Requirements for a Generic MPEG-1 Cipher to Function in an Existing Streaming Server Environment

Jason But
Centre for Advanced Internet Architectures. Technical Report 040426A
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 solution. Copyright protection should be implemented using both passive schemes – such as watermarking – and active schemes – such as encryption. While some algorithms for encryption of video have been proposed, they are in the main unsuitable for implementation in a streaming video environment. In this paper I describe a set of requirements that an MPEG-1 cipher must possess in order to function in a distributed streaming server environment using existing streaming server products and technologies.**

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

I. Introduction

Streaming of high-quality video over the Internet has not progressed far beyond the trial stages. While there are many technical limitations on a working implementation, there are also non-technical reasons why a commercial video streaming application will not be successful.

Copyright protection is one such issue that must be addressed. Copyright owners will not entrust their content to a video streaming implementation that cannot protect the content that it delivers. Online video delivery systems will live or die by their patronage, and consumers will only use a system if it offers content that they find "interesting". The more "interesting" the content, the more likely the consumer will pay to access it [1, 2].

Commercial reality means that the Copyright on "interesting" content is more valuable and its protection more important. This allows us to make the following statement.

> *Consumers will not use a video streaming service unless "interesting" contents is made available, but content owners will not make this material available unless their Copyright is protected.*

Copyright protection of online content is a complex issue and involves not only the protection of content against theft, but also the guarantee of payment and secure money transfers. The scope of this paper does not extend to secure monetary exchange with regard to purchase of viewing rights on content [1-3].

Protection of content against theft can be accomplished using two techniques. The first, Watermarking, is a passive form of protection, stolen watermarked content can still be freely viewed. However, once theft is detected, the stolen content can be checked to determine its original source and to aid in prosecution [4, 5].

The second technique is active protection through the form of encryption. In this case the content is modified such that it cannot be viewed unless the key to decryption is known. A complete system would involve secure delivery of the key upon completion of payment for access [1, 2].

This paper is concerned with the requirements for an MPEG-1 cipher that will allow the encrypted stream to be installed on and delivered from existing streaming server implementation.

II. Why be Compatible with Existing Streaming Servers?

An option that is available is for a streaming server developer to implement a cipher that is unique to their product. Indeed, some may see this as a means to differentiate their product from the competition. However there are a number of reasons why this is not a good idea.

A. Not Encryption Experts

Implementing a streaming server is a complex procedure. It involves the complex management of available disk bandwidth, memory bandwidth and network bandwidth in order to maximise the number of concurrent streams that can be delivered. Add the complexity of supporting indexed and high-speed playback modes and the development of a streaming video server is not an easy task [6-8].

Streaming server developers must be good software engineers with a great deal of experience in managing limited resources and concurrent programming techniques. They do not, however, need to be experts in development and implementation of ciphers.

Furthermore, should the cipher be implemented by the streaming server developers, they will also be

responsible for maintaining the software, as well as developing and maintaining an encrypted video playback application for clients to use.

B. Security and Scalability

If server developers use a cipher which is unique to their server, it is likely to be implemented such that encryption occurs in real-time as the content is being delivered to the consumer over the network. This approach has flaws in both security and scalability.

If video is encrypted in real-time, it will be stored on the server in plaintext form. This will make the server itself a target for attack. Any attack which renders the server open to an outside interest will mean that all content on that server is vulnerable. The more content stored on the server, the more tempting it is as a target.

Further, real-time encryption of streaming video is CPU intensive. Encryption of a small number of streams is not an issue, but CPU and memory requirements increase with the number of concurrent streams. This requires not only further resource management in a scenario where resources are already scarce, but places further limitations on the maximum number of concurrent streams that can be supported by a streaming server.

C. Minimising Competition

It may be considered that allowing streaming server developers to implement their own encryption solution will increase competition between suppliers. However, this will actually serve to minimise competition. Competition would exist only while a service provider is yet to decide which server product to use, once the decision is made, the server developer would have a monopoly with that service provider.

If the developer stopped upgrading their systems, the service provider would either have to stay with old technology or move to a different platform. Implementations that are standardised across platforms ensure that a service provider will always find competition amongst developers.

The standards should not be limited in how data is actually transmitted over the network (standardising playback applications to function with all servers), but also on the cipher used to encrypt the data. A standard cipher, where the encrypted bitstream can be installed and streamed from different server platforms, will maximise the competition between server developers.

III. Distributed Streaming Servers

A distributed server arrangement minimises the network resources required to stream video, offering a greater number of concurrent streams to a greater number of consumers. In a distributed server arrangement, video content would be cached at a local streaming server where it could be delivered to a number of local viewers [8-11].

With competition between server developers, it is possible (and probably desirable) that each streaming server in the server network is running a different platform. This adds credence to the argument that an MPEG-1 cipher for streaming video should be compatible with a range of streaming servers [2, 10, 11].

Ensuring that the video is encrypted prior to being installed on the distributed server becomes more important if we consider a multi-party distributed server scenario, where the streaming server network is owned and operated by a number of companies working together to provide a common streaming video service to a large area. In this environment, there is more delivery of content to untrusted persons and a greater potential for theft [2, 11].

Copyright protection in a distributed server environment would function as shown in Figure 1. The encrypted (and watermarked) content is made freely available by the content owner. Content is delivered over the network and cached at a local streaming server to the eventual client. The consumer requests access to the content and purchases the decryption key from the content owner. The consumer can then begin to stream the encrypted video from the local server and decrypt it using their purchased key.

IV. Implementation of Playback Modes on Streaming Servers

Before we can begin to design a cipher that functions correctly with a range of existing streaming server products, it is necessary to understand how these existing products function.

A streaming server offers a more complex application space than a basic data server such as a World Wide Web (WWW) or file server. This is due to a client being able to interact with the video server to control aspects of the media stream being requested [7].

Controls available to the client often emulate that available to a user directly operating playback of a video stored on a DVD, whereby they can pause and recommence playback, jump to random points within the video (index) and perform playback at both high and low speeds [12].

Most streaming video servers offer the capability of pausing and recommencing playback, as well as the option of indexing to a random point within the video stream. On the other hand, high-speed playback functionality is only provided by some streaming video server implementations, while slow-speed playback is generally not offered at all.

Implementation of different playback modes increase the complexity of the streaming server. While the CPU processing load of streaming video is not high, the complexity arises from managing the available disk, memory and network bandwidths. Streaming video at a constant rate (standard, paused and indexed playback) is easier than co-ordinating system management when streaming must also occur at different rates (during different speed playback modes) [7].

One common factor is that if indexed playback is implemented – as is the case in most server products – then the server must decode the installed video bitstream

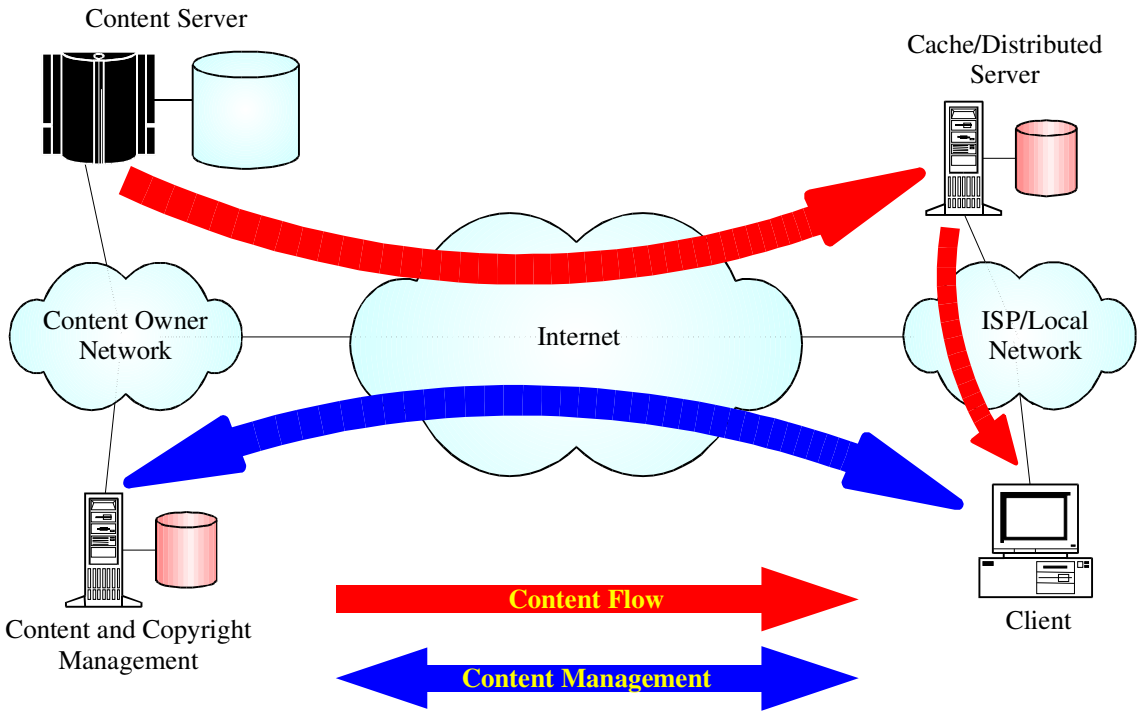

Figure 1: Content Protection in a Distributed Server Environment

to some degree in order to extract timestamps to implement this functionality.

Also common to streaming video is that the bitstream received at the client must be passed through a standard MPEG-1 decoder for playback. This means that if the server streams the data in a proprietary format, the original MPEG-1 bitstream must be reconstructed at the client prior to decoding and playback [9].

A. Constant BitRate Streaming

There are many different approaches to streaming video over an IP network, in general, it is only necessary that the MPEG-1 bitstream is somehow delivered to the end client, which then reconstructs the bitstream before passing it to an MPEG-1 decoder for playback [6].

MPEG-1 video is encoded at a variable bitrate, some frames are encoded using more bits that others, meaning that some segments of video are of unequal size than other segments of the same length [13, 14]. Streaming of stored video is rarely performed at a variable rate. Variable rate streaming is suitable for live video feeds – particular live interactive feeds such as video conferencing. In this instance, streaming data over the network as it is encoded minimises the transmission delay from source to destination and provides a more responsive interactive experience [9].

When streaming from a stored source, data is generally transmitted at a constant bit rate being equal to the average bitrate of the encoded bitstream (eg. Mediabase, Apple Quicktime, Microsoft NetShow Theatre). The data is then provided to the decoder at the required variable bitrate from a buffer at the client end, see Figure 2 (note that the graphs are for illustrative purposes only). This introduces a greater delay from the moment data leaves the server to when it is decoded.

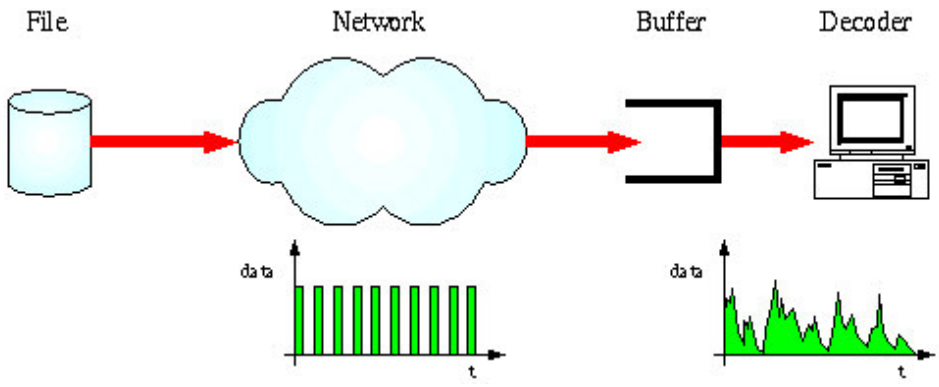

Figure 2: Constant Bitrate MPEG Streaming

Actual transmission time is not as important as the variation in transmission time, or jitter [9].

The buffer at the client is used to minimise the effects of jitter, as long as enough data is stored in the buffer, playback can continue even if some data arrives late at the client. Playback only suffers if the data arrives so late that all data in the buffer has been decoded and the buffer is empty. The level of acceptable network jitter can be increased by increasing the size of the buffer with the cost being an increased total transmission time to the decoder and subsequent degraded response time to interactive commands [9].

B. Standard Playback

Video streaming includes the reassembly of data at the client into a valid MPEG-1 bitstream for decoding and playback purposes. If we consider the MPEG-1 bitstream format, or more formally the MPEG-1 System Stream, in more detail (see Figure 3), we can see that the System Stream format allows encapsulation of one or more substreams, known as the MPEG-1 Video Stream and MPEG-1 Audio Stream [13, 14].

The MPEG-1 Video Stream defines a video sequence while the MPEG-1 Audio Stream defines audio data to be played back. A single System Stream can contain multiple (up to 16) different Video Streams and (up to 32) Audio Streams. Obviously, only one Video and one Audio Stream is selected for playback at any one time, the other streams encoded within the System Stream would be ignored [13, 14].

Streaming video can be performed in a number of ways:

- We can stream the System Stream as is, when the data is received at the client it is passed directly to an MPEG-1 decoder for playback [9]. As per the SGI Mediabase and Microsoft NetShow Theatre streaming servers.
- The server can select the encoded Video and Audio Stream from the System Stream and stream these two bitstreams separately. At the client, a new System Stream is constructed containing these two sub-streams and passed to the MPEG-1 decoder for playback [9].
- The server selects the encoded Video and Audio Streams from the System Stream which are then streamed separately. At the client, the Video Stream is passed directly to an MPEG-1 Video decoder while the Audio Stream is passed directly to an MPEG-1 Audio decoder for playback [9]. As per the Apple Quicktime Streaming Server.

Each of these approaches is a viable one for a streaming server to utilise. The first approach simplifies server implementation as less processing is performed on the stored bitstream, however this load can be minimised by extracting the sub-streams prior to installation on the server. This approach also minimises complexity in timing and delivering data packets onto the network. The primary cost is when the original bitstream actually does consist of multiple Video and Audio Streams, in this case more network bandwidth than necessary is used in transmitting data that will not be used by the client [9].

The second approach can potentially save network bandwidth by only transmitting the Video and Audio Streams that the user selects. Some effort is expended by the client in reconstructing a valid System Stream to pass to a decoder.

The final approach distributes the effort of decoding. The decoding and playback of the System Stream is performed by the server, extracting the selected Video and Audio Streams and delivering these over the

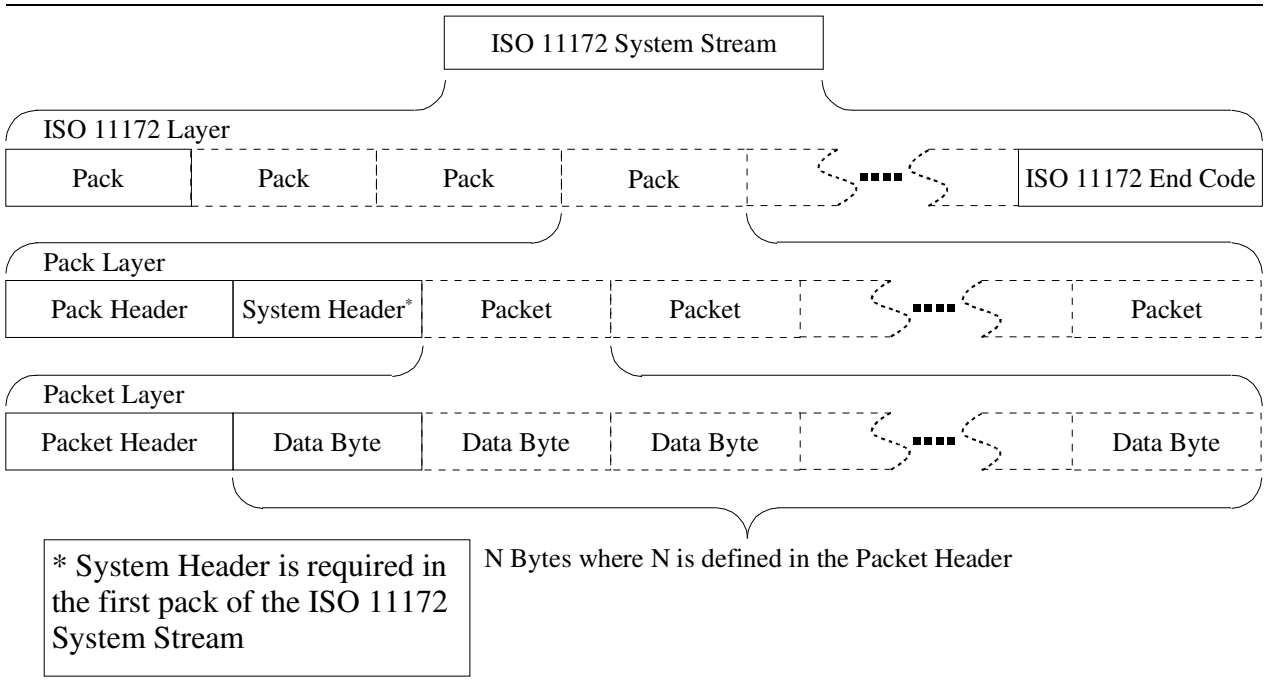

Figure 3: MPEG-1 System Stream Definition

network. Decoding is completed at the client by separate MPEG-1 Video and Audio decoders. Load at the server can be minimised by decoding the System Stream and extracting the Video and Audio Streams during installation of the content.

The exact size of individual IP datagrams and the time intervals at which these packets are placed on the network vary from implementation to implementation.

C. Pause

Pausing playback of a video stream is usually a simple matter of following a series of steps:

- Send a message to the server to stop streaming data, the server sends no more packets of data but remembers its position in the stored bitstream so that it can recommence streaming.
- The client decoder stops processing data from the buffer at the current frame being displayed. Any data remaining in the buffer is used once the user decides to recommence playback.
- Any packets that were in transit before the server received the message to stop streaming will arrive at the client. This data is stored in the buffer to be used once the user decides to recommence playback.

When playback resumes, the server is instructed to start sending data again, this continues from the next block of data in the bitstream from the last packet sent. When this arrives at the client it can be stored in the buffer immediately after the last packet that arrived when playback was initially paused. The client decoder can also continue processing data from the buffer [9].

D. Indexed Playback

Indexed playback is a feature provided on nearly all streaming video servers. It allows the user to randomly select the current playback position within the installed bitstream. This allows quick navigation through the content, especially if the timestamp of required scene is known.

Implementation of Indexed playback when playing back an asset stored locally on disk is basic. The approximate position in the bitstream is calculated based on both the timestamp and the average encoded bitrate. We then seek to this position in the file and start processing the data, looking for some System Stream information indicating that we may recommence playback [15].

Indexed playback of streamed video over a network is slightly more complex. The process of seeking for a suitable starting point is typically performed by the server, which then begins streaming only the data to be decoded to the client [12, 16].

Client playback applications typically reset their decoder during an indexed jump, treating the restart in data flow as a new bitstream to be decoded and displayed. As such, the newly received bitstream must conform to the MPEG-1 bitstream standards.

This task cannot be performed by simply streaming data over the network from a different point in the file

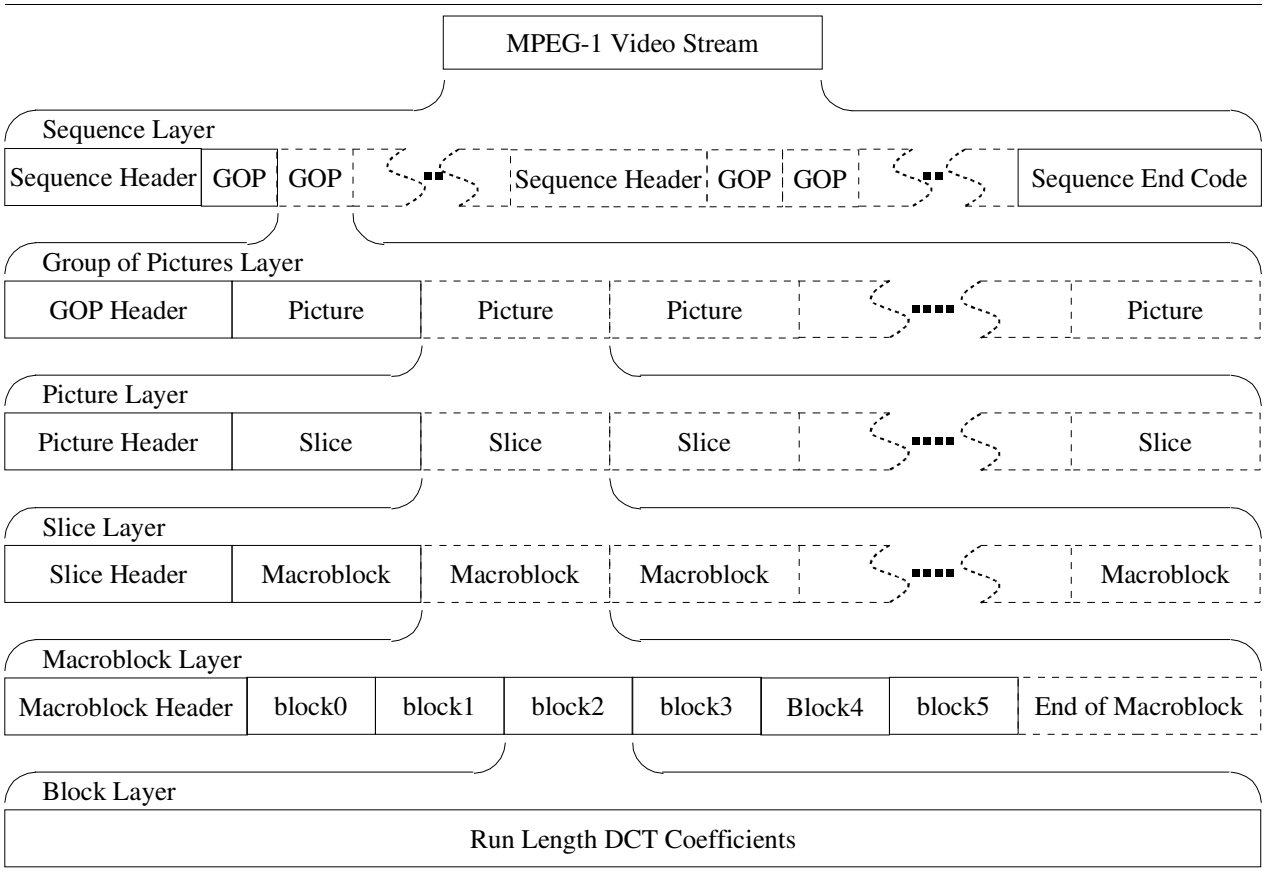

Figure 4: MPEG-1 Video Stream Definition

stored at the server. Instead, the server must execute some decoding of the MPEG-1 bitstream. The purpose of this decoding is to locate a valid starting point from which decoding can start [12, 16].

The definition of the MPEG-1 Video Stream, see Figure 4, shows that decoding can commence from the start of any GOP in the bitstream (the Sequence Header from the start of the original bitstream can be repeated) [13, 14]. This necessarily means decoding the bitstream to locate timestamp information as well as the location of the start of the nearest GOP to the selected indexed timestamp.

Once this data stream start point has been determined, the server can recommence standard streaming from this point in the stored bitstream. In order to provide indexed playback functionality, the streaming server must be able to locate individual GOPs and their timestamps within the stored bitstream.

Some servers, such as Apple QuickTime require that the indexing information be pre-processed through a process Apple calls "hinting". Others, such as Mediabase and NetShow Theatre will perform this task in real time.

E. High-Speed Playback

High-speed playback modes, in both forward and reverse directions, is a feature that is not provided by all streaming server implementations (eg. The Apple Quicktime Server does not support the high-speed playback modes). As a feature, it is complex to implement, and can consume server resources. The solution is not to stream the original bitstream at a faster rate as this consumes disk, memory and network bandwidth at an increased rate, to provide a series of images that are only viewed at high-speed with a view to locate a particular point in the media [17, 18].

Instead, implementation of high-speed playback is performed in a way to conserve resource usage. The typical approach used is to extract particular frames from the original bitstream and transmit only these frames over the network for the client to decode. In this way, a smaller portion of the bitstream is transmitted, decreasing resource usage and not impacting on the number of supported concurrent streams.

Within the video stream, see Figure 4, a GOP encodes a series of frames, the frames themselves are encoded at the Picture Layer, where each Picture makes up a single frame in the video sequence. These frames cannot be extracted at random, in order to achieve the compression ratio that it does, video frames are encoded as being derived from both previous and future frames in the sequence [14].

The video image itself is made up of a number of frame types, I-Frames (which can be decoded and displayed independently of other frames), P-Frames (which require decoding a previous I or P-Frame prior to decoding) and B-Frames (which require decoding a previous I or P-Frame as well as a following I or P-Frame prior to decoding) [14].

Of these frame types, only I-Frames can be independently extracted from the original bitstream and be decoded for display. The reason that I-Frames are used less often is that they consume the largest number of bits in the bitstream.

A typical high-speed playback mode will be implemented by extracting a sequence of I-Frames from the original bitstream and delivering these frames only to the client. These frames can be decoded and displayed in any order, allowing implementation of both forward and reverse playback modes [16, 18].

However, the MPEG-1 decoder cannot directly decode a series of I-Frames, only an bitstream conforming to the MPEG-1 Video Stream format [14]. As such, the server must reconstruct a new bitstream from the extracted I-Frames to deliver to the client for decoding and display. The procedure is shown in Figure 5.

As for indexed playback, the original Sequence Header at the beginning of the bitstream can always be repeated to construct the high-speed bitstream. A new sequence of GOPs can be constructed from the existing GOP Headers and the first Picture within the respective GOP.

The GOP Header does not indicate the number of pictures within the GOP and thus Pictures can be deleted from the data stream. The first Picture within the GOP is an I-Frame so that subsequent Pictures can be correctly decoded [13, 14].

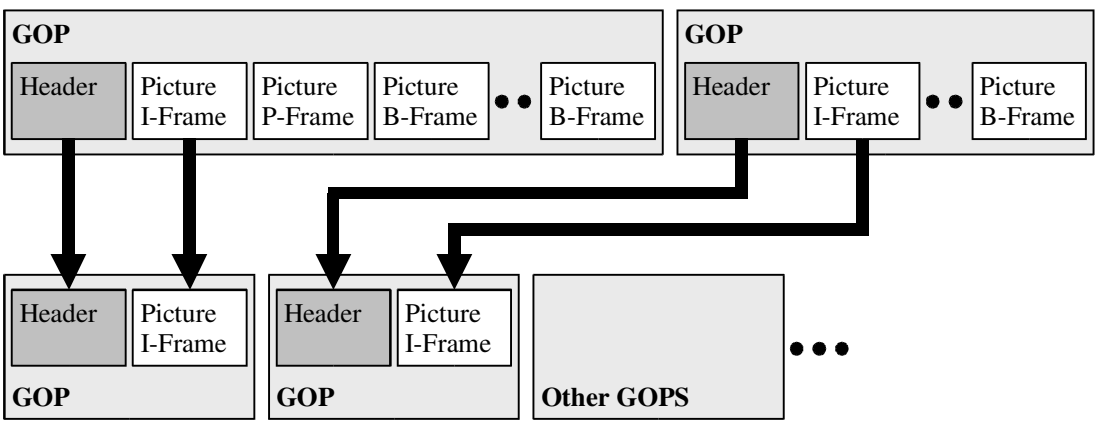

Figure 5: Generating a High-Speed Playback Bitstream

This procedure will construct a new, valid, MPEG-1 video bitstream that consists of only the first I-Frame stored in each GOP. While not enforced by the MPEG-1 standard, a GOP typically contains about 12 frames, resulting a sequence consisting of only every twelfth frame [14].

Further, a server will typically only deliver the video stream during high-speed playback rather than constructing a System Stream. This is because audio is not required during high-speed playback. A server can optimise its implementation by extracting the high-speed bitstreams upon installation of content – in this way the data is already assembled for delivery should a high-speed playback mode be selected by the client [18].

Indexed high-speed playback is also available if high-speed modes are provided. This involves indexing into the high-speed video bitstream to the specified timestamps and commencing playback from there [18].

To provide high-speed playback functionality, the streaming server must be able to locate and extract individual I-Frames and GOP Headers from the stored bitstream. These data blocks will be reassembled to form a new high-speed playback MPEG-1 video stream to be delivered in these modes. To provide indexed high-speed playback, the server must be able to locate the timestamps within this new high-speed playback bitstream.

Some servers, such as Mediabase pre-process the installed bitstream to generate the high-speed bitstreams during the installation process. In this way, the high-speed bitstream is always available. Other servers, such as NetShow Theatre perform this task in real time.

V. Cipher Requirements

The aim is to have an MPEG-1 Cipher that is compatible with a range of existing streaming video products. In order for this to occur, it must meet the following requirements:

1. **The encrypted bitstream must install onto an existing streaming video server.** Most streaming servers will perform a partial decode of the content to ensure that it conforms to the MPEG-1 bitstream format. Meeting other requirements of supporting indexed and high-speed playback modes should ensure that this check is passed as well.
2. **Servers that allow pausing during playback must continue to provide this feature.** Most servers will allow the user to pause playback of the video stream. This is often implemented by simply pausing and restarting delivery of data but can also require a re-index into the bitstream. The server must be able to relocate the current playback position.
3. **Pausing and resuming playback must be supported during decryption and playback at the client.** The decryption module must be able to cope with the data stream being delivered during a pause in playback. This is not typically an issue if pause is implemented through a basic stop and restart of data delivery. If a restart is implemented via re-indexing into the encrypted bitstream, the cipher must be able to resynchronise the decryption process such that the bitstream is correctly decrypted.
4. **Servers that provide indexed playback must be able to provide this feature with an encrypted bitstream.** Indexed playback by servers is provided by commencing playback at a given timestamp in the bitstream. To support this functionality, a server must be able to locate the same timestamps within the encrypted bitstream.
5. **Indexed playback, if provided by the server, must be supported during decryption and playback at the client.** The decryption module must be able to cope with the data stream being delivered during indexed playback. The cipher must be able to resynchronise decryption at any of the possible indexation timestamps such that the bitstream is correctly decrypted.
6. **Servers that provide high-speed playback modes must be able to provide this feature with an encrypted bitstream.** High-speed playback, if implemented, is typically provided through the extraction and delivery of separate I-Frames from the original MPEG-1 bitstream. To support this functionality, a server must be able to locate and extract the encrypted I-Frames within the encrypted bitstream.
7. **High-speed playback modes, if provided by the server, must be supported during decryption and playback at the client.** The decryption module must be able to cope with the data stream being delivered in the high-speed playback modes. The cipher must be able to resynchronise decryption at each I-Frame delivered during high-speed playback such that the bitstream is correctly decrypted.
8. **The encrypted bitstream should be secure against attack.** This requirement is obvious. The point of encrypting video is to protect it against theft. Obtaining the encrypted bitstream is assumed to be possible and therefore we must ensure that the cipher makes it unfeasible to retrieve the plaintext bitstream without purchase of the key.

VI. Digital Rights Management

Digital Rights Management (DRM) encompasses the entire Copyright protection argument [3]. MPEG-1 encryption, as discussed in this paper, forms a small piece of the puzzle that is DRM. A complete solution involves not only protection of content against theft through encryption of data, but also an addressing of the following issues:

- Watermarking (passive protection) to assist in prosecution after theft has occurred [4, 5].
- Key Management, or the distribution of the decryption key to those authorised to access the content.
- Licensing issues. Specification of digital licenses authorising the holder to certain privileges with content [3]. Examples include – Is editing allowed? Is copying allowed? Restrictions on access dates?

- Distribution issues. How are distributors of content allowed to access the content [2, 8, 11].
- Distribution of Monies. How do we ensure that payment is actually made? How do we specify how the costs of accessing digital content are spread amongst all shareholders of that content? How can this distribution be automated? [20, 21]

Encryption of digital video content fits into this scenario as a means of actually protecting the content. If we enforce the aforementioned restrictions on the cipher, it also answers a question of content distribution – content is provided in encrypted format to all distributors.

VII. Conclusion

Copyright protection of networked digital video is a complex issue that encompasses a wide area with many parts to address. This paper seeks to summarise the requirements to a possible solution to one of these parts, that of encryption of the video to safeguard it against theft.

In an ideal situation, a streaming video service would be implemented using a distributed server scenario. In such a scenario, competition is maximised if encryption of video is performed externally to the streaming video implementation – the streaming server should be able to stream content that is already encrypted and it should be able to decrypted at the consumer. In this way, content is in encrypted form at all stage while it is on the network.

By developing an MPEG-1 cipher that is compatible with a range of different streaming server products, content protection can be implemented independantly of any streaming digital video solution.

This paper describes how streaming servers transfer video data over the network to the consumer and how they provide functionality over and above simple video delivery, such as indexed and high-speed playback modes. By understanding how this functionality is provided, it places restrictions on the design of a suitable MPEG-1 cipher that can be used in existing streaming video implementations.

Apart from the obvious requirement that the encrypted bitstream should prove resistant to attack, the other requirements of an ideal MPEG-1 cipher include:

- All streaming server products should accept the encrypted bitstream for installation.
- The encrypted bitstream must be able to be streamed from the server in all playback modes supported by that server, standard, paused, indexed and high-speed.
- The cipher must be able to resynchronise itself such that the received bitstream at the client can always be correctly decrypted and decoded for playback purposes. This must be true in all supported playback modes.

Acknowledgments

This work in this paper has been developed as part of my PhD studies at Monash University, Melbourne, Australia.

Figures 3 and 4 have been drawn by the author but modified from diagrams found in Mitchell et al [14].

References

[1] Memon, N. and Wong, P. W., *"Protecting Digital Media Content"*, Communications of the ACM, vol. 41, no. 7, 1998, 35-43.

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

[3] 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

[4] Bao, F., Sun, Q., Hu, J., Deng, R. H. and Wu, J., *"Copyright protection through watermarking: towards tracing illegal users"*, The 6th IEEE International Workshop on Intelligent Signal Processing and Communications Systems (ISPACS'98), November

[5] Abdulaziz, N., *"Digital Watermarking and Data Hiding in Multimedia"*, PhD Thesis, Monash University

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

[7] Ramarao, R. and Ramamoorthy, V., *"Architectural Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem"*, IEEE International Conference on Communications

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

[9] Wu, D., Hou, Y. T., Zhu, W., Zhang, Y.-Q. and Peha, J. M., *"Streaming Video over the Internet: Approaches and Directions"*, In: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, 2001.

[10] Chan, S. and Tobagi, F., *"Caching schemes for distributed video services"*, IEEE International Conference on Communications (ICC'99), 1999.

[11] But, J. and Egan, G., *"Designing an Affordable Scalable Video On Demand System"*, 2nd ATcrc Telecommunications and Networking Conference and Workshop, pp. 16-21

[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] ISO, *"ISO/IEC 11172. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s"*, ITU, 1996

[14] Mitchell, J. L., Pennebaker, W. B., Fogg, C. E. and LeGall, D. J., MPEG Video Compression Standard, Chapman & Hall ISBN 0-412-08771-5.

[15] Anderson, M., *"VCR Quality Video at 1.5 Mbits/s"*, In: National Communication Forum

[16] Lin, C.-W., Zhou, J., Youn, J. and Sun, M.-T. (2001) "MPEG Video Streaming with VCR Functionality" IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, 2001, 415-425.

[17] Chen, H. J., Krishnamurthy, A., Little, T. D. C. and Venkatesh, D. (1995) "A Scalable Video-on-Demand Service for the Provision of VCR-like Functions", In: 2nd International Conference on Multimedia Computing and Systems, May 1995, pp. 65-72

[18] Frimout, E. D., Biemond, J. and Lagendick, R. L. (1995) "Extraction of a dedicated fast playback MPEG bit stream" Proceeding of the SPIE, Vol. 2501, 1995, 76-87.

[19] Shanableh, T. and Ghanbari, M., *"The Importance of Bi-Directionally Predicted Pictures in Video Streaming"*, In: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, 2001, 402-414.

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

[21] Aslam, T., *"Protocols for E-Commerce"*, Dr. Dobbs Journal, Vol. December, 1998, 52-58.