Limitations of Existing MPEG-1 Ciphers for Streaming Video

Jason But
Centre for Advanced Internet Architectures. Technical Report 040429A
Swinburne University of Technology
Melbourne, Australia
jbut@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. In this paper I consider the suitability of these ciphers in a streaming video context. The conclusion is that none of the existing ciphers are 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. In a previous paper I considered the requirements of a cipher to function in a distributed streaming server environment.

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].

In this paper I examine a number of techniques that could be used to protect streaming MPEG-1 video and apply them to the required criteria. The meeting of these criteria by an existing cipher allows the use of that cipher in an existing streaming server environment.

This paper shows that none of the existing ciphers and encryption techniques meet all of the criteria of a cipher for use in streaming video. As such, the conclusion is that another cipher that does meet this criteria needs to be developed.

II. Cipher Requirements

Ideally, the cipher should function separately from the streaming server implementation [1, 3]. This would provide the following benefits:

- The server platform can be chosen based on cost, features provided and implementation rather than on the security features. This allows competition between server manufacturers.
- We are not tied to one server implementation, further encouraging competition.
- Security is not implemented by streaming server developers but rather programmers who understand encryption.
- Future streaming video products should be supported.
- Existing streaming server products can be utilised without making any changes.
- Encrypted video can be cached and ultimately delivered by untrusted parties as long as the key exchange is secure.

For a cipher to function within all streaming server environments, it must possess the following properties:

- Successful installation onto a range of existing streaming video servers. The encrypted bitstream must appear to be a valid MPEG-1 bitstream to the implementation of the streaming server [3].
- The encrypted bitstream must be able to be streamed from the server in all playback modes (eg. pause, indexed, high-speed playback) provided in a streaming video environment [3].
- Resynchronise such that decryption and playback is successful in all supported playback modes [3].
- Is secure against attack [3].
- The solution must be scalable and must not limit the maximum number of concurrent streams that a server can deliver.

The key points are twofold. First, that the encrypted bitstream should appear as a valid MPEG-1 bitstream such that existing streaming server implementations will accept and deliver the content in all supported playback modes. Second, that the received bitstream be able to be correctly decrypted prior to playback, regardless of the playback mode in which the data was delivered [1, 3].

III. Existing MPEG-1 Ciphers

A range of different existing approaches are considered and found to be not 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. Partial Encryption involves the protection of segments of the original bitstream while other portions are left as plaintext.

Cipher	Encrypted bitstream can be stored on Server	Scalable to support multiple concurrent streams	No changes required to Server implementation	Encrypted bitstream can be served in Indexed Playback modes	Encrypted bitstream 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	✔	✘

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

The deficiencies of these ciphers are summarised in Table 1. The obvious conclusion being that a suitable cipher for encryption of streaming MPEG-1 video does not exist.

A. *Network and Transport Layer Encryption*

One approach uses existing network protocols such as IPSec or Secure Sockets Layer (SSL) to encrypt the video stream. This minimises any system design requirements as encryption is handled separately to streaming server infrastructure.

IPSec is a Network Layer protocol [4, 5] that provides secure communications between any two IP enabled stations, typically used to provide Virtual Private Networks over a public network infrastructure.

In a streaming video, 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. IPSec is an extension of the IP Protocol and IPSec datagrams can be routed by any IP Router on the network.

While IPSec is compatibile with all existing IP based software [4, 6], 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 [7].

Also, the original video must be stored as plaintext on the server as the server does not know about the encryption. This makes the server a larger target for attack to steal the content. The same problem exists at the client where decrypted packets are easily captured from the IP Stack.

This is typically not an issue since IPSec was 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).

SSL is a Transport Layer protocol [8] that provides secure communications between two applications running on IP enabled stations. As a TCP Protocol extension, SSL datagrams 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 also unsuitable for streaming video. Like IPSec, SSL suffers the same scalability and security problems. Further, SSL is usually implemented as part of the application, requiring existing streaming video applications to be modified. Also, SSL provides secure TCP-like communications, while many streaming servers use UDP – or related protocols like RTP. This requires the further modification of existing systems to support TCP based streaming [6, 8].

B. *Full Encryption*

Another approach is to encrypt the entire bitstream [7]. This can be performed in two ways. The first is similar in scope to IPSec and SSL based systems. Video is stored as plaintext on the streaming server, which then encrypts the entire bitstream during delivery. 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 [7, 9].

The second approach encrypts all video prior to installation on streaming servers. This minimises both processing requirements on the server and guards 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, these servers could not provide different playback modes unless further modifications were made to store information about specific index points within the bitstream. This is required as some decoding of the bitstream must be performed by the server in order to provide this functionality [10-12].

C. *SECMPEG*

The SECMPEG cipher was designed by Meyer and Gadegast [13]. The cipher 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 then modified to include extra information to allow correct decryption at a later stage.

SECMPEG is not suitable for streaming video. For all four algorithms, the changes made to the headers [13] make them non-compliant to the MPEG-1 bitstream format and as such they cannot be streamed by existing Streaming Server products.

Even if the encrypted bitstream could be delivered by existing streaming server products, there is also the inability 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. This further precludes the implementation of indexed or high-speed playback modes [10].

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 [14] proposed a cipher which uses 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.

Some modifications are suggested by the authors, one involves the pseudo-random selection of one of two permutation lists via a cryptographically secure random bit generator, the other groups blocks of 8 DC co-efficients and applies the DES cipher [14].

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

This cipher is not secure against a known plaintext attack. By comparing the decoded DCT co-efficients, they can be re-ordered and the permutation list retrieved. The list can then be applied to the remainder of the sequence to retrieve all data. The same procedure is also valid 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 [7, 15].

Tang [14] proposes a decoder with a built-in cipher module. This allows 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 and requires continuous maintenance of the software base to implement both the decoder and the cipher together.

E. Video Encryption Algorithm

Qiao and Nahrstedt [16] 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:

- First sub-dividing data into blocks of an even number of bytes. Each block is randomly divided 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 is modified by VEA, replaced with a new header block that contains information about the number and length of Slices encoded within the Picture. This procedure shortens rather than lengthens the bitstream [16].

VEA is secure, a statistical analysis shows that the second list can be considered as a unique one-time pad that is used to encrypt the first list. The ciphertext data 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 [16].

VEA is not suitable for streaming video, the format of the Picture Layer has been modified. High-speed playback modes cannot be implemented since individual I-Frames cannot be extracted from the bitstream.

Real-time decryption is also a problem. As the bitstream will be longer in length following decryption. Extra processor and memory requirements during decryption complicate software implementation.

F. Video Encryption Algorithm – Number 2

Shi and Bhargava [17] propose a different algorithm, also called Video Encryption Algorithm (VEA). In its initial incarnation [17], the cipher encrypts 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 [18, 19] to also encrypt the sign bits of motion vectors.

Since the encrypted bitstream does not modify the contents of the MPEG-1 headers, the encrypted bitstream 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. Regardless of the cryptographic strength of the random bit generator, the cipher is not secure and is susceptible to a known plaintext attack. The attack extracts the corresponding sign bits from the encrypted and plaintext streams, and determines the pseudo-random bit sequence. Since the same sequence is reused for each GOP, it can be used to decrypt the entire bitstream.

As for the Zig-Zag Permutation Cipher [14], it is imperative that the decryption of a VEA encrypted bitstream is performed within the MPEG-1 decoder. CPU utilisation is efficient as decryption involves only a simple XOR for each coefficient. This precludes using a third-party MPEG-1 decoder.

G. Frequency Domain Scrambling Algorithm

The Frequency Domain Scrambling Cipher proposed by Zeng and Lei [20, 21] operates on information encoded within the Macroblock layer, in particular the DCT co-efficients of the Macroblocks. This cipher is similar to the VEA cipher proposed by Shi and Bhargava [19], where sign bits of co-efficients are encrypted. The cipher is strengthened by also considering the following measures:

- **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 [20, 21]. Also, as modifications to the bitstream are performed on Macroblock data, header information used by streaming servers to provide indexed and high-speed streaming remains present and therefore the encrypted bitstream can be streamed.

The Frequency Domain Scrambling Cipher is more reliant on being implemented as part of the decoder than other ciphers [20, 21]. The cipher complexity means that CPU efficiency is only obtained if decryption occurs within the decoding cycle. This precludes the use of third-party MPEG-1 decoders.

H. A Unique Cipher

Griwodz et al [22] propose a unique algorithm with regard to protection of distributed video. A Poisson process is used to select bytes from the original plaintext stream at pseudo-random intervals. These bytes are extracted to form a new bitstream which is then encrypted. The corresponding bytes from the original bitstream are then corrupted, using nearby bytes from the bitstream to calculate a statistically similar value. This corrupted bitstream can then be freely distributed. Decryption is performed through the purchase of the new bitstream containing the un-corrupted bytes, which are then inserted back into the corrupted bitstream.

Experimentation by the authors show that only 1% of the original bitstream need be corrupted to render the file unplayable. They 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. There is no telling which bytes will be corrupted and there is the potential that the corrupted bitstream cannot be successfully installed or streamed from existing streaming servers.

There is also the issue of indexed and high-speed playback. To insert the un-corrupted bytes back into the bitstream, the current bitstream position must be known. This position, while readily available when decoding from disk, is usually not available during streaming. Indexed and high-speed playback modes ensure that the bitstream position does not change incrementally and thus makes it a unknown value. Locating the corrupted bytes in the bitstream, and therefore decryption and playback, becomes impossible.

I. Multi-Layer Encryption

Tosun and Feng [23, 24] propose a modification on the VEA cipher developed by Qiao and Narhstedt. This cipher looks at the 64 DCT co-efficients and breaks them into three separate layers. The Base Layer consists of the lowest frequency (most significant) co-efficients, the Middle Layer consists of the mid-range frequency co-efficients, and the Enhancement Layer consists of the highest frequency co-efficients.

The cipher assumes separate transmission of each layer using different transport characteristics, with guaranteed delivery of the Base Layer, high probability of delivery of the Middle Layer, while the Enhancement Layer gets the lowest priority. The three streams are recombined at the client prior to decoding and display.

The VEA Cipher is applied to the Base and Middle Layers only, the Enhancement Layer (containing minimal information on the actual content) is delivered as plaintext. The concept enables secure delivery of content over a network that potentially cannot cope with the required throughput – lower layers can be decrypted and displayed independently of higher layers, resulting in poorer quality video rather than discontinuities in playback.

The Multi-Layered Cipher is also not suitable for streaming video. It suffers from the same issues as the original VEA algorithm. Also, not all streaming server products offer Layered Streaming and those that do will not necessarily use the same approach to do so.

J. Selective Macroblock Encryption

Alattar, Al-Regib and Al-Semari [25, 26] propose a set of four ciphers which operate on the Macroblocks within the MPEG-1 video stream:

1. Encrypt I-Macroblocks and headers for predicted Macroblocks
2. Encrypt every n^{th} I-Macroblock
3. Encrypt every n^{th} I-Macroblock and headers for predicted Macroblocks
4. Encrypt every n^{th} I-Macroblock and every n^{th} header for predicted Macroblocks

For each method, selected data is encrypted using DES. The authors recommend resetting the count for determining the n^{th} block at the start of each slice as well as periodically changing the DES key. The first option ensures correct selection of Macroblocks within a slice, important if data is lost. A dropped Macroblock results in incorrect decryption until the count is reset. The second recommendation makes attacking the cipher more complex as the DES key is constantly changed.

The authors show that the encrypted video content is not viewable. Given that the DES Cipher is provably secure against all but a Brute Force Attack [9], coupled with regular changing of the DES key, makes attacking this cipher computationally infeasible.

The Selective Macroblock Cipher is not suitable for streaming video. While servers could stream the encrypted bitstream in indexed and high-speed 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 [27] propose an algorithm in which the entire contents of the I-Frame and the Video Sequence Headers are encrypted. This algorithm employs major changes to the format of the bitstream as extra information is inserted to locate start and end points. The resultant bitstream cannot be streamed from existing Streaming Server products due to the non-conformance to the MPEG-1 bitstream format. Also, while the authors suggest the encryption of I-Frames only will secure the entire video, others [7] have shown that it is also necessary to consider encryption of P and B-Frames. AEGIS is not suitable for streaming video – it is not compatible with existing streaming server products and does not totally protect the encoded content.

IV. Conclusions

The protection is streaming video is an important aspect of a streaming video implementation. Encryption is one means through which this could be achieved, another is watermarking of content. When considering encryption of streaming video, it is important that it 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 outlined in [3] is required.

Acknowledgments

The work in this paper has been developed as part of my 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] IETF, *"Security Architecture for IP"*, http://www.ietf.org/internet-drafts/draft-ietf-ipsecrfc2401bis- 01.txt, 1998

[5] IETF, *"IP Encapsulating Security Payload (ESP)"*, http://www.ietf.org/internet-drafts/draft-ietfipsec- esp-v3-06.txt, 1998

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

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

[8] Freier, A. O., Karlton, P., and Kocher, P. C., *"The SSL Protocol, Version 3.0 - Internet Draft"*, http://wp.netscape.com/eng/ssl3/ssl-toc.html, 1996

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

[10] 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.

[11] Gemmell, J., Vin, H. M., Kandlur, D. D., Rangan, P. V. and Rowe, L. A., *"Multimedia Storage Servers: A Tutorial"*, Computer, May 1995, pp. 40-49.

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

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

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

[15] 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

[16] 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

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

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

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

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

[21] 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

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

[23] 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

[24] 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

[25] 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

[26] 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.

[27] 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