16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

LOSSLESS COMPRESSION OF ENCRYPTED GREY-LEVEL AND COLOR IMAGES

Riccardo Lazzeretti, Mauro Barni

Department of Information Engineering (University of Siena)
Via Roma 56, 53100, Siena, Italy
email: lazzeretti2@unisi.it
web: http://www.dii.unisi.it/ vipp/

ABSTRACT

The feasibility of lossless compression of encrypted images has been recently demonstrated by relying on the analogy with source coding with side information at the decoder. However previous works only addressed the compression of bilevel images, namely sparse black and white images, with asymmetric probabilities of black and white pixels. In this paper we investigate the possibility of compressing encrypted grey level and color images, by decomposing them into bit-planes. A few approaches to exploit the spatial and cross-plane correlation among pixels are discussed, as well as the possibility of exploiting the correlation between color bands. Some experimental results are shown to evaluate the gap between the proposed solutions and the theoretically achievable performance.

1. INTRODUCTION

In many practical scenarios multimedia contents need both to be compressed and protected. Whereas the classical way to compress and protect data requires that data are first compressed end then encrypted, in some cases it may be desirable to operate the other way a round. This is the case, for example of an encrypted content that is transmitted without compression through the internet, but at a certain point needs to be compressed in order to match the characteristics of the transmission link. If the node is not trusted it may not be provided with the encryption key, then compression of the encrypted content is required. Though at first sight this seems an impossible task, in [1] Johnson et al. showed that by relying on the theory of source coding with side information (SCSI) at the decoder, it is indeed possible to compress the encrypted data, without supplying the key to the encoder and obtaining in theory the same results that would have been obtained by compressing the non-encrypted bitstream.

In addition to the theoretical analysis, Johnson et al. [1, 2] propose a practical approach to the lossless compression of encrypted black and white images[1]. In this paper we consider the extension of the work in [1, 2] to the case of lossless compression of grey level and color images. As it will be shown such an extension is not trivial, and several practical problems need to be tackled with. In a nutshell, we suggest to apply the bilevel algorithm by Johnson et al. to the image bit planes, by taking care to exploit the spatial correlation between pixels, and that between adjacent bit planes. With regard to color images, the correlation between different color bands is taken into account either by explicitly evaluating and exploiting the cross-band correlation or by first decorrelating the color bands and then applying the grey level algorithm separately to the single image bands.

The performance of the proposed schemes are evaluated experimentally on a few test images and compared with the theoretically achievable compression rates.

[1] In [1] the lossy compression of an i.i.d. Gaussian sequence is also analyzed, however we focus only on the lossless case.

2. CODING ENCRYPTED BLACK/WHITE IMAGES

In order to explain the principles that make the compression of an encrypted image possible, let us consider two correlated information sources X and Y. It is known that the rate required to represent each source exactly is $R_X \geq H(X)$ for source X and $R_Y \geq H(Y)$ for source Y, where $H()$ is the source entropy. At the same time, if the dependency between X and Y is exploited by a joint encoder, the total rate is lower-bounded by the joint entropy, i.e. $R_{tot} \geq H(X,Y)$. Slepian and Wolf showed [3] that the same asymptotic rate can be achieved by two separate encoders as long as the two coded streams are decoded jointly. For instance Y could be coded losslessly without knowledge of X while X could be coded at rate $H(X|Y)$, and recovered at the receiver with vanishing error probability. The *trick* is to use a joint decoder that decodes X by relying on the side information Y previously recovered. A similar result holds in the case of lossy coding [4]. To illustrate how SCSI can be used to enable signal compression directly in the encrypted domain, let Y be the secret key used by the cryptosystem, and X the encrypted signal. Let assume that the hypotheses behind the SCSI theory are verified. An encoder that only has access to the encrypted version of the signal can code it at the same rate obtainable by a coder that can operate on the non-encrypted signal. To recover the original signal, the decoder must have access to the side information (the secret key). This is the case, for example, of a distribution system where coding (or transcoding) has to be performed at intermediate, non-trusted, nodes, whereas decoding is performed by a trusted end-user.

In [1] Johnson et al. proposed a practical scheme (Fig. 1) to compress an encrypted stream with performance close to the theoretical results.

The image is scanned by rows and then encrypted by xoring it with a unique pseudorandom Bernoulli (1/2) key. The entropy of the encrypted stream conditioned to the key (the side information) is equal to the entropy of the plain image. If the percentage of black (white) pixels in the image is significantly lower than 1/2, compression is possible even if the encoder does not know the encryption key. By following the Slepian-Wolf coding paradigm, the parity check matrix of a linear channel code is applied to the encrypted stream whose compression is obtained by storing the syndrome instead of the bit plane itself. To be specific, if a code, characterized by a parity check matrix of size $(n-k) \times n$, is used to compress

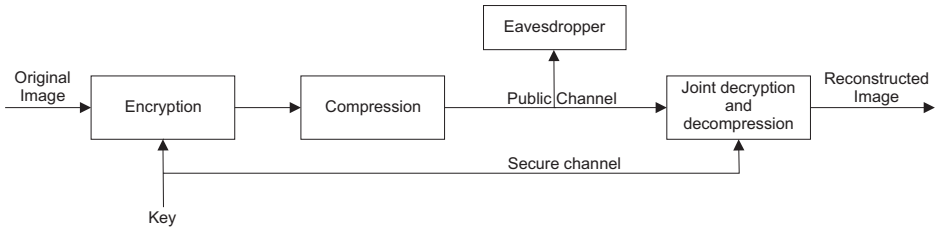


Figure 1: Scheme to compress encrypted images.

a stream of length n, a rate equal to $(n-k)/n$ is obtained and reconstruction is possible if $(n-k)/n$ is greater than or equal to the plain stream entropy. The decoder sees the key as a noisy version of the encrypted stream obtained by transmitting it over a binary symmetric channel and its goal is to find the nearest codeword to the key among those having the received syndrome. In order to approach the theoretical limit, a capacity achieving channel code needs to be used. In [1] the authors use LDPC codes (Low Density Parity Check codes) [5, 6] which are known to be very closed to the capacity for a BSC (Binary Symmetric Channel) and for which an efficient decoding algorithm exists based on belief propagation (BP).

To operate, the decoder needs to know the key, the syndrome and the conditional probabilities of the encrypted stream given the key. These probabilities are exploited by a modified BP algorithm to recover the encrypted sequence. Belief propagation is an iterative algorithm that converges exactly over trees, and performs quite well over sparsely loopy graphs, such as Tanner graphs representing LDPC codes [7]. Belief propagation needs some adaptations to work with non-null syndromes. Specifically each i^{th} check node update rule has to be modified to incorporate the knowledge of the i^{th} bit of the syndrome. The value of the key and the correlation between the key and the encrypted stream are used as a priori information to initialize the marginal distributions of the data nodes.

In [2], Shonberg et al. the above principles are extended so to take into account the spatial correlation of black and white images. Specifically, the authors model the images like Markov random fields and exploit the correlation between adjacent pixels. In this paper we use a different approach to exploit the spatial correlation, in addition we extend the analysis to grey and color images.

3. GREY-SCALE IMAGE REPRESENTATION

The extension of the algorithm described in the previous section to the case of grey level images is not trivial. Its direct extension in fact would require the use of a multilevel version of LDPC codes and the corresponding BP-based decoder. Unluckily, such extension does not exist, hence we need to decompose the image into a binary stream.

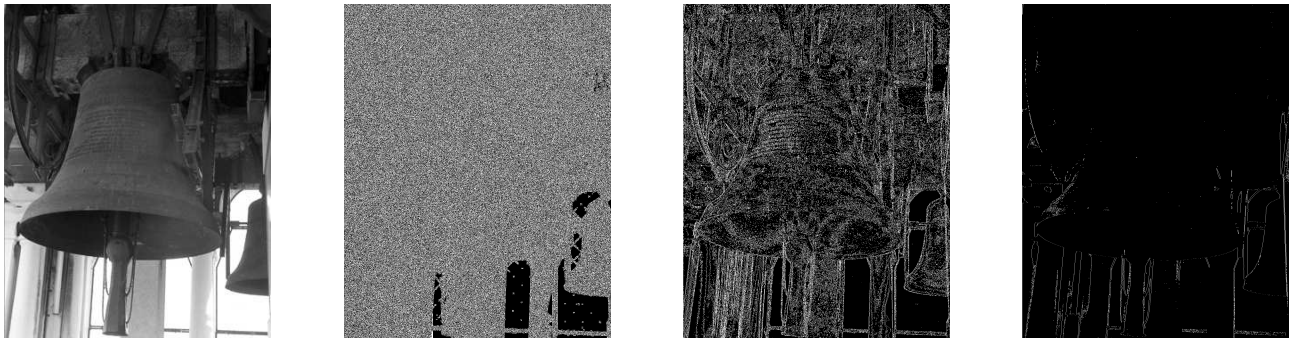
The easiest way to proceed is to subdivide the image into bit-planes and consider each of them as a black/white image. A problem with this approach is that bit-planes (and the whole image) can not be modeled as memoryless sources. As a matter of fact, by analyzing the bit-planes, it is easy to realize that for each of them the average percentage of white and black pixels is very close to 50%, hence no compression gain has to be expected unless the spatial correlation among pixels is taken into account. For this reason we suggest that before encryption the image is stored in such a way that the spatial correlation between bit-planes is (at least partially) removed.

A simple and fast way to remove the spatial correlation from the bit planes consists in scanning each row of the bit-planes from left to right and computing the xor between each bit and the previous one. In this way the vertical edges are extracted (see figure 2). The original bit plane can be easily recovered by xor-ing again the bit-planes along the rows. Of course, in the least significant bit planes the gain is almost null, however in the most significant bit-planes the gain is very large.

A more effective way to remove the spatial correlation is to act directly on the grey level values, i.e. scan from left to right sequentially each row of the image and predict the current pixel value as the average of the four adjacent pixels in a causal neighborhood. The pixel values are then replaced by the prediction error. A weighted mean can be considered to obtain better results, if we consider that the $(i-1,j)$ and $(i,j-1)$ pixels are at distance one from the (i,j) pixel, the $(i-1,j-1)$ and $(i-1,j+1)$ have distance $\sqrt{2}$. The image with the prediction error is then decomposed into bit-planes.

A problem with this approach is that the prediction error can assume value in the range [-255, 255], therefore requiring, at least in principle 9 bit-planes. To avoid this problem, we can observe that, since the mean is calculated on known pixels, we know that the difference assumes value in the range [-mean, 255-mean]. If $255-mean$ is greater than 127, the interval [128, 255-mean] can be mapped in the interval [-127, -mean-1]. The value 128 becomes $-mean-1$, the value 129 becomes $-mean-2$, and so on. We can proceed similarly if $-mean$ is less than -127. In this way it is possible to represent the differences by using only 8 bit-planes: one bit for the sign and seven bits for the module. The differences usually assume low values and the most significant bits are equal to 0. Rarely the differences are greater then 127, therefore usually there is no necessity to map. The efficiency of these methods will be shown in the result section. Most of the differences have low values and need few bits for their representation. As a consequence the higher bit-planes have many bits equal to 0, and allows large compression gains. Note that the probabilities of 0 and 1 in each bit plane must be stored in clear together with the encrypted bitplanes to allow the subsequent compression.

Processing each bit plane independently does not allow to exploit the correlation between bit-planes. As a matter of fact the conditional entropy between adjacent bitplanes is $H(p_i|p_{i+1}) \leq H(p_i)$. If spatial decorrelation is achieved by means of pixel prediction the sign bitplane is the most noisy, hence it has to be considered as the LSB bit-plane. To take into account the correlation among bitplanes the probabilities



(a) Original image (b) Plane 1 (LSB) (c) Plane 5 (d) Plane 8 (MSB)

Figure 2: Bit planes after xor-based decorrelation.

of a pixel value in a plane conditioned to the corresponding values in the upper bit plane are stored in the file. Such probabilities are exploited by the encoder to calculate the conditional entropy of each bit plane conditioned to the upper plane and decide the compression rate to be applied to each plane. At the decoder, the probabilities are fed into the BP algorithm as a-priori probabilities and used to decode the encrypted and compressed sequence (see section 5).

4. COLOR CASE

The simplest way to deal with color images is to treat each band as a gray-scale image. The problem is that by doing so we neglect the correlation among bands. To get a better compression rate, the correlation of a bitplane with the same bit plane of one of other color bands could be considered instead of the correlation with the upper bit plane. In practice, we select a reference color band, which is coded as a grey level image, and for the other bands the probabilities of each pixel value conditioned to the value of the same pixels in the same bit plane of the reference band are stored. We decided to use the green band as the reference band.

An alternative approach, consists in passing from the RGB color space to a color space whose coordinates are independent of each other (for instance the *YCbCr* color space). Unfortunately, the conversion from the RGB color space to *YCbCr* coordinates (or other coordinates ensuring band decorrelation) is an affine operation with non-integer coefficients, hence compromising the lossless nature of the scheme. To avoid this problem we used a reversible integer approximation of the correct transform in a way similar to that described in [8].

Let x be the vector that identifies a pixel in the RGB space. Its representation in the target color space is computed by the affine transformation $y = A * x + b$, where A is the conversion matrix and b a translation vector. We compute the LU factorization of A. L is a row permutation of a lower triangular matrix having identity elements on the principal diagonal. U is an upper triangular matrix, such that $L * U = A$. We compute a new matrix $\overline{U} = D * U$, where D is the diagonal matrix obtained by the inverse of the diagonal elements of U ($D = diag(U[1,1]^{-1}, U[2,2]^{-1}, U[3,3]^{-1})$). $\overline{U}$ is an upper triangular matrix having identity elements on its principal diagonal. We compute the new representation of y as $\overline{y} = L * \overline{U} * x + \overline{b}$ where $\overline{b}$ is a translation vector, not necessarily equal to b, computed to ensure that all the possible $\overline{y}$ assume positive values. This allows to compute the transformation and the inverse transformation without errors, but we can not use only 8 bits to represent each element of $\overline{y}$ and for this reason each band representation needs 9 bits.

Let us now analyze in more details the *YCbCr* transformation. This transformation is defined by the matrix

$$\mathbf{A} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix}.$$

Applying the LU decomposition we obtain

$$\mathbf{L} = \begin{bmatrix} 0.4862 & 1 & 0 \\ -0.2392 & -0.4921 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 0.6150 & -0.5150 & -0.1 \\ 0 & 0.8374 & 0.1626 \\ 0 & 0 & 0.4921 \end{bmatrix}$$

From the U matrix we construct

$$\mathbf{D} = \begin{bmatrix} 1.6260 & 0 & 0 \\ 0 & 1.1942 & 0 \\ 0 & 0 & 2.0321 \end{bmatrix}$$

to get

$$\overline{\mathbf{U}} = \begin{bmatrix} 1 & -0.8374 & -0.1626 \\ 0 & 1 & 0.1942 \\ 0 & 0 & 1 \end{bmatrix}.$$

Practically, given a pixel x, $\overline{y}$ can be computed as:

$$\begin{aligned} x_1[1] &= x[1] + round(-0.0.8374 * x[2] - 0.1626 * x[3]) \\ x_1[2] &= x[2] + round(0.1942 * x[3]) \\ x_1[3] &= x[3] \end{aligned}$$

$$\begin{aligned} x_2[1] &= x_1[2] + round(0.4826 * x_1[1]) \\ x_2[2] &= x_1[3] + round(-0.2392 * x_1[1] - 0.4921 * x_1[2]) \\ x_2[3] &= x_1[3] \end{aligned}$$

$$\overline{y}[1] = x_2[1],\ \overline{y}[2] = x_2[2] + 158,\ \overline{y}[3] = x_2[3] + 255$$

To compute the representation $\bar{y}$, a constant has been added to each element of the vector x_2 to obtain only positive values. From the transformed coordinates we can go back to the original pixel values by applying the same procedure in a reverse way. The rounding error introduced in the direct transformation is corrected by the inverse transformation. We have tested over all the possible values that the RGB vector can assume to be sure that only 9 bits are required to each band.

Both xor-based and prediction-based spatial decorrelation can be applied to the modified YCbCr, extending them to 9 bitplanes. Due to the low correlation between the *YCbCr* color bands, each band can now be treated independently as in the grey-scale case.

5. IMPLEMENTATION DETAILS

To encode, decode and decrypt we follow the scheme proposed by Johnson et al. with some important differences. Let us start with the encoder. In theory the achievable rate depends only on the source entropy, however for this to be possible an ideal channel coding step would be required. In our algorithm such a step is implemented by means of LDPC codes. Each LDPC code is characterized by a matrix having as many rows as the bit length of the bitplanes and as many columns as the length of the syndrome, both quantities should be adapted to the bit plane entropy and its size. In order to limit the number of matrices used by the encoder and the decoder, we decided to fix the number of the rows. This number needs to be large for a good performance of the code, hence we chose to work with matrices having 10^5 rows and the bitplanes are subdivided into blocks of 10^5 pixels. A possible choice is to subdivided them in rectangles 400 bit wide and 250 bit high. Clearly the probabilities and the entropies have to be calculated over each block and not over the entire bitplanes. A similar problem arises because of the impossibility of adapting the code length to the entropy of the block at hand[2]. Then we have to consider a finite number of rate values and construct a corresponding, fixed set of LDPC matrices. Each block will be compressed at the nearest available rate greater then its entropy. The more the available rates the better the compression rate (at the expense of simplicity).

The goodness of an LDPC code is related to the absence of cycles in the Tanner graph associated to it. Building codes without cycles of size 4 is simple, but finding and correcting cycles with greater length is difficult. Because of the possible presence of these cycles, the BP algorithm may not convergence. In practice, LDPC codes present an error region near to the maximum entropy they can handle. To avoid this problem, we pay attention to ensure that this critical convergence region is avoided. In practice we are using 4-long-cycles-free LDPC codes of length 100000 bits and compression rates 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. Because of these solutions, we can not exactly achieve a compression rate equal to the ideal entropy.

Moreover the encoder needs to know some information to work. Prior to encryption, it is necessary to compute the conditional probability to have 0 or 1 conditioned to the value that the reference bitplane assumes in the same position. For the first bitplane no reference is available hence we simply compute the probabilities of 0 and 1. This information has to be transmitted with the encrypted bit-planes. The encrypter creates a file composed by an header and a sequence of descriptor/block pairs. The header contains information about the image (height, width, color or gray-scale...). Each descriptor contains the probability values of the associated encrypted block, from which the encoder can calculate the correct compression rate. The format of the bit stream prior to encryption (and compression) is reported in figure 3. As we said only the block data is encrypted and then compressed.

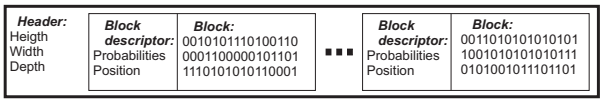


Figure 3: Scheme of the file containing the encrypted image.

Let us now analyze the decoder. Decoding and decrypting are performed jointly. The decoder uses the BP algorithm. Let $r[i]$ be the i^{th} bit of the reference bitplane, $k[i]$ the i^{th} bit of the key and $d[i]$ the i^{th} bit of the target plane. The i^{th} data node of the Tanner Graph associated to the LDPC code must be initialized with the a priori information $inf[i]$, that is:

$$inf[i] = a[i] * log(\frac{p(d[i]=1|r[i])}{1-p(d[i]=1|r[i])}$$

where

$$a[i] = \begin{cases} 1 & \text{if } k[i]=1 \\ -1 & \text{if } k[i]=0 \end{cases}$$

The first received bitplane (generally the 8^{th} bitplane of a grey-scale image or the 8^{th} bitplane of the green band of a color image) is decoded without a reference bitplane and the conditional probabilities $p(d[i]=1|r[i])$ are replaced by $p(d[i]=1)$. Each decoded bitplane has to be decrypted immediately to decode the subsequent bitplanes.

5.1 Security issues

The theoretical security of the scheme in figure 1 has been proved in [1]. For a perfectly secure system the key must be as long as the image. This is clearly impractical, hence in our system the key is produced by a pseudorandom number generator whose starting seed is transmitted to the decoder through a secure channel. Particular care must be given to the file header. This portion of the file contains sensible information that can help a pirate to decrypt the image. Ciphering this portions of the file with an encrypting algorithm increases the security. Unfortunately this is not possible because the encoder needs the probability values.

A possible solution could be to avoid encrypting the probabilities, insert in the descriptor the position of the block in the image, previously encrypted, and shuffle the descriptor and block pairs, so that the encoder can still compress each block since it can be done without any reference to the other blocks. At the same time, the pirate will learn the conditional probabilities without knowing the image portion the probabilities are conditioned to.

In [2] a different approach is used, since the characteristics of the image are estimated from doped bits. In this way a higher level of security is reached, however the system presented in [2] works only in the presence of a feedback channel. This is not always possible, for instance it decoding is

[2]Whereas with other coding strategies, e.g. turbo coding, it is possible to adapt the rate of the code on the fly, e.g. by means of proper puncturing techniques, this is not easy to do with LDPC codes.

not performed immediately after transmission, since in this case the source or the encoder may not have yet the image available.

6. EXPERIMENTAL RESULTS

We carried out several tests over grey-scale and color images. Here we propose the results obtained on a small set of ten grey-scale and color images.

Table 1 summarizes the results that we found for the grey level case. The table gives the conditional entropy of each bit plane (where conditioning is made with respect to the superior bit plane) and the actual coding rate achieved on each plane. Both xor- and prediction-based spatial decorrelation are considered.

The overall theoretical and practical coding rates are also given. As it can be seen a significant compression is obtained even by working in the encrypted domain, though a certain gap with respect to the ideal case exists.

Plane	Theoretical bit/rate with xor-based decorrelation	Obtained bit/rate with xor-based decorrelation	Theoretical bit/rate with prediction-based decorrelation	Obtained bit/rate with prediction-based decorrelation
1	0.9962	1.0000	0.8995	0.9889
2	0.9909	1.0000	0.9902	1.0000
3	0.9276	0.9889	0.9598	1.0000
4	0.7395	0.9667	0.7389	0.9444
5	0.5081	0.6833	0.2934	0.4111
6	0.3411	0.4778	0.1082	0.2278
7	0.2781	0.4000	0.0505	0.2000
8	0.0852	0.2056	0.0033	0.1889
Total	4.8668	5.7222	4.0439	4.9611

Table 1: Theoretical and practical bit/rates with xor- and prediction-based spatial decorrelation.

As to color images, table 2 shows the rate in bit/pixel that can be theoretically reached and the actual rate achieved by our system, both working with RGB and *YCbCr* coordinates.

	Xor-based decorrelaion		Prediction-based decorrelation	
Repres.	Theoretical	Obtained	Theoretical	Obtained
RGB	16.587	18.769	13.759	16.513
YCbCr	15.094	18.044	13.532	16.431

Table 2: Theoretical and practical bitrates (bit/pixel) reached with xor- and prediction-based decorrelation in RGB and YCbCr representations.

The results described above should be compared with the true entropy of the images (note that the tables report the plane by plane entropies whose sum is surely higher than the true image entropy, since it fails to remove completely spatial and cross-plane correlation). This is not an easy task since the actual entropy of the images is not known. In order to get a rough idea of the effectiveness of our schemes, we compared the performances of grey-level images with those of a general purpose entropy coder (the popular WinRar routine), an image-oriented lossless compression algorithm (JPEG-LS) and an estimation of the image entropy calculated as the multilevel entropy of the prediction error, where the simple prediction scheme described in section 3. A summary of the results we have got are reported in table 3.

WinRar	JPEG-LS	Estimation
4.224	3.503	3.920

Table 3: Comparison with WinRar compression, an image-oriented lossless compression algorithm (JPEG-LS) and an estimation of the image entropy calculated as the multilevel entropy of the prediction error.

As it can be seen the results we obtained are promising, even if the gap to state-of-the-art compression in the plain domain is still large.

7. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have introduced a few techniques to compress encrypted color and grey-level images. The best theoretical results are obtained by transforming color images in an approximated *YCbCr* domain. As to spatial decorrelation, working on the prediction error gives much better results, however the xor-based algorithm may lead to an interesting generalization of the proposed scheme towards lossy compression. It is only needed to erase the lower encrypted bitplanes to reduce the bit rate while keeping the quality of the reconstructed image acceptable (this is not possible when we work with the prediction error since such an error is computed before splitting the image into bitplanes).

REFERENCES

[1] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg and K. Ramchandran, "On Compressing Encrypted Data" *IEEE Tranaction on Signal Processing* vol.52 no.10 pp. 2992−3006, October 2004.

[2] D. Schonberg, S. Draper and K. Ramchandran, "On Compression of Encrypted Images" *Image Processing, 2006 IEEE International Conference on* pp. 269−272, October 2006.

[3] S.Slepian and J.K. Wolf, "A coding theorem for multiple access channels with correlated sources". *Bell Syst. Tech. J.*, 52:1037-1076, 1973.

[4] A.Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder", *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 1-10, Jan. 1976.

[5] M.J. Wainwright. "Sparse Graph Codes for Side Information and Binning", *IEEE Signal Processing Magazine*, vol. 24 no.5, pp. 47-57, 2007.

[6] R. G. Gallager. "Low density parity check codes", *IEEE Transaction on Information Theory*, vol. IT-8, pp. 21-28, January 1962.

[7] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498-519, Feb. 2001.

[8] Pei, S.C. and Ding, J.J. "Reversible Integer Color Transform with Bit-Constraint" *Image Processing, 2005. ICIP 2005. IEEE Int. Conf. on*, vol. 3, 2005.