Selective Video Encryption Of A Distributed Coded Bitstream Using LDPC Codes

Hwayoung Um and Edward J. Delp

Video and Image Processing Laboratory (VIPER)
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana USA

ABSTRACT

Selective encryption is a technique that is used to minimize computational complexity or enable system functionality by only encrypting a portion of a compressed bitstream while still achieving reasonable security. For selective encryption to work, we need to rely not only on the beneficial effects of redundancy reduction, but also on the characteristics of the compression algorithm to concentrate important data representing the source in a relatively small fraction of the compressed bitstream. These important elements of the compressed data become candidates for selective encryption. In this paper, we combine encryption and distributed video source coding to consider the choices of which types of bits are most effective for selective encryption of a video sequence that has been compressed using a distributed source coding method based on LDPC codes. Instead of encrypting the entire video stream bit by bit, we encrypt only the highly sensitive bits. By combining the compression and encryption tasks and thus reducing the number of bits encrypted, we can achieve a reduction in system complexity.

Keywords: Low Complexity Source Coding, Selective Encryption, LDPC Code, Sensor Networks

1. INTRODUCTION

Many applications, such as satellite television, video conferencing, medical and military imaging systems, require reliable security in storage and transmission. In recent years many potable devices, such as mobile telephones and personal digital assistants (PDAs), have the capability of exchanging multimedia messages. Such terminals increasingly include low to moderate resolution color displays, cameras, and moderate native processing power. However, battery size and battery life is always a concern for these mobile devices. Wireless systems are also limited by bandwidth and mobile terminal resources.

To meet the security and privacy needs in various applications, encryption of image and video data is necessary. The security of multimedia data is usually provided by a combination of encryption and data hiding methods [1–3]. Encryption transforms a plaintext message into a ciphertext message, which is unintelligible. Data hiding embeds information in the actual multimedia signal. The traditional way to secure multimedia applications is encrypt the entire multimedia bit stream using a secret key cryptography algorithm such as the Data Encryption Standard (DES) [4–6] or the newer Advanced Encryption Standard [7].

Selective encryption is a technique for encrypting parts of a compressed stream to minimize computational complexity [8]. Selective encryption is not a new idea. It has been proposed in several applications, especially in multimedia systems [9, 10]. Selective encryption can be used to reduce the power consumed by the encryption function for digital content when the content is protected by a digital rights management systems [1]. Since only parts of the bit stream are encrypted, selective encryption can also enable new system functionality such allowing previewing of content. For selective encryption to work, we need to rely not only on the beneficial effects of redundancy reduction described by Shannon [11], but also on a characteristics of the compression algorithm to concentrate important data relative to the original signal in a relatively small fraction of the compressed bitstream [10]. These important elements of the compressed data become candidates for selective encryption.

This work was supported by a grant from the Indiana 21st Century Research and Technology Fund. Address all correspondence to E. J. Delp, ace@ecn.purdue.edu.

Security, Steganography, and Watermarking of Multimedia Contents VIII, edited by Edward J. Delp III, Ping Wah Wong,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6072, 60721B, © 2006 SPIE-IS&T · 0277-786X/06/$15

Distributed source coding (DSC) [12–21] is a new paradigm for video compression, based on the work of Wyner-Ziv [22] and Slepian-Wolf [23]. DSC refers to separate encoding of correlated sources with joint decoding. DSC exploits the source statistics in the decoder and, hence, the encoder can be very simple, at the expense of a more complex decoder. The traditional approach in video compression of a complex encoder and simple decoder is essentially reversed. Although the theoretical result given by the Slepian-Wolf theorem has been known for more than 30 years, practical approaches to DSC did not appear until 1999. Since then, DSC has become a very active area of research [24].

The focus of this paper is to examine the use of selective encryption on a compressed bit stream that has been encoded using distributed source coding. We studied how selective encryption can achieve a high level of effectiveness. By this, we mean a strategy in which even a small fraction of encrypted bits can cause a video sequence to become useless if an attacker attempts to decode it without decrypting the secured portions. In this study, we examined which types of bits are most effective for selective encryption. Instead of encrypting the entire video sequence bit by bit, we encrypted only these highly sensitive bits.

2. DISTRIBUTED SOURCE CODING

The Slepian-Wolf theorem [23], which forms the basis of lossless distributed source coding (DSC), defines the achievable rate region when two physically separated and statistically correlated sources are independently encoded and jointly decoded. Distributed compression refers to the coding of two (or more) dependent random sequences, but with the special twist that a separate encoder is used for each. Each encoder sends a separate bit stream to a single decoder which may operate jointly on all incoming bit streams and thus exploit the statistical dependencies.

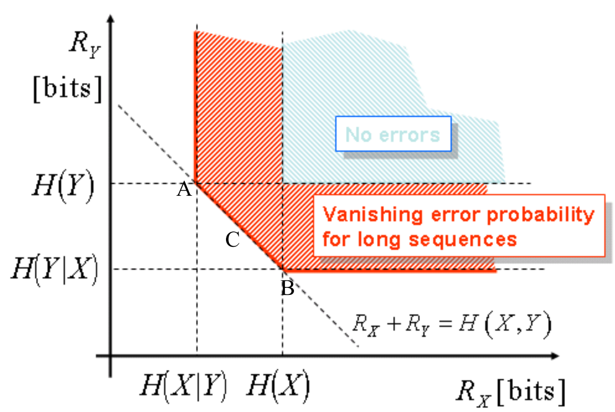

Figure 1. Slepian-Wolf rate region for two sources.

Consider two statistically independent identically distributed (i.i.d.) finite-alphabet random sequences **X** and **Y** (in Figure 1). With separate conventional entropy encoders and decoders, one can achieve $R_X \geq H(X)$, $R_Y \geq H(Y)$, where $H(X)$ and $H(Y)$ are the entropies of **X** and **Y**, respectively. However if two discrete signals, X and Y, are compressed using two independent encoders but are decoded by a joint decoder, then Slepian-Wolf theorem on distributed source coding states that even if the encoders are independent, the achievable rate region for probability of decoding error to approach zero is $R_X \geq H(X|Y)$, $R_Y \geq H(Y|X)$, and $R_x + R_y \geq H(X,Y)$ [25].

For practical Slepian-Wolf coding, we can try to design codes to approach the corner point **A** with $R_x + R_y = H(X|Y) + H(Y)$ in the Slepian-Wolf rate region of Figure 1. This is a problem of source coding of X with side information Y at the decoder. In other words, the signal X is compressed conventionally and sent at the full rate of $R(X) = H(X)$ and is recovered perfectly at the decoder, while the signal Y is compressed as close as possible to the Slepian-Wolf limit $H(Y|X)$ as shown in Figure 1, where the rate region for two arbitrarily correlated sources X and Y are displayed [15, 26]. It has been shown that the Slepian-Wolf boundary is achievable both asymptotically and with finite-length sequences [12]. Specifically, the corner points of the Slepian-Wolf boundary, where one source is losslessly available at the decoder (e.g. Y compressed to $H(Y)$ via a conventional entropy-compression method) and the other is maximally compressed utilizing the statistical correlation between the two sources (X compressed to $H(X|Y)$), may be modeled as an equivalent channel coding problem with decoder side information (SI) where the equivalent transmission channel is specified by the correlation of the sources. To get close to the theoretical limit, two key issues need to be resolved: (i) finding a capacity-approaching channel code for the equivalent transmission channel and (ii) bridging the practice and solution of channel coding with that of source coding. Although closely related, the two issues reflect different aspects of the DSC problem. While the former can take advantage of the rich literature available on channel coding, the latter is much less studied.

3. DISTRIBUTED VIDEO CODING BASED ON LDPC CODES

Low-density parity-check (LDPC) codes are a class of linear error-correcting codes [27]. Linear codes use a generator matrix $\mathbf{G}$ to map messages s to transmitted blocks $\mathbf{x}$, also known as codewords. They have an equivalent description in terms of a related parity-check matrix $\mathbf{H}$ with M rows and N columns. All codewords $\mathbf{x}$, of length N, satisfy $\mathbf{Hx} = 0$. Each row of $\mathbf{H}$ represents a parity check on a subset of the bits in $\mathbf{x}$; all these parity checks must be satisfied for $\mathbf{x}$ to be a codeword.

As their name suggests, low-density parity-check codes are defined in terms of parity-check matrices $\mathbf{H}$ that consist almost entirely of zeroes. Gallager [27] defined (n, p, q) LDPC codes to have a blocklength n and a parity-check matrix with exactly p ones per column and q ones per row, where $p \geq 3$. If all the rows are linearly independent then the rate of the code is $(q-p)/q$, otherwise the rate is $(n-p^{'})/n$ where $p^{'}$ is the dimension of the row space of $\mathbf{H}$.

We now define some basic notation we will use to describe low-density parity-check (LDPC) codes. LDPC codes are well represented by *bipartite graphs* in which one set of nodes, the *variable nodes*, corresponds to elements of a codeword (bits) and the other set of nodes, the *check nodes*, corresponds to the set of parity-check constraints which define the code. For a given length and a given degree distribution, we define an *ensemble* of codes by choosing edges, i.e., the connections between variable and check nodes, randomly. More precisely, we enumerate the edges emanating from the variable nodes in some arbitrary order and proceed in the same way with the edges emanating from the check nodes [28].

Definition 1: LDPC codes [27] are best described by their parity-check matrix $\mathbf{H}$ and the associated bipartite graph. The parity-check matrix $\mathbf{H}$ of a binary LDPC has a small number of ones. The way of spreading ones in $\mathbf{H}$ is described by the degree distribution polynomial $\lambda(x)$ and $\rho(x)$, which indicate the percentage of columns and rows of $\mathbf{H}$ respectively, with different Hamming weights. When both $\lambda(x)$ and $\rho(x)$ have only a single term, the LDPC code is *regular*, otherwise it is *irregular* [15].

Definition 2: The bipartite of an LDPC code is an equivalent representation of the parity-check matrix $\mathbf{H}$. Each column is represented with a *variable node* and each row with a *check node*. The graph has an *edge* between variable node j and check-node i if $H(i,j) = 1$ [15, 29].

In this paper, we used a distributed source coding method based on non-uniform LDPC coding [26] at the symmetric rates (point C of Figure 1), i.e., both of the encoders are compressed at the same rate. That is, $R_x = R_y = \frac{H(X,Y)}{2} = \frac{1}{2} + \frac{H(p)}{2}$, where $H(p) = H(X|Y) = -plog(p) - (1-p)log(1-p)$ and p is the crossover probability of $P[X \neq Y|X] = p$. We shall refer to this as a symmetric LDPC code. Here X and Y are assumed independent, identically distributed binary sequences of length k. X and Y are statistically dependent to each other and the dependency can be described by the conditional mass function $P[X_1|X_2]$. The correlation between X and Y can be modeled as the input and output of a binary symmetric channel with crossover probability of $P[X \neq Y|X] = p$. We assume that the length of the LDPC code of rate R is n.

3.1. Source Encoder

We will use the DSC coder described in [26] and shown in Figure 2 for compressing a video sequence. To use this method to encode video we will let $\mathbf{X}$ be one frame of video and $\mathbf{Y}$ be the next frame of video in a sequence. We will transmit the upper half of $\mathbf{X}$ and the lower half of $\mathbf{Y}$ along with parity bits associated with each frame. We will use the lower half of $\mathbf{Y}$ to reconstruct the lower half of $\mathbf{X}$ using the parity bits for both $\mathbf{X}$ and $\mathbf{Y}$. We will then do a similar operation to reconstruct the upper half of $\mathbf{Y}$. Our goal is to investigate which parts of the bit stream of this encoder needs to protected.

Using a linear binary (n,k) block code, such as a LDPC code, there are 2^{n-k} distinct syndromes, each indexing a set of 2^k binary words of length n. All sets are disjoint and in each set the Hamming distance properties of the original code are preserved, i.e., all codes have the same performance over the binary symmetric correlation channel. For the encoder, a sequence of input n bits is mapped into its corresponding syndrome $(n-k)$ bits. Thus, the compression ratio achieved with this scheme is n:(n-k) [16].

As shown in Figure 2, two half sets of each video frame are used to encode $\mathbf{X}$ and $\mathbf{Y}$, where $\mathbf{X}$ and $\mathbf{Y}$ represent even and odd number video frames respectively (the two video frames are correlated). The even video frame X is used an the input to a rate R_x systematic LDPC encoder. At the output of the encoder, the first half of the input video frame, x_1, and the corresponding parity check bits, p_1 are transmitted. This results in a source encoding rate of $R_{x1} = \frac{k/2+p_1}{k}$ bit per input bit. That is, the rate of the systematic LDPC code is equal to $R_x = \frac{k}{n} = \frac{1}{R_{x1}+1/2}$. A corresponding operation is performed on the odd video frame Y similar to that of the even video frame X. However, the second half of the video frame and the corresponding parity bits are used for this case. Since the compression rate of both video frames are the same, the rate of the systematic LDPC codes are identical. Hence only a single LDPC code is needed [26].

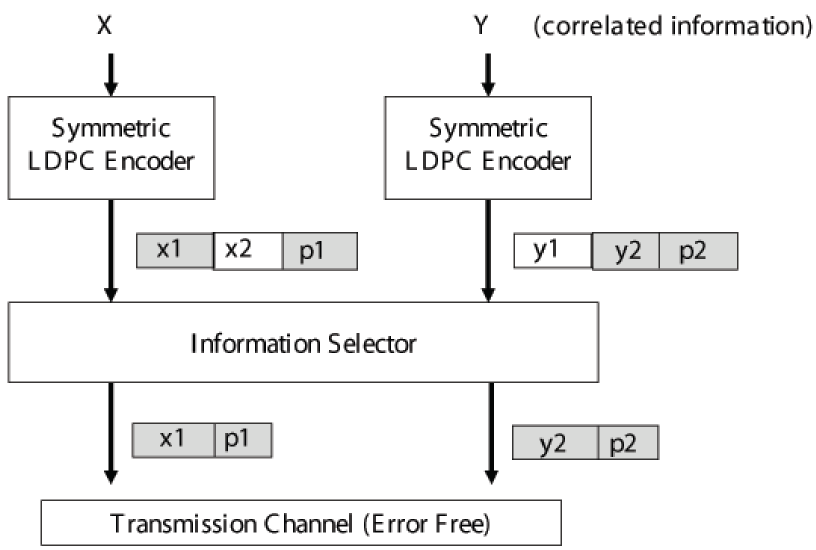

Figure 2. Symmetric LDPC Encoding.

3.2. Source Decoder

The decoder must estimate the n-length video frame $\mathbf{X}$ from its (n-k)-long syndromes and the side information, the half video frame of $\mathbf{Y}$. The transmitted codewords are decoded from the received data, the two half video frames and the parity bits, using the likelihood of the possible codewords. The likelihood of the possible codewords is the probability of receiving the data that was actually received if the codewords is question were the one that was sent. For decoding purposes, the most important issue is the relative likelihood for a bit to be 1 versus 0. This is captured by the likelihood ratio in favor of a 1, which is $P(data|bit = 1)/P(data|bit = 0)$.

Definition 3: For a Binary Symmetric Channel with error probability p, the likelihood ratio in favor of 1 bit is as follows: (a) if the received data was +1: $(1-p)/p$. (b) if the received data was −1: $p/(1-p)$.

The decoder of X has the first half of X perfectly, $x1$, (here we assume that the channel is error free). To construct the entire video frame for X, the decoder use the lower half of Y and the parity bits of X, p_1. The log likelihood ratios (LLRs) of all bits should be known in order to use the *message passing algorithm*, which will be described in the next section in detail. The LLRs of the parity bits and half video frames that passed through the channel are infinity. The lower half of Y is assumed to be the output of a binary symmetric channel (BSC) with cross over probability of p whose input is X. The LLRs of this fraction of the video frames are equal to $ln(\frac{1-p}{p})$. Then, by knowing the LLRs for all the bits, the message passing algorithm can decode the video frame X. The same process can be used to decode the video frame Y [26].

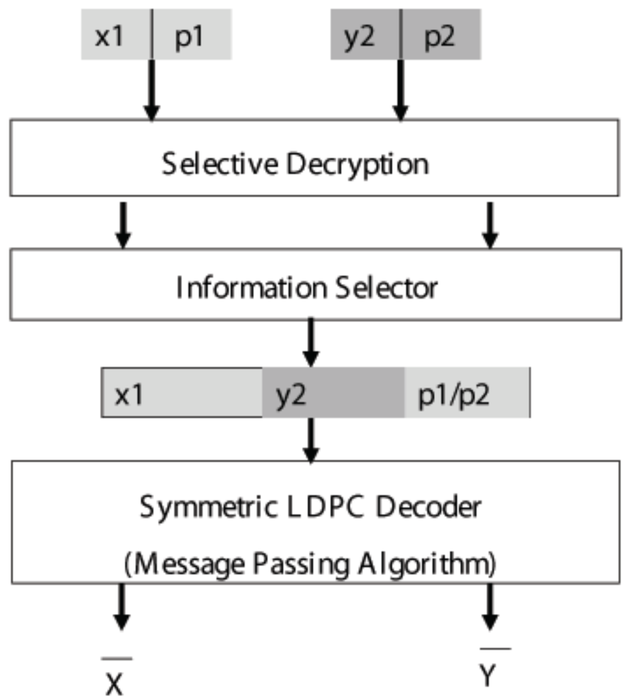

Figure 3. Symmetric LDPC Decoding.

3.3. LDPC Decoding Using The Message Passing Algorithm

The message passing algorithm is an efficient and iterative method used for decoding LDPC codes [27,29]. The algorithm is based on passing messages from the variable nodes to the check nodes and vice versa per iteration. It first collects the incoming messages, and then computes and sends out its own message.

The algorithm has two stages: *initialize* and *iterate*. Let $c = (c_1 c_2 \cdots c_N)$ be the codeword transmitted over the channel (channel input), $R = [r_1 r_2 \cdots r_N]$ be the received signal (noisy channel output), and δ^2 be the variance of the channel noise. Consider the Tanner graph of the parity-check matrix that defines the code. During initialization, each bit-node $1 \leq j \leq N$ computes its message $v_j = (2/\delta^2) \times r_j$ and sends it down to M_j, the set of all check-nodes connected to bit-node j. Each iteration has two different phases: check-node update and variable-node update. This phase updates the upward messages for each edge of the Tanner graph. A Tanner graph for a binary (n, k, d) code $\boldsymbol{C}$ is a bipartite graph whose adjacency matrix is the parity-check matrix of $\boldsymbol{C}$. The left part of the graph consists of *variable nodes* which correspond to symbol bits of the code, and the right part of the graph consists of *check nodes* which correspond to parity checks in the graph.

Check-node update. Assume that an edge connects check-node $1 \leq i \leq M$ and bit-node $1 \leq j \leq N$. The upward message corresponding to this edge $(u_{i,j})$ is a probability measure that indicates which value of c_j (c_j =1 or 0) satisfies parity-check i. This probability measure is computed based on the messages received from all the bit-nodes connected to check-node i, excluding bit-node j. Thus, during the check-node update, each check-node i computes its message using its received data, and sends the message up to N_i, the set of all the bitnodes connected to check-node i.

Variable-node update. This phase updates the downward messages for each edge of the Tanner graph. The downward message corresponding to the edge that connects check-node $1 \leq i \leq M$, and bit-node $1 \leq j \leq N$, is denoted by $v_{i,j}$. The downward message $v_{i,j}$ is a probability measure indicating if c_j is one or zero. This measure is computed based on the messages received from all the check-nodes connected to bit-node j, excluding check-node i. Thus, during the bit-node update, each bit-node j employs its received data to update its own message and then sends it down to all the check-nodes of the Tanner graph that are connected to it (M_j).

These messages iterate between bit-nodes and check-nodes to finally obtain the a posteriori log likelihood ratio (LLR), $\lambda_n = log(Pr(c_n = 1|R)/Pr(c_n = 0|R))$ for each bit-node n. The number of iterations is denoted by l_{max}. The algorithm converges when there are no considerable changes in the output, or when we are within the desired degree of accuracy. Practically, this happens after a constant number of iterations (usually no more than 10). Therefore, l_{max} is a constant. After convergence, we use λ_n to decode the sequence. Positive λ_n indicates that $Pr(c_n = 1|R) \geq Pr(c_n = 0|R)$, and so $c_n = 1$. Similarly, negative λ_n means c_n is zero [27, 29].

4. PROPOSED SELECTIVE VIDEO ENCRYPTION METHOD

In selective encryption, a bit stream is partially encrypted to minimize computational complexity or provide new functionalities for uses of the encrypted bit stream while at the same time providing "reasonable" security of the bit stream. One goal might be to provide additional error resilience in the case of a wireless network with packet losses and erasures.

The block diagram of our proposed selective encryption method for video compressed, using the LDPC-based DSC method described in the previous section, is shown in Figure 4. The video sequence, X, is first compressed with the LDPC-based DSC encoder. The seed K' is used as the input to a pseudo-random generator (PRG), whose output is denoted by K. If K is truly random, then the PRG forms a stream cipher. Recall that the output bitstream of the LDPC-based DSC encoder consists of two types of data, the half video frames (pixels) and the parity bits. The encoded bit stream, W, is partially encrypted by forming the bitwise binary sum $Z = E[X] = W \oplus K$ of parts of the compressed bit stream. Then, Z is transmitted over the channel. The adversary is assumed to be able to eavesdrop on the ciphertext Z. We assume that the seed has been transmitted to the decoder through a secure channel. By implementing an identical PRG, the decoder also has access to K. Our goal is to decode X, using the fact that K is available at the decoder and the half frame of Y, of the correlated video sequence, is available as side information. Because $Z = W \oplus K$, it follows that $W = Z \oplus K$.

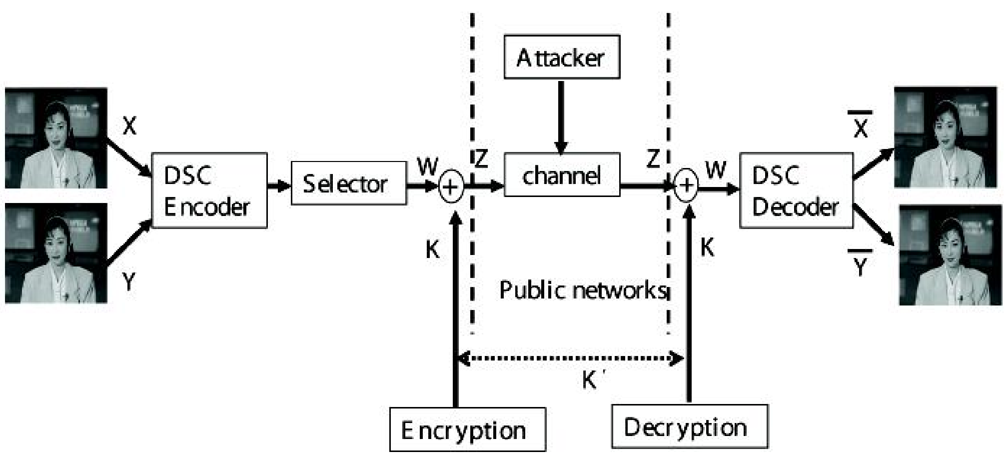

Figure 4. Diagram of the proposed selective encryption method.

In this scheme, the candidates for encryption are the half video frames (pixels) and the parity bits. Now we discuss the issue of the partial encryption for each candidate.

4.1. Parity Bits

Decoding can be done using only the parity check matrix defining the codewords, without reference to the generator matrix defining the mapping from the source messages to the codewords. Hence the parity bits must be encrypted. Otherwise an attacker can recover the original video frame directly. Below we describe how the parity bits are used by the decoder.

Assuming equal a priori probabilities for the codewords, the probability of correctly decoding an entire codeword is minimized by picking the codeword with the highest likelihood. One might instead wish to decode each bit to the value that is most probable. This minimizes the bit error rate, but is not in general guaranteed to lead a decoding for each block to the most probable complete codeword; indeed, the decoding may not be a codeword at all. Minimizing the bit error rate seems nevertheless to be the most sensible objective, unless block boundaries have some significance in a wider context. The begin, information about each bit of the codeword derived from the received data for that bit alone is expressed as a probability ratio, the probability of the bit being 1 divided by the probability of the bit being 0. This probability ratio is equal to the likelihood ratio for that bit, since 0 and 1 are assumed to be equally likely a priori. As the decoding algorithm progresses, these probability ratios will be modified to take account of information obtained from other bits, in conjunction with the requirement that the parity checks be satisfied. To avoid double counting of information, for every bit, the algorithm maintains a separate probability ratio for each parity check that bit participates in, giving the probability for that bit to be 1 versus 0 based only on information derived from other parity checks, along with the data received for the bit.

As indicated above, the parity bits are used to decide what are the correct information bits in the decoding process. This means that using encrypted parity bits for source decoding would render the decoded video useless because the decoder would generate the wrong codewords.

4.2. The Half Video Frames

The half video frames must be encrypted because they reveals parts of the original video. We consider each 8 bit pixel of the frame in the form of 8 bitplanes. Our approach is to encrypt a subset of the bitplanes, starting with the bitplane containing the most significant bit (MSB) of the pixel and increasing to the least significant bit (LSB) of the pixel. By doing this the encrypted pixels are less likely to show any of the original gray scale information. The minimal percentage of pixels to be encrypted is 12.5% when encrypting 1 bit. We increase the percentage of the pixels encrypted in steps of 12.5%.

5. SIMULATION RESULTS

Consider a single video frame (image) composed of $M \times N$ pixels (where M is the width and N the height of the image) where each image is in the YUV color space [30]. The YUV format is typically sub-sampled and for our work we will use the 4:1:1 format. We will also use Peak-Signal-to-Noise-Ratio (PSNR) for our measure of image quality.

In our simulation, we used QCIF video sequences with 176×144 pixels as shown in Figure 5. The video is compressed using the LDPC coder discussed in the previous section. Our selective encryption algorithm encrypts the parity bits and a subset of the bitplanes for each pixel, starting with the most significant bit (MSB). The encrypted bitplanes are transmitted as plaintext.

Figure 6 shows a reconstructed frame video after encrypting only the parity bits. We assumed that an attacker does not have access to the encryption key. Hence the attacker can access the unencrypted half video frames and reconstruct the video frames by combining the two half frames. The reconstructed video frames are very similar to the original frame. We note that the encryption of only the parity bits cannot guarantee the security of the video sequence.

Figure 7 shows four examples of reconstructed video frames after selectively encrypting MSBs of the pixels along with the parity bits. We encrypted the first MSB, the first two MSBs, the first four MSBs, and all 8 bits

Figure 5. Original Video Sequence: YUV 4:1:1 sub-sampled with 176×144 pixels.

(a) The upper half frame of X. (b) The lower half frame of Y. (c) The reconstructed video frame.

Figure 6. Results when only the parity bits are encrypted.

respectively. In the case of encrypting the MSB and the parity bits, structural information is still visible, but the encryption of two or more bits and the parity bits reveals no useful information in the reconstructed frames. The PSNR decreases steadily from 18dB to 9dB as we encrypt more MSBs.

(a) 1MSB and parity bits are encrypted, PSNR=18.18dB. (b) 2MSBs and parity bits are encrypted, PSNR=13.23dB. (c) 4MSBs and parity bits are encrypted, PSNR=10.82dB. (d) 8MSBs and parity bits are encrypted, PSNR=9.94dB

Figure 7. Visual examples of the selective encryption when MSBs and the parity bits are encrypted.

Figure 8 shows the case where the frame is reconstructed after the LSBs and the parity bits are encrypted. It shows that the PSNR decreases steadily from 117dB to 9dB as we encrypt more LSBs.

In these simulations, the rate of the systematic LDPC code is 1/2, $\frac{k}{n} = \frac{1}{2}$. We define the encryption ratio as the ratio of the number of encrypted bits $((i/8) \times (2/n) + (n-k))$ to the number of data bits $(2/n + (n-k))$. The percentage of encrypted bits are shown in Table 1.

The experiments indicate that at least 63% of the bits need to be encrypted. This is larger than has been reported for selective encryption methods for MPEG-2 [10] where the syntax of the bit stream has been exploited. Our results are also due to the relative simple scheme we used for distributed source coding.

(a) 1LSB and parity bits are encrypted, PSNR=117.98dB.
(b) 2LSBs and parity bits are encrypted, PSNR=98.49dB.
(c) 4LSBs and parity bits are encrypted, PSNR=82.08dB.
(d) 8LSBs and parity bits are encrypted, PSNR=9.94dB

Figure 8. Visual examples of the selective encryption when LSBs and the parity bits are encrypted.

Table 1. Encryption ratios.

Encryption methods	Encryption ratio
parity bits only	50%
1 bit and parity bits	56%
2 bits and parity bits	63%
4 bits and parity bits	75%
8 bits and parity bits	100%

6. CONCLUSION

In this paper, a framework for implementing selective video encryption for a LDPC-based distributed source coding method is proposed. We showed that the encryption of 63% or more bits reveals no useful information in the reconstructed video. Hence the proposed method has some advantages over conventional full data encryption with regard to complexity. We are investigating how the compressed bit stream can be exploited by imposing a syntax on the output of the DSC coder. We are also investigating other types of DSC coders.

REFERENCES

1. E. Lin, A. Eskicioglu, R. Lagendijk, and E. Delp, "Advances in digital video content protection," *Proceedings of the IEEE* **93**, pp. 171–183, 2005.
2. A. M. Eskicioglu and E. J. Delp, "An overview of multimedia content protection in consumer electronics devices," *Signal Processing : Image Communication* **16**, pp. 681–699, 2001.
3. A. M. Eskicioglu, J. Town, and E. J. Delp, "Security of digital entertainment content from creation to consumption," *Signal Processing : Image Communication* **18**, pp. 237–262, 2003.
4. "Data Encryption Standard(DES), National Bureau of Standards FIPS Publication 46," 1977.
5. "Data Encryption Standard(DES), National Bureau of Standards FIPS Publication 81," 1980.
6. D. R. Stinson, *Cryptography*, CRC Press LLC, Boca Raton, Florida, 1995.
7. "Advanced Encryption Standard(AES), National Bureau of Standards FIPS Publication 197," 2001.
8. L. Qiao and K. Nahrstedt, "Comparison of MPEG encryption algorithms," *Proceedings of the International Journal of Computers and Graphics, special issue: Data Security in Image Communication and Network* **28**, January 1998.
9. H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Transactions on signal processing* **48**, pp. 2439–2451, August 2000.
10. T. Lookabaugh and D. C. Sicker, "Selective encrytion for consumer applications," *IEEE Communications Magazine* , pp. 124–129, May 2004.
11. C. Shannon, "Coomunication theory of secrecy systems," *Bell Systems Techical Journal* **28**, pp. 656–715, October 1949.

12. S. Pradhan and K. Ramchandran, “Distributed source coding using syndromes (DISCUS): design and construction,” *Proceedings of the IEEE Data Compression Conference (DCC’99)*, March 1999.
13. R. Puri and K. Ramchandran, “PRISM: a reversed multimedia coding paradigm,” *Proceedings of the IEEE International Conference on Image Processing (ICIP 2003)*, **1**, pp. 617–620, Semtember 2003.
14. B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, “Distributed video coding,” *Proceedings of the IEEE* **93**, pp. 71–83, January 2005.
15. Z. Xiong, A. Liveris, and S. Cheng, “Distributed source coding for sensor networks,” *IEEE Signal Processing Megazine* , pp. 80–94, September 2004.
16. A. Liveris, Z. Xiong, and C. Georghiades, “Compression of binary sources with side information at the decoder using LDPC codes,” *IEEE Communication Letter* **6**, pp. 440–442, October 2002.
17. L. Liu, Y. Liu, and E. J. Delp, “Network-driven Wyner-Ziv video coding using forward prediction,” *Proceedings of the SPIE International Conference on Visual Communications and Image Processing*, pp. 641–651, (San Jose, CA), Januaray 2005.
18. L. Liu, P. Sabria, L. Torres, and E. J. Delp, “Error resilience in network-driven wyner-ziv video coding,” *Proceedings of the SPIE International Conference on Image and Video Communications and Processing, accepted*, (San Jose, CA), January 15-19 2006.
19. Z. Li and E. Delp, “Wyner-ziv video side estimator: Conventional motion search methods revisited,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 825–828, (Genova, Italy), September 11-15 2005.
20. Z. Li, L. Liu, and E. J. Delp, “Wyner-ziv video coding: A motion estimation perspective,” *Proceedings of the SPIE International Conference on Image and Video Communications and Processing, accepted*, (San Jose, CA), January 15-19 2006.
21. Z. Li, L. Liu, and E. J. Delp, “Wyner-ziv video coding with universal prediction,” *Proceedings of the SPIE International Conference on Image and Video Communications and Processing, accepted*, (San Jose, CA), January 15-19 2006.
22. A. Wyner and J. Zip, “The rate-distortion function for source coding with side information at the decoder,” *IEEE Transactions on Information Theory* **IT-22**, pp. 1–10, January 1976.
23. D. Slepian and J. Wolf, “Noiseless coding of correlated information sources,” *IEEE Transactions on Information Theory* **IT-19**, pp. 471–480, Semtember 1973.
24. Z. Li, *New methods for motion estimation with applications to low complexity video compression*. Ph.D. Thesis, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, December 2005.
25. T. Cover and J. A. Thomas, *Element of Information Theory*, John Wiley and Sons INC., New York, NY, 1991.
26. M. Sartipi and F. Feriki, “Distributed source coding in wireless sensor networks using LDPC coding: The entire Slepian-Wolf Rate Region,” *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2005)*, pp. 1939–1944, 2005.
27. R. G. Gallager, “Low density parity check codes,” *MIT Press* , 1963.
28. T. Richardson, M. A. Shokrollahi, and R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Transactions on information theory* **47**, pp. 619–637, Febuary 2001.
29. S. Babvey, A. Bourgeois, and S. W. McLaughlin, “An Efficient R-Mesh Implementation of LDPC Codes Message-Passing Decoder,” *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS’05)*, 2005.
30. K. Jack, *Video Demystified*, HighText Interactive, Inc., San Diego, CA, 1996.