Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2007, Article ID 52965, 15 pages
doi:10.1155/2007/52965

Research Article

Digital Video Encryption Algorithms Based on Correlation-Preserving Permutations

Daniel Socek,[1] Spyros Magliveras,[2] Dubravko Ćulibrk,[1] Oge Marques,[1] Hari Kalva,[1] and Borko Furht[1]

[1] *Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA*
[2] *Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, FL 33431, USA*

Correspondence should be addressed to Daniel Socek, dsocek@fau.edu

Received 28 February 2007; Accepted 19 June 2007

Recommended by Qibin Sun

A novel encryption model for digital videos is presented. The model relies on the encryption-compression duality of certain types of permutations acting on video frames. In essence, the proposed encryption process preserves the spatial correlation and, as such, can be applied prior to the compression stage of a spatial-only video encoder. Several algorithmic modes of the proposed model targeted for different application requirements are presented and analyzed in terms of security and performance. Experimental results are generated for a number of standard benchmark sequences showing that the proposed method, in addition to providing confidentiality, preserves or improves the compression ratio.

1. INTRODUCTION

Application-specific video encryption represents an important problem in multimedia security. In order to support a wide range of real-world video applications, an encryption algorithm should be designed within a specific video compression framework. Conventional encryption is designed for generic data, and as such, it does not support many specific video application requirements. For instance, video encryption algorithms that support one or more of the following application requirements are often needed.

(1) *Perceptual quality control.* An encryption algorithm can be used to intentionally degrade the quality of perception, but still keep the video visually perceivable.
(2) *Format compliance.* It could be desired that the encryption algorithm preserves the video compression format, so that the ordinary decoders can still decode the encrypted video without crashing.
(3) *Codec standard compliance.* A typical video system is likely to consist of a premanufactured standard-conforming encoder and decoder modules, and a video encryption method that requires no modification to either of the two modules is often desirable.
(4) *Minimal processing speed.* In many real-time video applications, it is important that the encryption and decryption algorithms are fast enough to ensure the minimal processing speed needed for the normal video system functioning.
(5) *Constant/near-constant bitrate.* It is often required that the encryption transformation preserves the size of a bitstream, where the output produced by an encryption-equipped encoder and the output produced by an ordinary encoder have same or similar sizes.

In general, two basic research methodologies for digital video encryption are used to provide support to aforementioned application requirements. *Selective encryption* algorithms perform conventional or nonconventional encryption only on certain selected parts of the video bitstream. In this type of algorithms the encryption step occurs either during or after encoding. For instance, Meyer and Gadegast [1] proposed to encrypt only the headers of the highest four layers in MPEG stream (sequence layer, GOP layer, picture layer, and slice layer), and optionally also the first macroblock after each slice header or all I-frames and all intracoded macroblocks. Spanos and Maples [2] suggested to encrypt I-frames of all MPEG groups of frames, the MPEG video sequence header (which contains all of the decoding initialization parameters such as the picture width, height,

frame rate, bit rate, and buffer size), and the ISO end code. Bhargava et al. [3] proposed to encrypt only the sign bits of the DCT coefficients and differential values of motion vectors in P- and B-frames of MPEG video. In approach by Li et al. [4] only the fixed length coded (FLC) data elements of a video stream are encrypted. However, the following security issues regarding selective encryption have been identified: (1) encrypting only I-frames of a video sequence does not provide enough security against ciphertext-only attacks, since the unencrypted B- and P-frames can reveal partial visible information [5]; (2) neither encrypting the sign bits nor encrypting multiple significant bits of the DCT coefficients is secure enough against ciphertext-only attacks utilizing the unencrypted bits [6]; (3) if all encrypted DCT coefficients are set to fixed values, it is possible to recover a rough view of the plaintext frame [7, 8]. In addition, many selective approaches require modification to both standard encoder and decoder, and a number of approaches result in a format defiant video stream.

The second type of algorithms use a *nonconventional full encryption* methodology, where the encryption is performed on the entire bitstream using a nonconventional encryption algorithm. Most of these algorithms are targeted for speed. Methods relying on fast chaotic maps are promising due to their fast performance. Although many chaotic encryption approaches were shown to be insecure, there are chaotic encryption algorithms that, up to date, remain unbroken, such as the method of Li et al. [9]. An excellent overview of these approaches, along with their comparative security analysis is presented in [10, 11]. There are a few recently proposed fast, hardware-friendly full encryption methods that are based on a class of neural networks [12]. However, these methods were later shown to be less secure than originally anticipated [13]. There are also nonconventional full approaches based on other mathematically hard problems, but most of them have been shown insecure due to oversimplification. For example, Yi et al. [14] proposed a new fast encryption algorithm for multimedia (FEA-M), which bases the security on the complexity of solving nonlinear Boolean equations. The scheme was shown insecure against several different attacks [15, 16]. In addition to questionable security, most nonconventional full encryption approaches are applied after encoding which does not support format compliance requirements. Also many of these algorithms are obsolete in recent years after the wide adoption of advanced encryption standard (AES) that offers much faster performance in comparison to the previous conventional cryptosystems, including data encryption standard (DES).

An approach to video encryption where the encryption step occurs before encoding is attractive since many of the application requirements are inherently supported. However, most encryption algorithms have a property to randomize the source data and thus negatively affect compression performance of an encoder. The first attempt to creating an encryption scheme that preserves the compressibility of the source was made by Pazarci and Dipçin [17], in which the encryption occurs in the RGB color space using four secret linear transforms before the video is compressed by the MPEG-2 encoder. However, in [4] it was shown that the scheme is not secure against brute force attacks where searching complexity is estimated to a computationally feasible number of possibilities, and that the scheme is not secure against known/chosen-plaintext attacks. Also, the scheme by Pazarci and Dipçin necessarily produces a perceivable output with degraded quality but does not offer a mode where the output encrypted video is nonperceivable.

An encryption scheme of much stronger security based on permutations of video frames was proposed in [18], and in this work we extend this approach to a family of algorithms that can be used for a variety of video applications. The scheme from [18] is based on the correlation-preserving "almost sorting" permutations which are derived from the previous frames. The proposed methodology is based on the fact that both sorting and "almost sorting" permutations can serve to preserve or improve compressibility of frames, and at the same time to disguise the frames to a nonperceivable form. This duality represents the main principle upon which our proposed algorithms rely.

The rest of the paper is organized as follows. Section 2 introduces permutations and examines their dual role in areas of data compression and data encryption. The proposed video encryption algorithms based on permutation transformations are presented in Section 3, while in Section 4 a thorough security analysis is performed. Experimental results showing the performance of proposed algorithms are given in Section 5. Finally, Section 6 serves to present our conclusions and ideas for future work related to this research.

2. DUALITY OF PERMUTATIONS

Permutation-based transformations are basic building blocks for many compression and encryption techniques. However, the dual use of these transformations has not been extensively studied. In this work we analyze the compression-encryption duality of permutations and develop actual methodologies for the dual use of certain permutation-based transformations in domain of digital video compression and encryption. To set the stage for the later discussion, some preliminary definitions are established next.

2.1. Permutations on sequences

A video frame can be represented in a one-dimensional finite sequence using raster scan order. A *permutation* of a finite sequence s is a bijection from s onto itself. Permutation P is often represented by its *Cartesian form* or *brackets form* denoting the indices for the rearrangement of s:

$$P = \begin{bmatrix} i_1 & i_2 & \cdots & i_n \end{bmatrix}, \tag{1}$$

where i_j, $1 \le j \le n$, is a sequence of n unique indices of elements of s, and n is the size of s. The family of all permutations on a sequence of size n forms an algebraic *group* under functional composition, denoted by S_n. P is called *sorting permutation* of s if it rearranges s in ascending order. We use $s_1^{f_1}, \ldots, s_k^{f_k}$ to denote the *histogram* of s, where $s_1, \ldots, s_k$

are distinct elements of s in the ascending order and f_i the frequency of element s_i.

Theorem 1. *If a histogram of finite sequence s is $s_1^{f_1}, \ldots, s_k^{f_k}$, there are exactly $f_1! \times \cdots \times f_k!$ sorting permutations of s.*

Proof. Let P be a sorting permutation of s. The indices corresponding to the positions of s_1 appear in the first f_1 places of the Cartesian form of P, the indices of s_2 appear in the second f_2 places, and so on. Thus, one can partition P into k segments of indices of sizes $f_1, \ldots, f_k$, and rearranging the indices within each segment results in another sorting permutation of s since the indices correspond to same values. At the same time, exchanging elements across segments disrupts ascending order of the resulting rearrangement of s, and the corresponding permutation is not a sorting permutation. Since there are $f_i!$ ways of rearranging indices of the ith segment of the Cartesian form of P, there are exactly $f_1! \times f_2! \times \cdots \times f_k!$ sorting permutations of s. □

Thus, if a frame F is of dimension $w \times h$, rearrangements of pixel values from F are achieved when permutations from $S_{w\times h}$ act on the corresponding raster scan sequence. If F has k colors, and $f_1, \ldots, f_k$ are frequency values of the color histogram of F, then according to Theorem 1 there are $f_1! \times \cdots \times f_k!$ permutations in $S_{w\times h}$ that sort frame F.

2.2. Permutations and compression

Permuting a sequence affects the correlation of the neighboring samples. If a random permutation acts on a sequence, the correlation of the neighboring samples is likely destroyed and the compressibility is decreased. On the other hand, if a sorting permutation acts on a sequence, the sample-to-sample correlation of the symbols is the best possible, and thus very suitable for run-length encoding (RLE) and similar compression primitives that exploit such correlation.

Many compression algorithms that assume neighboring sample correlation in the source, such as the image and video coding methods, are likely to take advantage of the sorted signal and produce very good compression. Figure 1 illustrates how compressibility of a natural image dramatically changes when pixel values are rearranged according to either a random permutation or a sorting permutation.

2.2.1. Compressing a sorting permutation

Even though certain permutations, such as sorting permutations, can affect the compression of source data in the positive way, compression of the permutation itself is usually not efficient. If a permutation P of degree n, that is, $P \in S_n$, is to be transmitted, an obvious way is to represent P as a sequence of length n, consisting of unique $\log_2 n$-bit indices corresponding to a Cartesian form of P. Total transmission cost is in that case $n \log_2 n$ bits. If an ordering of permutations from S_n is fixed, such as the lexicographic ordering, each permutation have its own index according to that ordering. For permutations with small indices transmitting the index itself could be more efficient, but in the worst case the cost of this transmission is $\log_2 n!$. This approach is analogous to a fixed dictionary compression approach. Unfortunately, sorting permutations of a natural image usually do not have lexicographically small index to compress well. Furthermore, for frames with k-bit color palette, where $k < \log_2 n$, it is cheaper to send an uncompressed frame (nk bits) from which the sorting permutation can be calculated, than to directly transmit the sorting permutation using $n \log_2 n$ bits.

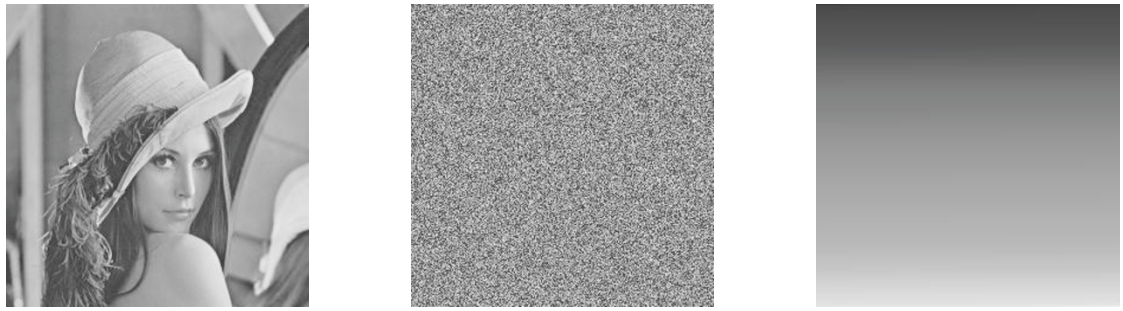
Thus, directly compressed source data is usually smaller in size than compressed permuted source data plus the compressed permutation that is used to recover the original order of the source. If efficient compression can be performed on a source data, which is the case for natural images and video frames, it is likely that the cheapest way to transmit a sorting permutation is to transmit the compressed source from which the receiver can calculate the sorting permutation after uncompressing the received data. This reveals the rationale used in the proposed algorithms.

2.3. Role of permutations in data compression

Permutation-based transformations were considered to serve as a compression primitive in the past. In [19], Burrows and Wheeler introduced one such transformation, which is referred to as the *Burrows-Wheeler transformation* (BWT). The authors presented an approach called block sorting lossless data compression algorithm, which combined BWT with move-to-front coding and a standard compressor such as Huffman coding or arithmetic coding. The algorithm reportedly achieves compression rates similar to that of content-based lossless methods, but at execution times comparable to that of the fast general-purpose lossless compressors, such as Ziv-Lempel techniques. The work by Burrows and Wheeler was further investigated and improved by Deorowicz [20]. Using the concept of permutation codes, Arnavut and others also studied applications of permutations and permutation codes to the compression of digital images [21]. According to study by Arnavut and Otu a good compression is achieved when BWT is used in lossless compression of color-mapped images where pixel values represent indices that point to color values in a look-up table [21]. In [22], Arnavut and Magliveras introduced lexical permutation sorting algorithm (LPSA), a more generalized version of BWT which has better performance than BWT when transmitting permutations. Sample reordering is used in many transform-based image and video coding methods. Specifically, in JPEG and MPEG type of image and video compression, a special reordering (permutation) is used to reorder transform coefficients (e.g., DCT coefficients) in a fixed order that allows for a more efficient symbol entropy coding. Although some alternate reorderings exist for certain applications, best performance on average is expected when the coefficients are permuted according to a zigzag ordering.

2.4. Permutations and encryption

Permutations are used extensively as an encryption primitive in modern symmetric-key cryptography. In addition, there is a significant number of permutation-only encryption



(a) (b) (c)

Figure 1: Compressibility of a natural image affected by permutations: (a) the original 256 × 256 greyscale image *Lena* [GIF = 66.5 KB], (b) randomly permuted image *Lena* [GIF = 85.3 KB], and (c) sorted image *Lena* [GIF = 7.38 KB].

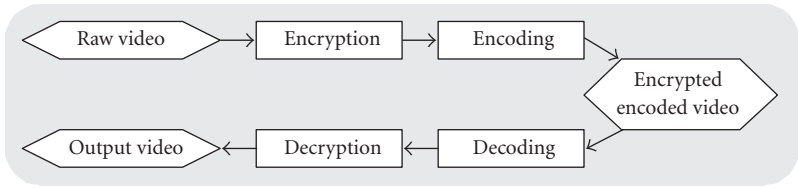


Figure 2: Block diagram of an approach where encryption occurs before video encoding (compression).

algorithms proposed for both analog and digital image and video encryption.

In most modern symmetric-key cryptosystems, permutations are used for data diffusion. Systems such as AES or DES are essentially a substitution-permutation networks, or shortly S-P networks, where permutation transformations are employed in every round. In fact, most symmetric-key block ciphers rely on permutations of symbols (e.g., bits) in order to provide data diffusion [23]. In addition, there are cryptosystems based solely on transformations that use permutation groups. For instance, cryptosystem PGM (permutation group mapping) is based on logarithmic signatures of finite permutation groups [24].

Permutations are extensively used in analog video encryption. Techniques such as scan line shuffling [25] or pixel position shuffling [26–28] represent common approaches for analog video encryption. Similarly, in digital video encryption domain, secret permutations are widely used to shuffle the positions of pixels [29], but also to shuffle DCT/wavelet coefficients [30, 31], Huffman table codewords [3], and even blocks or macroblocks [32]. These algorithms are based solely on secret permutations that are generated by a secret key.

Video encryption algorithms based solely on secret permutations often receive harsh criticism. In [32] it is pointed out that these algorithms are inherently and necessarily insecure against several types of cryptanalysis, including known-plaintext, chosen-plaintext, and chosen-ciphertext attacks. The authors even discuss cryptanalytic techniques that are universally applicable to all permutation-only encryption algorithms. While the methods proposed in this work technically belong to this category of algorithms, there is a crucial difference between the algorithms proposed here and the previously proposed permutation-only encryption algorithms. In Section 4 it is discussed in detail why this difference makes the proposed algorithms robust against the various attacks presented in [32].

3. CORRELATION-PRESERVING VIDEO ENCRYPTION

Most encryption algorithms have a randomization effect on the source data, and as such, cannot be effectively applied before the compression stage. In this section we present a set of encryption algorithms for spatial-only video coding based on permutation transformations that have a correlation-preserving property. Using these algorithms, one can perform encryption prior to video encoding, as illustrated in Figure 2.

The basic idea behind the permutation-based methodology for correlation-preserving video encryption is as follows. Sorted, as well as "almost sorted" frames are strongly spatially correlated. Such permuted frames are in many instances even more compressible in terms of spatial-only coding than the original source frames. When a sorting permutation of the previous frame acts on the current frame, it produces what we refer to as an "almost sorted" frame. Transmitting a compressed frame from which the initial permutation can be computed is efficient. Once an initial permutation is transmitted through a secure channel, the sender uses it to "almost sort" the next frame. In Section 4 it is shown that, except in rare circumstances, a sorted or "almost sorted" frame can be safely sent through the regular, nonsecure channel. By calculating a sorting permutation of the received frame, the receiver uses it to recover the next frame, and so on. This way the spatial correlation within frames of a video sequence is expected to be preserved, if not improved, when

Initialization: Set a to a copy of $w \times h$ frame F, p to $[0\ 1\ 2\ \cdots\ (w \times h) - 1]$ (the identity permutation with zero-based index), l to 0, and r to $(w \times h) - 1$.
Input: a, p, l and r.
(1) Set $i = l - 1$, $j = r$, and $v = a[r]$
(2) If $r \leq l$ return from the algorithm
(3) Start an infinite loop and do the following:
  (a) Set $i = i + 1$
  (b) While $a[i] < v$ do the following:
    (i) Set $i = i + 1$
  (c) Set $j = j - 1$
  (d) While $v < a[j]$ do the following:
    (i) If $j = l$ break from this while loop
    (ii) Set $j = j - 1$
  (e) If $i \geq j$ break from the infinite loop
  (f) Exchange $a[i]$ and $a[j]$
  (g) Exchange $p[i]$ and $p[j]$
(4) Exchange $a[i]$ and $a[r]$
(5) Exchange $p[i]$ and $p[r]$
(6) Recursively call this algorithm with $a = a$, $p = p$, $l = l$ and $r = i - 1$
(7) Recursively call this algorithm with $a = a$, $p = p$, $l = i + 1$ and $r = r$.

Algorithm 1: Modified recursive quicksort algorithm for computing the unique sorting permutation of a given frame.

Input: Raw video sequence (or scene) $F_1, \ldots, F_m$.
(1) Alice first computes the permutation P_1 from frame F_1.
(2) Alice calculates $E(F_1)$ and transmits it through *ChS*.
(3) For each subsequent frame F_i, $i = 2, \ldots, m$, Alice does the following:
  (a) She computes the permutation P_i and the frame $P_{i-1}(F_i)$;
  (b) Alice then applies the standard encoder to the frame $P_{i-1}(F_i)$ and transmits the encoded frame $E(P_{i-1}(F_i))$ to Bob through *ChR*.

Algorithm 2: Basic encryption algorithm for lossless spatial-only video coding.

static-camera low motion sequences (e.g., video conferencing or telephony) and spatial-only video codecs (e.g., motion JPEG) are used.

3.1. Global system settings

The system is assumed to have two channels of communication (in physical or abstract sense). *ChR* denotes a regular, nonsecure channel where all messages are plain and open for eavesdropping, while *ChS* denotes a secure channel that can also be eavesdropped, however, the messages are encrypted using a secure communication protocol based on a conventional cryptosystem such as AES. In our model, a video consists of one or more scenes and each scene consists of a sequence of frames $F_1, F_2, \ldots, F_m$. For a given frame, there are likely a large number of sorting permutations of it (see Theorem 1). The system must fix a method by which a *unique* sorting permutation is always selected for a given image. Algorithm 1 illustrates a method that we used for computing a unique sorting permutation for a given frame. This particular method is based on a modification to a recursive quicksort algorithm, however, similar approach can be used with other sorting methods.

3.2. Basic algorithms

If F is a frame of size $n =$ width of $F \times$ height of F, let $P(F)$ be a frame obtained by permuting the elements of F according to a permutation P from S_n. The inverse of permutation P is denoted by P^{-1}. For a given frame F_i, let P_i denote the unique sorting permutation obtained by the modified quicksort method from Algorithm 1. The encoding of frame F is denoted by $E(F)$, and $D(F)$ denotes the decoding of F. The basic algorithm for lossless video coding is described in Algorithms 2 and 3 (encryption and decryption, resp.). The algorithm for spatial-only lossless video encryption faithfully

Input: Encoded first frame $E(F_1)$ and encrypted encoded subsequent frames of a video sequence (or scene) $E(P_1(F_2)), \ldots, E(P_{m-1}(F_m))$.

(1) Bob computes $D(E(F_1)) = F_1$ and obtains the permutation P_1.

(2) For each successive received frame $E(P_{i-1}(F_i))$, $i = 2, \ldots, m$, Bob does the following:

(a) Computes $D(E(P_{i-1}(F_i))) = P_{i-1}(F_i)$ and calculates $F_i = P_{i-1}^{-1}(P_{i-1}(F_i))$ where P_{i-1}^{-1} is the inverse permutation of P_{i-1};

(b) Calculates the permutation P_i of F_i.

Algorithm 3: Basic decryption algorithm for lossless spatial-only video coding.

Input: Raw video sequence (or scene) $F_1, \ldots, F_m$.

(1) Alice first computes $E(F_1)$ and then $F_1' = D(E(F_1))$ from which she obtains the unique sorting permutation P_1'.

(2) Alice sends $E(F_1)$ through *ChS* to Bob.

(3) She computes $E(P_1'(F_2))$ and sends it through *ChR* to Bob.

(4) Next, she computes $F_2' = D(E(P_1'(F_2)))$ and then $F_2'' = (P_1')^{-1}(F_2')$ from which she calculates the unique sorting permutation P_2''.

(5) For each subsequent frame F_i, $i = 3, \ldots, m$, Alice does the following:

(a) Computes $E(P_{i-1}''(F_i))$, and sends it to Bob through *ChR*;

(b) Computes $F_i' = D(E(P_{i-1}''(F_i)))$;

(c) Applies $(P_{i-1}'')^{-1}$ to get $F_i'' = (P_{i-1}'')^{-1}(F_i')$;

(d) Calculates the canonical sorting permutation P_i''.

Algorithm 4: Basic encryption algorithm for lossy spatial-only video coding.

corresponds to the model from Figure 2 where encryption completely precedes video encoding. This is achieved by creating "almost sorted" frames that are sent through open channel *ChR*. In spatial-only lossless video encoding, adaptive dictionary-based compression primitives are often used to exploit neighboring pixel correlation prior to applying entropy coding. In particular, this technique is employed in animated GIF and motion PNG coding. Sorted and "almost sorted" data is well suited to this type of compression. Compression with a pixel prediction model such as the one used in motion JLS (lossless JPEG) also relies on correlation of the currently encoded pixel and the pixels in the neighborhood area. In motion JLS, for instance, a current pixel is predicted in raster order, from pixels directly on top, to the diagonal and to the left of the current pixel. Sorted and "almost sorted" data are also suitable for this compression model.

Similar, but slightly different approach to video encryption can be taken when dealing with lossy spatial-only video coding. However, to compensate for the loss of data and to prevent error propagation issues, "almost sorting" permutations must be calculated on the compressed frames which results in somewhat more involved encryption step. The algorithm (encryption and decryption) targeted for lossy video coding is depicted in Algorithms 4 and 5, respectively. This algorithm requires a compression stage as a preprocessing to the encryption, so technically it does not exactly correspond to Figure 2. When compression is seen as a preprocessing step, the algorithm should still be considered to be a precompression encryption approach, and as such, inherently possesses the nice properties such as codec-standard compliance and format compliance.

In lossy transform-based coding of digital images and video frames, typically a block of pixels undergoes the transformation such as DCT or wavelet. For instance, this is the case with motion JPEG (M-JPEG) coding. The given block of pixels represents a small subimage of the image or frame, thus containing a set of two-dimensional neighboring pixels. In this setting sorted and "almost sorted" images and frames compress well. If the sorted and "almost sorted" data is grouped in to the blocks of the same size that is used in transform coding, the compression is further improved, as indicated by an example in Figure 3.

The computational complexity of the proposed method is very low at the decoder side for both lossless and lossy video coding, since the only additional computation that has to be performed involves the calculation of a sorting permutation. The algorithm from Algorithm 1 used to calculate the unique sorting permutation of a given frame has a computational complexity of only $\mathcal{O}(N \log N)$. Inverting or applying a permutation is equivalent to a table lookup.

INPUT: Encoded first frame $E(F_1)$, encrypted encoded second frame $E(P_1'(F_2))$ and encrypted encoded subsequent frames of a video sequence (or scene) $E(P_2''(F_3)), \ldots, E(P_{m-1}''(F_m))$.

(1) Bob calculates $D(E(F_1)) = F_1' \approx F_1$ and sorting permutation P_1'.

(2) From $E(P_1'(F_2))$ he computes $F_2' = D(E(P_1'(F_2)))$.

(3) Bob approximates $F_2 \approx F_2'' = (P_1')^{-1}(F_2')$.

(4) He then recovers the unique sorting permutation P_2'' of F_2''.

(5) For each received frame $E(P_{i-1}''(F_i))$, $i = 3, \ldots, m$, Bob:

  (a) Decodes $E(P_{i-1}''(F_i))$ into $F_i' = D(E(P_{i-1}''(F_i)))$;

  (b) Approximates $F_i \approx F_i'' = (P_{i-1}'')^{-1}(F_i')$;

  (c) If $i < m$ he calculates a sorting permutation P_i'' of F_i'.

ALGORITHM 5: Basic decryption algorithm for lossy spatial-only video coding.

(a) (b) (c)

FIGURE 3: Improving compressibility by adhering to block size used in transform-based coding: (a) 256×256 greyscale image *Lena* [JPEG = 6.95 KB], (b) sorted image *Lena* in raster order [JPEG = 1.81 KB], and (c) image *Lena* fully sorted and arranged according to 8×8 blocks conforming to the encoder's transform coding block size [JPEG = 1.09 KB]. The compression quality parameter of JPEG encoder was set to 50 (where 0 is the best quality and 100 the worst).

The basic algorithms proposed in this section can be extended to accommodate for additional application requirements. For instance, these algorithms do not offer perceptual quality control, cannot handle global camera motion such as translation, and do not support VCR-like functionality. Next, we introduce several extensions to the basic algorithms in order to support these additional application requirements.

3.3. Extensions to basic algorithms

The following extensions to the basic algorithms from Algorithms 2, 3, 4, and 5 are established in order to broaden their applicability.

(i) Block-based extension for perceptual quality control.

(ii) Extension for handling global camera translational motion.

(iii) Extension for hiding the histogram.

(iv) Extension for enabling VCR-like functionality and better error resilience.

3.3.1. Block-based approach

The proposed algorithm can be applied on individual blocks within a frame the same way it is applied on the entire frame. By doing so, two different features are achieved: (1) the algorithm is more robust to high motion within a frame as long as the motion is limited to small number of blocks, and (2) by controlling the block size one can also control the degree of perception in the sense that the video becomes degraded (blocky) but perceivable for smaller blocksizes. This algorithmic mode is illustrated in Figures 6(g), 6(h), and 6(i).

3.3.2. Extensions for handling global camera motion

Unfortunately, the basic algorithms cannot handle camera motion well, since the sorting permutation of the previous frame will, in the case of global motion, not create almost sorted data when applied to the data of the current frame. However, if a global translational camera motion is known, for instance, by using some motion estimation methods as a preprocessor, it is possible for the receiver to *readjust* the sorting permutation accordingly by sending this information to the receiver's side.

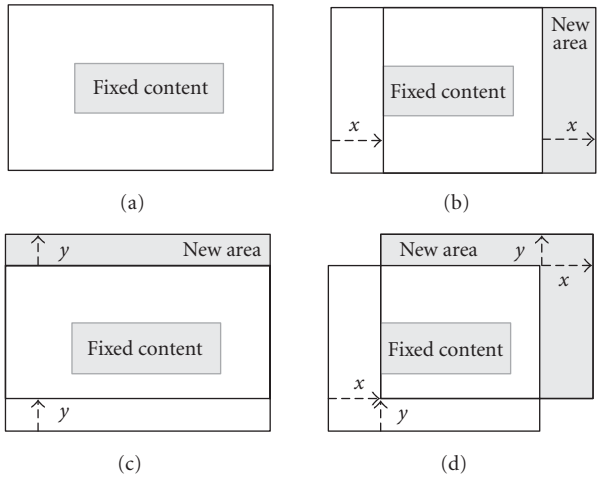


Figure 4: Translational camera motion.

Assuming that the camera moves in a simple translational motion, as illustrated in Figure 4, where x and y represent the amount of pixels that camera moved within x-axis and y-axis, respectively, the sorting permutation can be readjusted to almost sort the current frame provided that the values of x and y are given.

Suppose a scene in which no movement occurred is captured with a camera that solely moved horizontally on x-axis a distance that translates to exactly x pixels and vertically on y-axis a distance that translates to exactly y pixels. Note that the value of x is positive if the camera moves to the right, and negative if it moves to the left, while value of y is positive if the camera moves down, and negative if it moves up. The algorithm presented in Algorithm 6 is used for readjusting the sorting permutation of frame F_i, represented with zero-based index and denoted by P_i, into P_i' to make it more suitable "almost sorting" permutation of the next frame F_{i+1}.

3.4. Histogram-hiding extension

Histogram information in the basic model is known when the "almost sorted" frames are sent through the regular channel. Thus, from a security point of view, it is a good idea to hide the histogram from the adversary. Since the original histogram is actually secret, it is possible to hide the rest of the video histograms by subtracting the sorted image (the histogram) of the previous frame from the currently "almost sorted" frame, which introduces some computational overhead in order to compute the differences. This extension can be combined with a block-based extension to either provide some limited perceptual encryption and to restrict the motion-related permutation noise to the block where motion occurred, as illustrated in Figures 6(k) and 6(l). This transformation is equivalent to applying a secret permutation (or secret permutations in the case of block-based approach with histogram-hiding extension) on the ordinary frame differences, where a given permutation changes significantly from frame to frame.

Given two $w \times h$ video frames I and J, the *frame difference* between I and J, denoted by $\Delta(I, J)$, is defined as follows:

$$\Delta(I,J)[x,y] = \operatorname{clip}\left(I[x,y] - J[x,y] + \left\lfloor \frac{x_{\text{peak}}}{2} \right\rfloor\right), \quad 1 \le x \le w,\ 1 \le y \le h, \tag{2}$$

where $I[x, y]$ denotes the pixel value of I at coordinates (x, y), x_{peak} is the maximum pixel value (e.g., $2^n - 1$ for n-bit-per-pixel frames), and $\operatorname{clip}(\cdot)$ is the following function:

$$\operatorname{clip}(x) = \begin{cases} x_{\text{peak}}, & x > x_{\text{peak}}; \\ 0, & x < 0; \\ x, & 0 \le x \le x_{\text{peak}}. \end{cases} \tag{3}$$

One should note the following property regarding frame differencing and permutations. For two given $w \times h$ frames I and J and a permutation $P \in S_{w \times h}$, the following holds:

$$\Delta(P(I), P(J)) = P(\Delta(I,J)). \tag{4}$$

In the proposed extension, it is more efficient from the computational point of view to perform the transformation $P_i(\Delta(F_i, F_{i+1}))$ than the transformation $\Delta(P_i(F_i), P_i(F_{i+1}))$.

In the histogram-hiding extension, the spatial correlation is likely improved over the base approach (see Section 5). When additional computation is allowed, this extension usually reduces bitrate. A sole frame differencing technique can be used to achieve a limited form of perceptual encryption provided that the initial frame is kept secret, however, it is not an effective perceptual encryption mechanism since difference frames carry too much visible information about the content and additional encryption transformation is necessary to provide confidentiality. When combined with block-based extension, histogram-hiding approach achieves perceptual encryption with a considerably limited quality control. Thus, the recommended use of this extension is with the basic algorithms where entire frames are permuted.

3.5. Extension for enabling VCR-like functionality and improved error resilience

Just like in MPEG video coding, there is a need for having self-decodable frames, ones that are independent of previous or future frames. In the base scheme, the current frame is always recoverable from the sorting permutation of the previous frame, and as such, the scheme cannot handle VCR-like functionality or frame dropping caused by noisy channels or other communication errors. However, these functionalities can be achieved in the following way. The sorting permutation of the first frame (the key frame) can be used to "almost sort" every kth frame. The loss in compression gain is expected to be small since the assumption that all frames are part of a single scene holds. By doing so, the receiver can fast forward or rewind the video up to a kth frame, and frame dropping will affect only frames up to the next kth frame. This strategy is analogous to the strategy used in MPEG-like algorithms, where GOP (group of pictures) with repetitive I-frames are utilized.

INPUT: Sorting permutation P_i of $w \times h$ frame F_i, and a global horizontal and vertical camera translational motion from frame F_i to frame F_{i+1}, denoted by x and y, respectively.

(1) Set $c = 0$ and $d = w \times h$

(2) For $0 \le k < w \times h$ do the following:

(a) Set $i = x + P_i[k] \bmod w$

(b) Set $j = y + \lfloor P_i[k]/w \rfloor$

(c) If $j < h$, $j \ge 0$, $i < w$ and $i \ge 0$ then set $P'_i[c] = j \times w + i$ and increment c by 1

(d) Otherwise, set $P'_i[d] = (j \bmod h) \times w + (i \bmod w)$ and decrease d by 1.

ALGORITHM 6: Permutation readjustment algorithm to handle global translational motion.

(a) (b) (c) (d)

FIGURE 5: Readjustment of the sorting permutation: (a) previous frame, (b) current frame with global motion $x = 6$, $y = -4$, (c) frame sorted with a sorting permutation of the previous frame, and (d) frame sorted with a readjusted sorting permutation of the previous frame.

4. SECURITY ANALYSIS

This section serves to analyze security aspects of the proposed methods. The security strengths and weaknesses are pointed out.

Brute-force attack

Brute-force attack is based on exhaustive key search, and is feasible only for the cryptosystems with relatively small key space. In our case, the brute-force attack consists of two possible venues: one could either attack the underlying conventional cryptosystem used for encryption in channel *ChS*, or the proposed permutation-based method used in channel *ChR*. For that reason, it is recommended to use a strong conventional symmetric-key cryptosystem such as AES with 128-bit or stronger keys. The size of the key space related to our permutation-based method is equivalent to the following: given a color histogram of a $w \times h$ image F, how many different images can be formed out of the histogram color values? Note that F is just one of these images.

Let $s_1^{f_1}, \ldots, s_k^{f_k}$ be the histogram of frame F. In [18] it was shown that the number of different images that can be formed by permuting F is equal to the size of the $S_{w\times h}$-orbit of F, denoted by $S_{w\times h}(F)$, under the group action of $S_{w\times h}$ on the set of all possible images of dimension $w \times h$. Since

$$\left| S_{w\times h}(F) \right| = \frac{(wh)!}{\prod_{i=1}^{k} f_i!}, \tag{5}$$

there are exactly $(wh)!/\prod_{i=1}^{k} f_i!$ different images with the same color histogram $s_1^{f_1}, \ldots, s_k^{f_k}$. These distinct images determine the effective key space of our method. If one uses an n-bit conventional cryptosystem to encrypt key frames in channel *ChS*, the actual key space of the proposed method is

$$\min\left(2^n, \frac{(wh)!}{\prod_{i=1}^{k} f_i!}\right). \tag{6}$$

The size of the key space depends on the color histogram of the encrypted frame. As one can see, this number is extremely large when considering any meaningful images of reasonable dimensions, and it is usually much larger than brute-forcing 2^n keys of the used conventional symmetric-key cryptosystem.

In the case of block-based algorithmic mode, the attacker is faced with a smaller key space. If a blocksize of $b \times c$ is used, there are wh/bc blocks within a frame. Suppose that each ith

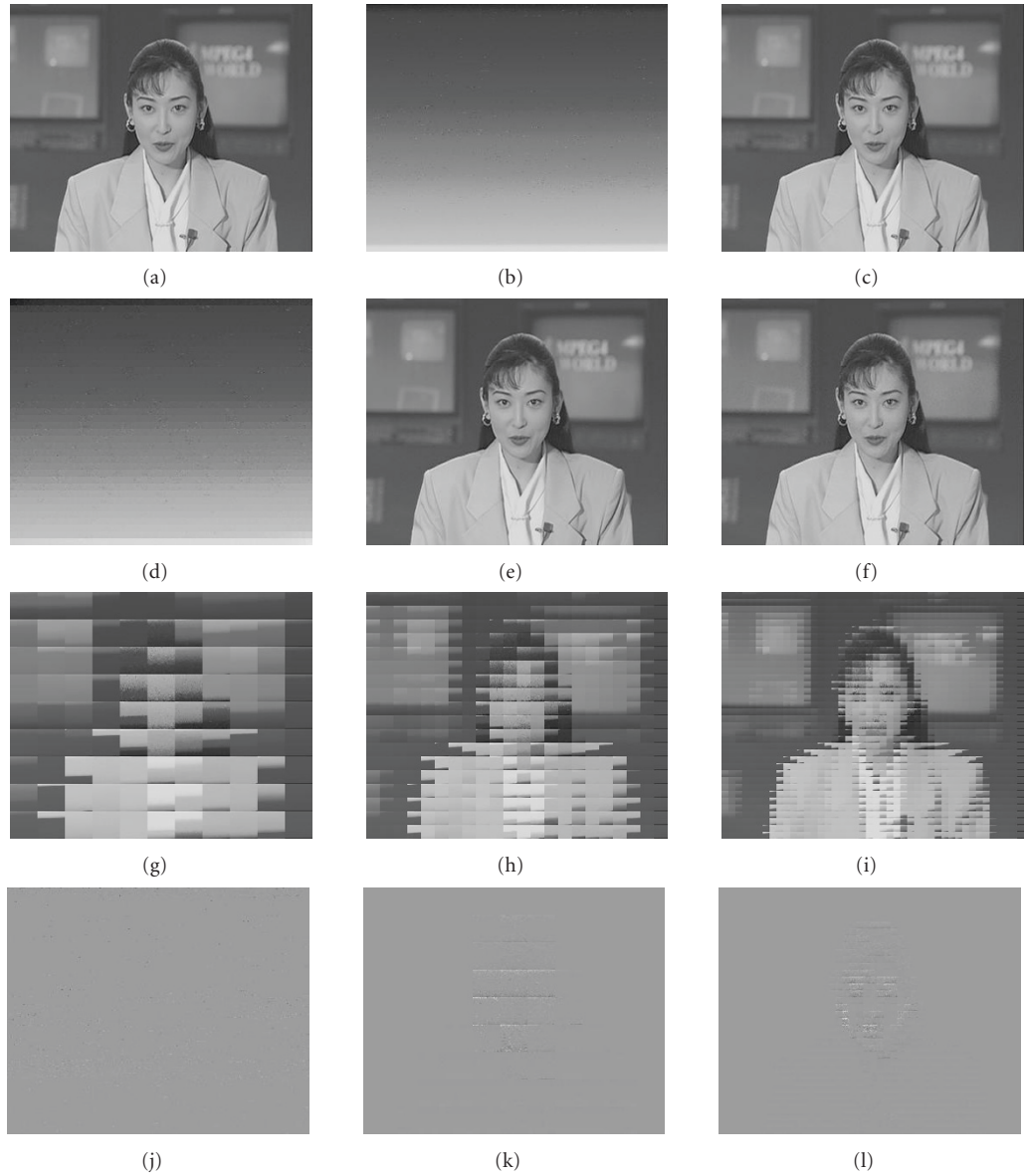


Figure 6: 150th frame of the following sequences: (a) original Akiyo, (b) sequence obtained by encrypting Akiyo with the basic encryption algorithm for lossless coding and decoding it without decryption, (c) sequence obtained by properly decrypting an encrypted Akiyo with lossless coding, (d) sequence obtained by encrypting Akiyo with the basic encryption algorithm for lossy MJPEG coding (with quality 90) and decoding it without decryption, (e) sequence decoded from a regular, not encrypted encoded Akiyo (compressed size 16 KB, PSNR 45.198 dB), (f) sequence obtained by properly decrypting an encrypted Akiyo using the proposed basic algorithm with M-JPEG coding (compressed size 12 KB, PSNR 41.737 dB), (g) (h) (i) sequence obtained by encrypting Akiyo with the block-based approach (blocksizes 32×32, 16×16, and 8×8, resp.) for lossless coding and decoding it without decryption, and (j) (k) (l) sequence obtained by encrypting Akiyo with the histogram-hiding approach combined with the basic encryption algorithm and the block-based approach (blocksizes 32×32 and 8×8, resp.) for lossless coding and decoding it without decryption.

Table 1: Sequences used in the experiments.

Sequence	No. of frames	Format	Bits/pixel
Hall monitor	250	CIF	8
Akiyo	250	CIF	8
Mother daughter	250	CIF	8
Grandma	100	QCIF	8
Claire	100	QCIF	8
Miss America	100	QCIF	8

Table 2: Compression performance of the proposed encryption algorithms with lossless spatial-only video coding.

Animated GIF						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss Am
Basic	0.746	0.433	0.745	0.685	0.776	0.867
blk32	0.959	0.597	0.955	—	—	—
blk16	0.976	0.655	0.968	0.845	0.908	0.981
blk8	0.980	0.745	0.975	0.877	0.941	0.990
Basic+hh	0.633	0.254	0.634	0.438	0.475	0.683
blk32+hh	0.657	0.264	0.654	—	—	—
blk16+hh	0.657	0.262	0.652	0.444	0.487	0.686
blk8+hh	0.658	0.259	0.650	0.445	0.487	0.688
Motion PNG						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss Am
Basic	0.813	0.421	0.789	0.695	0.765	0.891
blk32	0.993	0.555	0.969	—	—	—
blk16	1.007	0.631	0.982	0.869	0.941	0.995
blk8	1.011	0.738	0.993	0.915	0.990	1.011
Basic+hh	0.699	0.302	0.709	0.510	0.571	0.729
blk32+hh	0.713	0.314	0.713	—	—	—
blk16+hh	0.715	0.310	0.710	0.517	0.587	0.740
blk8+hh	0.716	0.307	0.708	0.518	0.583	0.743
Motion JLS						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss Am
Basic	1.113	0.693	1.135	0.757	0.954	1.131
blk32	1.120	0.691	1.122	—	—	—
blk16	1.128	0.750	1.132	0.818	1.037	1.142
blk8	1.150	0.854	1.147	0.882	1.135	1.163
Basic+hh	1.053	0.441	1.085	0.662	0.789	1.069
blk32+hh	1.049	0.445	1.056	—	—	—
blk16+hh	1.048	0.440	1.047	0.660	0.794	1.064
blk8+hh	1.046	0.433	1.035	0.664	0.797	1.064

$b \times c$ block in F has the color histogram $s_{i1}^{f_{i1}}, \ldots, s_{ik}^{f_{ik}}$. Then, the size of the key space is

$$\min\left(2^n, \prod_{j=1}^{wh/bc} \frac{(bc)!}{\prod_{i=1}^{k} f_{ji}!}\right), \tag{7}$$

which is for reasonable blocksizes, such as 8×8 or larger, still computationally infeasible.

Known/chosen-plaintext and chosen-ciphertext attacks

Permutation-only video encryption is considered weak against known/chosen-plain-text attack, and a chosen-ciphertext attack [32]. However, all of the previously proposed methods rely on generating the secret permutation using a secret key. Under this scenario, all of the aforementioned attacks are trying to recover the secret key (or a part of it) that was used for the current or future encryptions. Our algorithms do not rely on such a principle, and there is no secret key upon which a permutation is generated. The proposed approaches rely on the sorting permutation of *previous* frame, and thus, a key is directly dependant of the plaintext. Under a chosen-plaintext attack, the adversary can compute the sorting permutation for the chosen frame, but this gives no information about the sorting permutations for the unknown frames. Under a chosen-ciphertext attack, the adversary can recover the unsorting permutation for the chosen encrypted frame, but this gives no information regarding other unknown ciphertexts.

Known weaknesses and consequential applicability limitations

A limited known-plaintext attack is applicable to our methods, because the adversary can recover all frames that follow the known frame until the scene changes and key frame is updated. This, however, only reveals that one scene, since the key is completely changed as soon as the scene changes. This is a feature of all systems whose key depends on the plaintext. In addition, if the adversary has the information on the

Table 3: Compression performance of the proposed encryption algorithms with lossy spatial-only video coding.

Motion JPEG (quality 90)						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss Am
Basic	0.923	0.797	1.035	0.602	0.756	1.015
blk32	0.904	0.826	1.010	—	—	—
blk16	0.925	0.863	0.999	0.716	0.947	1.003
blk8	0.945	0.876	0.987	0.779	0.954	0.995
Basic+hh	0.741	0.373	0.801	0.349	0.397	0.732
blk32+hh	0.709	0.340	0.700	—	—	—
blk16+hh	0.690	0.327	0.666	0.324	0.361	0.691
blk8+hh	0.664	0.305	0.636	0.317	0.346	0.658
Motion JPEG (quality 70)						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss Am
Basic	0.825	0.796	0.968	0.589	0.687	0.999
blk32	0.803	0.870	0.972	—	—	—
blk16	0.845	0.886	0.955	0.740	0.913	0.984
blk8	0.888	0.893	0.937	0.805	0.945	0.976
Basic+hh	0.524	0.314	0.527	0.261	0.328	0.570
blk32+hh	0.487	0.309	0.460	—	—	—
blk16+hh	0.465	0.301	0.434	0.256	0.321	0.538
blk8+hh	0.449	0.288	0.418	0.255	0.316	0.522
Motion JPEG (quality 50)						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss Am
Basic	0.764	0.750	0.889	0.580	0.657	0.938
blk32	0.763	0.849	0.917	—	—	—
blk16	0.822	0.867	0.911	0.757	0.895	0.962
blk8	0.882	0.885	0.893	0.814	0.939	0.951
Basic+hh	0.403	0.340	0.442	0.291	0.356	0.550
blk32+hh	0.374	0.332	0.409	—	—	—
blk16+hh	0.364	0.325	0.395	0.291	0.356	0.542
blk8+hh	0.372	0.320	0.390	0.293	0.355	0.535

possible videos to be encrypted, he or she may be able to recognize which video sequence is being transmitted from Alice to Bob by observing the publicly given pixel value histograms of frames. Another related problem is the adversary's ability to analyze the properties of a given histogram for rough clues about the content. Namely, cartoon pictures and real photos have different histograms, and photos of human faces usually have narrower histograms than photos of natural scenes [10]. Finally, since the encrypted frames look less smooth when a fast motion occurs within a scene, it is possible to gain a limited knowledge about the scene dynamics by observing the "almost sorted" frames. Although limited, these attacks are unavoidable in the proposed methodology and our video encryption algorithms should not be used in applications where these attacks are of interest to an adversary.

Histogram-hiding extension of the proposed algorithms is much more robust against the second type of the known histogram attack since the adversary does not have the actual color histograms of the frames needed to analyze the properties of a given histogram for rough clues about the content. Source recognition attack still holds since the adversary have the access to the distribution of frame differences in the encrypted video, a statistics that can reveal the previously known video.

Even though some weaknesses have been identified, the proposed video encryption algorithms are applicable to a majority of real-word video security scenarios.

5. EXPERIMENTAL RESULTS

To evaluate the performance of our method in terms of compression, experiments are performed on six benchmark sequences in CIF (352×288) and QCIF (176×144) formats. Table 1 shows the technical summary of the used sequences. The following is a legend of acronyms used in Tables 2, 3, and 4:

(i) basic—basic algorithm operating on entire frames;

(ii) blk32—block-based extension operating on 32×32 blocks;

(iii) blk16—block-based extension operating on 16×16 blocks;

Table 4: Resulting PSNR (in dB) with lossy spatial-only video coding.

Motion JPEG (quality 90)						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss am
None	43.03	45.10	45.05	40.62	44.74	45.18
basic	38.76	41.50	39.61	41.14	41.81	41.14
blk32	39.03	43.04	40.64	—	—	—
blk16	39.94	43.17	40.93	41.66	43.16	42.10
blk8	40.23	43.25	41.22	41.40	43.16	42.37
Basic+hh	39.98	46.15	41.35	45.05	46.12	43.47
blk32+hh	40.37	46.66	42.13	—	—	—
blk16+hh	40.57	47.00	42.47	45.44	46.80	43.96
blk8+hh	40.85	47.52	42.82	45.63	47.18	44.32
Motion JPEG (quality 70)						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss am
None	37.97	40.23	40.73	36.11	39.46	41.10
basic	33.38	36.04	34.27	35.98	36.23	35.61
blk32	34.90	37.42	35.42	—	—	—
blk16	35.42	37.70	35.88	36.34	37.26	36.70
blk8	35.72	38.03	36.39	36.19	37.35	37.18
Basic+hh	36.09	42.91	38.36	43.47	43.53	40.67
blk32+hh	36.74	43.19	38.90	—	—	—
blk16+hh	37.13	43.56	39.27	43.72	44.11	41.03
blk8+hh	37.64	44.15	39.79	43.90	44.49	41.49
Motion JPEG (quality 50)						
Algorithm	Hall monitor	Akiyo	Mother	Grandma	Claire	Miss am
None	35.80	38.04	38.71	34.46	37.13	39.32
basic	31.27	33.84	32.33	33.72	34.01	33.49
blk32	32.76	35.03	33.47	—	—	—
blk16	33.25	35.43	33.98	34.05	34.83	34.55
blk8	33.63	35.81	34.61	33.99	35.04	35.10
Basic+hh	34.95	41.02	37.51	42.35	42.09	39.78
blk32+hh	35.42	41.14	37.88	—	—	—
blk16+hh	35.82	41.39	38.21	42.51	42.49	39.93
blk8+hh	36.43	41.82	38.73	42.65	42.78	40.32

(iv) blk8—block-based extension operating on 8×8 blocks;
(v) basic+hh—basic algorithm with histogram-hiding extension operating on entire frames;
(vi) blk32+hh—block-based extension operating on 32×32 blocks with histogram-hiding extension;
(vii) blk16+hh—block-based extension operating on 16×16 blocks with histogram-hiding extension;
(viii) blk8+hh—block-based extension operating on 8×8 blocks with histogram-hiding extension;
(ix) none—encoding without encryption.

Each sequence from Table 1 was encrypted with the proposed video encryption algorithms and encoded with both lossless and lossy spatial-only video coding. The encrypted sequences are evaluated in terms of compression performance, and in case of lossy coding, also in terms of resulting PSNR. The goal was to measure how the proposed correlation-preserving algorithms affect compressibility. Tables 2 and 3 show the ratio of encrypted encoded bitsize and the unencrypted encoded bitsize.

Finally, Table 4 shows the resulting PSNR for the lossy coding case. There is often a modest loss of quality due to slight salt-and-pepper noise that is added to the reconstructed (decrypted) video due to quantization, as depicted in Figures 6(e) and 6(f).

6. CONCLUSIONS AND FUTURE WORK

In this work, we present a methodology for encrypting a video content before the compression phase, without significantly impacting the compression ratio. In its core, the proposed approaches are based on permuting the current frame with a specific sorting permutation of a previous frame. The proposed algorithms preserve, and in some in-

stances even improve the spatial correlation of the source data since "almost sorted" frames on average have a better spatial sample-to-sample correlation than the actual frames. Therefore, spatial-only video codecs can be enhanced with an encryption-equipped preprocessor and a decryption-equipped postprocessor that can result in similar or improved compression performance but in the same time providing a significant level of computationally provable security. In effect, the algorithms produce fully application-friendly output and require no modification to the codec modules. Both security and performance analysis of the proposed methodology show that the algorithms are computationally efficient and resistant to typical cryptanalytic attacks.

There are several obvious applications in practice for which the proposed algorithms are suitable. Industrial and business-related video telephony and videoconferencing often requires confidentiality. The types of videos and price of spatial-only codecs, such as M-JPEG, clearly suit the requirements of the proposed encryption approaches. Surveillance video monitoring is also a suitable application. In this application it is important not to use motion-based coding since motion coding tends to reduce the spatial resolution, but more importantly, interpolated video is inadmissible as evidence in many legal jurisdictions. Due to recent terrorist attacks on western countries such as United States and United Kingdom, airplane cockpits and other public transportation places are monitored by their governments. Here, M-JPEG is currently the preferred method as it allows each picture to be wholly compressed into a data file, whereas MPEG derives frames from a sequence, storing mainly the differences and use interpolated frames which really do not stand up in court [33]. Digital multimedia captured for medical purposes is likely to be encoded with lossless video coding and without any kind of interpolation since the highest precision is necessary for correct diagnosis, and many medical multimedia videos are indeed represented by a single scene with low-motion.

Future directions should include an investigation of extending or modifying the proposed principle to achieve efficiency in exploiting temporal correlation as well, in order to achieve applicability to more advanced video codecs such as H.26x and MPEG family. The proposed algorithms could be combined with an object segmentation approach to improve their robustness against movement of objects within the scene, for instance, by using the sorting permutations of each object from the previous frame (including background). Another possibility for future research include the extension to spatial selectivity (e.g., to achieve anonymity) where only objects of interest (such as human face, etc.) are encrypted, while other objects are kept unencrypted.

REFERENCES

[1] J. Meyer and F. Gadegast, "Security mechanisms for multimedia data with the example MPEG-1 video," Project Description of SECMPEG, Technical University of Berlin, 1995.

[2] G. Spanos and T. Maples, "Performance study of a selective encryption scheme for the security of networked, real-time video," in *Proceedings of the 4th International Conference on Computer Communications and Networks (ICCCN '95)*, pp. 2–10, IEEE Press, Las Vegas, Nev, USA, September 1995.

[3] B. Bhargava, C. Shi, and S.-Y. Wang, "MPEG video encryption algorithms," *Multimedia Tools and Applications*, vol. 24, no. 1, pp. 57–79, 2004.

[4] S. Li, G. Chen, A. Cheung, B. Bhargava, and K.-T. Lo, "On the design of perceptual MPEG-Video encryption algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 214–223, 2005.

[5] I. Agi and L. Gong, "An empirical study of secure MPEG video transmissions," in *Proceedings of the Symposium on Network and Distributed System Security (SNDSS '96)*, pp. 137–144, IEEE Computer Society, San Diego, Calif, USA, February 1996.

[6] C.-P. Wu and C.-C. J. Kuo, "Fast encryption methods for audiovisual data confidentiality," in *Multimedia Systems and Applications III*, vol. 4209 of *Proceedings of SPIE*, pp. 284–295, Boston, Mass, USA, November 2000.

[7] J. Wen, M. Severa, W. Zeng, M. Luttrell, and W. Jin, "A format-compliant configurable encryption framework for access control of multimedia," in *Proceedings of the 4th IEEE Workshop on Multimedia Signal Processing (MMSP '01)*, pp. 435–440, Cannes, France, October 2001.

[8] T. Lookabaugh, D. C. Sicker, D. M. Keaton, W. Y. Guo, and I. Vedula, "Security analysis of selectively encrypted MPEG-2 streams," in *Multimedia Systems and Applications VI*, vol. 5241 of *Proceedings of SPIE*, pp. 10–21, Orlando, Fla, USA, September 2003.

[9] S. Li, X. Zheng, X. Mou, and Y. Cai, "Chaotic encryption scheme for real-time digital video," in *Real-Time Imaging VI*, vol. 4666 of *Proceedings of SPIE*, pp. 149–160, San Jose, Calif, USA, January 2002.

[10] S. Li, G. Chen, and X. Zheng, "Chaos-based encryption for digital images and videos," in *Multimedia Security Handbook*, vol. 4 of *Internet and Communications Series*, pp. 133–167, CRC Press, Boca Raton, Fla, USA, 2004.

[11] B. Furht, E. Muharemagic, and D. Socek, *Multimedia Encryption and Watermarking*, vol. 28 of *Multimedia Systems and Applications*, Springer, Berlin, Germany, 2005.

[12] D. Guo, L. M. Cheng, and L. L. Cheng, "A new symmetric probabilistic encryption scheme based on chaotic attractors of neural networks," *Applied Intelligence*, vol. 10, no. 1, pp. 71–84, 1999.

[13] D. Socek and D. Ćulibrk, "On the security of a clipped hopfield neural network-based cryptosystem," in *Proceedings of the 7th ACM Workshop on Multimedia and Security (MM-Sec '05)*, pp. 71–76, New York, NY, USA, August 2005.

[14] X. Yi, C. H. Tan, C. K. Siew, and M. R. Syed, "Fast encryption for multimedia," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 1, pp. 101–107, 2001.

[15] A. M. Youssef and S. E. Tavares, "Comments on the security of fast encryption algorithm for multimedia (FEA-M)," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 1, pp. 168–170, 2003.

[16] M. J. Mihaljević, "On vulnerabilities and improvements of fast encryption algorithm for multimedia FEA-M," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1199–1207, 2003.

[17] M. Pazarci and V. Dipçin, "A MPEG2-transparent scrambling technique," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 2, pp. 345–355, 2002.

[18] D. Socek, H. Kalva, S. Magliveras, O. Marques, D. Ćulibrk, and B. Furht, "A permutation-based correlation-preserving encryption method for digital videos," in *Proceedings of the 3rd International Conference on Image Analysis and Recognition (ICIAR '06)*, vol. 4141 of *Lecture Notes in Computer Science*, pp. 547–558, Springer, Póvoa de Varzim, Portugal, September 2006.

[19] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Research Report 124, Digital Systems Research Center, Palo Alto, Calif, USA, 1994.

[20] S. Deorowicz, "Improvements to Burrows-Wheeler compression algorithm," *Software-Practice and Experience*, vol. 30, no. 13, pp. 1465–1483, 2000.

[21] Z. Arnavut and H. Otu, "Lossless compression of color-mapped images with Burrows-Wheeler transformation," in *Proceedings of the IASTED Signal Processing Conference*, pp. 185–189, Acta-Press, Rhodes, Greece, July 2001.

[22] Z. Arnavut and S. Magliveras, "Lexical permutation sorting algorithm," *The Computer Journal*, vol. 40, no. 5, pp. 292–295, 1997.

[23] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Fla, USA, 1996.

[24] S. Magliveras, "A cryptosystem from logarithmic signatures of finite groups," in *Proceedings of the 29th Midwest Symposium on Circuits and Systems*, pp. 972–975, Elsevier, Lincoln, Neb, USA, August 1986.

[25] L. Brown, "Technical options for implementing pay-TV in Australia," *Australian Telecommunication Research Journal*, vol. 24, no. 2, pp. 1–8, 1990.

[26] A. D. Wyner, "An analog scrambling scheme which does not expand bandwidth—part I: discrete time," *IEEE Transactions on Information Theory*, vol. 25, no. 3, pp. 261–274, 1979.

[27] A. D. Wyner, "An analog scrambling scheme which does not expand bandwidth—part II: continuous time," *IEEE Transactions on Information Theory*, vol. 25, no. 4, pp. 415–425, 1979.

[28] Y. Matias and A. Shamir, "A video scrambling technique based on space filling curves," in *Proceedings of the Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology (CRYPTO '87)*, vol. 293 of *Lecture Notes in Computer Science*, pp. 398–417, Springer, Santa Barbara, Calif, USA, August 1987.

[29] J.-C. Yen and J.-I. Guo, "Efficient hierarchical chaotic image encryption algorithm and its VLSI realisation," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 147, no. 2, pp. 167–175, 2000.

[30] L. Tang, "Methods for encrypting and decrypting MPEG video data efficiently," in *Proceedings of the 4th ACM International Multimedia Conference*, pp. 219–229, ACM Press, Boston, Mass, USA, November 1996.

[31] A. Uhl and A. Pommer, *Image and Video Encryption: From Digital Rights Management to Secured Personal Communication*, vol. 15 of *Advances in Information Security*, Springer, Berlin, Germany, 2005.

[32] S. Li, C. Li, G. Chen, N. G. Bourbakis, and K.-T. Lo, "A general cryptanalysis of permutation-only multimedia encryption algorithms," IACR's Cryptology ePrint Archive: Report 2004/374, 2004.

[33] M. Horne, "Future video accident recorder," in *Proceedings of the International Symposium on Transportation Recorders*, The National Transportation Safety Board and International Transportation Safety Association, Arlington, Va, USA, May 1999.