Presented at ACM Multimedia 1998, September 12-16, 1998, Bristol Uk

Protecting VoD the Easier Way

Carsten Griwodz, Oliver Merkel
Industrial Process and System Communications
Dept. of Electrical Eng. & Information Technology
Darmstadt University of Technology
Merckstr. 25 • D-64283 Darmstadt • Germany
Tel.: (49) 6151 / 166159
{griff,merkel}@kom.tu-darmstadt.de

Jana Dittmann, Ralf Steinmetz
GMD IPSI
German National Research Center
for Information Technology
Dolivostr. 15 • D-64293 Darmstadt • Germany
Tel.: (49) 6151 / 869845
{dittmann,steinmetz}@darmstadt.gmd.de

1. ABSTRACT

Various on-demand systems require that large numbers of customers are provided with the same multimedia stream content or different but closely related content in short temporary sequence but not at exactly the same time. This includes video on demand and news on demand. A typical approach to increase the performance of such systems is caching. However in current commercial on-demand streaming applications in the Internet caches are used very rarely because a mechanism to protect the content from resale by the cache owners does not exist. A typical solution is to transfer all content via protected unicast transmissions, which is an approach that does not scale.

We want to present a trivial scheme that provides similar protection for the content but be used efficiently with multicasting and caching. In this approach, the major part of the video is intentionally corrupted and can be distributed via multicast connections, while the part for reconstruction of the original is delivered to each receiver individually. We propose also means to discourage resale of the multimedia content by customers. One proposal introduces receiver-sided introduction of watermarks into the video, the other uses infrequent corrupt bytes to achieve uniqueness of each copy.

1.1 Keywords

Multimedia, video-on-demand, copyright protection, corruption

2. INTRODUCTION

Many on-demand applications require that the same content is delivered to many different receivers in short sequence. In both VoD and NoD applications, the goal of the content provider is the frequent and rapid sale of contents in the most popular phase of their life cycle. The restriction of access to a small group of receivers is not intended for this type of content redistribution. In video-on-demand systems, for example, advertisements and current trends raise the popularity of certain video titles, and thus, the rate of requests to the specific title. In fact, videos exhibit certain long-term aging characteristics that are verified in [5]. This could be exploited in a distribution system by the introduction of caching and prefetching techniques.

However, to prevent data theft, the delivery of commercially relevant content from the sender to the receiver is currently protected by encrypted point-to-point (unicast) transmissions. As we can see in current not-quite digital applications such as broadcast television or video rental, ensuring copyrights is not necessarily a technical issue but it is sufficient for the content provider to be able to verify and prove that the copyright has been violated after the actual violation. This is typically the case for applications that concern distribution of non-confidential content to mass markets, such as news-on-demand, audio- or video-on-demand. To support the tracking and proof of data theft, watermarking techniques are under development which identify the individual receiver of the content.

For the wide-spread sale of a multimedia content, the use of caching is necessary because neither the network infrastructure nor any content provider's servers can support an arbitrary number of unicast connections with content that is encrypted in real-time.

This conflict between copyright protection and scalability inhibits effectively the deployment of caching and prefetching in the area of commercially sold multimedia content, although the profit that could be gained from reducing the network constraints would be considerable (as demonstrated by the sex business, which is protected from redistribution on a large scale by the privacy needs of its customers).

In this paper, we do not try to provide a generic solution but we address a specific application area. For some commercial multimedia applications such as video conferencing, perfect protection is a major requirement of the communicating partners. In some multimedia applications, not the perfect protection of content is a necessity but instead, ensuring the copyright of the content owner is the primary issue.

For such multimedia applications, which can take major advantage of caching or prefetching mechanisms, we want to provide a simple approach that is able to protect the content owner from the service provider in a wide area network while protecting the infrastructure from an unnecessary number of retransmissions.

We want to provide a straightforward mechanism for these applications which can interoperate with simple caching systems in the service providers' domain as well as reasonably powerful multimedia servers of the content providers. The mechanism should be computationally cheap, in order not to overloaded the server with the task of modifying (e.g. watermarking or encryption) the content for an arbitrary number of concurrent unicast transmissions which would make the application unscalable. We propose the unencrypted transmission of the bulk of data, which allows the use of caching, and the additional encrypted point-to-point transmission of a minimal amount of data that is necessary to reconstruct the complete multimedia content. Furthermore, ideas are presented for the introduction of personal watermarking into the reconstructed video at the client side.

The digital watermarking technology enables the possibility to add copyright or customer information into digital data, like video, to protect the content from resale by an original customer. There are two major approaches: adding visible, perceptual marks or embedding invisible, imperceptible (hidden) information using steganographical schemes. Usually digital watermarks are additional hidden information, labels, holographically inlaid in the multimedia data and perceptually undetectable (invisible). Those concepts result from a direct work on the data. Ideally it is impossible to remove the embedded information (owner identification, customer information or other information about the multimedia data) without damaging the data. The embedded information is embedded with a secret key and can be retrieved using the same secret key. The technique de-motivates the illegal re-use of distributed digital data. Labelling-based protection strategies intend to enable the proof of ownership on copyrighted material, to detect the originator of illegal reproductions, to monitor the usage of the copyrighted multimedia data and to analyse the spread spectrum of the data over networks and servers. It should be noted, though, that the embedded signalling can be used for a variety of purposes other than copyright control.

We combine the theft and resale protection of data during transmission with a watermarking scheme that embeds specific customer information into the video to track the use of the reconstructed video, and to prove data theft and to de-motivate resale by regular customers.

Our ideas were primarily directed at MPEG-1 video but tests (on AVI, Quicktime and MPEG audio layer 3) show that other audio and video formats are protectable in the same way.

3. RELATED WORK

Current approaches to wide area video distribution in the Internet assume either that content is free (and probably worthless) or alternatively, that the distribution system is strictly controlled. These alternative commercial video delivery approaches use encrypted point-to-point delivery to ensure that only paying users receive the service. To ensure that copyright violations can be proved, the delivered content is watermarked with information about the content's seller as well as the customer.

The approaches to encryption that we have found in the literature are specifically implemented for MPEG-1, which was also in the focus of our tests. However, we have found only examples that can not be combined with partial or complete reuse of the video data that is stored in caches. For each transmission, encryption keys are selected and the manipulation of the video is repeated, which requires a relevant amount of resources at the sender side.

The initial approach towards video encryption was the simple DES or RSA encryption of the whole stream. Various more efficient encryption algorithms were implemented recently, typically in a way that makes an involvement of an MPEG parser necessary. Maples and Spanos present in [9] an approach of partial encryption exclusively of I-frames of MPEG movies. Tang proposes in [11] a scheme of reordered DCT coefficients, which is considered insecure as well as criticized because of the penalty to the compression ratio in later papers. Qiao et.al. propose in [10] a scheme called VEA (video encryption algorithm) that works exclusively on the data bytes and does not interpret the MPEG data. They exploit the entropy in the MPEG data stream to reduce the number of XOR operations in comparison to a full encryption by 47%. Still, each byte of the video data is manipulated once for each transmission. Kunkelmann et.al. present in [8] a variety of approaches to the partial encryption of the complete video stream, for use with a security gateway, and come to the conclusion that a mix of partial bit stream encryption and variable length code encryption is the most efficient for their application. They consider a partial encryption of 10% of the data appropriate for VoD applications, while full protection requires a major part of all data to be encrypted to prevent reconstruction.

All of these approaches are relatively computing intensive and would put a heavy strain when executed by a VoD server. Since encrypted content can not be re-used for any two receivers, the operations mentioned above have to be performed for each receiver of a stream independently. Kunkelmann et.al. report an increase of CPU utilization by 10.5% for the playback of the video stream when decryption is necessary. Obviously, another drawback of these approaches is the effect that optimizations that have been investigated in video server work such as batching ([1]) are reduced to schemes for unloading the servers'

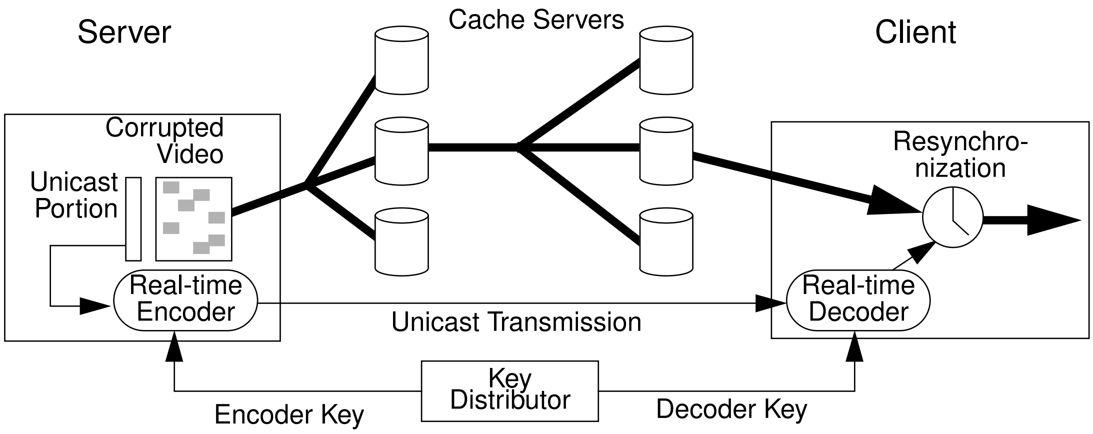

Figure 1. Distribution System

disks while memory, CPU and networking requirements grow linearly with the number of concurrent requests..

We favour an approach that is addressed by researchers in restoration and reconstruction of images: we don't encrypt publicly accessible data. Instead, the unavailable information is not present in the publicly accessible data at all.

4. PROTECTION FROM DATA THEFT

One of the basic assumptions of our paper is that we have to consider the computing speed of video servers as well as their capacity. Typical design considerations in the slow-starting video server business are concerned with the production of machines with high internal I/O throughput, disk throughput and network throughput because these are supposed to limit the performance of current video server applications. The supported number of transactions and the computing power, on the other hand, are considered sufficient for the envisioned scenarios. Referring to other video encryption work, we understand that the assumption of sufficient server computing power does not hold when the server is supposed to re-encrypt the content for each customer of the service in real-time. However, this kind of protection of the content owners' copyright and the maintenance of information about the final receiver of the information is considered necessary.

4.1 General Approach

Our basic idea is to unload the servers and networks from the necessity of encrypting and unicasting complete videos by the use of caching and pre-distribution in wide-area networks with an acceptable compromise to protection. In our approach, reconstruction can not break an encryption algorithm to decode the missing information. Instead, we destroy the data entirely in the freely distributed part of the multimedia content. Figure 1 shows a sketch of the distribution system that we envision for our approach.

The video transmission is performed in two phases. The bigger part of the video is corrupted and it is made (from the content provider's point of view) publically available in cache servers ("corrupted video") in the first phase. In the second phase, a point-to-point transmission is used to deliver the missing bytes to the customer, encrypting this missing information when a video is actually requested ("unicast portion"). This provides the content owner with the information that a request has taken place and thus the billing option, and also with customer identification about the receiver of a perfect copy of the video. The latter information can be used to trace copyright violators in conjunction with the appropriate personalized techniques that we present in Section 5.

The data that is distributed into the wide-area network and cached on arbitrary nodes along the path is made unusable by content-independent corruption of data in the file. Correct replacements for the corrupted part of the data are transmitted to the customer by means of a point-to-point connection. Content-independence provides two advantages.

- First, it allows for a means to a much simpler, on-the-fly corruption of the data than content-sensitive encryption techniques.
- Second, no reconstruction strategy can be applied based on knowledge about the data stream itself.

The unicast portion is encrypted on the server side using a personal key of the receiver, e.g. a key provided by a trusted third party. If the unicast portion is small in comparison the complete video, the computational load of encrypting this portion of the video is relevantly below that which is induced by using an MPEG parser. Also, less interaction with the optimized output paths of video servers or video cache servers is necessary.

At the receiver's side, the unicasted data is decrypted using the personal decryption key, and established synchronization approaches ([7]) can be applied to synchronize the unicasted partial transmission with the

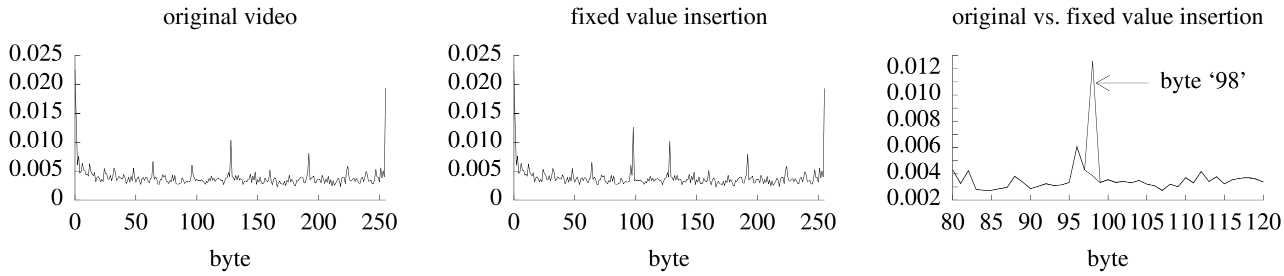

Figure 2. Byte value frequencies in original movie and after fixed corruption

main part of the data which is received from a nearby cache server.

The decision whether the encrypted data can be consumed directly from the data stream or whether a download is necessary depends mainly on the observed throughput. It can be made independently by the client while the stream is being received. Assuming that the unicast portion makes up 1% of an MPEG-1 video, the necessary point-to-point throughput is approximately 2 kilobyte/s, which is streamable in large parts of the Internet nowadays. For the bulk of the video data we assume the presence of a cache server to which a 1.5 MBit/s streaming connection can be established.

4.2 Specifcs of MPEG-1

To the extent described so far, our approach can be used with all kinds of streamed media data, but the small subset of data that we intend to corrupt is not generally sufficient to confuse an arbitrary encoding algorithm. The reason for this is not the generic use of encoding formats but the reduction of the knowledge about the stream which is necessary to corrupt the data sufficiently to remove perceptible information. We have applied the scheme to MPEG-1, for which it is feasible. For other, especially plainly intra-coded formats, reconstruction by comparison of neighbour frames can be automatized more easily. In typical MPEG-1 groups of pictures, however, I-frames, which are the basis for repairing frames, are sufficiently far apart in time to make this auttomatic reconstruction difficult for large parts of the video. D'Ardia et. al. have presented results in [2] that show a relevant variation between consecutive I-frames in most kinds of video transmission such as sports, news or movies. The scheme could probably not by applied to talk-shows because the low variation between consecutive I-frames allows reconstruction of the sequence.

We have investigated the percentage of data that needs to be corrupted in an MPEG-1 video stream to reduce the video quality to a 'teaser' or worse quality. In contrast to other approaches, which work on the uncompressed images, we make use of the two vulnerabilities of MPEG to data corruption or data loss.

The first vulnerability is that the destruction of an MPEG-1 I-frame affects all in the following group of pictures. In a video that has been compressed with a typical group of pictures lengths of 15 frames, the error is expanded in time by the relative decoding in P- and B-frames and affects all 15 frames.

The second vulnerability is introduced by the compression scheme. MPEG-1's Huffmann encoding improves the effectiveness of our intentional corruption of single bytes of data. Since the Huffmann algorithm is bit-oriented rather than byte-oriented, a typical Huffmann decoder implementation is unable to recover from the error for the rest of a data segment. Furthermore, a complete Huffmann decoding of the data is necessary before the corruption is detected because all bytes except for the special values 0xff and 0x0 are meaningful to the Huffmann decoder. As a result of this error propagation from a corrupted byte to the rest of a data segment in a frame, the number of bytes that need to be destroyed to corrupt a compressed MPEG frame completely is much lower than the number of bytes necessary for an uncompressed frame. To verify the second vulnerability, we have tested various clips and parameter sets.

Because of the error propagation, the destruction of larger blocks with the same overall ratio of corrupt to correct bytes turns out not to be feasible. The reason for this is the effect that Huffmann decoders generate corrupted data from the bytes immediately after the first corrupted bit. This effect is not increased by longer series of corrupted bytes. The corruption of single bits may be as efficient as the corruption of bytes, but it is not feasible in our case because the bit changes increase rather than decrease the number of CPU operations.

4.3 Hiding Errors in the Stream

A question that arises concerns the option that are available to a data pirate. Most probably, we have to add additional security mechanisms. We assume that the encryption algorithm and the key exchange mechanisms protect the data from being stolen by an eavesdropper during transmission and thus, that the encrypted part of the content remains safe. The primary concern is then whether the unencrypted data is protected from restauration.

In our experiments we can distinguish the selection of fixed or variable byte values used for the corruption of the

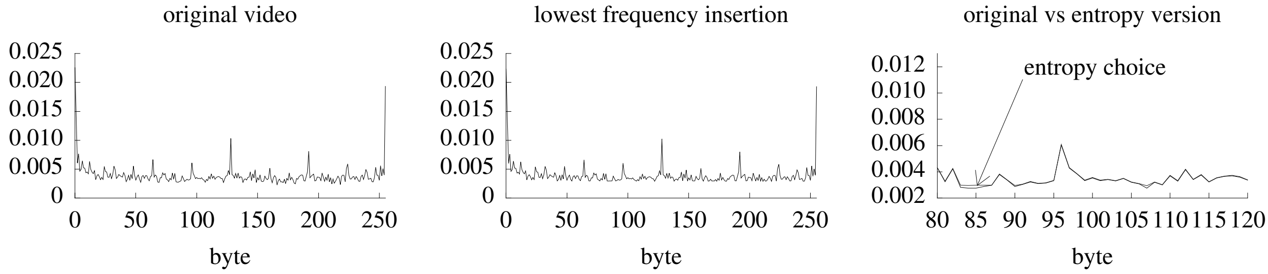

Figure 3. Byte value frequencies in original movie and after statistical corruption

original stream, and the applications of this corruption at periodic or variable offsets from each other.

An attacker may easily identify both a fixed byte value (by gathering statistics on frequencies, see Figure 2) and a periodic offset (by the use of auto-correlation). Both information should be concealed as good as possible, obviously. To prevent the possibility of identification it is necessary and essential to vary both the replacement distance and the replacement value.

Some of our tests were conducted by corrupting bytes at an equal distance throughout the video. Such a sequence is easily detected; this is a potential problem. To eliminate this problem, we apply the Poisson distribution and a random seed per movie. We write bytes from the original video to a file (the unicast portion) and afterwards, destroy those bytes in the original video. The unicast portion is stored on the server and encrypted once per request. The seed value is distributed to the receiver before all other content that is sent in the unicast transmission. The receiver's implementation of the distribution function must be identical to the sender's, e.g. we have used the rather limited, but floating-point free, implementation of the libg++ with 32 bit signed integers. Under this assumption, the receiver is able to replace the corrupted bytes with the decrypted bytes when it receives them over the unicast connection.

In our first approaches we evaluated the effect of inserting constant values (bytes) in video streams for simplicity reasons. However, we found that such bytes were too easily identified by an attacker. We respond to this by trying to replace the correct byte with a corrupt byte such that the chances of identifying this byte as corrupt by statistical analysis of the stream or part of the stream is minimal. The most effective approach seems to be to increase the "chaotic" appearance of the data to an attacker.

To have a measure for "chaos", we concentrated on the entropy value of the data and the changes to this value by the introduction of errors. To achieve the highest possible entropy in a file, the probability to find a single byte value in any set of bytes taken from the stream would be equal to any other value taken from the set.

The entropy is computed as follows: the relevance I of a byte value depends on the frequency h_i of that byte i in a data set ([6]).

$$I(h_i) = K \cdot \log h_i \qquad ;K = -\frac{1}{\log 256} \qquad (1)$$

Thus, we calculate the entropy H as the average relevance of all bytes by

$$H = \bar{I} = \sum_{i=1}^{n} h_i \cdot I(h_i) = K \cdot \sum_{i=1}^{n} h_i \cdot \log h_i$$

$$;K = -\frac{1}{\log 256} \qquad (2)$$

Figure 2 is also a demonstration of these results. If the information difference $\Delta I = I(h_{new}) - I(h_{old})$ between old and new value is negative the entropy decreases. If it is positive the entropy increases. We can apply this formula to the data of a video stream as it is streamed. Thus, we are able to control whether we want a higher or a lower entropy because we can choose the value of the corrupted byte in the output stream freely. We want to use this change of entropy as a possibility to hide the manipulated byets containing error values.

Control or padding bytes have high frequencies. Thus, their relevance in the calculation of the entropy value H is low according to (2). If the goal is to present a less informative stream you have to present something like a blank paper. The optimum would be that some special chosen bytes are presented very often and the frequency of the other values converge to 0. The probability of finding less frequent values is low but especially those values should be changed. However, the most frequent bytes in MPEG are header and padding bytes. Thus, a byte that assumes either of these values that is obviously in the wrong place could be identified quickly, which simplifies reconstruction of the original. We concluded that lowering the entropy in this way is ineffective.

To increase entropy is an easier task. Because the high entropy of compressed video streams is typically very high from the strart, it is most effective to choose the least frequent byte from the stream to replace the original value

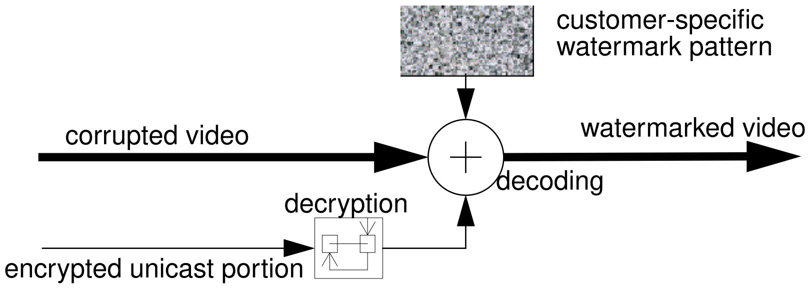

Figure 4. Watermarking at the client side

at any position. To increase entropy means that all values should come close to an identical frequency.

Thus, choosing a value with high frequency (and it is very probable to be hit) and replacing it by a value with low frequency (optimum would be the value with lowest frequency according to (2)) has the greatest effect on the overall entropy value, which reaches its maximum with a uniform distribution of all byte values.

For our scheme, we have decided to simplify the finding of these least frequent values. Instead of collecting statistics of the complete video stream, we select the value for insertion by identifying the byte i with the lowest relevance $I(h_i)$ by calculating $I(h_i) = k \cdot \log h_i$ on all bytes that have been observed previously in the stream.

Figure 3 demonstrates the nearly invisible changes in byte frequencies that can be achieved by this means. Thus we assume that the optimal fill byte value for hiding the replacement inside the stream can be found by observing the byte frequencies. Note that the probability to change a byte value with high frequency is high, although not optimal in our case since we do not search actively for frequent bytes in the stream.

A potential danger to this approach would be the use of exactly the same formula in an attack to identify the positions that have been manipulated by us. Since the entropy of MPEG streams is extremely high from the start (we found entropy values between 97.4% - 99% in our example videos), this approach can also not be used to identify the positions of these bytes once that they have been inserted following our approach.

4.4 Experimental Results

All of our test clips were approx. 1.5 MBit/s MPEG system streams from various sources and decoders. Movie lengths were between 17 and 615 MBytes.

As mentioned above, our approach is content-insensitive. It is generally assumed that reconstruction of headers for MPEG-1 is relatively simple because current encoders produce CBR streams and use always the same header data anyway. Thus, in all our experiments we took care that all headers of a video are correctly re-inserted. The remaining errors are disturbing enough to yield results that are unacceptable for commercial exploitation. We have also reconstructed the audio parts of the stream in most of the tests for a few reasons. First and foremost, the players synchronize their timing to the audio clock and we wanted to remove the effect of timing errors from our experiments. Second, Windows would frequently hang when the audio device drivers received too much corrupted data. Third, measuring the level of destruction in an audio track requires devices that we did not have available.

We started experiments with a destruction ratio of 1%, assuming that this would not be sufficient to destroy the video sufficiently to render it unviewable. However, using an error size of 1 byte, the error propagation rendered movies unplayable to two MPEG players (ActiveMovie, VideoCharger Player) and showed only artifacts in other (MpegTV). This observation was made even after full reconstruction of all headers and the complete audio stream.

Subsequently, we considered even lower error rates and found 0.5% to deliver bad quality and 0.1% to provide a quality sufficient to read blocks of text that remain immovable for several seconds, e.g. in the trailers. All of our numbers are adequately above current capabilities of restoration to good quality which handle bit error rate of about to 10^{-4} well.

5. DISCOURAGING RESALE

In the described commercial on-demand applications with caches we first transmit a corrupted version of the multimedia content via the caching and prefetching mechanisms, but without encryption. The contents' quality is bad enough to make resale not worthwhile. The missing quality is encoded in the unicast portion and delivered encrypted to the individual user via unicast transmission. By itself, this transmission scheme might discourage the resale of the content that is present in the cache, but it does not inhibit a cache owner from buying the multimedia content by himself legally just once, decoding the missing information and re-selling the complete multimedia content afterwards.

5.1 Watermarking

To discourage such a scheme, too, we propose to add a watermark to the content that is generated personally for each receiver of the unicast portion of the stream. To perform a customer specific watermarking of the transmitted video we want to apply a watermarking algorithm to the transmitted encrypted minimal part which delivers the high quality difference. A customer-specific watermarking pattern with the watermarking information, for example the name of the customer, is generated. This pattern in transferred to the customer's decoder at the head of the unicast transmission. There, it is DCT transformed and stored in the decoder at the client side. While the following correct data is inserted into the corrupt video stream, the watermark is added to the DCT coefficients of the video stream. Figure 4 illustrates the scheme.

Our watermarking algorithm is implemented for MPEG video and is based on overlaying a pattern with its power concentrated mostly in low frequencies. The pattern is created using a pseudo random number generator and a cellular automaton with voting rules. This idea was described for still images by Fridrich ([4]) at first, but had the disadvantage, that the original image was necessary for retrieval and there was no detailed information about the author or producer embedded. To ensure the last requirement instead of one overlaying pattern over the whole frame we add an 8x8 pattern over every 8x8 Block of the frame. To embed binary code words we define additional modification rules of the overlaying 8x8 pattern described in Dittmann et.al. ([3]). Furthermore we use statistical properties to find the label in the retrieval procedure without the original frame and error correcting codes to ensure more robustness.

The watermarked frames show no visible degradation caused by the overlaid pattern, yet the pattern is embedded in a robust sense. It is possible to prove the presence of the pattern in single video frames after filtering, MPEG compression, cropping, re-sampling, blurring, down-sampling, and noise adding. The watermark also appears to be resistant with respect to the collusion attack (averaging several watermarked images to remove the watermark). The watermarked video can be tracked and used to prove the theft of the video data.

5.2 Error Insertion

As an alternative to user-sided watermarking of a data stream, we are now investigating yet another corruption approach to discourage resale of the content by customers. We consider the insertion of random sequences of very scarce bit errors into the unicast portion of the stream. Like watermarking, this can be exploited to prove copyright violation in a way that makes the danger of manipulation to the decoder software irrelevant.

The unauthorized reseller may decide to request the video multiple times in order to use a voting mechanism and eliminate the bit errors (since we assume that the technique is known). It is relevant to find a scheme that will yield a sufficiently large number of remaining bit errors to single out the unauthorized reseller and take further measures to prove the contract violation. Bit errors that remain after the execution of voting steps to eliminate bit errors can be identified by the content provider using a brute force approach of computing these values based on the seed values on file.

We have examined a couple of schemes that insert infrequent bytes into the video stream randomly and found that completely random errors are easily fixed by applying voting mechanisms. Our current idea is to choose for each video a random sequence of intervals of the unicast portion. For each delivery of the stream, a uniform distribution is applied to put one byte error into each interval. Similar to the distortions of a watermark, each copy can be identified by these randomly inserted errors when the provider keeps the random seed values in a database. If unauthorized copies of the video are uncovered, the bit error sequences can be compared with the series of bit errors which are generated by the seed value on file using a brute force approach.

If the attacker chooses a 3-copy voting to eliminate the bit errors, errors remain with some probability that can be used to identification the original customer. Let the length of the video be S_f, the unicast portion S_u, with $T = S_f/S_u$. If the average offset is O and the length of each interval is I, there is a probability of $S_f/(OTI^2)$ that a least one byte error remains. For a 1GB MPEG-1 video, 0.5% encrypted transmission, $O = 1000$ bytes (resulting in a $0.5 \cdot 10^{-6}$ byte error rate in the video) and $I = 100$, this computes to 0.537. Smaller intervals increase this probability considerably.

However, the necessary length of the video for the application of this idea is large, so further investigations are necessary to understand whether this is feasible.

6. CONCLUSION

We have presented a way of distributing multimedia content in wide-area networks that allows to make use of caching and prefetching for the distribution of the larger part of the video data. By corrupting part of the original video data and transmitting this data in an additional very low bitrate, secure point-to-point transmission to the customer, we ensure that reconstruction and resale of the video by data thieves in the network is not worthwhile. Specifically, this approach allows the content provider to co-operate in a non-centralized distribution system with cache owners with the need to trust them.

We have further proposed two ways to combine this approach with a means of identifying re-sellers of multimedia content. One approach uses customer-specific watermarks that are insertion into the content during the recombination of the corrupted part and the unicast portion of the content. The other approach reapplies the corruption idea on top of the original corruption, but instead of fixing the errors before presentation to the user, these very infrequent errors remain in the stream for identification purposes.

In conjunction with any of the two approaches to make resale of the data by the customer tractable, the partial encryption scheme that we presented for freely distributing major parts of a multimedia content protects the content providers copyright sufficiently for several applications that require mass distribution of non-confidential, commercial multimedia data.

7. REFERENCES

[1] A. Dan, D. Sitaram, P. Shahabuddin: "Dynamic Batching Policies for an On-Demand Video Server", Multimedia Systems 4(3), pp. 112-121, 1996

[2] L. D'Ardia, B. Fadini, G. Ventre: "Analysis, Classification and Simulation of Multimedia Traffic Sources", Proceedings of DDCN '97, Disitributed Computer Communication Networks, Tel Aviv, November 1997

[3] J. Dittmann, M. Stabenau, R. Steinmetz: "Robust MPEG Watermarking Technologies", 1998, submitted to ACM Multimedia '98

[4] J. Fridrich, J.: "Methods for data hidung", Center for Intelligent Systems & Department of Systems Science and Industrial Engineering, SUNY Binghamton, Methods for Data Hiding", working paper, 1997

[5] C. Griwodz, M. Bär, L. C. Wolf: "Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie", ACM Multimedia 1997, November Seattle, WA, USA, November 1997

[6] W. Hilberg: Digitale Speicher 1, Oldenburg Verlag, ISBN 3-486-20270-7 (1987)

[7] J. Jarmasz, N. D. Georganas: "Designing a Distributed Multimedia Synchronization Scheduler", Proc.IEEE Multimedia Systems'97, Ottawa, June 1997

[8] T. Kunkelmann, R. Reinema, R. Steinmetz, T. Blecher: Evaluation of Different Video Encryption Methods for a Secure Multimedia Conferencing Gateway, Proc. of the 4th Int'l COST237 Workshop, Lisboa, Portugal, Dezember 1997, pp. 75-89

[9] T. B. Maples, G. A. Spanos: "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video", Proc. of the 4th Int'l Conf. on Computer Communications and Networks, Las Vegas, Nevada, September 1995

[10] L. Qiao, K. Nahrstedt, I. Tam: "Is MPEG Encryption Using Random Lists instead of Zig Zag Order Secure?", IEEE International Symposium on Consumer Electronics, December 1997, Singapore

[11] L. Tang: "Methods for Encrypting and Decrypting MPEG Video Data Efficiently", Proc. of the 4th ACM Multimedia Conference, Boston, MA, November 1996, pp. 219-230