Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

PERGAMON

Pattern Recognition 37 (2004) 725–737

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Image and video encryption using SCAN patterns

S.S. Maniccam[a], N.G. Bourbakis[a,b,c,*]

[a] *AIIS Inc., 73100 Chania Crete, Greece*
[b] *Wright State University, ITRI, College of Engineering and Computer Science, 3640 Glenn Hwy, Dayton, OH 45435, USA*
[c] *TUC, ECE Department, 73100 Chania Crete, Greece*

Received 26 December 2002; accepted 18 August 2003

Abstract

This paper presents a new method for image and video encryption and a first stage lossy video compression based on frames difference before the encryption. The encryption methods are based on the SCAN methodology which is a formal language-based two-dimensional spatial accessing methodology which can generate very large number of scanning paths or space filling curves. The image encryption is performed by SCAN-based permutation of pixels and a substitution rule which together form an iterated product cipher. The video encryption is performed by first lossy compressing adjacent frame differences and then encrypting the compressed frame differences. The main characteristics of the proposed methods are image encryption, first stage compression-based frames differences and encryption of video whose compression error can be bounded pixelwise by a user specified value, very large number of encryption keys, and ability to encrypt large blocks of any digital data. Results from the use of the methods proposed here are also provided.
© 2003 Published by Elsevier Ltd on behalf of Pattern Recognition Society.

1. Introduction

Security is an important issue in communication and storage of images and videos, and encryption is one of the ways to ensure security. Image and video encryption have applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication, etc. There are several image and video encryption methods. They include SCAN-based methods [1–3], chaos-based methods [4,5], tree structure-based methods [6–8], MPEG-based methods [9–11], and other miscellaneous methods [12–16]. The main advantage of the encryption method presented here is the higher complexity of encryption versus the other methods.

The proposed image encryption method is based on permutation of the pixels of the image and replacement of the pixel values. The permutation is done by scan patterns (encryption keys) generated by the SCAN methodology and the pixel values are replaced using a simple substitution rule. The permutation and substitution operations are applied in intertwined manner and iteratively. The image encryption is lossless. The proposed video encryption method is based on finding the difference between adjacent frames and lossy compressing the differences and then encrypting the compressed frame differences. The video encryption is lossy but the pixelwise difference between original and recovered video can be bounded by a user specified value.

The proposed encryption method belongs to a general framework called iterated product cipher [17,18] which is based on repeated and intertwined application of permutation and substitution. This framework has been well studied and developed in terms of cryptographic strengths and attacks and this forms the basis of many modern encryption methods including Data Encryption Standard [18], Advanced Encryption Standard [19], and chaos-based encryption methods [4,5]. The proposed encryption method has additional features such as key-dependent permutation, larger key space, variable length keys, and encryption of larger blocks. Note that the proposed encryption methods are symmetric private key encryptions, meaning that the

* Corresponding author. Wright State University, ITRI, College of Engineering and Computer Science, 3640 Glenn Hwy, Dayton, OH 45435, USA. Tel.: +1-937-775-5158.
E-mail address: bourbaki@cs.wright.edu (N.G. Bourbakis).

0031-3203/$30.00 © 2003 Published by Elsevier Ltd on behalf of Pattern Recognition Society.
doi:10.1016/j.patcog.2003.08.011

same key is needed for both encryption and decryption and the key must be known to both sender and receiver before the communication of image or video.

It is important to mentioned that the new implementation of the SCAN methodology proposed here has eliminated the criticisms made by some authors [20,21] of the first version of the SCAN methodology implemented in 1988 [1,22]. The main reasons are that the new version offers an unbreakable security due to the high volume of keys (10^{76000}) that require 10^{75000} in years from a supercomputing machine to cover all possible cases, and the SCAN confusion function that converts the image histogram flat (see Section 3.2) by making the decryption process absolutely impossible.

This paper is organized into five sections. Section 2 presents a brief introduction to the SCAN methodology and defines an encryption-specific SCAN language. Section 3 describes the image encryption method. Section 4 describes the video compression and encryption method. Section 5 presents conclusions about the proposed encryption methods.

2. SCAN methodology

2.1. Basic idea of SCAN

A scanning of a two-dimensional array $P_{m\times n} = \{p(i,j): 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$ is a bijective function from $P_{m\times n}$ to the set $\{1, 2, \ldots, mn-1, mn\}$. In other words, a scanning of a two-dimensional array is an order in which each element of the array is accessed exactly once, or a permutation of the array elements. The terms scanning, scanning path, Scan pattern, and Scan word are used interchangably in this paper.

The SCAN [1–3,23] represents a family of formal languages-based two-dimensional spatial accessing methodology which can represent and generate a large number of wide variety of scanning paths easily. The SCAN family of formal languages includes several versions, such as Simple SCAN, Extended SCAN, and Generalized SCAN, each of which can represent and generate a specific set of scanning paths. Each SCAN language is defined by a grammar and each language has a set of basic scan patterns, a set of transformations, and a set of rules to compose simple scan patterns to obtain complex scan patterns. The rules for building complex scan patterns from simple scan patterns are specified by the production rules of the grammar of each specific language. The reader is referred to Refs. [1–3,23] for detailed descriptions of syntax and semantics of various SCAN languages and their applications.

2.2. SCAN for encryption

The basic idea of the proposed encryption method is to rearrange the pixels of the image and change the pixel values. The rearrangement is done by a set of scanning patterns (encryption keys) generated by an encryption-specific SCAN language which is formally defined by the grammar $G = (\Gamma, \Sigma, A, \Pi)$ where non-terminal symbols $\Gamma = \{A, S, P, U, V, T\}$, terminal symbols $\Sigma = \{c, d, o, s, r, a, e, m, y, w, b, z, x, B, Z, X, (,),$ *space*, $0, 1, 2, 3, 4, 5, 6, 7\}$, start symbol is A, and production

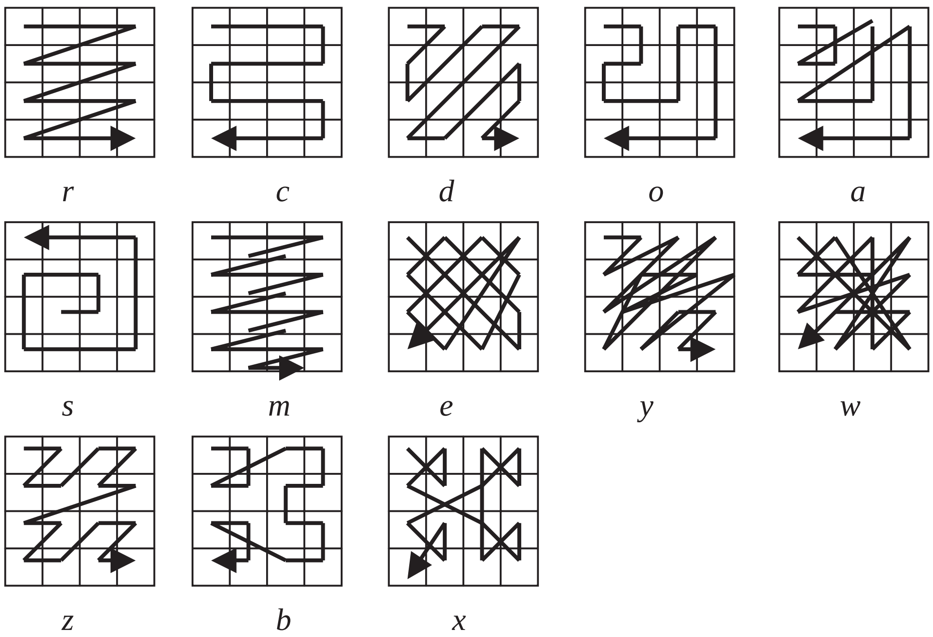

Fig. 1. Basic scan patterns.

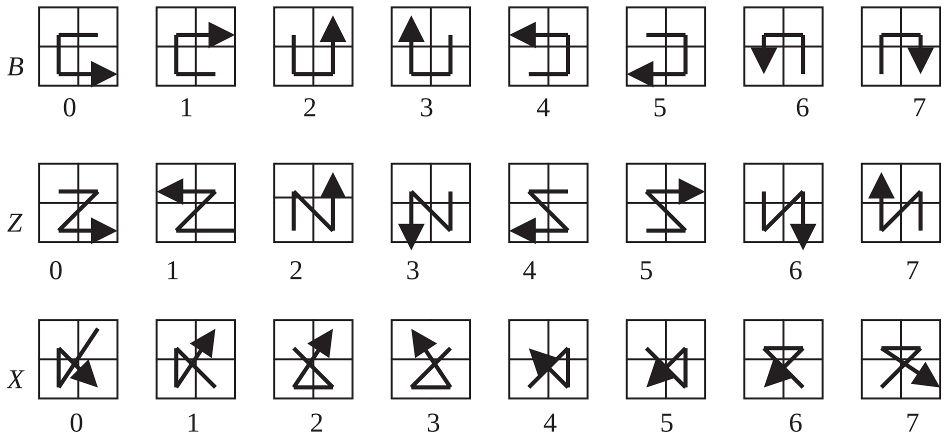

Fig. 2. Partition patterns and transformations.

rules Π is given by

$A \rightarrow S \mid P,$

$S \rightarrow UT,$

$P \rightarrow VT(A\ A\ A\ A),$

$U \rightarrow c \mid d \mid o \mid s \mid r \mid a \mid e \mid m \mid y \mid w \mid b \mid z \mid x,$

$V \rightarrow B \mid Z \mid X,$

$T \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7.$

The semantics of this encryption-specific SCAN language is described next.

(a) $A \rightarrow S \mid P$ means process the region by scan S or partition P.

(b) $S \rightarrow UT$ means scan the region with scan pattern U and transformation T.

(c) $P \rightarrow VT(A\ A\ A\ A)$ means partition the region with partition pattern V and transformation T, and process each of the four subregions in partition order using As from left to right.

(d) $U \rightarrow c \mid d \mid o \mid s \mid r \mid a \mid e \mid m \mid y \mid w \mid b \mid z \mid x$ means scan with continuous raster or diagonal or continuous orthogonal or spiral out or raster or right orthogonal or diagonal parallel or horizontal symmetry or diagonal symmetry or diagonal secondary or block or zeta or xi, respectively. These scan patterns are shown in Fig. 1.

(e) $V \rightarrow B \mid Z \mid X$ means partition with letter B or letter Z or letter X, respectively.

(f) $T \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7$ means use one of the eight transformation with scan or partition. For partition, these transformations are shown in Fig. 2. For scan, these transformations are defined as follows. For all scan patterns, 0 means identity transformation shown in Fig. 1, and 2 means 90° clockwise rotation. For scan patterns $c, o, s, a, e, m, y, w, b,$ and x, 4 means 180° clockwise rotation and 6 means 270° clockwise rotation. For scan patterns r and z, 4 means vertical reflection and 6 means vertical reflection followed by 90° clockwise rotation. For scan pattern d, 4 means 90° clockwise rotation followed by horizontal reflection and 6 means 180° clockwise rotation followed by vertical reflection. For all scan patterns, 1, 3, 5, and 7 are reverses of scanning paths specified by 0, 2, 4, and 6, respectively.

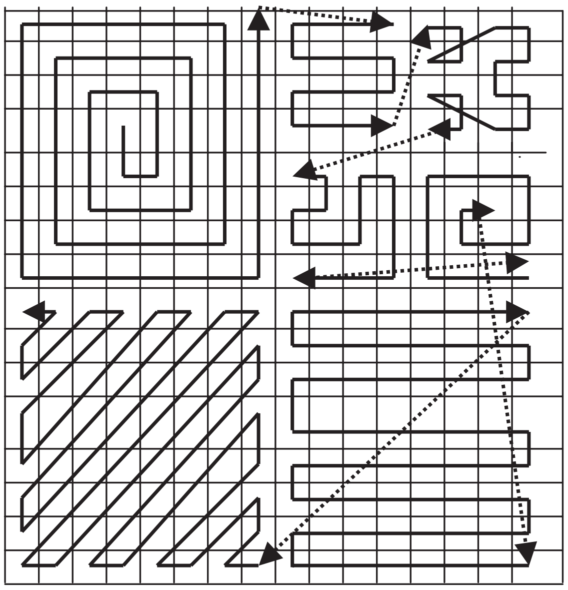

Fig. 3. Example of scan key pattern—$B5(s2Z0(c5\ b0\ o0s5)\ c4\ d1)$.

As an example, consider the scan key $B5(s2\ Z0(c5\ b0\ o0\ s5)\ c4\ d1)$ for a 16×16 image. The scanning path which corresponds to this scan key is shown in Fig. 3. The image is first partitioned into four subregions using $B5$

partition order. These four subregions are scanned using $s2$, $Z0(c5\ b0\ o0\ s5)$, $c4$, and $d1$. The second subregion is further partitioned into four subregions using $Z0$ partition order and these four subregions are scanned using $c5$, $b0$, $o0$, and $s5$.

3. Image encryption

The basic idea of the proposed image encryption method is to rearrange the pixels of the image and change the pixel values. The pixel rearrangement is done by scan keys. The pixel values are changed by a simple substitution mechanism which adds confusion and diffusion properties to the encryption method. The permutation and substitution operations are applied in intertwined and iterative manner. First, the encryption algorithm is described in detail. Next experimental results are shown to prove various properties of the encryption method which include pixel rearrangement, confusion, and diffusion properties. Finally, various extensions of the encryption method and the size of the encryption key space are discussed.

3.1. Encryption algorithm

The encryption is done by *Encrypt*() function which is described next and illustrated in Fig. 4.

```
$Encrypt(I, N, k_1, k_2, p, m, J)$
Inputs: Image $I$, Image size $N \times N$ ($N = 2^n, n \geqslant 2$),
        Encryption keys $k_1$ and $k_2$, Random seed
        integer $p$, Number of encryption iterations $m$
Output: Encrypted image $J$
{
  Let $A$, $D$, $G$ be two-dimensional arrays of size
  $N \times N$ and let $B$, $C$, $E$, $F$, $R$ be one-dimensional
  arrays of length $N \times N$
  Generate $N \times N$ random integers between 0
  and 255 using random seed $p$ and assign to $R$
  Copy $I$ into $A$
  Repeat $m$ times
  {
      Read pixels of $A$ using key $k_1$ and write into $B$
      $C[1] = B[1]$, $C[j] = (B[j] + ((C[j-1]+1)R[j])$
      $\bmod 256) \bmod 256$ for $2 \leqslant j \leqslant N \times N$
      Read pixels of $C$ and write into $D$ using spiral key $s0$
      Read pixels of $D$ using diagonal key $d0$ and write
      into $E$
      $F[1] = E[1]$, $F[j] = (E[j] + ((F[j-1]+1)R[j])$
      $\bmod 256) \bmod 256$ for $2 \leqslant j \leqslant N \times N$
      Read pixels of $F$ and write into $G$ using key $k_2$
  }
  Copy $G$ into $J$ and return $J$
}
```

The encryption key actually consists of four components, namely, the two scan keys k_1 and k_2, the random seed integer p, and the number of encryption iterations m. These four encryption key components are known to both the sender and the receiver before the communication of encrypted image. The random numbers needed by *Encrypt*() can be obtained by any method such as linear congruential generator with seed p. The encryption algorithm uses four scan keys to increase the complexity of pixel rearrangement. The keys k_1 and k_2 are specified by the user as part encryption key. The other two keys spiral $s0$ and diagonal $d0$ (shown in Fig. 1) are fixed as part of encryption algorithm. These two keys $s0$ and $d0$ are chosen because they have opposite directions of scanning and hence increase the complexity of pixel rearrangement caused by the user specified keys k_1 and k_2.

There are two fundamental properties which every secure encryption method must satisfy [17,18]. The first is confusion property which requires that ciphertexts (encrypted data) have random appearance (uniformly distributed pixel values). The second is diffusion property with respect to plaintexts (original data) and keys which requires that similar plain texts produce completely different ciphertexts when encrypted with the same key and similar keys produce completely different ciphertexts when encrypting the same plaintext. The proposed encryption method satisfies both confusion and diffusion properties.

The confusion and diffusion properties are achieved by transforming the sequence B into sequence C using $C[j] = (B[j] + ((C[j-1]+1)R[j]) \bmod 256) \bmod 256$, and similarly, sequence E into sequence F. The sequence C gets uniformly distributed pixel values because uniform random sequence R is used in the transformation. Therefore, G also gets uniformly distributed pixel values and gets confusion property. The sequence C gets diffusion property because, a single change in value $B[j]$ changes $C[j]$ which changes $C[j+1]$ which changes $C[j+2]$ and changes propagate up to the end of the sequence $C[N \times N]$. It can be shown that a single pixel change in A causes all pixels in G to be changed in one iteration as follows. Suppose a single pixel is changed in A. Then the corresponding pixel at some location j in B changes. Then all pixels between j and $N \times N$ in C change. Then the corresponding pixels in D including the top left pixel change (because spiral scan ends at top left corner). Then the corresponding pixels in E including the first pixel change (because diagonal scan begins at top left corner). Then all pixels in F change and then all pixels in G change. A single change in encryption scan key also changes all pixels in G in one iteration, because a change in scan key causes at least one pixel at some location in B to be changed which causes all pixels in G to be changed as shown above.

The decryption is done by reversing the operations of encryption. Note that the decryption requires the encryption key which consists of k_1, k_2, p and m. Decryption is done as follows. Read pixels of G using key k_2 and write into F. Then, transform F into E by $E[1] = F[1]$,

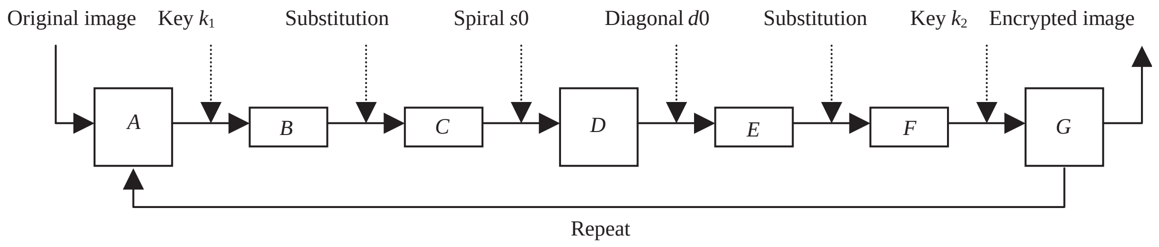

Fig. 4. Illustration of encryption.

$E[j] = (F[j] - ((F[j-1] + 1)R[j]) \bmod 256) \bmod 256$ for $2 \leqslant j \leqslant N \times N$. Then read pixels of E and write into D using diagonal scan $d0$. Then read pixels of D using spiral scan $s0$ and write into C. Then transform C into B by $B[1] = C[1]$, $B[j] = (C[j] - ((C[j-1] + 1)R[j]) \bmod 256) \bmod 256$ for $2 \leqslant j \leqslant N \times N$. Then read pixels of B and write into A using key k_1. Repeat this process m times to get the decrypted image. Note that the random array R is obtained with random seed p.

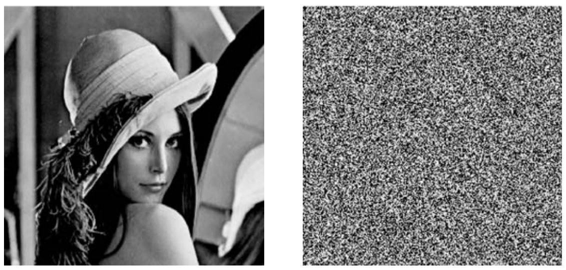

Fig. 5. Encryption of Lena image using *Encrypt*().

3.2. Experimental results

Several experiments were conducted to test various properties of the proposed image encryption method which include pixel rearrangement, confusion, and diffusion properties. Note that in all the following experiments, a fixed sequence of random numbers was generated by the C language library random number generator with seed 100 and used in the *Encrypt*() function. Note also that all images are of size 256 × 256.

(a) The *Encrypt*() algorithm was first used to encrypt Lena image with encryption scan keys $B2(x0\ y5\ s6\ r3)$, $c5$ and number of encryption iterations 5. The original and encrypted images are shown in Fig. 5.

(b) In order to determine how well *Encrypt*() rearranges the pixels, the confusion and diffusion parts (i.e. transforming B to C and E to F) of *Encrypt*() were eliminated and only rearrangement parts of *Encrypt*() were used with keys $B2(x0\ y5\ s6\ r3)$, $c5$. The encrypted images of Lena after 1, 2, and 5 iterations are shown in Fig. 6. The pixels are rearranged in completely random looking manner in just a few iterations.

(c) In order to determine how well *Encrypt*() disperses a small region, a white image with a 10 × 10 black block was encrypted (without confusion and diffusion) using *Encrypt*() with scan keys $B2(x0\ y5\ s6\ r3)$, $c5$ and number of iterations 10. The original and encrypted images are shown in Fig. 7. The pixels of the block are dispersed all over the image after encryption.

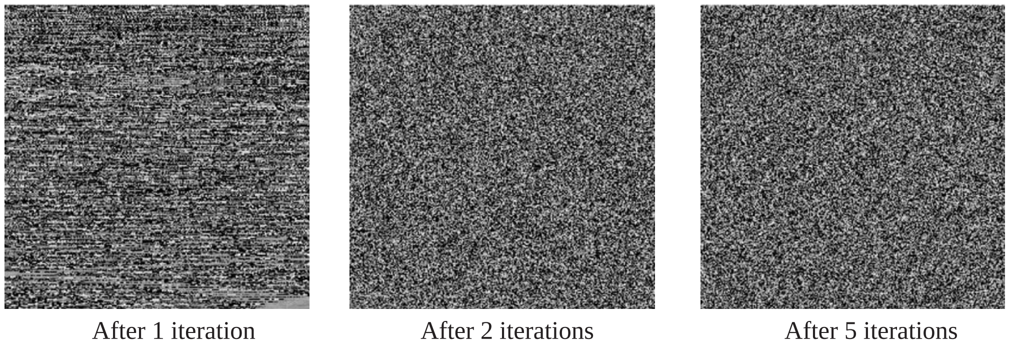

Fig. 6. Pixel rearrangement property of *Encrypt*().

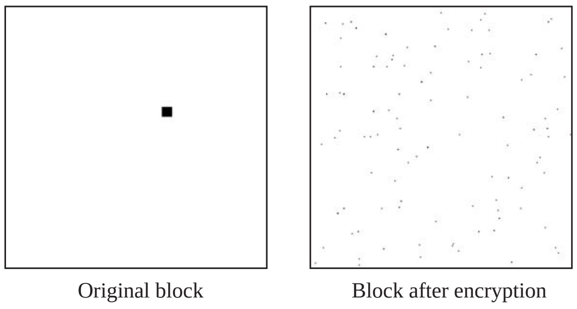

Fig. 7. Dispersion of a small block.

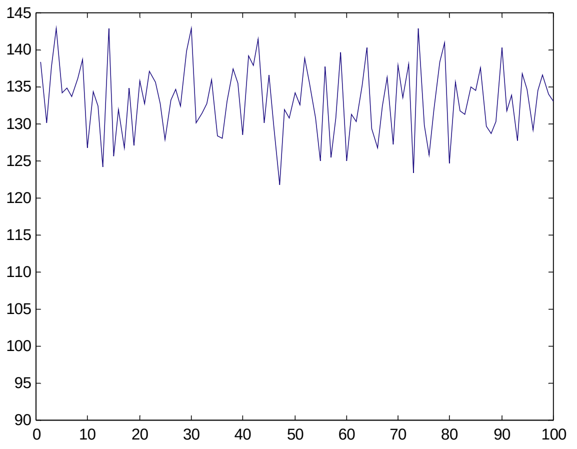

Fig. 9. Spread of blocks at 100 random locations with random keys.

(d) In order to measure the dispersion at a point and at a block, two measures are defined as follows:

$$PSpread_{k,n}(p) = (\Sigma dist(E(p), E(q)))/|N(p)|, q\varepsilon N(p),$$

$$BSpread_{k,n}(B) = (\Sigma PSpread_{k,n}(p))/|B|, p\varepsilon B,$$

where p is any point in the original image, $N(p)$ is the set of neighboring points of p in the original image, k is a set two encryption scan keys, n is the number of encryption iterations, $E()$ function is the permutation induced by scan key k and iteration n, $dist()$ is the Euclidean distance function, B is any region in the original image, and $|\ |$ is the magnitude in number of pixels. The spread of the 10×10 block in Fig. 7 was computed for various iterations with scan keys $B2(x0\ y5\ s6\ r3)$, $c5$ and the spread versus iteration plot is shown in Fig. 8. The spread increases significantly during the first few iterations and then randomly oscillates between 120 and 150.

(e) In order to determine whether large spreads occur for blocks at any locations and with any scan keys, 10×10 blocks were chosen at 100 random locations and at each location two random scan keys were generated, and spread was computed for these blocks and keys after 10 iterations. The plot of spread versus 100 random locations and keys is shown in Fig. 9. Large spreads between 120 and 150 occur regardless of locations or keys.

(f) In order to determine the confusion property of *Encrypt*(), the Jet image and a pure black image were encrypted using *Encrypt*() with scan keys $B2(x0\ y5\ s6\ r3)$, $c5$ and number of iterations 10. The original and encrypted images and their histograms are shown in Fig. 10. The encrypted images have uniform histograms regardless of the original images, thus proving the confusion property of *Encrypt*().

(g) In order to determine the diffusion property of *Encrypt*() with respect to images, the jet image was modified by incrementing the value of one randomly chosen pixel by 1. The value of pixel (100, 23) was incremented from 239 to 240. Both the original Jet and modified Jet were encrypted using *Encrypt*() with keys $B2(x0\ y5\ s6\ r3)$, $c5$ and number of iterations 10.

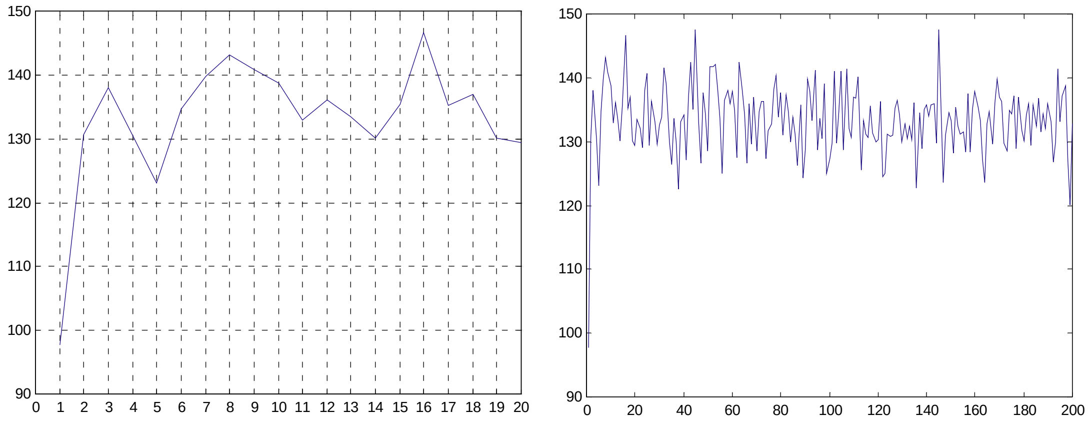

Fig. 8. Spread of a block versus iterations (first 20 and 200 iterations).

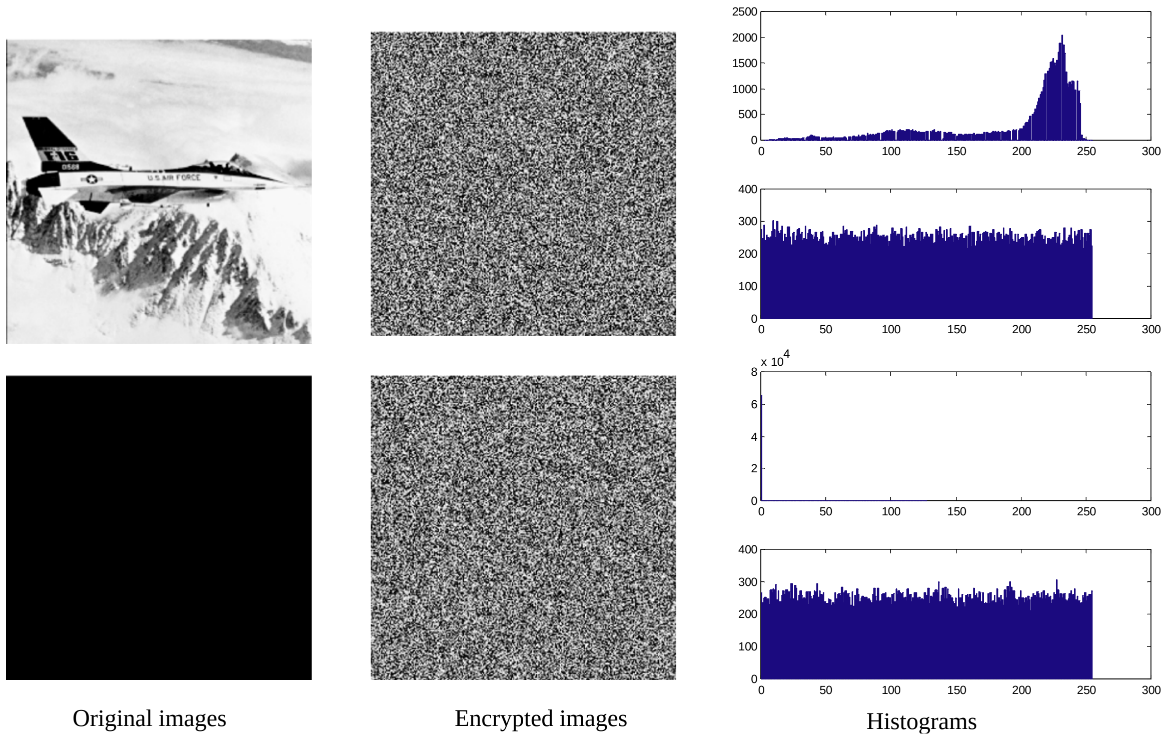

Fig. 10. Confusion property of *Encrypt*().

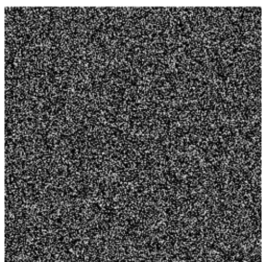

Fig. 11. Diffusion property of *Encrypt*() with respect to image.

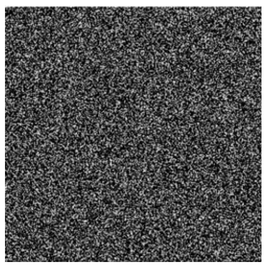

Fig. 12. Diffusion property of *Encrypt*() with respect to key.

Fig. 11 shows the pixelwise difference of two encrypted images which shows that the two encrypted images have no similarities even though their original images differ by only one pixel, thus proving the diffusion property of *Encrypt*() with respect to images.

(h) In order to determine the diffusion property of *Encrypt*() with respect to encryption keys, the Jet image was encrypted using *Encrypt*() with two different pairs of keys $B2(x0\ \ y5\ \ s6\ \ r3)$, $c5$ and $B2(x0\ \ y5\ \ s6\ \ r4)$, $c5$ which differ only by one digit, and number of] iterations 10. Fig. 12 shows the pixelwise difference of two encrypted images which shows that the two encrypted images have no similarities even though the same original image was encrypted by two encryption keys which differ by only one digit, thus proving the diffusion property of *Encrypt*() with respect to keys.

(i) In order to determine the effect of small difference between encryption and decryption keys, the Jet image was encrypted with keys $B2(x0\ \ y5\ \ s6\ \ r3)$, $c5$ and decrypted with keys $B2(x0\ \ y5\ \ s6\ \ r4)$, $c5$ in 10 iterations. The resulting images are shown in Fig. 13. Even though encryption and decryption keys differ by only one digit, the decrypted image is completely different from original image. Thus, having an almost perfect guess of the encryption key makes decryption impossible.

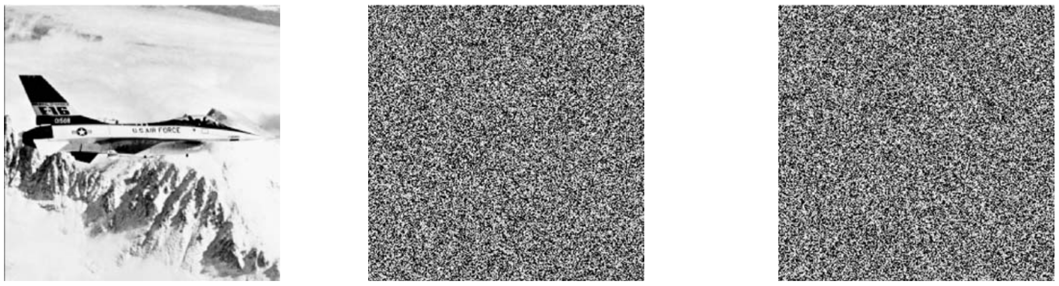
Original image Encrypted with $B2(x0y5s6r3),c5$ Decrypted with $B2(x0y5s6r4),c5$

Fig. 13. Effect of small difference between encryption and decryption key.

Table 1
Number of encryption keys

Image size	Number of encryption keys is greater than
4×4	10^{4}
8×8	10^{18}
16×16	10^{76}
32×32	10^{300}
64×64	10^{1200}
128×128	10^{4800}
256×256	10^{19000}
512×512	10^{76000}
1024×1024	10^{304000}

3.3. Number of encryption keys

Let $S(n)$ be the number of scan patterns of a $2^n \times 2^n$ two-dimensional array generated by the encryption-specific SCAN defined in Section 3.2. Let $T(n)$ be the number encryption key pairs which can be used to encrypt a $2^n \times 2^n$ image. Then $S(n)$ and $T(n)$ are given by the formulae below. Table 1 illustrates the magnitude of number of encryption key pairs. From the table, it is clear that attack by searching the encryption key pairs exhaustively is impossible.

$$S(2) = 104,$$

$$S(n) = 104 + 24(S(n-1))^4 \quad \text{for } n \geqslant 3,$$

$$T(n) = (S(n))^2 \quad \text{for } n \geqslant 2.$$

The above formulae can be derived as follows. For a $2^n \times 2^n$ $n \geqslant 2$ image, there are 13 basic scan patterns shown in Fig. 1 each with eight transformations resulting in 104 basic scan-transformation patterns. When $n \geqslant 3$, there are additionally 24 ways shown in Fig. 2 to partition the image into subregions of size $2^{n-1} \times 2^{n-1}$ each having $S(n-1)$ scan patterns recursively. This results in $S(2)=104$, $S(n)=104+24(S(n-1))^4$ $n \geqslant 3$. A scan key pair has two scan keys each of which can be any of $S(n)$ scan patterns, resulting in $T(n) = (S(n))^2$.

3.4. Extensions of encryption method

The encryption method can be extended to encrypt any one-dimensional array of bytes or digital data as follows. Decompose the one-dimensional array into one-dimensional arrays of length 2^{2n} $n \geqslant 2$ and transform decomposed one-dimensional arrays into two-dimensional arrays of size $2^n \times 2^n$ and apply *Encrypt*() to these $2^n \times 2^n$ two-dimensional arrays and transform encrypted two-dimensional arrays into one-dimensional arrays and append them to obtain the final encrypted image. Note that the transformation between one- and two-dimensional arrays can be done by simply scanning two-dimensional arrays in raster scan order. Note also that if decomposition produces a remainder array of length less than 16 then this small array can be kept not encrypted and appended to other encrypted parts.

A rectangular image can be encrypted by first transforming it into one-dimensional array of pixels and then applying the above procedure. Similarly, rectangular part(s) of an image can be encrypted. Binary files such as executable programs can be encrypted similarly. A compressed image can be encrypted by transforming compressed bit stream into byte stream and then applying the above procedure. Color images with k-bit pixels can be encrypted by modifying the substitution parts of *Encrypt*() with random number range 0 to $2^k - 1$ and modulus value 2^k.

4. Video encryption

The basic idea of the proposed video encryption method is to find the difference between adjacent frames and compress and encrypt these frame differences. Frame difference operation is lossy resulting in lossy video compression. But the maximum pixelwise difference between the original and recovered video can be chosen by the user. The video encryption is illustrated in Fig. 14. The video encryption algorithm and experimental results are described in the following sections.

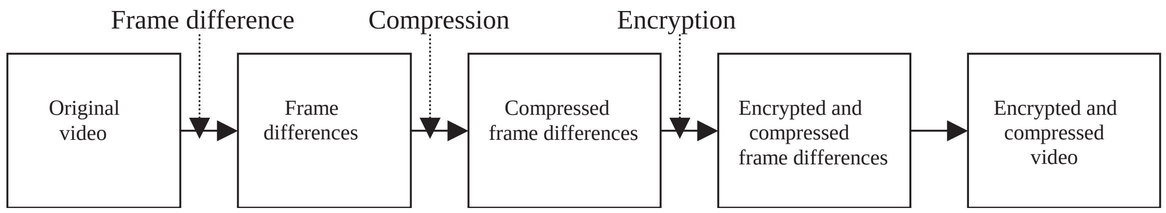
Frame difference
Compression
Encryption
Original video
Frame differences
Compressed frame differences
Encrypted and compressed frame differences
Encrypted and compressed video

Fig. 14. Illustration of video encryption.

4.1. Encryption algorithm

The video encryption is done by *VideoEncrypt*() function. This function and its auxiliary functions are described next. In all these functions, symbols N, w, and h denote number of video frames, frame width, and frame height, respectively. The symbols m, n, and k denote number of bits needed to represent x-, y-coordinates, and pixels of frames, respectively. The symbol $B_{p,q}$ denotes binary representation of integer p in q bits. $C = Append(A,B)$ means appending arrays A and B and assigning the result to C.

```
$VideoEncrypt(V,K)$
Inputs: Original video $V$, Encryption key $K$
Output: Encrypted video is sent to receiver
{
    Let $F_1 \dots F_N$ be the frames of $V$
    Let $G_1 \dots G_N = Difference(F_1 \dots F_N)$
    Set $Buffer = G_1$
    For ($i = 2$–$N$)
    {
        $Comp = Compress(G_i)$
        $Buffer = Append(Buffer, Comp)$
        If size of $Buffer \geqslant 256 \times 256$ bytes
            Encrypt first $256 \times 256$ bytes of $Buffer$ with $K$ and
            send to receiver and empty those bytes from $Buffer$
    }
    Encrypt remaining bytes in $Buffer$ and send to receiver
}
$G_1 \dots G_N = Difference(F_1 \dots F_N)$
Input: Video frames $F_1 \dots F_N$
Output: Difference frames $G_1 \dots G_N$
{
    Let $G_1 = F_1$, $T = F_1$
    Let $TH$ = user specified value
    For ($k = 2$–$N$)
    {
        For $1 \leqslant i \leqslant h$, $1 \leqslant j \leqslant w$
            If $|F_k[i][j] - T[i][j]| \leqslant TH$
                $G_k[i][j] = -1$
            Else
                $G_k[i][j] = F_k[i][j]$
                $T[i][j] = G_k[i][j]$
    }
    Return $G_1 \dots G_N$
}
```

```
$X = Compress(D)$
Input: Difference frame $D$
Output: Compressed difference frame $X$
{
    $X = \Phi$, $Count = 0$
    Decompose $D$ into $4 \times 4$ windows
    For each window $W$
        If $W$ has an element other than $-1$
            $X = Append(X, EncodeWindow(W))$
            $Count = Count + 1$
    $X = Append(B_{Count,m+n-4}, X)$
    Return $X$
}
$U = EncodeWindow(W)$
Input: $4 \times 4$ window $W$ from difference frame
Output: Encoded window $U$
    Let $U = \Phi$ and let $(x, y)$ be the top left corner of $W$
    $U = Append(U, B_{x/4,m-2})U = Append(U, B_{y/4,n-2})$
    For each pixel $p$ in $W$ in raster order
        If $p$ equals $-1$
            $U = Append(U, 0)$
        Else
            $U = Append(U, 1)$, $U = Append(U, B_{p,k})$
    Return $U$
}
```

Note that *Difference*() function involves lossy operation which makes the video compression lossy. But the *Difference*() function guarantees that the magnitude of difference between a pixel value in the original video and its corresponding pixel value in the recovered video at the receiver is at most threshold value TH which is chosen by the user depending on the application. Higher threshold value produces higher compression ratio and more degradation of video. Fig. 15 illustrates compression of a 16×16 frame difference. In this example, compression ratio between the original frame and compressed frame difference is $(16 \times 16 \times 8)$:$(4 + 68 + 36 + 32)$ or about 8:1. By encrypting the compressed frame differences, both fast encryption and compression are achieved. To recover the video at the receiver, the compressed-encrypted video is first decrypted to obtain the compressed video which is then decompressed to obtain the frame differences. These frame differences and the first frame are then used to reconstruct the video. Note that the parameters N, w, h, m, n, k (defined

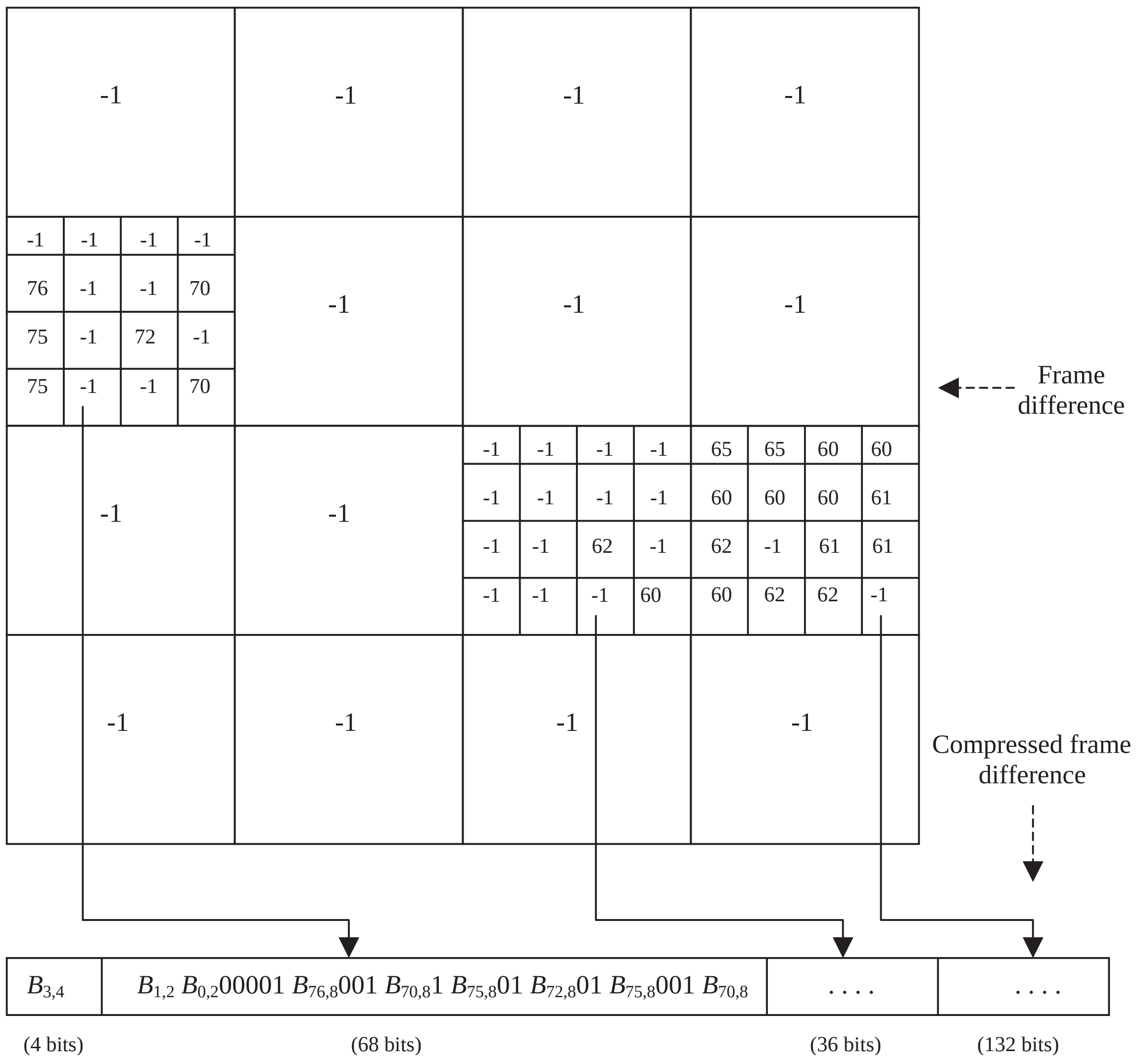
Frame difference
Compressed frame difference
$B_{3,4}$
$B_{1,2}$ $B_{0,2}$00001 $B_{76,8}$001 $B_{70,8}$1 $B_{75,8}$01 $B_{72,8}$01 $B_{75,8}$001 $B_{70,8}$
. . . .
. . . .
(4 bits)
(68 bits)
(36 bits)
(132 bits)

Fig. 15. Illustration of compression of frame difference.

at the beginning of Section 4.1) are sent to the receiver separately before the transmission of compressed-encrypted video.

4.2. Experimental results

The proposed video encryption method (which performs a first stage compression before encryption, see Fig. 15) was tested with several videos. The compression scheme here is based on the differences between consecutive frames, and it does not include the lossless compression scheme obtained by the SCAN methodology [24]. Fig. 16 shows representative frames of four such videos each with 20 consecutive frames. Table 2 shows the compression percentage, i.e. ((original size − compressed size)/(original size))×100 and the mean square error between original and recovered videos for various threshold values. Fig. 17 shows five frames from Claire video and compressed-encrypted video of all 20 frames and corresponding five recovered frames when threshold value is 5.

Note that the threshold value is the maximum allowable pixelwise difference between original and recovered videos whereas the mean square error measures the average difference between original and recovered videos. Higher threshold values produce higher compression and more degradation of videos, but compression grows faster than average error. For example, on the average, threshold value 5 produces about 78% video compression with average error only about 1.67 (and maximum error 5).

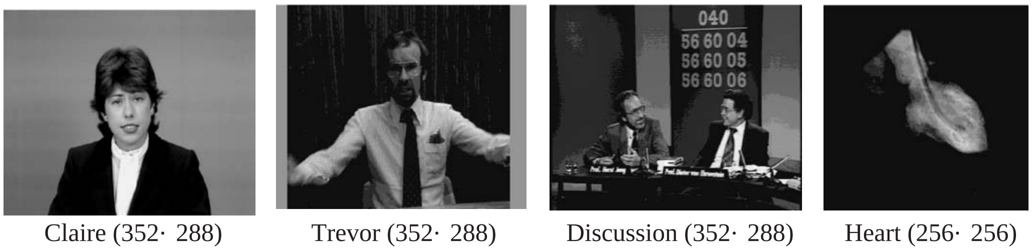

Fig. 16. Test videos.

Table 2
Compression and mean square error of videos versus threshold values

Threshold value	Claire		Trevor		Discussion		Heart	
	Comp (%)	Error	Comp (%)	Error	Comp (%)	Error	Comp (%)	Error
0	11.05	0.0000	3.74	0.0000	4.82	0.0000	37.33	0.0000
1	52.26	0.6363	31.73	0.5041	29.27	0.5732	45.47	0.0265
2	73.09	1.0413	48.20	0.9023	50.96	1.0693	50.24	0.4610
3	82.65	1.3416	59.99	1.1752	66.67	1.4481	73.30	1.5104
4	87.24	1.5500	65.31	1.3684	75.70	1.7228	74.65	1.6293
5	89.60	1.7065	67.79	1.5469	80.02	1.9384	77.32	1.8410
6	90.96	1.8374	69.57	1.7417	82.38	2.1317	85.19	2.4320
7	91.89	1.9648	71.12	1.9534	83.95	2.3177	86.92	2.6047
8	92.61	2.0912	72.52	2.1785	85.25	2.5058	88.01	2.8732
9	93.24	2.2191	73.83	2.4140	86.30	2.6978	89.49	2.9241
10	93.77	2.3532	75.07	2.6637	87.21	2.8995	90.17	3.2005

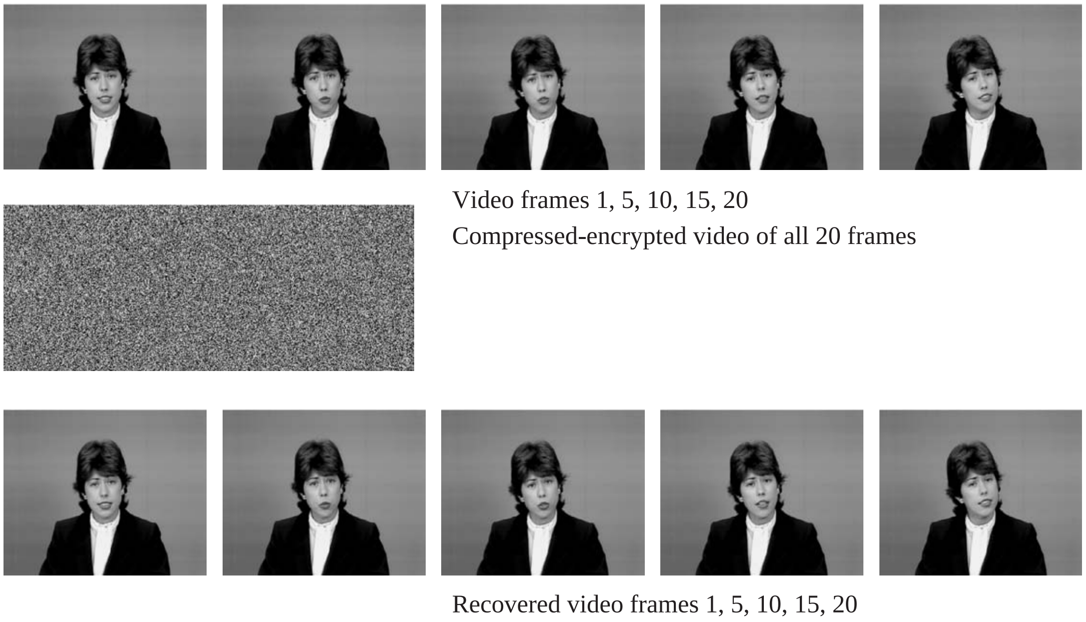
Video frames 1, 5, 10, 15, 20
Compressed-encrypted video of all 20 frames
Recovered video frames 1, 5, 10, 15, 20

Fig. 17. Compression encryption of Claire video.

5. Conclusions

This paper presented new methods for image and video encryption based on the SCAN methodology. The new implementation of the SCAN methodology proposed here has eliminated the criticisms made by some authors [20,21] of the first version of the SCAN methodology implemented in 1988 [1,22]. In addition, the generalized G-SCAN method represents the most complete version of the SCAN family of languages, but its complexity is too high for a regular use [3]. Summarizing the main characteristics of the proposed encryption methods are:

(a) Lossless encryption of image.
(b) Lossy compression and encryption of video whose pixelwise compression error is bounded by user specified value.
(c) Number of encryption keys is larger than that of most existing image or video encryption methods, and encryption keys have variable lengths.
(d) Encryption method belongs to the framework of iterated product cipher which has been well studied and developed.
(e) Confusion and diffusion properties are satisfied, and almost perfect guess of encryption key makes decryption impossible.
(f) Capability to encrypt large blocks of any one-dimensional digital data.
(g) Encryption uses only integer arithmetic and it can be easily implemented in hardware.

The video compression method is particularly useful when it is important to control or guarantee that pixelwise difference between original and reconstructed video is always less than a user specified value. The compression works well when there is little and smooth change between adjacent frames. These factors and encryption capability make the proposed video compression-encryption method especially useful in applications such as telemedicine. Even though lossless compression can be achieved by setting the threshold value to 0, this approach produces only low compression. Future work includes incorporating existing SCAN-based lossless image compression method to achieve lossless video compression and encryption.

It is important to be mentioned here that the complexity of the SCAN methodology is of the order of $O(n^2 + \log_2 n)$, for images of $n \times n$ pixels.

The software implementation (in C and Pentium II) takes about 3 s to encrypt a grayscale image in Section 3.2 and about 5 s to encrypt a 20-frames video in Section 4.2 and about same amount of time for decryption as encryption. The software implementation is not suitable for real-time applications. Both image and video encryption algorithms have been implemented in hardware by using FPGAs. Thus, the use of FPGAs eliminates the disadvantage and make the methodology a real-time system for encryption of images and video.

Acknowledgements

This work is supported by AIIS Inc. and a patent has been filled.

References

[1] N. Bourbakis, C. Alexopoulos, Picture data encryption using SCAN patterns, Pattern Recognition J. 25 (6) (1992) 567–581.
[2] C. Alexopoulos, N. Bourbakis, N. Ioannou, Image encryption method using a class of fractals, J. Electron. Imag. 4 (1995) 251–259.
[3] N. Bourbakis, Image data compression encryption using G-SCAN patterns, IEEE Conference on SMC, Orlando, FL, October 1997, pp. 1117–1120.
[4] J. Scharinger, Fast encryption of image data using chaotic Kolmogorov flows, J. Electron. Imag. l7 (2) (1998) 318–325.
[5] F. Pichler, J. Scharinger, Finite dimensional generalized Baker dynamical systems for cryptographic applications, Lecture Notes in Computer Science, Vol. 1030, Springer, Berlin, 1995, pp. 465–476.
[6] L. Chang, Large encrypting of binary images with higher security, Pattern Recognition Lett. 19 (5) (1998) 461–468.
[7] X. Li, Image compression and encryption using tree structures, Pattern Recognition Lett. 18 (11) (1997) 1253–1259.
[8] H. Chang, J. Liu, A linear quadtree compression scheme for image encryption, Signal Process.: Image Commun. 10 (4) (1997) 279–290.
[9] S. Changgui, B. Bhargava, A fast MPEG video encryption algorithm, Proceedings of ACM Multimedia Conference, 1998, pp. 81–88.
[10] T. Lei, Methods for encrypting and decrypting MPEG video data efficiently, Proceedings of ACM Multimedia Conference, 1996, pp. 219–229.
[11] L. Qio, K. Nahrstedt, A new algorithm for MPEG video encryption, First International Conference on Imaging Science, Systems, and Technology, 1997, pp. 21–29.
[12] T. Chuang, J. Lin, New approach to image encryption, J. Electron. Imag. 4 (1998) 350–356.
[13] X. Wu, P. Moo, Joint image/video compression and encryption via high order conditional entropy coding of wavelet coefficients, Proceedings of IEEE International Conference on Multimedia Computing and Systems, 1999, pp. 908–912.
[14] T. Chuang, J. Lin, A new multiresolution approach to still image encryption, Pattern Recognition Image Anal. 9 (3) (1999) 431–436.
[15] C. Kuo, Novel image encryption technique and its application in progressive transmission, J. Electron. Imag. 2 (1993) 345–351.
[16] B. Macq, J. Quisquater, Cryptology of digital TV broadcasting, Proc. IEEE 83 (6) (2000) 944–957.
[17] B. Schneier, Applied Cryptography, Wiley, New York, 1993.

[18] G. Brassard, Modern Cryptology, Springer, New York, 1988.
[19] S. Landau, Standing the test of time: the data encryption standard, Notices of American Mathematical Society, March 2000, pp. 341–349.
[20] S. Li, X. Zheng, Cryptanalysis of a chaotic image encryption method, IEEE International Symposium on Circuits and Systems, Arizona, May 26–29, 2002.
[21] H. Cheng, X. Li, Partial encryption of compressed images and videos, IEEE Trans. Signal Process. 48 (8) (2000) 2439–2451.
[22] C. Alexopoulos, SCAN a formal methodology for accessing and processing 2D data, Ph.D. Dissertation, Department of Computer Engineering and Informatics, University of Patras, Greece, 1988.
[23] N. Bourbakis, C. Alexopoulos, A fractal based image processing language—formal modeling, Pattern Recognition J. 32 (2) (1999) 317–338.
[24] S. Maniccam, A new methodology for compatible compression, encryption and information hiding of image and video, Ph.D. Thesis, BU-EE, IPMV-Lab, 2000.

About the Author—DR. NIKOLAOS BOURBAKIS received his B.S. degree in Mathematics from the National University of Athens, Athens, Greece, and his Ph.D. in Computer Science and Computer Engineering, from the Department of Computer Engineering & Informatics, University of Patras, Patras, Greece, in 1983. He currently is a Distinguished Professor in CSE and the Director of the Information Technology Research Institute (ITRI) at Wright State University, OH. Previous Academic positions: Associate Director of the Center on Intelligent Systems, Director of the Image-Video- Vision & Applied AI Research Lab, Professor of Electrical Engineering with joint appointment to the Computer Science Department at SUNY-Binghamton, Professor and Lab Director at Technical University of Crete, Greece, S. Scientist at IBM S. Jose CA and Assistant Professor at GMU. Dr. Bourbakis' industrial experience includes service to IBM, CA and Soft Sight, NY. He is the founder and Vice President of the AIIS, Inc., NY. He contacts research in Applied AI, Machine Vision, Bioinformatics/Bioengineering, Information Security, and Parallel/Distributed Processing funded by USA and European government and industry. He has published more than 230 articles in refereed International Journals, book-chapters and Conference Proceedings, and 10 books as an author, co-author or editor. He has graduated 10 Ph.D.s and 30 Masters students. He is the founder and the EIC of the *International Journal on AI Tools*, the EIC of a new *International Journal on Computational Bioinformatics Engineering* (upcoming) the Editor-in-Charge of a *Research Series of Books in AI* (WS Publisher), the Editor-in-Charge of a *Research Series of Books in Bioinformatics* (upcoming, KAP Publisher), the Founder and General Chair of several International IEEE Computer Society Conferences, Symposia and Workshops (*Tools with AI*, *Intelligence in Neural and Biological Systems*, *Intelligence*, *Image*, *Speech and Natural Language Systems*, *Information and Systems*, *Bioinformatics and Bioengineering*, etc). He is or was an Associate Editor in *IEEE Trans. on KDE*, IEEE Multimedia and in International Journals (*EAAI*, *PRAI*, *PR*, *PAA*, *ISR*, *COIS*, etc.) and a Guest Editor in 13 special issues in IEEE and International Journals related to his research interests. He is an IEEE Fellow, a distinguished IEEE Computer Society Speaker, an NSF University Research Programs Evaluator, an IEEE Computer Society Golden Core Member, an External Evaluator in University Promotion Committees, an Official Nominator of the National Academy of Achievements for Computer Science Programs, and a keynote speaker in several International Conferences. He is also listed in many organizations (WHO'S WHO in Engineering, in Science, in Education, in Intellectuals, in Computer Engineering, AMWS, List of Distinguished Editors, etc.). His research work has been internationally recognized and has won several prestigious awards. Some of them are: IBM Author recognition Award 1991, IEEE Computer Society Outstanding Contribution Award 1992, IEEE Outstanding Paper Award ATC 1994, IEEE Computer Society Technical Research Achievement Award 1998, IEEE I& S Outstanding Leadership Award 1998, IEEE ICTAI 10 years Research Contribution Award 1999.

About the Author—DR. S. MANICCAM, received his B.S. and M.S. degrees in Mathematics from the State University of New York, and his Ph.D. in Computer Science and Engineering, from the Department of Computer Engineering Department, SUNY, 2001. He currently is an Assistant Professor in CS Department at Eastern Michigan University, MI. His research interests are on image processing, complex systems, formal methods. He has published several Journal and Conference papers.