MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

On Compression of Encrypted Video

Daniel Schonberg, Chuohao Yeo, Stark C. Draper, Kannan Ramchandran

TR2007-059 March 2007

Abstract

We consider video sequences that have been encrypted uncompressed. Since encryption masks the source, traditional data compression algorithms are rendered ineffective. However, it has been shown that through the use of distributed source-coding techniques, the compression of encrypted data is in fact possible. This means that it is possible to reduce data size without requiring that the data be compressed prior to encryption. Indeed, under some reasonable conditions, neither security nor compression efficiency need be sacrificed when compression is performed on the encrypted data. In this paper we develop an algorithm for the practical lossless compression of encrypted gray scale video. Our method is based on considering the temporal correlations in video. This move to temporal dependence builds on our previous work on memoryless sources, and on- and two-dimensional Markov sources. For comparison, a motion-compensated lossless video encoder can compress each unencrypted frame of the standard F̈oremanẗest video sequence by about 57%. Our algorithm can compress the same frames, after encryption, by about 33%

Data Compression Conference (DCC)

On Compression of Encrypted Video*

Daniel Schonberg
EECS Dept.
U. of California, Berkeley
Berkeley, CA 94720-1772
dschonbe@eecs.berkeley.edu

Chuohao Yeo
EECS Dept.
U. of California, Berkeley
Berkeley, CA 94720-1772
zuohao@eecs.berkeley.edu

Stark C. Draper
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139
sdraper@eecs.berkeley.edu

Kannan Ramchandran
EECS Dept.
U. of California, Berkeley
Berkeley, CA 94720-1772
kannanr@eecs.berkeley.edu

Abstract

We consider video sequences that have been encrypted uncompressed. Since encryption masks the source, traditional data compression algorithms are rendered ineffective. However, it has been shown that through the use of distributed source-coding techniques, the compression of encrypted data is in fact possible. This means that it is possible to reduce data size without requiring that the data be compressed prior to encryption. Indeed, under some reasonable conditions, neither security nor compression efficiency need be sacrificed when compression is performed on the encrypted data (Johnson et al., 2004).

In this paper we develop an algorithm for the practical lossless compression of encrypted gray scale video. Our method is based on considering the temporal correlations in video. This move to temporal dependence builds on our previous work on memoryless sources, and one- and two-dimensional Markov sources. For comparison, a motion-compensated lossless video encoder can compress each unencrypted frame of the standard "Foreman" test video sequence by about 57%*. Our algorithm can compress compress the same frames, after encryption, by about* 33%*.*

1 Introduction

The work of Johnson et al. [1] establishes that it is theoretically possible to compress encrypted data to the entropy rate of the unencrypted source. Since *good* encryption makes a source look completely random, traditional algorithms are unable to compress encrypted data. For this reason, traditional systems have to compress before they encrypt. Johnson et al. [1] show that the problem of compressing encrypted data is related to source coding with side information. It was also shown that neither compression performance nor security need be impacted under some reasonable conditions. For example, the scenario focused on in this work, lossless compression of data encrypted with a stream cipher theoretically exhibits no performance loss. A block diagram of this novel system structure is presented in Fig. 1.

Consider the problem of wireless video content delivery systems. Providers of video content place considerable value on content security. For systems such as home wireless networks, transport efficiency is essential for providing maximum range. Often though,

*This research was supported in part by the NSF under grant CCR-0325311. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

2007 Data Compression Conference (DCC'07)
0-7695-2791-4/07 $20.00 © 2007 IEEE

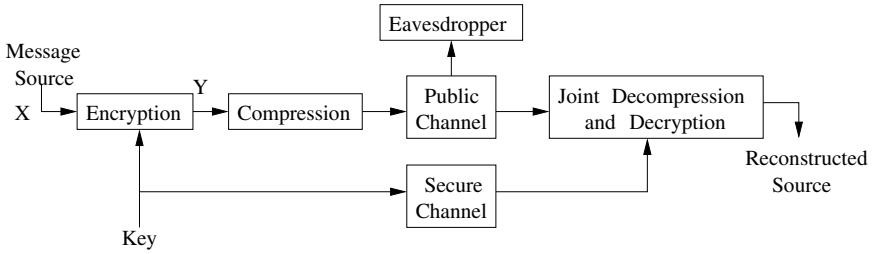


Figure 1. *The source is first encrypted and* then *compressed. The compressor does not have access to the key used in the encryption step. At the decoder, decompression and decryption are performed jointly.*

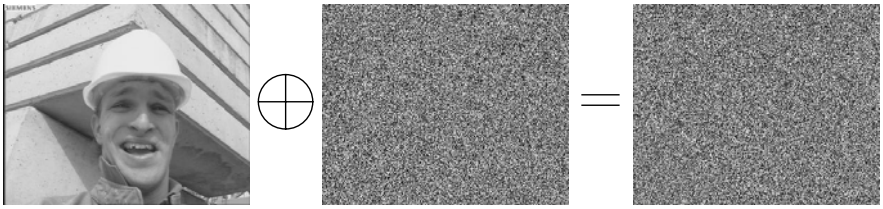


Figure 2. *A sample frame from the "Foreman" video is on the left (*$811,008$ *bits total). To encrypt this image, the* $811,008$ *bit random key in the center is added to the unencrypted image on the left (using a bit-wise exclusive-OR).*

video providers are highly concerned about content security, and encrypt their sources prior to distribution. The HDCP [2] (High-Bandwidth Digital Content Protection) protocol for example, requires encryption of uncompressed video data[1]. This transmission of full-rate, high-bandwidth video creates a conflict with end users who wish to be spectrally efficient. Ideally, we would like content providers to first compress their data. In practice though, we cannot enforce rigorous best practices with regard to compression. Instead, we would like to improve practical performance. We focus here on lossless compression to operate in conjunction with such protocols as HDCP, though our construction can be extended to lossy video coding.

To better understand the problem we present, consider the image in Fig. 2. On the left side of Fig. 2 is a single frame from the "Foreman" video sequence (we refer to this frame as the plain-text since it is unencrypted). This image has 288×352 pixels with each pixel value in the range of 0 to 255. Only 8 bits are needed to represent each pixel value, thus this image has a total of $811,008$ bits. The middle image of the figure is an image of equal size and range as the first image, but consisting of bits selected uniformly at random (i.e., they are IID Bernoulli random variables, each with parameter 0.5). We use the bits of this image as the key bits for our encryption system. We encrypt the image by applying a bitwise exclusive-OR (XOR) between the key bits and the source. The resulting encrypted image is shown at right (referred to as the cipher-text since it is encrypted). Though several algorithms exist today for compressing the highly structured unencrypted image on the left, there are no image compression algorithms that can compress the *marginally random* image on the right.

To understand why the encrypted image on the right of Fig. 2 is compressible, note that while the unencrypted plain-text and cipher-text are independent, compression is

[1]Video providers often encrypt their data uncompressed with the motivation that the resulting deluge of data will overwhelm attackers. An unfortunate side effect is that the deluge can also overwhelm many communication links over which we would like to distribute the video, e.g., home wireless networks.

achieved *by leveraging the dependence between the cipher-text and the key.* This dependance can be understood by viewing the cipher-text as a noisy version of the key stream. For example, if we know the image has a low Hamming-weight, then the possible cipher-texts are clustered around the key sequence. Since the key stream is available at the decoder, we can reconstruct the cipher text with the compressed data. Reconstruction is achieved via Slepian-Wolf decoding [3, 4] (as discussed in [1]). In order to develop systems that can compress encrypted data, we develop distributed source coding schemes whose inter source correlations match the unencrypted source's statistics.

To date, practical schemes for compressing encrypted data have focused on simple source models. Johnson et al. [1] consider a memoryless model. Practical systems for sources with 1-D [5] and 2-D [6] Markov memory are described. Unfortunately, extending these models to practical gray-scale images proves difficult. Two of the most common techniques used for image compression are inapplicable when the image is encrypted. The first method is the application of transforms, such as the DCT (Discrete Cosine Transform). As can be seen in Figure 2 though, bitwise image encryption is a non-linear operation. That is, the transform of the encrypted image lacks the exploitable structure of the transform of the unencrypted image. The second method is the use of localized predictors. Without access to the local information available to compressors of unencrypted images though, these localized predictors simply fail.

In contrast, video supports a larger opportunity for compression (and arguably a larger need). Presuming a frame by frame compression and decompression, the decoder has access to temporal (as opposed to just spatial) predictors. Our goal then is to present a compression scheme focused largely on leveraging the temporal correlation of video rather than the spatial correlation of images. Specifically, in this work we present an encrypted video compression scheme based on prior decoded frames. We present a model of the bits of each image based on the pixels of prior frames, and a scheme based on sparse linear transforms for compressing the frames. We discuss this algorithm in detail and describe the tradeoffs considered. Finally, we present results for compression on the "Foreman," "Garden," and "Football" sequences.

This paper is organized as follows. Section 2 provides a discussion of the source model we use and the way we generate predictors and statistics of each frame for the decoder. In Section 3, we discuss our method for compressing encrypted data. We then present results in Section 4 and finally provide concluding notes and future plans in Section 5.

2 Source Model and Prediction Scheme

In this work, we focus on gray-scale images whose pixel values range from 0 to 255. Pixels are assumed to lie on a grid with N_h rows and N_v columns. We focus on a bitwise decomposition of each pixel, using 8 bits per pixel. We describe a bit plane as every pixel in a frame of equal significance in the pixel representation. The bit planes range from most to least significant. Due to the bitwise focus of this model, we find it convenient to consider each bit plane separately.

For compressing a single bit plane, we leverage the model of [6] for representing binary images[2]. In this work, the correlation between each pixel $x_{i,j}$ and its 4 nearest neighbors is considered; up & down, left & right. A section of the model is illustrated via a factor

[2]The source coding with side information construction of [7, 8], developed concurrently, is related but considers an image and a noisy version of the same image. Since here neither the key nor the cipher-text are images, their construction does not directly apply.

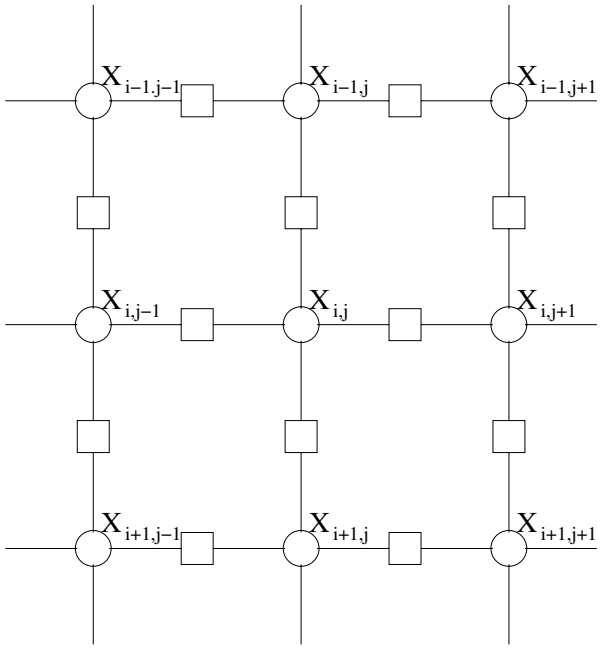


Figure 3. *A factor graph for the spatial source model considered in [6]. The circles and squares on the grid represent the Markov field model. As used in this paper, a copy of this model is applied to every bit plane of every frame. We omit frame and bit-plane notation from this figure for clarity.*

graph [9] in Fig. 3. Factor graphs are bipartite graphs consisting of variables (represented by circles) and constraints on those variables (represented by squares). In the graph in Fig. 3, the circles labeled $x_{i,j}$ represent the bits of the image and the squares represent the dependance between pixels.

In [6], the model is assumed to have a stationary distribution. The marginal probability on each bit is denoted $p = Pr(x_{i,j} = 1)$. The horizontal correlation parameters are denoted $h_0 = Pr(x_{i,j} = 1|x_{i,j-1} = 0) = Pr(x_{i,j} = 1|x_{i,j+1} = 0)$ and $h_1 = Pr(x_{i,j} = 1|x_{i,j-1} = 1) = Pr(x_{i,j} = 1|x_{i,j+1} = 1)$. Vertical correlation parameters are denoted $v_0 = Pr(x_{i,j} = 1|x_{i-1,j} = 0) = Pr(x_{i,j} = 1|x_{i+1,j} = 0)$ and $v_1 = Pr(x_{i,j} = 1|x_{i-1,j} = 1) = Pr(x_{i,j} = 1|x_{i+1,j} = 1)$.

We now extend this model to the bit plane model of videos. Specifically, we will relabel each bit as $x^t_{i,j}[l]$ to represent the l-th most significant bit of the pixel at row i and column j of frame t. A dropped index implies that we are considering the entire range over that index. For example, $x^t[l]$ represents the entire l-th bit plane of the frame t, while x^t represents the entire frame at time t with elements ranging from 0 to 255 (since we include every bit plane).

Using the methodology introduced in [6] for compressing encrypted images, we find only insignificant gains for all but the two most significant bit planes. Instead, we consider the following approach, diagramed in Fig. 4. For the sake of convenience, we assume our system works on one frame at a time. This allows our decoder to have a full copy of the previous frames available for processing. We operate by having a predictor generated of each frame based on the two previous frames. That is, we generate predictor $\hat{x}^t = g(x^{t-1}, x^{t-2})$. This function is represented by the "Predictor" boxes in Fig. 4 below.

Given that the previous two frames, x^{t-2} and x^{t-1}, were decoded successfully, we would like to generate a prediction, $\hat{x}^t$ of the current frame x^t. In this work, we implemented a simple predictor operating across the bit planes that uses motion compensation together

with motion extrapolation to generate $\hat{x}^t$. x^{t-1} is divided into sub-blocks of $N \times N$ pixels ($N = 8$ in this work), and for each block, motion estimation is performed to find the best matching block in x^{t-2}. In other words, for the (a, b) block, with $a \in 1, 2, ..., N_h/N$ and $b \in 1, 2, ..., N_v/N$, we find $(v_x(a, b), v_y(a, b))$ such that:

$$(v_x(a,b), v_y(a,b)) = \arg \min_{(v_x,v_y)\in\mathcal{N}} \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} d\left(x^{t-1}_{aN+i,bN+j}, x^{t-2}_{aN+i+v_y,bN+j+v_x}\right)$$

where $\mathcal{N}$ denotes the allowable search range, and $d(\cdot, \cdot)$ is a metric that measures the difference between two pixel intensities. Here, we simply used the square difference, i.e. $d(\alpha, \beta) = |\alpha - \beta|^2$. Assuming that there is little change in the motion fields over one frame instance, we estimate the motion vector for each block in x^t as the motion vector of the co-located block in x^{t-1}. Therefore, if (v_x, v_y) is the motion vector of the (a, b) block of x^{t-1}, the (a, b) block in $\hat{x}^t$ is then given by:

$$\hat{x}^t_{aN+i,bN+j} = x^{t-1}_{aN+i+v_y,bN+j+v_x} \ , \forall \quad 0 \le i, j \le N-1$$

We recognize that a more sophisticated predictor could lead to a better $\hat{x}^t$ and hence a better model performance, but this simple implementation suffices for a proof of concept.

It is clear that the predicted frame $\hat{x}^t$ can be useful for understanding the actual frame x^t. To make use of the predictor though, we must model how accurate of a predictor it is. We do this by considering the empirical distribution $P(x^{t-1}, \hat{x}^{t-1})$ of the previous frames and their predictors. That is, we assume the joint distribution of our current frame and its predictor is $\hat{P}(x^t, \hat{x}^t) = P(x^{t-1}, \hat{x}^{t-1})$. In Fig. 4, we represent this function with the box labeled "P(Actual,Predicted)." In practice, we actually use all previous frames (with a forgetting factor) and their predictors to generate the empirical distribution, but we omit this from Fig. 4 for clarity.

The overall model consists of a predicted frame $\hat{x}^t$ based on the two previous frames and an empirical distribution $\hat{P}(x^t, \hat{x}^t)$ based on the previous frame and its predictor. We use these to alter the parameters (p, h_0, h_1, v_0, v_1). We no longer assume them to be stationary, instead we make them dependent upon their context ($p(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $h_0(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $h_1(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $v_0(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $v_1(x^t_{i,j}[l]|\hat{x}^t_{i,j})$). Specifically[3], we calculate each of these parameters using the marginalized empirical distribution $\hat{P}(x^t, \hat{x}^t)$ conditioned on $\hat{x}^t_{i,j}$.

3 Encoder and Decoder

In this section we present a practical encoder and decoder for compressing encrypted data. As stated above, we assume the decoder is supplied with an estimate of the statistics ($p(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $h_0(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $h_1(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $v_0(x^t_{i,j}[l]|\hat{x}^t_{i,j})$, $v_1(x^t_{i,j}[l]|\hat{x}^t_{i,j})$). Further, we assume that the encoder knows what rate to use (we discuss this assumption in more detail below). We compress the encrypted source using a sparse linear transformation implemented with a matrix multiplication. A detailed description of the design of the linear transformation matrix (and the basis for this codec) can be found in [10]. In particular, the design of the transform matrix (with a modification discussed below) is based on LDPC (Low-Density Parity-Check) codes [11].

[3]In practice we also condition these parameters on the bit planes of greater significance, since we process the bit planes successively from most to least significant. For example, we condition the parameters for the third most significant bit plane based on the two most significant bit planes. We omit this from the notation for clarity.

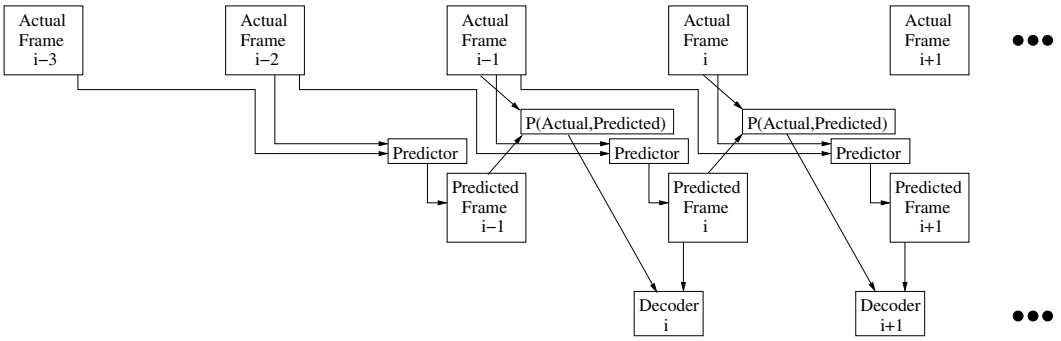


Figure 4. *A diagram of the way by which statistics are generated and fed to the decoder for each frame. Here, predicted frames are developed from the two previous frames. A measure of the predictor's accuracy is generated by measuring the accuracy of the previous frame's predictor.*

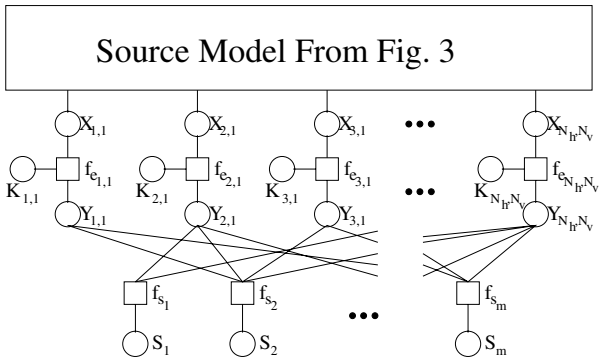


Figure 5. *The full graphical model for compressing encrypted spatially correlated sources. The model consists of the source model on top (abstracted here but shown in detail in Fig. 3), the encryption model in the middle, and the code on the bottom. Note that we omit the frame and bit plane indices for clarity.*

The decoder operates by running belief propagation over the factor graph [9]. We thus proceed by describing the appropriate factor graph. The graphical model consists of three components connected together; the model for the source, the encryption, and the code. Details of the source graphical model were described in Section 2 and shown in Fig. 3.

We form the encryption model and attach it to the source model as shown in Fig. 5[4]. Since we consider only stream ciphers here, we can model the encryption process as $y_{i,j}^t[l] = x_{i,j}^t[l] \oplus k_{i,j}^t[l]$, where $y_{i,j}^t[l]$ is the cipher-text, $k_{i,j}^t[l]$ is the bits of the key, and $\oplus$ indicates the XOR operation. We represent the constraint between these three variables in the graphical model with a square node labeled $f_{e_{i,j}^t[l]}$. The circles representing the variables $x_{i,j}^t[l]$, $k_{i,j}^t[l]$, and $y_{i,j}^t[l]$ are all connected to the encryption constraint $f_{e_{i,j}^t[l]}$.

The code model consists of a representation of the linear transformation matrix $\mathbf{H}$, the cipher bits $y_{i,j}^t[l]$, and the compressed bits s_i. In [10] it was shown that good performance can be achieved when the transformation matrix is designed as an LDPC code. This structure is represented graphically in Fig. 5. The squares labeled f_{s_i} represent the linear transformation $\mathbf{H}$, and the results of that transformation are represented by the circles labeled s_i (i.e., the compressed bits).

Decoding is achieved using the sum-product algorithm on the factor graph of Fig. 5. The sum-product algorithm is an inference algorithm designed to be exact on trees. Al-

[4]In this figure, we omit the frame and bit plane indices for clarity.

though not exact on "loopy" graphs (such as the graph in Fig. 5), empirical performance is very good. Strong performance is due both to the code sparsity (thus its loops are long on average) and the source being smooth (thus there is strong dependency between adjacent bits). The algorithm iteratively updates an estimate of the distribution for each of the variables. In the first half of each iteration, the constraints (squares) update their messages while in the second half of each iteration, the variables (circles) respond by updating their messages. Messages represent the current distribution estimate.

In addition to the bits calculated using the sparse linear transformation, a portion of the source bits are transmitted uncompressed. We refer to these as "doped" bits. In practice, due to the strength of the predictors used, the doped bits are typically between 5% and 10% of the output compressed bits. These bits are used in two ways. First, since these doped bits are known unambiguously at the decoder they anchor the iterative decoder by catalyzing the process. Second, they provide a mechanism for additional estimation about the compressed data. The decoder empirically estimates the source statistics and incorporates these estimates into the decoder We demonstrate algorithm performance below.

As mentioned above, we have assumed here that the encoder knows at which rate to transmit. In a practical implementation, since the encoder has no access to the source data, some feedback is required for the encoder to know at which rate to transmit. One approach would be to use an iterative feedback scheme [5]. We can see that feedback need only be minimal for this system. Assuming consistent amounts of motion between frames, then each compressed bit plane requires a similar amount of rate as the same bit plane in the previous frame. Further, there is likely to be a predictable reduction in compressibility between successive bit planes of a single frame. The sum-product algorithm presented here has the added benefit as being able to provide a decoding error indicator. If the decoder fails to converge after a specified number of iterations, we record a decoding error. By reporting decoding errors back to the encoder with feedback, end to end performance is further improved.

4 Results

In this section, we present results from the compression of a sequence of 12 encrypted frames (i.e., a group of pictures (GOP) size of 12) of some typical test sequences; "Foreman" (low motion), "Garden" (high motion), and "Football" (high motion). In this group, we only compress the last nine frames (frames 4 through 12). As mentioned above, the first three frames are used to initialize the predictors. With a larger GOP size, the rate effect of the three uncompressed frames diminishes. Further, for greater ease of implementation, we divided each frame into 9 regions (arranged 3×3) as a grid. Each region, having 1/9 of the pixels, has on the order of $\sim 10^4$ pixels (and each bit plane has approximately $\sim 10^4$ bits). Better performance would be seen by considering each frame as a whole.

Our overall results are presented in Table 1. The numbers in this table represent the average compression ratio in terms of the number of output bits per source bit. For example, the number 0.6700 in the upper right most cell of the table indicates that on average only 0.6700 output bits were necessary to represent each bit of the source video. As a reminder, this number applies only to the last 9 frames in our group of pictures, ignoring the first 3 frames of the video for initialization as in Fig. 4. As can be seen in this chart, though we cannot perform as well as we would have on unencrypted data,

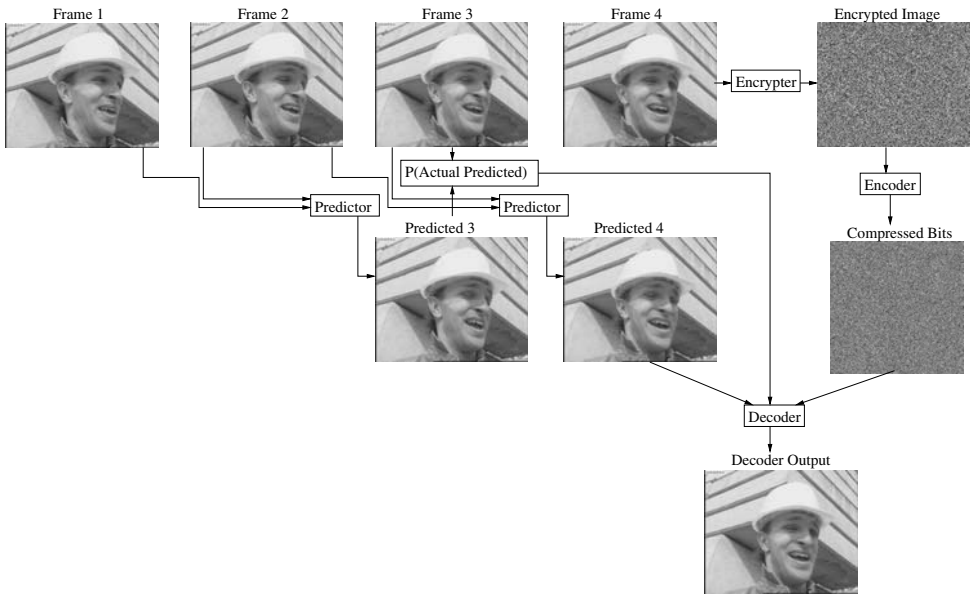


Figure 6. *A sample compression of the fourth frame of the "Foreman" sequence. This block diagram demonstrates the generation of each of the elements available to the encoder and to the decoder. Here, the decoder is able to reconstruct frame 4 from the predicted frame 4, the probability model, and the compressed bits.*

significant compression gains can still be achieved. Further, we can see the effect of the predictor on performance, as performance is far better for the low motion "Foreman" frames (and thus strong predictor quality) than the other sequences (large deviations between predicted and actual frames).

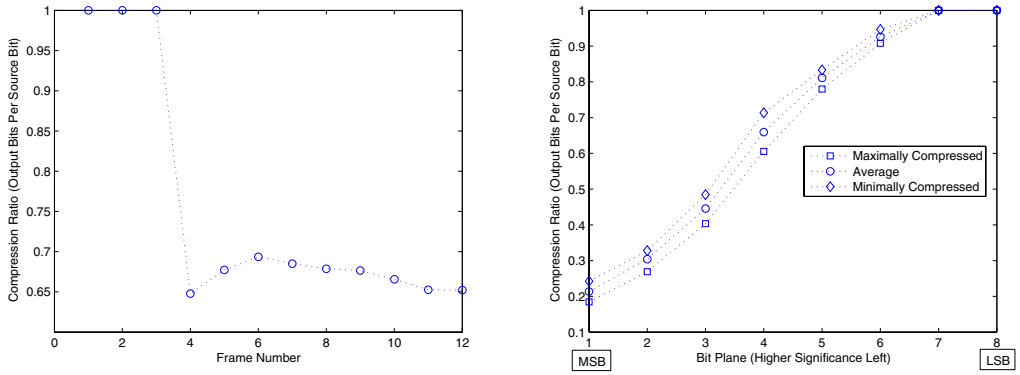
	JPEG-LS	MSU lossless	Encrypted Video
Foreman Frames 4 - 12	0.4900	0.4275	0.6700
Garden Frames 4 - 12	0.7300	0.6062	0.8515
Football Frames 4 - 12	0.6712	0.6138	0.9844

(1)

For purposes of comparison, we also compressed the videos losslessly using publicly available software. The first system uses JPEG-LS [12] to perform pure intra-frame video compression. JPEG-LS not only has low encoding complexity, but also demonstrates exceptional compression performance relative to other lossless image coding standards [13]. In our study, we compressed each video frame with the publicly available JPEG-LS coder [14]. The second system, MSU lossless video codec [15], is a publicly available lossless inter-frame video codec, which is claimed by its authors to have the highest lossless video compression performance [16]. Due to its proprietary nature, the details of their video codec is not known. However, its performance does seem to be comparable to past results in the literature that used either fixed spatio-temporal predictor [17] or motion compensation [18].

As an example of the operations performed for a particular frame, consider Fig. 6. This figure demonstrates everything that goes into the compression of frame 4. Here we can see that the encoder has access to only the encrypted version of frame 3. In contrast, the decoder is given access to the predicted frame 4, the probability model estimating the reliability of the predicted frame, and the compressed bits. In this example, the frame is recovered without error.

In Fig. 7, we present more detailed results from our compression of the encrypted "Foreman" sequence. The vertical axis of these plots represent the compression ratio from above (output bits per input bit). In the plot on the left (Fig. 7(a)), we plot the rate as a function of the frame number. Note that we made no attempt to compress the first three frames for this proof of concept. This plot shows that the overall performance varies as the video progresses but each frame is compressed to at least 70% of the source rate. In the plot on the right (Fig. 7(b)), we present the rate used for each bit plane (across the 9 compressed frames). The horizontal axis ranges from the most significant bit plane (1) at left to the least significant bit plane (8) at right. For reference, we were able to compress the most significant bit plane by 78% on average. Frame 12 was most compressible (81%) while frame 5 was least compressible (76%). This gives a good indication of how much correlation our system can exploit in each bit plane. As can be seen in this plot, we were able to obtain no gains in the two least significant bit planes of this sequence.



(a) (b)

Figure 7. Compression results for the 9 frames considered by our system. (a) The average rate (in source bits per output bit) used on each frame. (b) The rate used on each bit plane from most significant (left) to least significant (right). Maximum and minimum taken across frames.

5 Conclusions & Future Directions

In this work we have presented a practical scheme for compressing encrypted video. We described our model which uses powerful codes for compressing encrypted data. Additionally, we have also presented empirical results for compressing encrypted frames of the "Foreman," "Garden," and "Football" sequences. We have shown that although a price is paid for the lack of access to the data itself, significant compression gains can still be achieved.

The work presented herein is a proof of concept. It suggests a number of areas for further study. First, we will implement a system using feedback such that the system is fully automated. The influence of the quality of predictors on compression performance will be further studied and we would also develop techniques to compress the initial frames in order to reduce their influence on the overall results. Finally, though we consider lossless video coding here, our inability to compress the least significant bit planes suggests a way to apply our scheme to lossy video coding. Namely, by simply dropping the less significant bit planes, we will improve system performance while providing good signal

quality. Since these planes play little significance in the predictors, their loss would not strongly inhibit algorithm performance.

References

[1] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," in *IEEE Trans. Signal Processing*, vol. 52, pp. 2992–3006, Oct. 2004.

[2] Wikipedia, "High-Bandwidth Digital Content Protection." `http://en.wikipedia.org/wiki/High-Bandwidth_Digital_Content_Protection`, October 2006.

[3] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. IT*, vol. 19, pp. 471–480, Jul. 1973.

[4] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, New York, 1981.

[5] D. Schonberg, S. Draper, and K. Ramchandran, "On blind compression of encrypted correlated data approaching the source entropy rate," in *43rd Annual Allerton Conf.*, (Allerton, IL), Sep. 2005.

[6] D. Schonberg, S. Draper, and K. Ramchandran, "On compression of encrypted images," in *Proc. Int. Conf. Image Processing*, Oct. 2006.

[7] D. Varodayan, A. Aaron, and B. Girod, "Exploiting spatial correlation in pixel-domain distributed image compression," in *Proc. Picture Coding Symp.*, (Beijing, China), April 2006.

[8] S. S. Pradhan and K. Ramchandran, "Enhancing analog image transmission systems using digital side information: A new wavelet based image coding paradigm," in *Proc. Data Compression Conf.*, Mar. 2001.

[9] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," in *IEEE Trans. IT*, vol. 47, pp. 498–519, Feb. 2001.

[10] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "LDPC codes can approach the Slepian Wolf bound for general binary sources," in *40th Annual Allerton Conf.*, pp. 576–585, Oct. 2002. Submitted To IEEE Trans. Commun.

[11] R. G. Gallager, *Low Density Parity Check Codes*. PhD thesis, MIT, Cambridge, MA, 1963.

[12] M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles andstandardization into JPEG-LS," *IEEE Trans. Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[13] D. Santa-Cruz, T. Ebrahimi, J. Askelof, M. Larsson, and C. Christopoulos, "JPEG 2000 still image coding versus other standards," *PROC SPIE INT SOC OPT ENG*, vol. 4115, pp. 446–454, 2000.

[14] HP Labs, "HP Labs LOCO-I/JPEG-LS." `http://www.hpl.hp.com/loco/`.

[15] MSU Graphics & Media Lab Video Group, "MSU lossless video codec." `http://www.compression.ru/video/ls-codec/index_en.html`.

[16] MSU Grpahics & Media Lab Video Group, "MSU lossless video codecs comparison." `http://www.compression.ru/video/codec_comparison/lossless_codecs_en.html`.

[17] D. Brunello, G. Calvagno, G. Mian, and R. Rinaldo, "Lossless compression of video using temporal information," *IEEE Trans. Image Processing*, vol. 12, no. 2, pp. 132–139, 2003.

[18] I. Matsuda, T. Shiodera, and S. Itoh, "LOSSLESS VIDEO CODING USING VARIABLE BLOCK-SIZE MC AND 3D PREDICTION OPTIMIZED FOR EACH FRAME," *European Signal Processing Conf.*, pp. 1967–1970, 2004.