ON COMPRESSION OF ENCRYPTED IMAGES

Daniel Schonberg, Stark Draper, Kannan Ramchandran

UC Berkeley, EECS Department, 211 Cory Hall #1772, Berkeley, CA 94720-1772
{dschonbe,sdraper,kannanr}@eecs.berkeley.edu

ABSTRACT

Coding schemes for secure and efficient communication over noiseless public channels traditionally compress and then encrypt the source data. In some cases reversing the ordering of compression and encryption would be useful, e.g., in enabling the efficient distribution of protected media content. Indeed, not only is it possible to reverse the order, but under some conditions neither security nor compression efficiency need be sacrificed. In earlier work on this problem we have assumed that the source data is either memoryless or has a 1-D Markov structure. Such models are poor matches for the 2-D structure of images. In this work, we use a 2-D source model, and develop a scheme to compress encrypted images based on LDPC codes. We present practical simulation results for compressing bi-level images. In tests, we are able to compress an encrypted $10,000$ bit bi-level image to $4,299$ bits and successfully recover the image exactly. In previous works, the best analogous 1-D model (operating on a raster scanned data sequence of the same source) could only compress the image to $7,710$ bits.

***Index Terms*—** Image coding, Data security, Data compression, Source coding, Message passing

1. INTRODUCTION

In [1] it was shown that it is theoretically possible to compress encrypted[1] 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 make sure 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 shown that neither compression performance nor security need be impacted under some reasonable conditions. A block diagram of this system structure is in Fig. 1.

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

[1]In [1] it was shown that with a stream cipher there is no loss in either compressibility or security. As an example, Shannon's one time pad is a stream cipher. We focus solely on stream ciphers here.

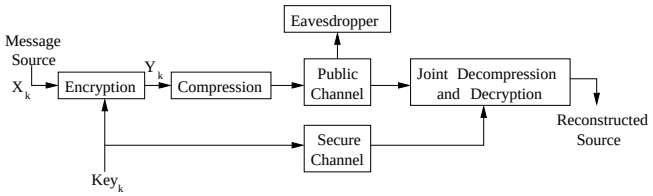

Fig. 1. *The source is first encrypted and* then *compressed. The compressor does not have access to the key used in the encryption step. At the decoder, decompression and decryption are performed jointly.*

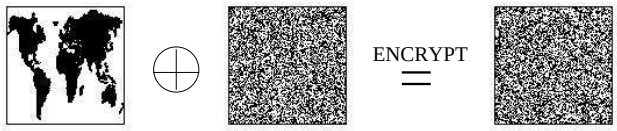

Fig. 2. *A 100x100 sample binary image is on the left* ($10,000$ *bits). To encrypt this image, the* $10,000$ *bit random key in the center is added to the unencrypted image on the left.*

For an example, consider the images of Fig. 2. In this figure, the world map image on the left and the key in center are added via bitwise exclusive-OR to produce the encrypted image on the right. Though several algorithms exist today for compressing the highly structured unencrypted image on the left, no image compression algorithms exist that can compress the *marginally random* image on the right.

To understand why the image on the right is compressible, note that while the unencrypted plain-text and cipher-text are independent, compression is achieved *by leveraging the dependance 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. 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 coding [2, 3]. 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. In [4], a practical system for sources with a Markov *chain* memory structure is described. There is still a significant gap between these models and "real-world" images[2], which are better modeled by their natural 2-D structure. As can be seen with Fig. 2, a 1-D model is insufficient to capture all the structure in the image.

In this work we describe how to decode using a model designed to capture the underlying 2-D structure of images. The result is more efficient compression of encrypted images. We implement a practical scheme, based on LDPC (Low Density Parity Check) codes (for compressing encrypted images). We describe how to apply our scheme to binary images.

This paper is organized as follows. We describe the source model in Section 2 and the encoder and decoder in Section 3. We present simulation results in Section 4, and conclude in Section 5.

2. SOURCE MODEL

In this section, we discuss our model for spatially correlated sources. We consider binary images, wherein each pixel $x_{i,j}$ takes on one of two values, i.e., $x_{i,j} \in \{0, 1\}$. Images are sampled on a rectangular grid with N_h rows and N_v columns. In order to model this source, in prior work [4], the bits were raster scanned and only the correlation between successive bits in the scan were considered. This forces a 1-D model on the data. In this paper, we consider the correlation between each pixel and its 4 nearest neighbors; up & down, left & right. We consider here a Markov *field* model for spatially dependant sources instead of a Markov chain.

We illustrate the model by way of factor graphs [6]. Factor graphs are bipartite graphs consisting of variables (represented by circles) and constraints on those variables (represented by squares). A section of the factor graph for the 2-D Markov field model for binary images is presented in Fig. 3. In the graph in Fig. 3, the circles labeled $x_{i,j}$ represent the bits of the image and the squares labeled $f_{th_{i,j}}$ and $f_{tv_{i,j}}$ represent the dependance between pixels.

We consider the following parameterizations of the stationary distribution over the Markov field. We denote the marginal probability on each bit as $p = Pr(x_{i,j} = 1)$. We take the correlations to be symmetric for both the horizontal and vertical dimensions. The horizontal 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 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 $v_1 = Pr(x_{i,j} = 1|x_{i-1,j} = 1) = Pr(x_{i,j} = 1|x_{i+1,j} = 1)$.

Though relatively simple, this model allows us to take into account spatial correlations previously ignored. Though correlation between a greater number of pixels exists, the nearest neighbor Markov model captures the strongest inter-pixel correlations. As we shall see, the result is a significant performance improvements over the 1-D model.

[2]The source coding with side information construction of [5], 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.

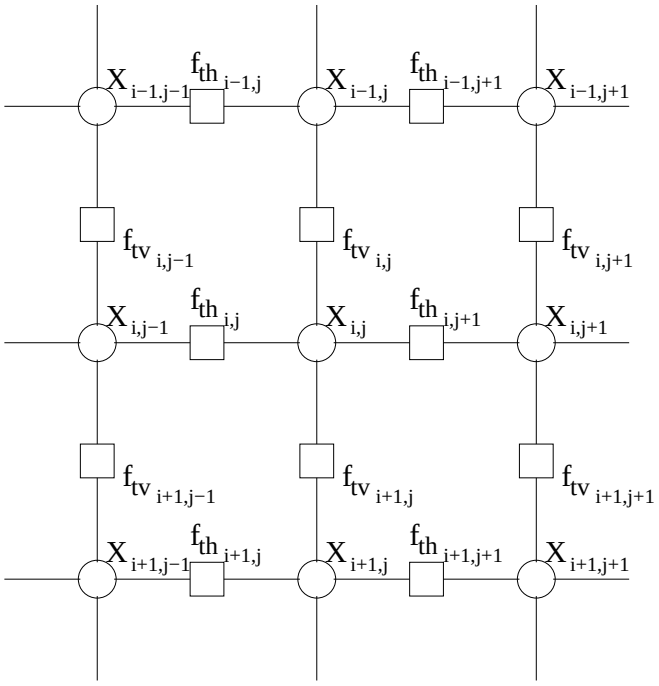

Fig. 3. *A factor graph for the spatial source model considered in this paper. The circles and squares on the grid represent the Markov field model.*

3. ENCODER AND DECODER

In this section we present a practical encoder and decoder for compressing encrypted images. We begin by assuming that full knowledge of the source statistics (p, h_0, h_1, v_0, v_1) is available to both encoder and decoder, and then relax this assumption. 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 [7]. In particular, the design of the transform matrix is based (with a modification discussed below) on LDPC codes [8].

The decoder operates by running belief propagation over the factor graph [6]. We thus proceed by describing the appropriate factor graph. The graphical model consists of three components connected together; the models 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. 4. Since we consider only stream ciphers here, we can model the encryption process as $y_{i,j} = x_{i,j} \oplus k_{i,j}$, where $y_{i,j}$ is the cipher-text, $k_{i,j}$ is the bits of the

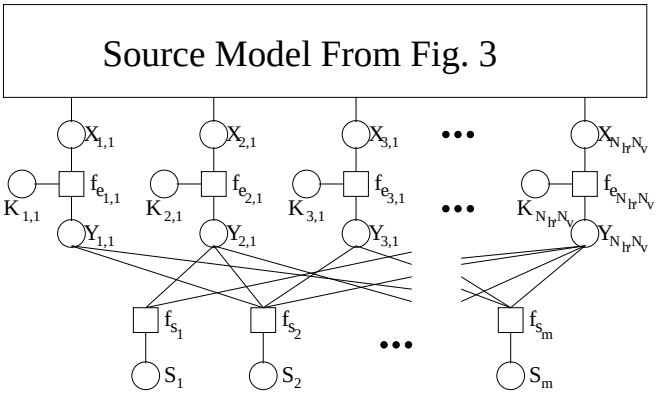

Fig. 4. *The full graphical model for compressing encrypted spatially correlated sources. The model consists of the source model on top (abstracted here but shown in detail in Fig. 3), the encryption model in the middle, and the code on the bottom.*

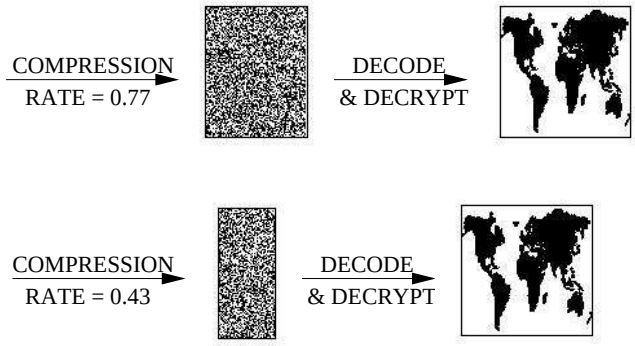

Fig. 5. *A comparison of the compressed bits and reconstructed image using the 1-D memory model from [4] and the 2-D memory model presented here. The 1-D model compressed the encrypted data to* $7,710,3,411$ *more bits than the* $4,299$ *bits used for the 2-D model. Clearly, the 2-D model achieves greater compression.*

key, and $\oplus$ indicates the exclusive-OR operation. We represent the constraint between these three variables in the graphical model with a square node labeled $f_{e_{i,j}}$. The circles representing the variables $x_{i,j}$, $k_{i,j}$, and $y_{i,j}$ are all connected to the encryption constraint $f_{e_{i,j}}$.

The code model consists of a representation of the linear transformation matrix $\mathbf{H}$, the cipher bits $y_{i,j}$, and the compressed bits s_i. In [7] 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. 4. 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. 4. The sum-product algorithm is an inference algorithm designed to be exact on trees. Although not exact on "loopy" graphs (such as the graph in Fig. 4), 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 the sparse linear transformation a portion of the output bits are designed to be exactly equal to the source bits. We refer to these as "doped" bits. The doped bits are our modification of the LDPC code based design. Typically between 30% and 50% of the compressed bits are doped bits. These bits are used in two ways. First, since these doped bits are known unambiguously at the decoder they anchor the iterative decoding process by catalyzing the process. Second, they provide a mechanism for estimating the statistics of the masked source. By selecting the doped bits to come in adjacent pairs, the decoder can empirically estimate the source statistics and use the estimates for decoding[3]. We demonstrate algorithm performance below.

4. RESULTS

As a demonstration, we compress the encrypted version of the binary image leftmost in Fig. 2. The unencrypted 100x100 binary image ($10,000$ bits) is a binary map of the globe. We use the doped bits in order to calculate the source statistics for use in the sum-product algorithm run at the decoder. This image is encrypted (right image in Fig. 2) by adding a pseudo-random Bernoulli-$1/2$ string (center image in Fig. 2).

In this example, our method compresses the encrypted data to $4,299$ bits, of which $2,019$ are doped bits. The decoder empirically estimates $(p, h_0, h_1, v_0, v_1) = (0.3935, 0.0594, 0.9132, 0.0420, 0.9295)$ and then reconstructs the original image using 81 iterations. The compressed bits and the reconstruction are presented in Fig. 5. For comparison, we present the 1-D memory model used in [4], where the encrypted image could only be compressed to $7,710$ bits. In that simulation, the image was reconstructed in 27 iterations[4]. The 2-D source model allows for greater compressibility.

To see the effects of the source model on the decoding, we present the estimates of the 2 decoders at the end of three iterations in Fig. 6. The 1-D decoder estimates can be seen to exhibit artifacts resulting from the north-south raster scanning. These artifacts are the several visible up and down lines. In contrast, such artifacts do not show up with the 2-D decoder. Instead, the estimates seem to result in "clumped" areas, areas

[3]To relax the assumption that the encoder also know the source statistics, feedback could be used by the decoder after estimating the source statistics [4].

[4]Study of the number of iterations required for convergence versus model structure is a part of ongoing work.

that grow from iteration to iteration. For both decoders, after a handful of iterations (typically under 10), these artifacts disappear.

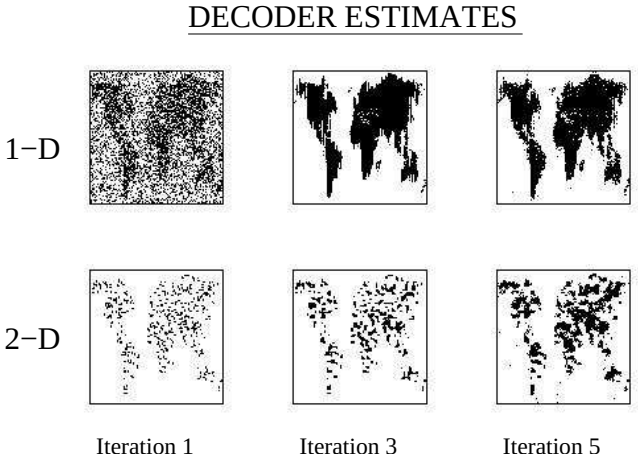

Fig. 6. *A comparison of the intermediate estimates obtained at the decoder. On the top are the estimates obtained from the 1-D model of [4] while on the bottom are estimates obtained from the 2-D model. The estimates from the 1-D model exhibit raster scan artifacts. In contrast, the 2-D model decoder exhibits a localized clumping nature.*

5. CONCLUSIONS & FUTURE DIRECTIONS

In this paper we have used a 2-D source model for spatially correlated data sources. We have leveraged this algorithm to allow us to efficiently compress encrypted images. Further, we presented the results of compressing an encrypted binary image without access to the source statistics at the decoder. We showed how the 2-D source model allows greater compression gains than the 1-D source model.

This work naturally suggests an extension to gray-scale and other larger-alphabet images. A first approach is to break an image up into a series of bit-planes where each bit-plane represents all the bits of equal significance in the binary expansion of the pixel values. Image structure is typically highly concentrated in the most significant bit-planes though. As a result, little compression gain is available with this approach. Accurate image models are necessary to be able to achieve significant gains when compressing encrypted data.

Modeling images presents several new challenges though. Most existing image compression algorithms are based on transforms, e.g., the DCT (Discrete Cosine Transform) in JPEG. Transforms aim to convert the image into a domain where it may be represented with only a few coefficients. Using a bit-wise stream cipher though, it becomes impossible to consider transforms since encryption is a non-linear process. In contrast, image coders which use pixel domain models use highly non-stationary predictors. For example, JPEG-LS (lossless) compresses each pixel based on 4 of the adjacent pixels. Since this data is unavailable when the image is encrypted, application is not straightforward. Instead, compression of encrypted gray-scale image will require greater use of the doped bits and other learning techniques.

By contrast, encrypted video offers advantages unavailable to single encrypted images. As an example, consider 3 frames of video. At the decoder, after the first 2 frames are decoded, the third frame can be estimated from the two previous frames with high reliability by considering "motion" models. Since the difference between this estimate and the actual frame is likely to be small, compression gains could be significant. Temporal dependance (as with most popular video coders) may offer greater compressibility then the spatial dependance considered in this paper, and is a promising area of future study.

6. REFERENCES

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

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

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

[4] 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.

[5] 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.

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

[7] 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.*, Oct. 2002, pp. 576–585, Submitted To IEEE Trans. Commun.

[8] R. G. Gallager, *Low Density Parity Check Codes*, Ph.D. thesis, MIT, Cambridge, MA, 1963.