing MPEG-1 Ciphers

We consider a range of existing approaches and find none of them suitable for protection of streaming video. These include using existing secure network protocols, encrypting the entire bitstream at the application layer, and a range of existing MPEG-1 Partial Encryption Ciphers.

The deficiencies of these ciphers are summarised in . None of these ciphers provide a suitable solution for encryption of streaming MPEG-1 video.

### *A. Network and Transport Layer Encryption*

Existing network protocols such as IPSec or Transport Layer Security (TLS) could be used to encrypt the bitstream independently from the streaming server infrastructure, minimising system design requirements.

IPSec is an IP layer solution [11, 12] providing secure links between any two IPsec enabled hosts (often utilized for Virtual Private Networks over public IP networks). In a streaming video scenario IPSec would be installed either on the streaming server or its gateway router. All traffic is encrypted as it leaves the site and decrypted at the client computer. While IPSec is compatible with all existing IP based software [11, 13], it is unsuitable for use with streaming video. IPSec is processor intensive, leading to scalability problems and limiting the maximum number of concurrent streams that can be supported [14].

Another concern is that the content server itself stores the video content in plaintext, making the server an attractive target for attempts to steal content. In addition, decrypted packets are easily captured from the IP Stack at the client.

IPSec was originally designed for secure communications between trusted parties. However, a streaming video service requires distribution of content from one trusted party (the central server) to an untrusted party (the client), potentially via a second untrusted party (the streaming server operator).

TLS – based on Netscape's original implementation of the Secure Sockets Layer (SSL) protocol – is a sockets layer solution [15] providing secure communications between two applications running on IP enabled hosts. As SSL runs on top of TCP/IP, an SSL session can be routed by any IP Router on the network. SSL is typically used to provide secure web services such as Internet banking.

SSL is unsuitable for streaming video, having similar scalability and security problems as IPsec. Further, because SSL provides secure communications over TCP, many existing UDP- or RTP-based content streaming applications would require modification to utilise SSL [13, 15].

### *B. Full Encryption*

Another approach is to encrypt the entire bitstream [14]. One approach is to store video as plaintext on the streaming server, and then encrypt the entire bitstream during delivery.

Table 1 Summary of Suitability of MPEG-1 Encryption Techniques with Streaming Video

| **Cipher** | **Encrypted bit-stream can be stored on Server** | **Scalable to support multiple concurrent streams** | **No changes required to Server implementation** | **Encrypted bit-stream can be served in Indexed Playback modes** | **Encrypted bit-stream can be served in High-Speed Playback modes** | **Cipher can be resynchronised at client during Indexed Playback** | **Cipher can be resynchronised at client during High-Speed Playback** | **Cipher can be efficiently implemented externally to the MPEG-1 decoder implementation** | **Cipher is secure against attack** |
|---|---|---|---|---|---|---|---|---|---|
| IPSec | N/A | ✘ | ✔ | N/A | N/A | N/A | N/A | N/A | N/A |
| SSL | N/A | ✘ | ✘ | N/A | N/A | N/A | N/A | N/A | N/A |
| Full Encryption | N/A | ✘ | ✘ | N/A | N/A | N/A | N/A | N/A | N/A |
| SECMPEG | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ |
| Zig-Zag Permutation Algorithm | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ |
| Video Encryption Algorithm | ✘ | ✔ | ✘ | ✔ | ✘ | Unknown | Unknown | ✔ | ✔ |
| Video Encryption Algorithm – Number 2 | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✘ | ✘ |
| Frequency Domain Scrambling Algorithm | ✔ | ✔ | ✔ | ✔ | ✔ | Unknown | Unknown | ✘ | ✔ |
| A Unique Cipher | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ |
| Multi-Layer Encryption | ✘ | ✔ | ✘ | ✔ | ✘ | Unknown | Unknown | ✔ | ✔ |
| Selective Macroblock Encryption | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ |
| AEGIS | ✘ | ✔ | ✘ | ✘ | ✘ | Unknown | Unknown | ✔ | ✘ |

***Unknown** – The cipher designers have not specified how key resynchronisation takes place.

ISBN 0-646-44190-6

The drawbacks are similar to IPSec and SSL based systems – processing requirements for real-time encryption of multiple streams and potential for server attack. Further, it requires modifications to existing streaming servers to ensure that the video is streamed in encrypted form [14, 16].

A second approach encrypts all video prior to storage on streaming servers, minimising processing requirements on the server and guarding against theft from the server. However, the encrypted bitstreams do not conform to the MPEG-1 bitstream specifications and cannot be streamed by existing streaming servers. Also, existing servers could not provide different playback modes unless additional information is stored about specific index points within the bitstream [8-10].

### *C. SECMPEG*

The SECMPEG cipher (Meyer and Gadegast [17]) applies one of four algorithms (Encrypt all headers; Encrypt all headers and DC co-efficients of I-Macroblocks; Encrypt all I-Frames and I-Macroblocks in P and B Frames; Encrypt entire bitstream). The selected data is encrypted using either the DES or RSA ciphers. The headers are augmented with extra information that allows correct decryption at a later stage.

SECMPEG is unsuitable for streaming video. The header changes make all four modes incompatible with existing MPEG-1 Streaming Server products. It is also not possible to index into the bitstream due to the non-resynchronisation of the cipher employed. SECMPEG encrypted video can only be decrypted from beginning to end at normal playback speed, precluding the implementation of indexed or high-speed playback modes [8].

### *D. Zig-Zag Permutation Algorithm*

In an MPEG-1 video stream, each block of 8x8 pixels in a Macroblock is encoded using a Discrete Cosine Transform (DCT) and processed in a zig-zag pattern. Tang [19] proposed using a random permutation list to map the 8x8 block rather than the fixed zig-zag pattern. The algorithm also splits the DC co-efficient to hide its relatively large value amongst smaller AC co-efficients. The same permutation list is applied to all Macroblocks.

Tang suggests additional modifications. One involves the pseudo-random selection of one of two permutation lists via a cryptographically secure random bit generator, the other applies the DES cipher to blocks of 8 DC co-efficients [19].

As the cipher only modifies the Macroblock contents, the bitstream can be processed by a streaming server and all playback modes are supported. Resynchronisation of the cipher in the indexed and high-speed playback modes is not required because each frame is encrypted with the same list. However, if the permutation list is pseudo-randomly selected the random bit generator must be resynchronised.

This cipher is vulnerable to a known plaintext attack. By comparing the decoded DCT co-efficients, they can be reordered and the permutation list retrieved, which can then be applied to the remainder of the sequence to retrieve all data. This also works if two permutation lists are used – apply both patterns and select the most likely of the two Macroblocks – typically the Macroblock with larger DC and low order AC co-efficients. The cipher is also vulnerable to a ciphertext only attack as described by Qiao [14, 19].

Tang [18] proposes a decoder with a built-in cipher module, allowing the random permutation list to be applied after the co-efficients have been extracted from the bitstream, but before they are decoded into individual pixel values. CPU resource requirements are low as re-organising DCT co-efficients is a simple task. However, incorporating the cipher into the decoder precludes the use of third-party MPEG-1 decoders.

### *E. Video Encryption Algorithm*

Qiao and Nahrstedt [20] propose the Video Encryption Algorithm (VEA). VEA encrypts individual frames – all data at the Picture Layer within the Video Stream is selected for encryption. The Picture Layer is encrypted by:

- Sub-dividing data into blocks of an even number of bytes. Randomly divide each block into two lists of equal length.
- The two lists are XORed to form a third list.
- The encrypted block is constructed from the third list and the second list encrypted using a cipher such as DES.

Decryption involves the decryption of the second list which is then XORed with the third list to retrieve the original first list. The original plaintext stream is then reconstructed. The Picture Layer bitstream format uses a new header block encoding the number and length of Slices within the Picture, shortening rather than lengthening the bitstream [20].

VEA is secure - the second list acts as a unique one-time pad, encrypting the first list. The ciphertext consists of the one-time pad ciphertext and an encrypted copy of the one-time pad. VEA is as secure as the cipher used to protect the second list [20]. However, VEA is not suitable for streaming video - the Picture Layer format has been modified, and High-speed playback modes cannot be implemented since individual I-Frames cannot be extracted from the bitstream. (Since the bitstream expands following decryption, real-time decoders also need to account for extra processor and memory requirements.)

### *F. Video Encryption Algorithm – Number 2*

Shi and Bhargava [21] propose a different form of VEA. In its initial incarnation the cipher encrypted the sign bits of all AC and DC co-efficients within the bitstream. Each bit of a binary key is XORed with the sign bits. When the key bits are exhausted, the key is reused. The authors suggest regular resynchronisation at the beginning of each Group Of Pictures (GOP) by re-starting encryption from the beginning of the key. The cipher was later modified [22, 23] to also encrypt the sign bits of motion vectors.

The encrypted bitstream retains the MPEG-1 headers, so this VEA can be successfully streamed in all playback modes on existing servers.

In both algorithms the key is used directly in the XOR operation (although it is possible to use a random bit generator). The cipher is susceptible to a known plaintext attack. An attacker extracts the corresponding sign bits from the encrypted and plaintext streams, determines the pseudo-random bit sequence (common to each GOP) and decrypts the entire bitstream.

As for the Zig-Zag Permutation Cipher [18], decryption of a VEA encrypted bitstream must be performed within the MPEG-1 decoder. CPU utilisation is efficient as decryption involves only a simple XOR for each coefficient. This precludes use of a third-party MPEG-1 decoder.

### *G. Frequency Domain Scrambling Algorithm*

The Frequency Domain Scrambling Cipher proposed by Zeng and Lei [24, 25] operates on information encoded within the Macroblock layer, in particular the DCT co-efficients. (similar to the Shi and Bhargava VEA cipher [23]). The cipher is strengthened by also considering:

- **Encrypting refinement bits within coefficients** – The refinement (or least significant) bits of the coefficients tend to have an even distribution and can be encrypted without impacting on the compression rate.
- **Block Shuffling** – Divide the bitstream into a series of blocks which are shuffled using a changing table. As only the positions of Macroblocks within the stream are changed, compression remains high.
- **Block Rotation** – Macroblocks are rotated pseudo-randomly. The actual pixel values are unchanged and the compression ratio is not affected.

The cipher is secure [24, 25]. The encrypted content can be streamed as only the Macroblock data is modified, thereby retaining header information required by servers to provide indexed and high-speed streaming.

The Frequency Domain Scrambling Cipher is more reliant on being implemented as part of the decoder than other ciphers [24, 25]. The cipher complexity means that CPU efficiency is only obtained if decryption occurs within the decoding cycle, precluding use of third-party decoders.

### *H. A Unique Cipher*

Griwodz et al [26] propose a unique algorithm for protection of distributed video. A Poisson process is used to select bytes from the original bitstream at pseudo-random intervals. The selected bytes form a new bitstream which is encrypted. Corresponding bytes from the original bitstream are then corrupted, using nearby bytes to calculate a statistically similar value. The corrupted bitstream is then freely distributed. Decryption is performed by purchasing the second bitstream of un-corrupted bytes and re-inserting them into the corrupted bitstream.

Only 1% of the original bitstream need be corrupted to render the file unplayable [26]. The authors envisage the corrupted bitstream being freely available on local caches while the smaller, encrypted bitstream is delivered from a central server.

This system functions well in a 'download now and play later scenario', but will not function in a streaming video implementation. Existing streaming servers may not be able to handle the randomly corrupted source bitstream. Indexed and high-speed playback is also problematic. To insert the un-corrupted bytes back into the bitstream, the current bitstream position must be known. This position is usually not available during streaming. Indexed and high-speed playback modes ensure that the bitstream position does not change incrementally, and locating the corrupted bytes in the bitstream becomes impossible.

### *I. Multi-Layer Encryption*

Tosun and Feng [27, 28] modify the VEA cipher of Qiao and Narhstedt, breaking the 64 DCT co-efficients into three separate layers. Lowest (most significant), mid-range and highest frequency co-efficients are mapped into the Base, Middle and Enhancement Layers respectively. Each layer receives different transport characteristics - guaranteed delivery for the Base Layer, high probability of delivery of the Middle Layer, and low priority for the Enhancement Layer (due to its low information content). The three streams are recombined at the client prior to decoding and display.

Only the Base and Middle Layers are encrypted. This enables secure delivery over networks with limited capacity – lower layers can be decrypted and displayed independently of higher layers, resulting in poorer quality video rather than discontinuities in playback. Unfortunately, the Multi-Layered Cipher is not suitable for streaming video, suffering the same issues as the original VEA algorithm. Also, not all streaming server products support Layered Streaming.

### *J. Selective Macroblock Encryption*

Alattar, Al-Regib and Al-Semari [29, 30] propose a set of four ciphers which operate on the Macroblocks within the MPEG-1 video stream:

1. Encrypt I-Macroblocks and predicted Macroblock headers
2. Encrypt every nth I-Macroblock
3. Encrypt every nth I-Macroblock and predicted Macroblock headers
4. Encrypt every nth I-Macroblock and every nth predicted Macroblock header

Selected data is encrypted using DES, ensuring that the video content is secure against all but Brute Force Attack [16]. The authors recommend resetting the count for determining the nth block at the start of each slice (ensuring correct selection of Macroblocks within a slice if data is lost) as well as periodically changing the DES key to make attacking this cipher computationally infeasible.

This cipher is not suitable for streaming video. While servers could stream the encrypted bitstream in all playback modes, resynchronisation of the DES cipher in these playback modes is not possible as we cannot determine which frame is currently being played back.

### *K. AEGIS Algorithm*

Spanos and Maples [31] propose AEGIS, which encrypts the entire contents of the I-Frame and the Video Sequence Headers. They change the bitstream by including extra start and end point locational information. AEGIS is not suitable for streaming video. Existing MPEG-1 streaming servers cannot handle their non-standard bitstream format. Also, while the authors suggest the encryption of I-Frames only will secure the entire video, others [14] have shown that it is also necessary to consider encryption of P and B-Frames.

## IV. Conclusions

The protection of streaming video is an important aspect of a streaming video implementation, encryption is one means through which this could be achieved. When considering a cipher for use in streaming video, it must be compatible with existing streaming video implementations [1, 3].

This paper explores the range of existing MPEG-1 encryption techniques and examines their suitability for use with a range of existing streaming video servers. The conclusion drawn is that none of the existing approaches are suitable for use in streaming video and that a new MPEG-1 cipher that meets the requirements is required.

The requirements outlined for an MPEG-1 encryption technique by But [3] are not incompatible, however existing techniques do not meet these requirements. In future publications I intend to describe a new MPEG-1 Partial Encryption technique that does meet all of these requirements and can be used to:

- Store encrypted video on existing streaming video servers.
- Is scalable to support multiple concurrent streams.
- Stream in a variety of playback modes.
- Resynchronise decryption such that all playback modes are supported.
- Be able of executing in real-time.
- Be secure against attack.

While MPEG-1 is an older video compression technique, it shares many commonalities with MPEG-2, and thus MPEG-1 ciphers can also be applied with little modification to MPEG-2 bit streams. The MPEG-4 compression algorithm differs from that used by MPEG-1 and MPEG-2, but the conceptual cipher requirements outlined in [3] still apply and should be kept in mind when considering encryption of these bitstreams for Copyright protection purposes.

## Acknowledgments

The work in this paper has been developed as part of the primary author's PhD studies at Monash University, Melbourne, Australia.

## References

[1] But, J., *"Implementing Encrypted Streaming Video in a Distributed Server Environment"*, Submitted to IEEE Multimedia, April 2004

[2] But, J. and Egan, G., *"Designing a Scalable Video On Demand System"*, International Conference on Communications, Circuits and Systems (ICCCAS'02), pp. 559-565

[3] But, J., *"Requirements for a Generic MPEG-1 Cipher to Function in an Existing Streaming Server Emvironment"*, CAIA Technical Report 040426A, CAIA Swinburne University, Australia, April 2004, http://caia.swin.edu.au/reports/040426A/CAIA-TR-040426A.pdf

[4] Lee, J., Hwang, S. O., Jeong, S-W., Yoon, K. S., Park, C. S. and Ryou, J-C., *"A DRM Framework for Distributing Digital Contents through the Internet"*, ETRI Journal, vol. 25, no. 6, December 2003, pp 423-436

[5] Reisslein, M., Hartanto, F. and Ross, K. W., *"Interactive video streaming with proxy servers (extended version)"*, Tech. Rep., GMD FOKUS, June 1999

[6] Wu, D., Hou, Y. T., Zhu, W., Zhang, Y-Q. and Peha, J. M., *"Streaming Video over the Internet: Approaches and Directions"*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11 no. 3, 2001, pp 402-414

[7] Gemmell, J., Vin, H. M., Kandlur, D. D., Rangan, P. V. and Rowe, L. A., *"Multimedia Storage Servers: A Tutorial"*, IEEE Computer, vol. 28 no. 5, May 1995, pp 40-49

[8] Lin, C-W., Zhou, J., Youn, J. and Sun, M-T., *"MPEG Video Streaming with VCR Functionality"*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11 no. 3, 2001, pp 415-425

[9] Frimout, E. D., Biemond, J. and Lagendick, R. L., *"Extraction of a dedicated fast playback MPEG bit stream"*, Proceedings of the SPIE, vol. 2501, 1995, pp 76-87

[10] Anderson, D. B., *"A Proposed Method for Creating VCR Functions using MPEG Streams"*, IEEE 12th International Conference on Data Engineering, 1996, pp. 380-382

[11] Kent, S. and Atkinson, R., *"Security Architecture for the Internet Protocol"*, IETF RFC 2401, http://www.ietf.org/rfc/rfc2401.txt, November 1998

[12] Kent, S. and Atkinson, R., *"IP Encapsulating Security Payload (ESP)"*, IETF RFC 2406, http://www.ietf.org/rfc/rfc2406.txt, November 1998

[13] Bozoki, E., *"IP Security Protocols"*, Dr. Dobb's Journal, December, 1999, pp. 42-55.

[14] Qiao, L. and Nahrstedt, K., *"Comparison of MPEG Encryption Algorithms"*, Computers and Graphics, Vol. 22, 1998, pp. 437-448.

[15] Dierks, T. and Allen, C., *"The TLS Protocol, Version 1.0"*, IETF RFC 2246, http://www.ietf.org/rfc/rfc2246.txt, January 1999

[16] Schneier, B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons ISBN 0-471-11709-9.

[17] Meyer, J. and Gadegast, F., *"Security Mechanisms for Multimedia with Examp[16]17le MPEG-1 Video"*, Tech. Uni. of Berlin, 1995

[18] Tang, L., *"Methods for Encrypting and Decrypting MPEG Video Data Efficiently"*, ACM International Multimedia Conference 96, November 1996, pp. 219-222

[19] Qiao, L., Nahrstedt, K. and Tam, M.-C., *"Is MPEG Encryption by using Random List instead of Zig-Zag order secure?"* IEEE International Symposium on Consumer Electronics, 1997

[20] Qiao, L. and Nahrstedt, K., *"A New Algorithm for MPEG Video Encryption"*, 1st International Conference on Imaging Science, Systems and Technology (CISST97), 1997, pp. 21-29

[21] Shi, C. and Bhargava, B., *"Light-weight MPEG Video Encryption Algorithm"*, Multimedia98, 1998, pp. 55-61

[22] Shi, C. and Bhargava, B., *"An Efficient MPEG Video Encryption Algorithm"*, 17th IEEE Symposium on Reliable Distributed Systems, October 1998, pp. 381-386

[23] Shi, C. and Bhargava, B., *"A Fast MPEG Video Encryption Algorithm"*, ACM Multimedia '98, 1998, pp. 81-88

[24] Zeng, W. and Lei, S., *"Efficient Frequency Domain Video Scrambling for Content Access Control"*, ACM Multimedia99, 1999, pp. 285-294

[25] Zeng, W., Wen, J. and Severa, M., *"Fast Self-Synchronous Content Scrambling by Spatially Shuffling Codewords of Compressed Bitstreams"*, IEEE International Conference on Image Processing, 2002, pp. 169-172

[26] Griwodz, C., Merkel, O., Dittmann, J. and Steinmetz, R., *"Protecting VoD the Easier Way"*, ACM Multimedia98, 1998, pp. 21-28

[27] Tosun, A. S. and Feng, W.-C., *"Efficient Multi-layer Coding and Encryption of MPEG Video Streams"*, IEEE International Computing Expo, 2000, pp. 119-122

[28] Tosun, A. S. and Feng, W.-C., *"A Light-weight Mechanism for Securing Multi-Layer Video Streams"*, IEEE International Conference on Information Technology: Coding and Computing, 2001, pp. 157-161

[29] Alattar, A. M. and Al-Regib, G. I., *"Evaluation of Selective Encryption Techniques for Secure Transmission of MPEG Video Bit-Streams"*, IEEE Symposium on Circuits and Systems, 1999, pp. 340-343

[30] Alattar, A. M., Al-Regib, G. I. and Al-Semari, S. A., *"Improved Selective Encryption Techniques for Secure Transmission of MPEG Video Bit-Streams"*, International Conference on Image Processing, 1999, pp. 256-260.

[31] Spanos, G. A. and Maples, T. B., *"Security for Real-Time MPEG Compressed Video in Distributed Multimedia Applications"*, IEEE 15th Annual International Conference on Computers and Communications, 1996, pp. 72-78