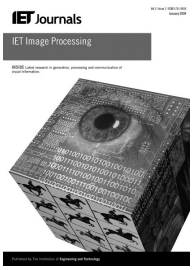
Lossless chaos-based crypto-compression scheme for image protection

Atef Masmoudi, William Puech

Published in IET Image Processing
Received on 26th September 2012
Revised on 17th January 2014
Accepted on 23rd February 2014
doi: 10.1049/iet-ipr.2013.0598



ISSN 1751-9659

Lossless chaos-based crypto-compression scheme for image protection

Atef Masmoudi[1], *William Puech*[2]

[1]*Department of Mathematics and Computer Science, Sfax Preparatory Engineering Institute, University of Sfax, 3018 Sfax, Tunisia*
[2]*LIRMM, UMR CNRS 5506, University of Montpellier II, 34392 MONTPELLIER CEDEX 05, Montpellier, France*
E-mail: atef.masmoudi@lirmm.fr

Abstract: In this study, the authors proposed a new scheme which performs both lossless compression and encryption of images. Lossless compression is done by arithmetic coding (AC) while encryption is based on a chaos-based pseudorandom bit generator. Hence, they proposed to incorporate recent results of chaos theory into AC in order to shuffle the cumulative frequency vector of input symbols chaotically to make AC secure and the decoding process completely key-dependent. Many other techniques based on varying the statistical model used by AC have been proposed in literature, however, these techniques suffer from losses in compression efficiency that result from changes in entropy model statistics and are weak against known attacks. The proposed compression–encryption techniques were developed and discussed. The numerical simulation analysis indicates that the proposed scheme is highly satisfactory for image encryption without any AC compression efficiency loss. In addition, it can be incorporated into any image compression standard or algorithm employing AC as entropy coding stage, including static, adaptive and context-based adaptive models, and at any level, including bit, pixel and predictive error pixel levels.

1 Introduction

In this paper, the combination of compression and encryption by using arithmetic coding (AC) and chaotic systems for image protection was particularly given much interest. Several chaotic systems have been recently used for image encryption [1–6] and some of these novel chaotic systems have designed pseudorandom bit generators (PRBGs) for stream cipher applications [7–9] since they have interesting cryptographic properties such as ergodicity, sensitivity to initial values and control parameters.

The AC has been widely used as an efficient entropy coding technique by most interesting image and video standards such as JBIG2, JPEG2000 and H.264/AVC, and by a variety of lossless data compression methods operating in bit- [10, 11], pixel- [12–15] or predictive error pixel-level [16–18].

The AC is also considered for data encryption. Grangetto *et al.* [19] proposed a novel multimedia security framework by means of the randomised AC (RAC). This framework is based on a random organisation of the encoding intervals using a secret key. A chaos-based zeroth-order adaptive AC (AAC-0) [20] technique was proposed. The statistical model is made varying in nature according to a coupled chaotic systems-based PRBG which makes AC compression efficiency loss of about 6%. Wen *et al.* [21] designed the binary AC (BAC) with key-based interval splitting. They proposed to use a key to split the interval associated with the symbol to be encoded. Thus, the traditional assumption in AC that a single continuous interval is used for each symbol is not preserved. The repeated splitting at each encoding/decoding step allows both encryption and compression. Kim *et al.* [22] demonstrated the insecurity of the interval splitting AC against a known plain-text attack and a chosen plain-text attack. They also provided an improved version called the secure AC applying a series of permutations at the input symbol sequence and output codeword. It should be noticed that because of the permutations process, it is difficult to extend the secure AC to the adaptive and context-based adaptive models which exploit the input symbol redundancy to encode messages. Mi *et al.* [23] proposed a chaotic encryption scheme based on RAC using the logistic map for PRBG. Zhou *et al.* [24] presented a scheme for joint security and performance enhancement of secure AC. They proposed to incorporate the interval splitting AC [21] scheme with the bit-wise XOR operation. This scheme can be extended to any adaptive model because of the elimination of the input symbol permutation step, and its implementation is lower in complexity than the original secure AC. A secure BAC scheme based on non-linear dynamic filter with changeable coefficients was presented and discussed in [25]. In a previous work, Masmoudi *et al.* [26] proposed a joint lossless compression and encryption scheme combining BAC with a PRBG. They proposed to randomly exchange the two intervals of the cumulative distribution vector which conserves the BAC compression efficiency while using the same probabilities with and without the exchanging intervals process. Duan *et al.* [27] introduced a RAC based on order-1 Markov model with a higher compression efficiency and good security. Recently, the proof of insecurity of the RAC

based on Markov model has been proved and an improvement has been suggested in [28].

In this paper, we suggested a new scheme to achieve encryption by controlling certain operations in the AC compression algorithm using a secret key. The proposed scheme can be applied with any statistical model, including static, adaptive and context-based adaptive models, and to a variety of lossless data compression methods.

The rest of this paper is organised as follows. In Section 2, we briefly overview AC. Section 3 details the proposed AC-encryption scheme. In Section 4, we analyse the compression efficiency and the security of the proposed scheme and discuss the experimental results. Finally, our conclusion is discussed in Section 5.

2 Overview of AC

The AC [29, 30] is a very efficient statistical coding technique for data compression. The AC principle consists to assign one codeword to the entire input data symbols. To calculate the appropriate codeword for input data symbols, the AC works with a statistical model that estimates the probability of each symbol at the encoding/decoding process. The model used by AC can be static (SAC) [31, 32], adaptive (AAC) [31, 32] or context-based adaptive [33, 10]. In the static model, AC uses a fixed frequency count vector to encode all the symbols, however, in adaptive and context-based adaptive models, the AC works with a variable frequency count vector which is estimated/updated before/after encoding each symbol.

Let X be a sequence of m events $X = x_1, \ldots, x_m$ taking values from an alphabet of n symbols $\{\alpha_1, \ldots, \alpha_n\}$. We denoted by $\boldsymbol{Freq} = [f_1, \ldots, f_n]$, in Algorithm 1 (see Fig. 1) line 11, the frequency count vector where f_k represents the frequency count of the corresponding symbol α_k from the total length of the sequence X.

During the encoding process, AC firstly estimates the probability (frequency count) of each symbol and then calculates the cumulative frequency vector ***cum_freq***, in Algorithm 1 (Fig. 1) line 12, by assigning a subinterval for each symbol α_k with a size proportional to its frequency count in the interval [0, R) (R in Algorithm 1 Fig. 1) line 13). The ***cum_freq*** vector is calculated as

$$\boldsymbol{cum\_freq}[\alpha_k] = \sum_{k=1}^{\alpha_{k-1}} f_k \tag{1}$$

Next, for any new symbol α_k from the input sequence, AC selects the subinterval for α_k and defines it as the new current interval. The AC iterates this step until all input sequence would be processed and finally generates the codeword that uniquely identifies the input data, which corresponds to any number within the final coding interval.

Let range be the size of the current subinterval, range in Algorithm 2 (see Fig. 2) line 12 and Algorithm 3 (see Fig. 3) line 15. The range is always equal to the product of the probabilities of the individual symbols encoded so far, and can be mathematically defined by (2). The 'range' value monotonically decreases at each iteration and therefore the number of bits needed to specify the founded interval of that length increase. It corresponds to the likelihood function given the observed data X in a specific model M

$$\text{range} = P_M(X) = \prod_{i=1}^{m} P_M(x_i) \tag{2}$$

If we let $L(X, M)$ denotes the likelihood function given the observed data X in the model M, then $L(X, M) = \text{range}$. Therefore we can bound the number of bits required to represent the message X by the following equation [29]

$$-\left\lfloor \log_2(L(X, M)) \right\rfloor + 2 \text{ bits} \tag{3}$$

where $\lfloor \cdot \rfloor$ is the floor function. The AC is therefore most useful when there are large probabilities in the corresponding model.

Algorithm 3

1: $y_1 \leftarrow y_1^0$
2: $y_2 \leftarrow y_2^0$
3: $y_3 \leftarrow y_3^0$
4: $Top\_value \leftarrow 2^N - 1 \; //N = 16bits$
5: $First\_qtr \leftarrow Top\_value/4 + 1$
6: $Half \leftarrow 2 * First\_qtr$
7: $Third\_qtr \leftarrow 3 * First\_qtr$
8: $low \leftarrow 0$
9: $high \leftarrow Top\_value$
10: $S \leftarrow [0, ..., 255]$
11: $Freq \leftarrow [f_0, ..., f_{255}]$
12: $cum\_freq \leftarrow [0, Freq[S[1]], Freq[S[1]] + Freq[S[2]], ..., \sum_{i=1}^{n} Freq[S[i]]]$
13: $R \leftarrow \sum_{i=1}^{n} Freq[S[i]]$

Fig. 1 *Initialise_AC()*

doi: 10.1049/iet-ipr.2013.0598

Algorithm 1

1: Initialise_AC()
2: $bits\_to\_follow \leftarrow 0$
3: **for** $i \leftarrow 1$ to m **do**
4: $y_1 \leftarrow 4 * y_1 * (1 - y_1)$
5: **if** $y_1 < 0.5$ **then**
6: $y_2 \leftarrow 4 * y_2 * (1 - y_2)$
7: $j \leftarrow \lfloor y_2 * (n - 1) \rfloor + 2$
8: $S \leftarrow [S[j], S[j + 1], ..., S[n], S[1], ..., S[j - 1]]$
9: $cum\_freq \leftarrow [0, Freq[S[1]], Freq[S[1]] + Freq[S[2]], ..., \sum_{i=1}^{n} Freq[S[i]]]$
10: **end if**
11: $j \leftarrow position\_of\_symbol(S, x_i)$
12: $range \leftarrow high - low + 1$
13: $high \leftarrow low + range * cum\_freq[j + 1]/R$
14: $low \leftarrow low + range * cum\_freq[j]/R$
15: **for** (; ;) **do**
16: **if** $high < Half$ **then**
17: $y_3 \leftarrow 4 * y_3 * (1 - y_3)$
18: $output\_bit(0 \oplus (y_3 < 0.5))$
19: $low \leftarrow 2 * low$
20: $high \leftarrow 2 * high + 1$
21: **else if** $low \geq Half$ **then**
22: $y_3 \leftarrow 4 * y_3 * (1 - y_3)$
23: $output\_bit(1 \oplus (y_3 < 0.5))$
24: $low \leftarrow 2 * (low - Half)$
25: $high \leftarrow 2 * (high - Half) + 1$
26: **else if** $(low \geq First\_qtr)\ and\ (high < Third\_qtr)$ **then**
27: $bits\_to\_follow \leftarrow bits\_to\_follow + 1$
28: $low \leftarrow 2 * (low - First\_qtr)$
29: $high \leftarrow 2 * (high - First\_qtr) + 1$
30: **else**
31: $break$;
32: **end if**
33: **end for**
34: **if** adaptive **then**
35: $update\_model(x_i)$
36: **end if**
37: **end for**

Fig. 2 *AC-encryption algorithm*

3 Proposed AC-encryption scheme

To make AC secure and the decoding process completely key-dependent, we can distinguish two different methods namely:

- the model-based encryption,
- the coder-based encryption.

The techniques, which secretly modifying the statistical model, suffer from losses in compression efficiency. In fact, our work which consists in randomly permuting the symbols' positions in the cumulative frequency count vector used by AC to make it secure and the decoding process completely key-dependent is within this framework. The proposed scheme can be seen as a coder-based encryption method.

In the encryption and decryption algorithms, we suggest to use new variable $\boldsymbol{S}$ called the symbols vector, in Algorithm 1 (Fig. 1) line 10, which is initially rearranged into the ascii order. For example, for grey-scale image data, where each pixel takes a value from {0, …, 255}, the symbols vector $\boldsymbol{S}$ is initially equal to

$$\boldsymbol{S} = [0, \ldots, 255]$$

Therefore, the ***cum_freq*** vector, used by AC during the encoding/decoding process, is initially equal to

$$\text{cum\_Freq} = [0, \text{Freq}[S[1]], \text{Freq}[S[1]] + \text{Freq}[S[2]], \ldots, \sum_{i=1}^{n} \text{Freq}[S[i]]$$

If we suggested swapping randomly the position of only two symbols in $\boldsymbol{S}$, and the two symbols happen to be of the same probability, then nothing would change in the ***cum_freq*** vector and consequently all the other symbols would keep the same lower and upper bounds. In this case, the new symbol would be encoded without encryption and the generated bitstream remains the same with and without the swapping step which leads to a non-secured scheme. Similarly, if we wish to encrypt a data source and a part of its distribution

Algorithm 2

Initialise_AC()
$value \leftarrow 0$
for $i \leftarrow 1$ to N **do**
  $y_3 \leftarrow 4 * y_3 * (1 - y_3)$
  $value \leftarrow 2 * value + input\_bit() \oplus (y_3 < 0.5)$
end for
for (; ;) **do**
  $y_1 \leftarrow 4 * y_1 * (1 - y_1)$
  if $y_1 < 0.5$ **then**
    $y_2 \leftarrow 4 * y_2 * (1 - y_2)$
    $j \leftarrow \lfloor y_2 * (n - 1) \rfloor + 2$
    $S \leftarrow [S[j], S[j+1], ..., S[n], S[1], ..., S[j-1]]$
    $cum\_freq \leftarrow [0, Freq[S[1]], Freq[S[1]] + Freq[S[2]], ..., \sum_{i=1}^{n} Freq[S[i]]]$
  end if
  $range \leftarrow high - low + 1$
  $cum \leftarrow ((value - low + 1) * R - 1)/range$
  $j \leftarrow 1$
  while $cum\_freq[j] \leq cum$ **do**
    $j \leftarrow j + 1$
  end while
  $high \leftarrow low + range * cum\_freq[j+1]/R$
  $low \leftarrow low + range * cum\_freq[j]/R$
  for (; ;) **do**
    if $high < Half$ **then**
      $value \leftarrow 2 * value$
      $low \leftarrow 2 * low$
      $high \leftarrow 2 * high + 1$
    else if $low > Half$ **then**
      $value \leftarrow 2 * (value - Half)$
      $low \leftarrow 2 * (low - Half)$
      $high \leftarrow 2 * (high - Half) + 1$
    else if $(low \geq First\_qtr)\ and\ (high < Third\_qtr)$ **then**
      $value \leftarrow 2 * (value - First\_qtr)$
      $low \leftarrow 2 * (low - First\_qtr)$
      $high \leftarrow 2 * (high - First\_qtr) + 1$
    else
      $break$;
    end if
    $y_3 \leftarrow 4 * y_3 * (1 - y_3)$
    $value \leftarrow value + input\_bit() \oplus (y_3 < 0.5)$
  end for
  $symbol \leftarrow S[j]$
  if $symbol = EOF\_symbol$ **then**
    $break$
  end if
  $out\_put(symbol)$
  if adaptive **then**
    $update\_model(symbol)$
  end if
end for

Fig. 3 *AC-decryption algorithm*

was uniform, then we risk that the swap step would fail each time we swap two symbols with the same probability.

It is also possible to randomly choose a position j and place its corresponding symbol at this position $S[j]$ in the beginning of the vector $\boldsymbol{S}$ and simultaneously shift all the other symbols $S[1]$, …, $S[j-1]$ to the right. This allows the modification of the ***cum_freq*** vector. However, all the symbols that were initially situated at on the right of $S[j]$, which are $S[j+1]$, …, $S[n]$, would keep the same low and high limits. Therefore, if the next symbol to encoded is within $S[j+1]$, …, $S[n]$, it will be encoded without encryption even after modifying the ***cum_freq*** vector. Thus, the adequate solution would be to apply a permutation of all the symbols positions at each encoding/decoding step,

doi: 10.1049/iet-ipr.2013.0598

which will obviously need a great deal of computation time mainly for data of a huge size.

The idea presented in this section consists in chaotically reordering the symbols' positions in $\boldsymbol{S}$, in order to randomly modify the lower and upper bound of all symbols prior to performing the AC encoding/decoding process. In fact, we propose to calculate a random position j to partition the symbols vector $\boldsymbol{S}$ in two parts to be interchanged. Thus, $\boldsymbol{S}$ becomes after permutaion step

$$\boldsymbol{S} = [S[j], S[j+1], \ldots, S[n], S[1], \ldots, S[j-1]$$

and the corresponding ***cum_freq*** vector becomes

$$\begin{aligned} &\textbf{\textit{cum\_freq}} \\ &\quad = [0, \text{Freq}[S[j]], \text{Freq}[S[j]] + \text{Freq}[S[j+1]], \ldots, \\ &\quad \times \sum_{i=j}^{n} F\text{req}[S[i]] + \text{Freq}[S[1]], \ldots, \sum_{i=1}^{n} \text{Freq}[S[i]]] \end{aligned}$$

Thanks to this solution, we chaotically modify the lower and upper bounds of each symbol which introduces randomisation in the AC procedure. The position j is calculated from a chaos-based PRBG, which leads to perform maximum randomisation in the proposed encryption approach. We also propose to use two other chaos-based PRBGs in the proposed compression–encryption scheme. The first PRBG is used to changing dually the $\boldsymbol{S}$ and ***cum_freq*** vectors in order to reduce the additional processing time needed by updating the ***cum_freq*** vector after the random organisation of the symbols vector $\boldsymbol{S}$, and to provide well protected results. The second PRBG is used to avoid XOR operation of output bits to enhance security and to make the proposed AC-encryption scheme more resistant against known attacks.

In this work, we have used, with no loss of generality, the logistic map (LM) as chaos-based PRBG, since it is fast, easy to implement and widely used to design chaotic-based cryptographic system. The known one-dimensional LM is defined as

$$y_{n+1} = \lambda y_n * (1 - y_n)$$

where $\lambda \in [0, 4]$, the control parameter, and $y_n \in (0, 1)$. The true chaotic behaviour of LM appears for $\lambda = 4$, this is why we suggest to use this value when designing LM-based PRBGs. It should be noted that multiple chaotic systems, high-dimensional chaotic systems, multiple iterations of chaotic systems and many other techniques can be used to design PRBG.

In this work, there are three PRBGs used to carry out the dual random organisation and the XOR operation:

- PRBG_1: randomly generates a controlling bit to control whether the symbols vector $\boldsymbol{S}$ and the ***cum_freq*** vector are modified or not.
- PRBG_2: randomly generates a value j that partition the symbols vector $\boldsymbol{S}$ in two parts to be interchanged.
- PRBG_3: randomly generates a bit to do XOR operation with the AC output bits.

Three initial values (or seeds) y_1^0, y_2^0 and y_3^0 [Algorithm 1 (Fig. 1) lines 1, 2 and 3, repectively] are needed to initialise the three PRBGs. Therefore the proposed AC-encryption scheme works with a key $K = (y_1^0, y_2^0, y_3^0)$ composed of three floating value numbers. Given the same K, both the encoder and the decoder generate the same controlling bit sequence. However, as long as one of the three key components is unknown or incorrectly given, the decoder cannot decode the compressed and encrypted data properly and the decoded data will be completely useless.

The encryption and decryption procedures are illustrated in Algorithms 2 and 3 (Figs. 2 and 3), respectively. Algorithm 1 presents the Initialise_AC() function used both by the encoder and the decoder to initialise all necessary variables. For more details about the AC implementation used in this work, please refer to [31]. In addition, AC is very sensitive to errors in the compressed data, and this undesired property ameliorates the security of the encryption-based AC method.

The proposed method for random interchanging the two parts of the symbols vector is very important as it guarantees the modification of the lower and upper bounds of all the symbols in the ***cum_freq*** vector.

Our compression–encryption scheme can be applied at bit-, pixel- or predictive error pixel-level. Even in the case of a binary source, we do not really need to omit the PRBG_2 generator as the permutation position for a binary source noted as j will usually be equal to 2 which guarantees the modification of the lower and upper bounds of the two binary symbols. Therefore our encryption algorithm remains the same whether the source is binary or not.

Fig. 4*a* presents a short example to illustrate the swapping position procedure of the input symbols and their corresponding frequencies. The conventional SAC encoding of the sequence 'CACBCC', obtained from an alphabet $\{A, B, C\}$ with known probabilities $\{0.2, 0.3, 0.5\}$, is depicted in Fig. 4*b*. The encoding of the same sequence using our approach is shown in Fig. 4*c*. The dual swapping of the cum_freq vector is determined according to two random and secret keystreams '010110' and '$-2-23-$', which are generated by PRBG_1 and PRBG_2, respectively. It should be noticed that, when randomly interchanging the symbols' positions in $\boldsymbol{S}$, we need to calculate the ***cum_freq*** vector according to these new positions, prior to performing the AC encoding process.

It should be noted that the proposed scheme conserves the compression efficiency of the AC because we use the same probabilities when encoding the input sequence with and without the dual swapping process described above. In addition, Figs 4*b* and *c* show that AC encodes the input sequence with and without encryption in two different intervals, but with the same size.

4 Experimental results

4.1 Analysis of compression efficiency

We propose to analyse the compression efficiency of our method on some well-known 8-bit grey-scale images, which are available from the University of Waterloo Greyset2 collection (http://links.uwaterloo.ca/repository.html), both on static and adaptive order-0 models. These 12 images range in size from 464×352 to 672×498 pixels.

To verify the effect of the interchanging symbols' positions procedure in compression efficiency, Table 1 gives the compression results of all test images, both on SAC and AAC-0.

When encoding an input sequence with SAC, the exact frequency count vector of all the existing symbols will be used. Consequently, the likelihood function given an image X and the static model based on true symbols probabilities

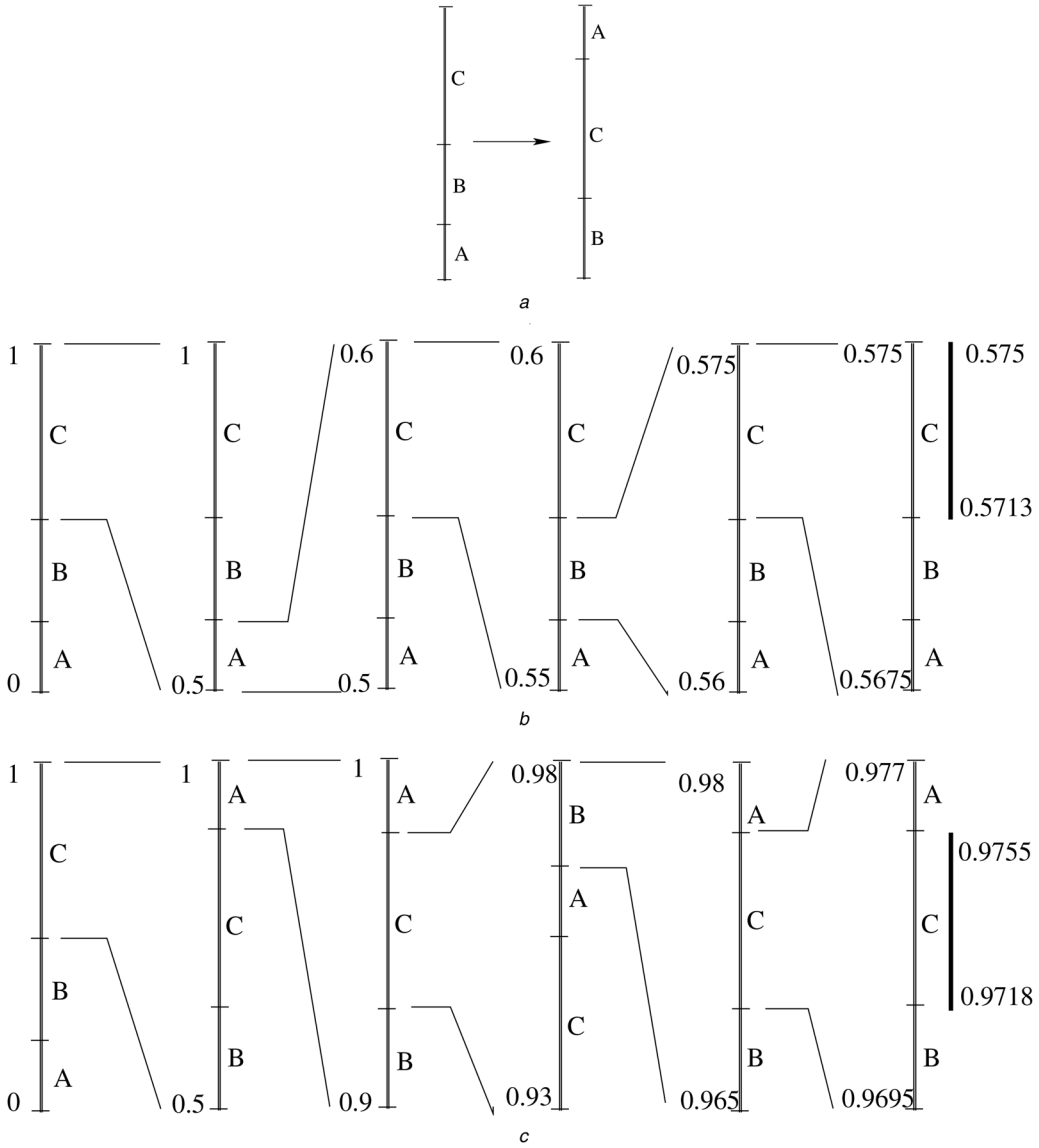


Fig. 4 *Swapping position procedure of the input symbols and their corresponding frequencies*

a Symbols and their corresponding probability positions are exchanged in the same order and are key-dependent ($y_1 = 1$ and $j = 2$)
Interval partitions for
b Conventional SAC
c AC with the proposed dual permutation scheme of symbol positions

(called P) is calculated as

$$L(X, P) = \prod_{i=1}^{m} P(x_i) = \prod_{k=0}^{255} \left(\frac{f(k)}{m}\right)^{f(k)} \quad (4)$$

However, AAC-0 assumes that all pixels have initially an equal frequency count, and after encoding/decoding each new pixel, it updates this pixel occurrence by adding one to its frequency count. Let $U_{[a,b]}$ be a discrete uniform distribution in the integer interval $[a, b]$ composed of n

Table 1 Compression results (in byte) of different images both on static and adaptive order-0 models

Image	SAC	SAC-encryption	AAC-0	AAC0-encryption
Lena	244 278	244 278	231 622	231 621
Goldhill	245 269	245 269	237 882	237 882
Barbara	244 944	244 945	238 748	238 749
Boat	233 705	233 705	227 049	227 050
Zelda	238 565	238 564	232 816	232 816
Mandrill	241 387	241 387	237 049	237 050
Peppers	248 285	248 284	238 376	238 376
France	261 626	261 623	258 217	258 216
Mountain	239 836	239 837	232 293	232 294
Library	119 564	119 564	112 108	112 109
Washsat	95 287	95 286	93 071	93 071
Frog	193 203	193 201	180 982	180 981

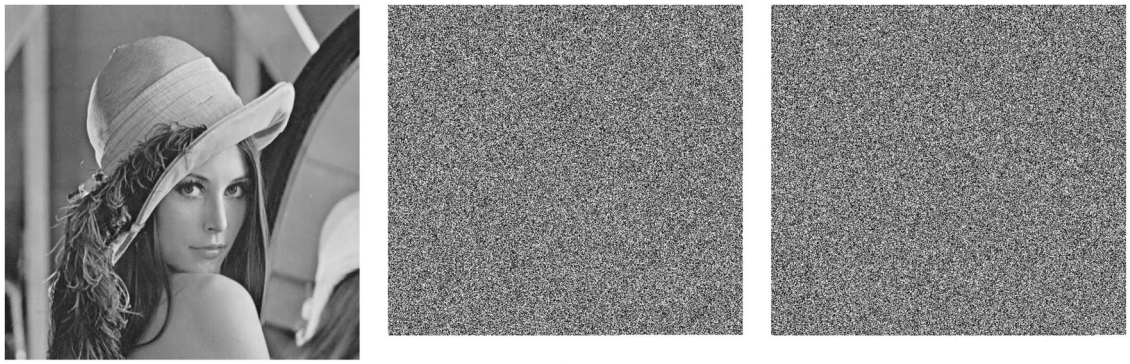
doi: 10.1049/iet-ipr.2013.0598

positive integers, so that $n = b - a + 1$ and each of the n values has an equal probability $(1/n)$. The probability of observing an image X whith respect to the model used by AAC-0, which corresponds to the likelihood function given that image, is calculated as

$$L(X, U_{[0,255]}) = \prod_{i=1}^{m} P(x_i|x_1, \ldots, x_{i-1}) = \frac{\prod_{k=0}^{255} \left\{ \prod_{j=1}^{f(k)} (j) \right\}}{\prod_{i=1}^{m} (n+i-1)} \quad (5)$$

where m is the total image size and f is the vector of true frequency counts. The number of bits required to represent the message X, either in SAC or AAC-0, can be bounded according to (3).

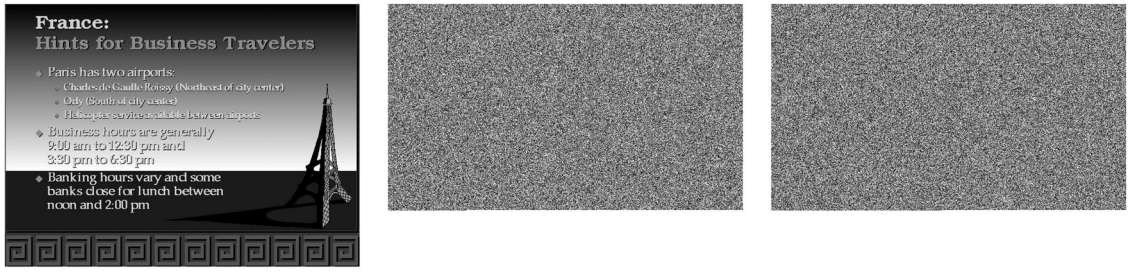
The proposed encryption scheme has the effect of finding randomly a position j between all possible positions in the symbols vector, which will be subdivided in two parts, and then swapping the order of these two parts both in symbols vector and ***cum_freq*** vector. The swapping procedure does not make any change to the subinterval width associate with each symbol, thereby allowing encryption without any loss in coding efficiency.



a b c

Fig. 5 *Three grey-scale images of Lena*

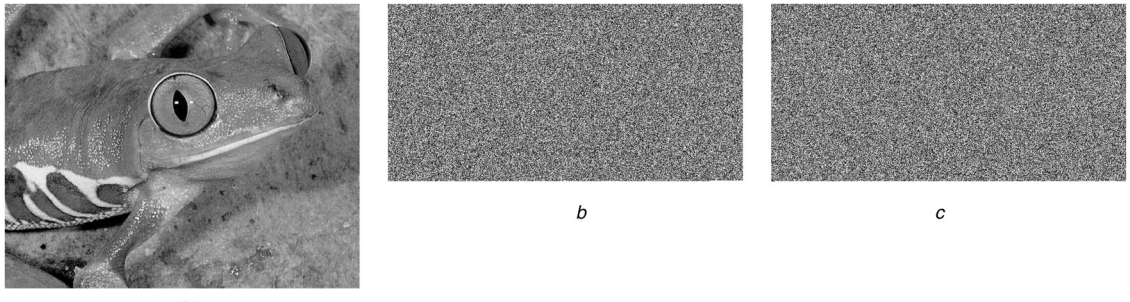
a Original image of Lena and its corresponding compressed-encrypted images using the key K both on
b SAC
c AAC-0



a b c

Fig. 6 *Three grey-scale images of France*

a Original image of France and its corresponding compressed-encrypted images using the key K both on
b SAC
c AAC-0



a b c

Fig. 7 *Three grey-scale images of Frog*

a Original image of Frog and its corresponding compressed-encrypted images using the key K both on
b SAC
c AAC-0

Table 2 Keys used for security analysis

	K	K_1	K_2	K_3
y_1^0	0.23951648742195	0.23951648742194	0.23951648742195	0.23951648742195
y_2^0	0.54397486939831	0.54397486939831	0.54397486939832	0.54397486939831
y_3^0	0.83215648972136	0.83215648972136	0.83215648972136	0.83215648972135

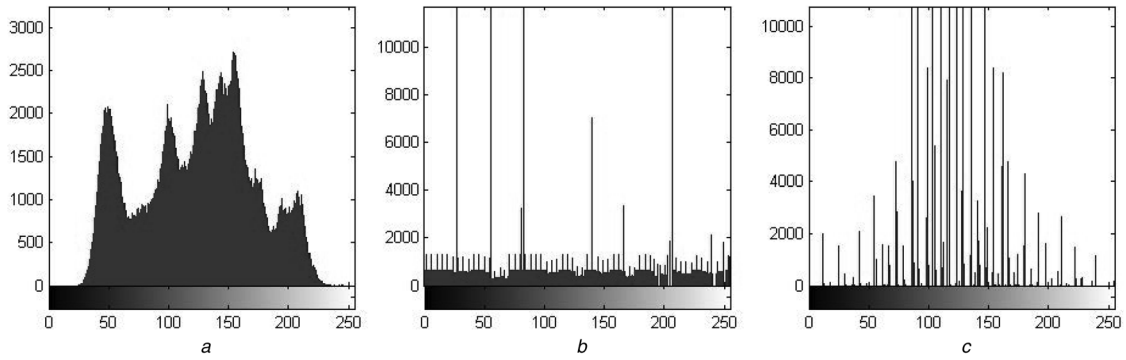


Fig. 8 *Histograms of the original images*

a Lena
b France
c Frog

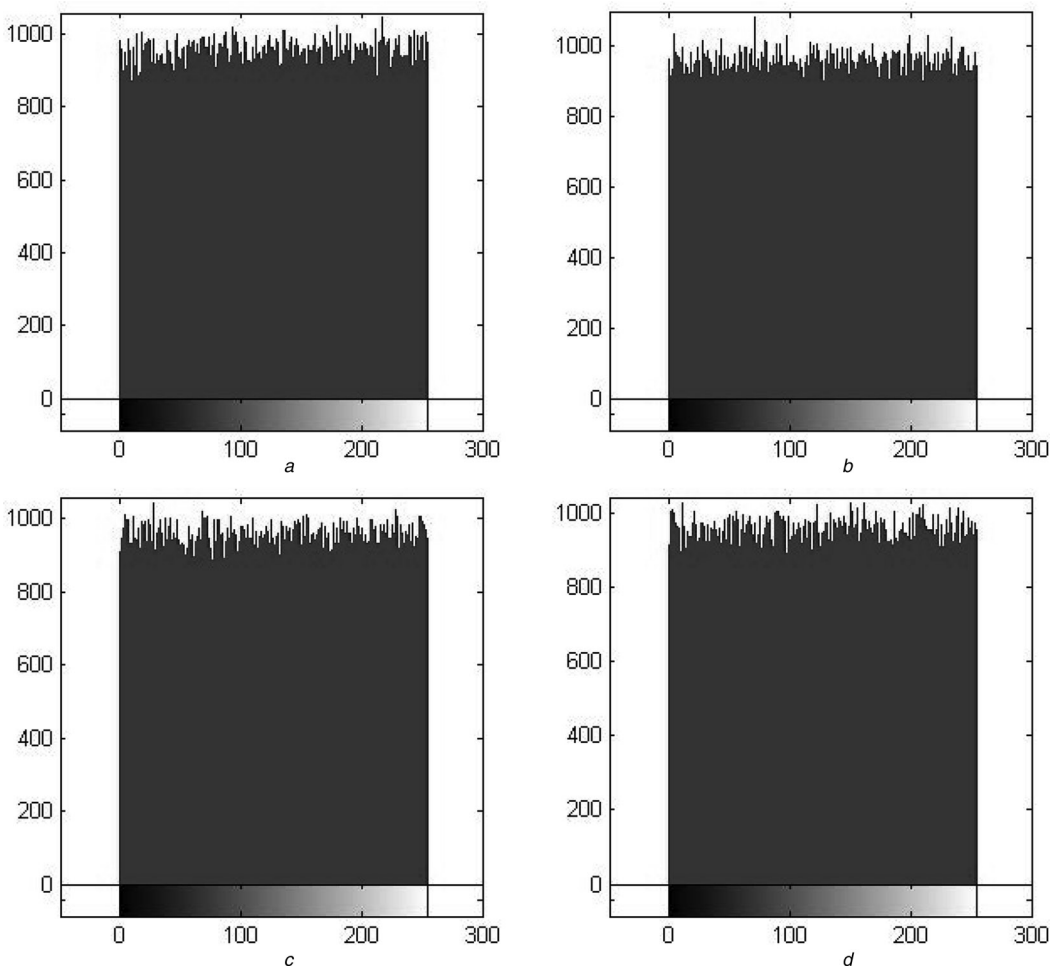


Fig. 9 *Histogram analysis of the AC outputs with and without encryption both on SAC and AAC-0 for original image of Lena*

Histogram of
a Output of SAC
b Output of AAC-0, compressed-encrypted output obtained by using the key K on
c SAC
d AAC-0

doi: 10.1049/iet-ipr.2013.0598

It should be noticed that during experiments, we used an integer AC implementation with 32-bit precision. In this case, changing the frequency counts positions in cum_freq vector may cause a slight difference in AC encoding and decoding procedures. From Table 1, the obtained compression results, both on SAC and AAC-0, with and without encryption were slightly different but there is no loss in compression efficiency.

In addition, Figs. 5–7 show three grey-scale images and their corresponding compressed-encrypted images in grey-scale form. The compressed-encrypted images are obtained by using the key K, see Table 2, both on SAC and AAC-0 models.

4.2 Security analysis

The security analysis of the proposed AC-encryption scheme includes the key sensitivity test, the statistical analysis, the entropy analysis and the key space analysis. Table 2 lists four different secret keys used for the security analysis.

4.2.1 Statistical analysis: An ideal encryption algorithm should resist statistical attacks. This has been shown using the histogram analysis, correlation analysis and information entropy analysis of the proposed scheme for both SAC and AAC-0.

(a) The histogram analysis: We have analysed the histograms of all the test images and their corresponding compressed-encrypted images using the key K. Fig. 8 presents the histograms of three original images. We presented the histogram analysis of the Lena, France and Frog images in Figs. 9, 10 and 11, respectively. It should be noticed that the unit of output acts as byte and therefore the range of output is $0 - 255$.

From Figs. 9*a* and *b*, we can see that for Lena image, the histogram of the AC output, both on SAC and AAC-0, has uniform distribution. However, for some images, such as France and Frog images, the output value of SAC has a non-uniform distribution, as illustrated in Figs. 10*a* and 11*a*, respectively. Similar results were obtained for AAC-0. The histograms of the compressed-encrypted outputs were also calculated. From Figs. 9*c*, 10*c* and 11*c*, we can see that the histograms of the compressed-encrypted outputs for SAC have always uniform distribution. We have found similar results for the histogram analysis of the all test images.

(b) The χ^2 test: We have also computed the χ^2 values of both AC output and ciphertexts, on SAC and AAC-0. The χ^2 value is described as follows

$$\chi^2 = \sum \left(f_i - \frac{n_c}{256}\right)^2 \Big/ \frac{n_c}{256}$$

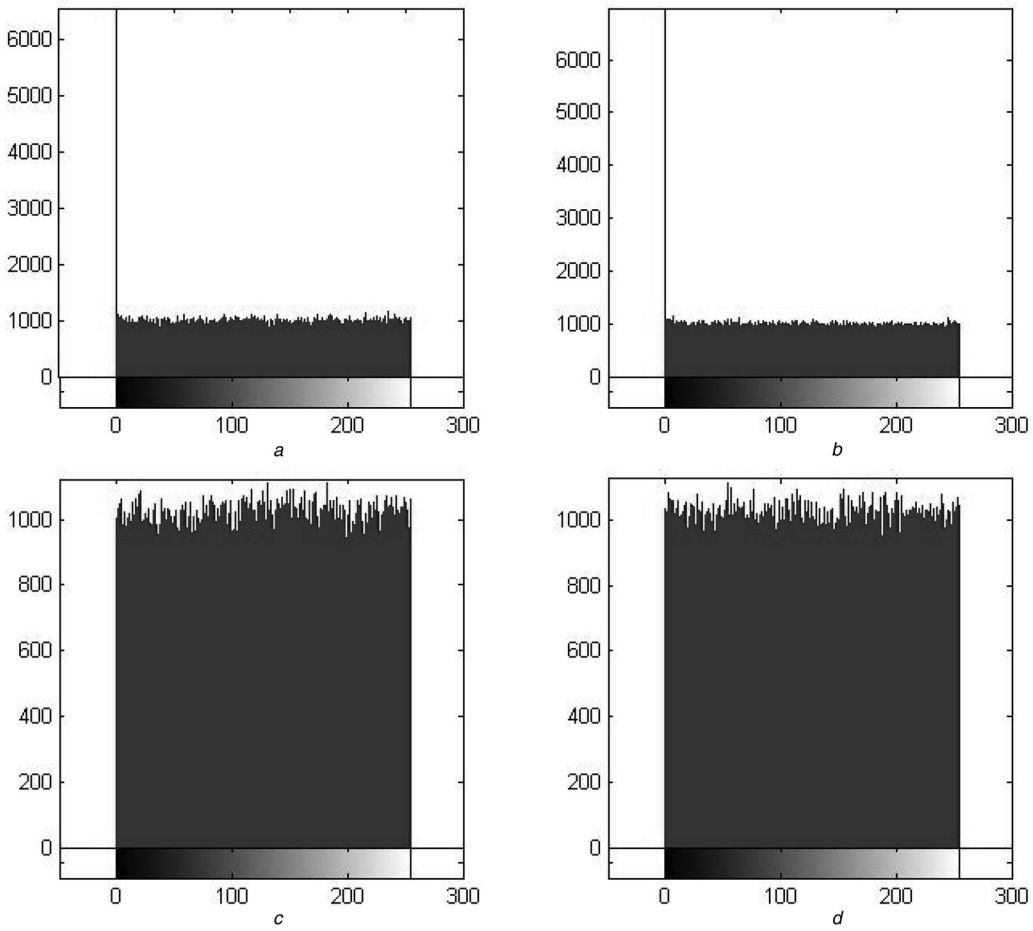


Fig. 10 *Histogram analysis of the AC outputs with and without encryption both on SAC and AAC-0 for original image of France*

Histogram of
a Output of SAC
b Output of AAC-0, compressed-encrypted output obtained by using the key K on
c SAC
d AAC-0

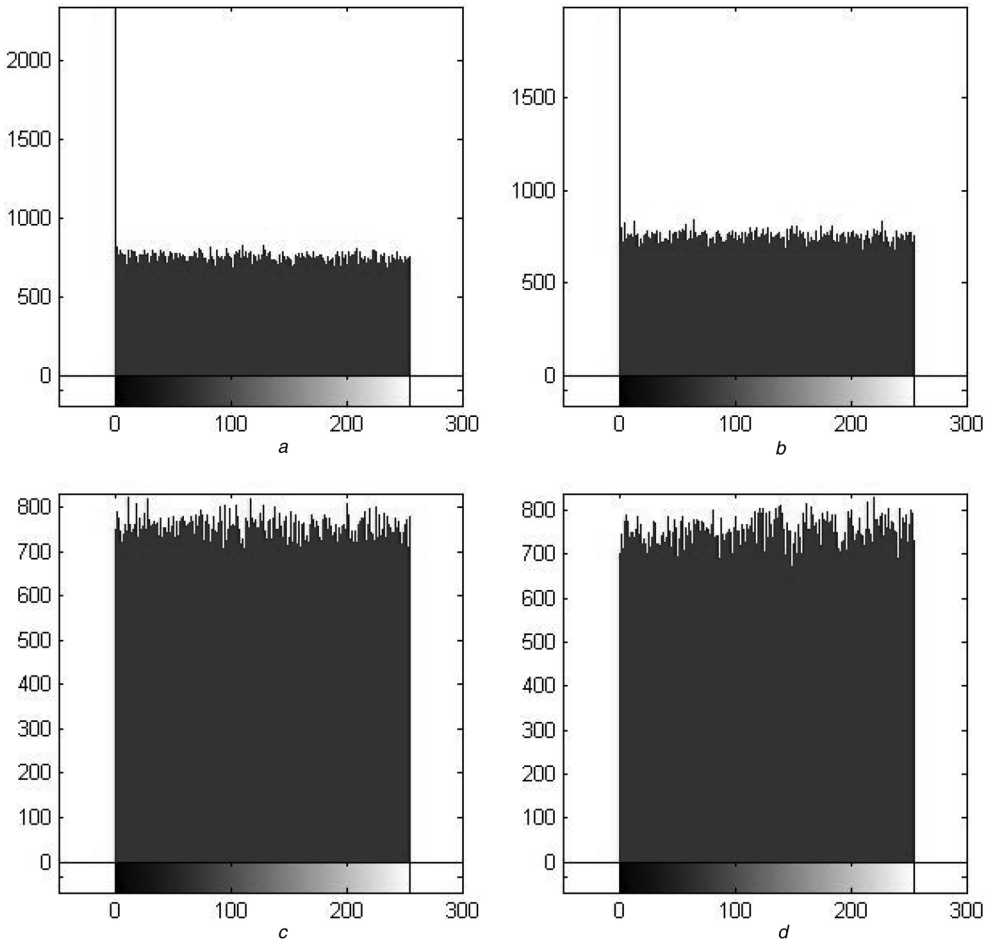


Fig. 11 *Histogram analysis of the AC outputs with and without encryption both on SAC and AAC-0 for original image of Frog*

Histogram of
a Output of SAC
b Output of AAC-0, compressed-encrypted output obtained by using the key K on
c SAC
d AAC-0

where i is the number of grey levels (0, …, 255), f_i is the observed frequency count of each grey level and n_c is the length, in bytes, of the obtained bitsream when applying AC. Assuming a significant level of 0.05, then $\chi^2(255,\ 0.05) = 293$. The χ^2 value for all test images and their corresponding bitstream with and without encryption both on static and zero-order adaptive models are presented in Table 3. The results shown in boldface are the worst. We clearly show that all the observed values of the ciphertext, both on static and adaptive models, are less than 293. This implies that the distribution of the compressed-encrypted histogram is uniform which points out good quality of the AC-encryption scheme. Therefore the encrypted image does not provide any clue to employ any statistical attack on the proposed image encryption scheme, which makes any statistical attacks difficult.

Table 3 χ^2 values for test images of the AC output and ciphertext both on static and adaptive models

	SAC	SAC-encryption	AAC0	AAC0-encryption
Lena	272	250	260	260
Goldhill	293	290	250	251
Barbara	264	268	261	232
Boat	260	238	270	257
Zelda	223	216	271	249
Mandrill	242	266	264	244
Peppers	256	284	309	241
France	18101	259	24734	245
Mountain	3859	283	447	261
Library	51292	269	510	290
Washsat	240	222	259	278
Frog	1348	272	1038	258

(c) The correlation test: For real images, each pixel is usually highly correlated with its adjacent pixels either in horizontal, vertical or diagonal directions. However, an efficient encryption scheme should generate a ciphertext with low correlation between adjacent values [3]. Horizontal, vertical and diagonal correlation coefficients γ_{xy} of two adjacent pixels can be calculated using the following equation

$$
\begin{aligned}
E(x) &= \frac{1}{N}\sum_{i=1}^{N} x_i \\
D(x) &= \frac{1}{N}\sum_{i=1}^{N}\big(x_i - E(x)\big)^2 \\
\operatorname{cov}(x, y) &= \frac{1}{N}\sum_{i=1}^{N}\big(x_i - E(x)\big)\big(y_i - E(y)\big) \\
\gamma_{xy} &= \frac{\operatorname{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}
\end{aligned}
\tag{6}
$$

doi: 10.1049/iet-ipr.2013.0598

Table 4 Correlation of the original Lena image and its corresponding ciphertext found using the key K

	Direction	Correlation coefficients
original images	horizontal	0.9714
	vertical	0.9725
	diagonal	0.9531
output of SAC	horizontal	0.0156
ciphertext of SAC	horizontal	0.0003
output of AAC-0	horizontal	0.0054
ciphertext of AAC-0	horizontal	0.0009

where N is the total number of duplets (x_i, y_i) obtained from the image, and $E(x)$ and $E(y)$ are the mean values of x and y, respectively.

Table 4 shows the correlations of the original Lena image and its corresponding ciphertext found using the key K. Similar results for all test images were obtained.

In addition, Figs. 12 and 13 show the correlation distribution of 1000 pairs of adjacent pixels randomly selected from the original images of France and Frog, and that in their compressed-encrypted images produced using the key K.

Fig. 14 reports the correlation achieved by a decoder which attempts to decode the Lena image using slightly different initial values, of which only one is correct. It can be seen that a very poor correlation is obtained among the decoded images obtained using slightly different initial values.

Both the correlation coefficients and the figures show that the proposed AC-encryption scheme decorrelates effectively the neighbouring pixels of the plain images, either in SAC or AAC-0.

(d) The information entropy test: The information entropy is initially defined by Shannon [34]. To calculate the entropy H of a source of length n, we should use the

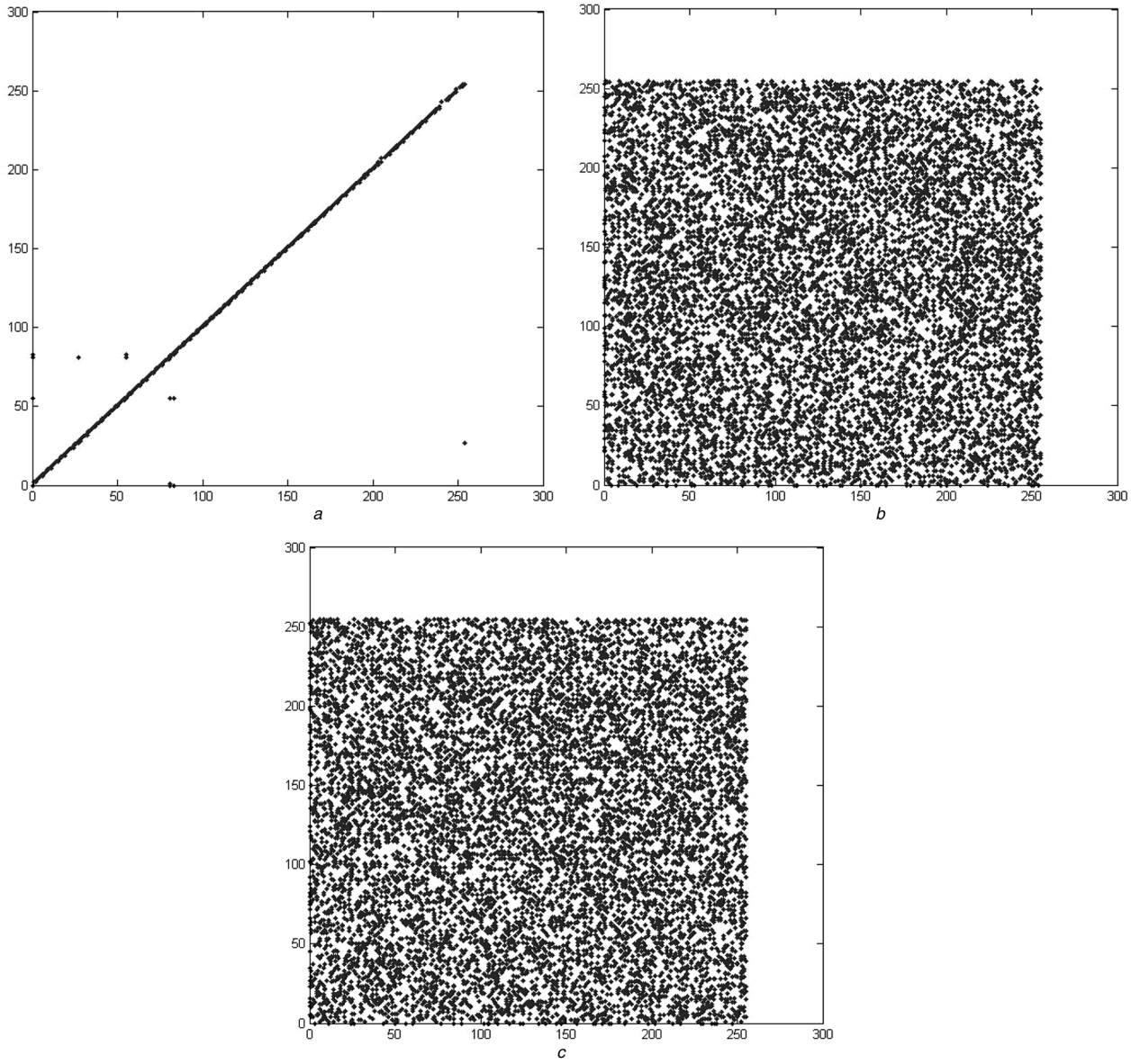


Fig. 12 *Original images of France*

a Correlations of 1000 pairs of adjacent pixels randomly selected from the original image of France and their corresponding compressed-encrypted images obtained by
b SAC
c AAC-0
The unit of output acts as byte, so the range of output is 0–255

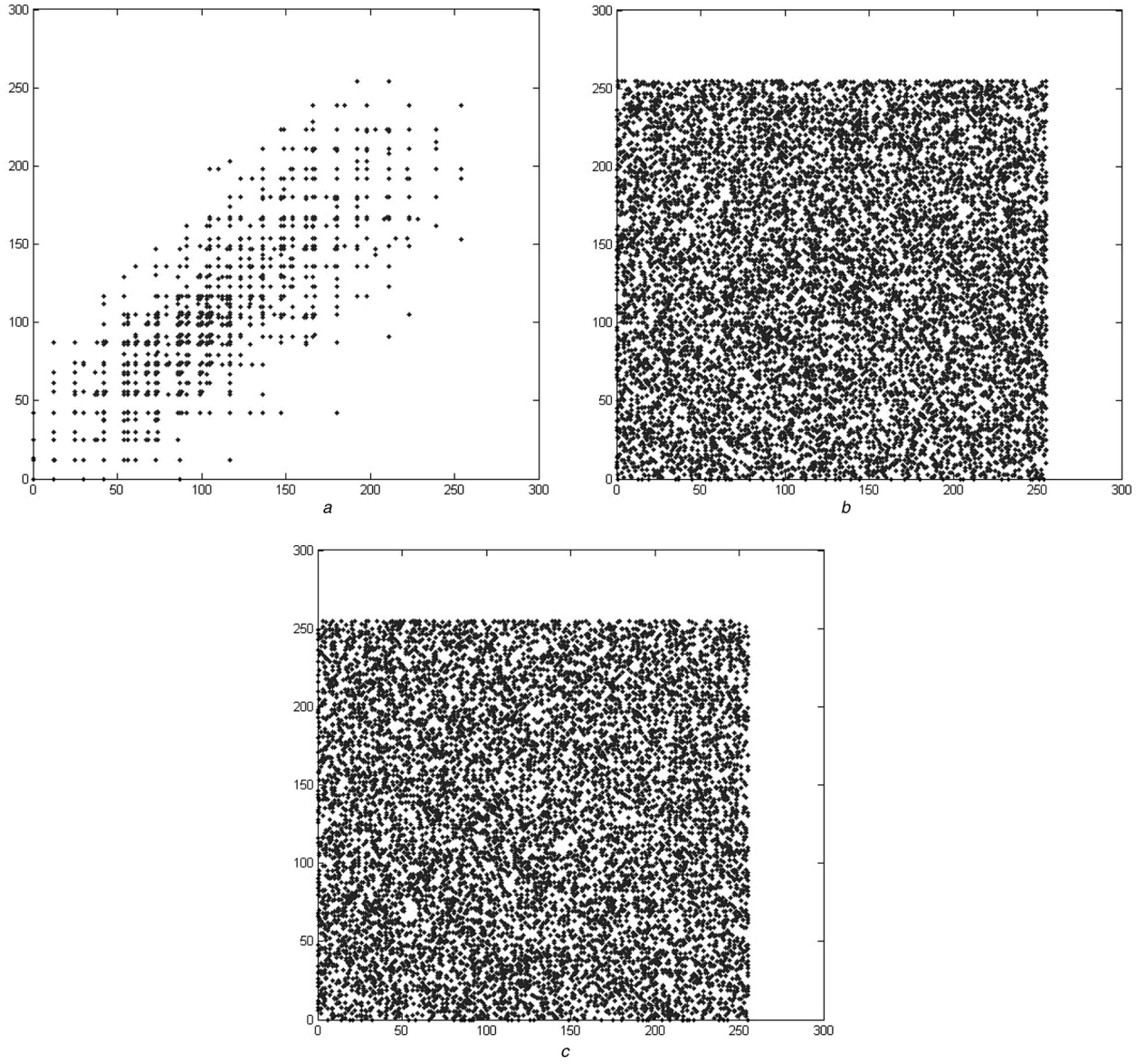


Fig. 13 *Original images of Frog*

a Correlations of 1000 pairs of adjacent pixels randomly selected from the original image of Frog and their corresponding compressed-encrypted images obtained by
b SAC
c AAC-0
The unit of output acts as byte, so the range of output is 0–255

following equation

$$H = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \tag{7}$$

When an image is encrypted, its entropy, expressed in bits/pixel, should ideally be equal to 8 bits/pixel. If the output of an image encryption algorithm [35] generates symbols with entropy less than 8 bits/pixel, there exists certain degree of predictability and the encryption algorithm will be threatened in its security. Table 5 presents the average entropy obtained for the ciphertexts of all test images, both on SAC and ACC-0, which is equal to 7.9990 and 7.9991 bits/pixel, respectively. The average entropy found is very close to the theoretical value $\simeq 8$, which confirms that the proposed AC-encryption scheme is secure against the entropy attack. In Table 5, the entropies shown in boldface represent the worst results, which confirms that the AC cannot be considered as an encryption scheme.

4.2.2 Key space: From a strong cryptographic point of view, the data security should be based on an algorithm where a brute force attack has a minimum of 2^{128} different keys that can be used in the encryption [36]. In our scheme, the secret key K includes three floating point numbers with precision of 10^{14}, and so the key space of the key is $10^{42} \simeq 2^{140}$ which is far enough to resist the brute-force attack.

4.2.3 Key sensitivity test: According to the basic principle of cryptology, a cryptosystem should be sensitive to the key. Thus, we propose the following tests [37]:

(a) Assume that the encryption key used is K presented in Table 2. First, the original image of Lena is encrypted using

doi: 10.1049/iet-ipr.2013.0598

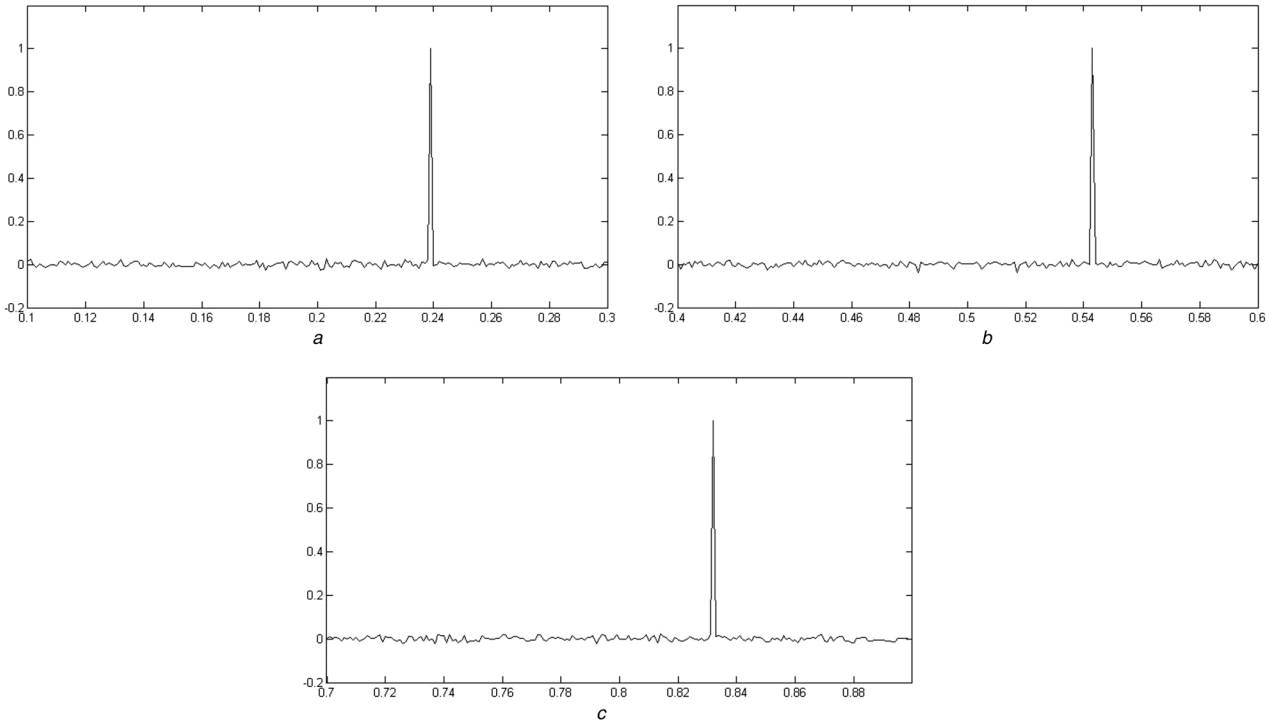


Fig. 14 *Correlation test obtained by decoding the compressed-encrypted Lena image with different combinations of the keys*

Generated using different initial values
a y_1^0
b y_2^0
c y_3^0 on AAC-0
The plots are measured against initial values obtained from the key K

the test key. Next, the same original image is encrypted with three slightly different keys described in Table 2.

We propose for each of the used secret keys, to conserve two parts of K and to change the third one by 10^{-14}. Suppose C^1 and C^2 are two ciphertexts having the same length N, then the number of pixel change rate (NPCR) and the unified average changing intensity (UACI) [38, 6] can be mathematically defined by (9) and (10), respectively, where |.| denotes the absolute value function. The NPCR and UACI between various ciphertexts produced using slightly different keys on SAC, are calculated and the results are given Table 6. Similar results were found when using AAC-0. Table 6 shows that ciphertexts obtained with keys K and K_1 have 99.63% pixels which are not identical.

Table 5 Entropy of the AC outputs with and without encryption for both static and adaptive models

	SAC	SAC-encryption	AAC0	AAC0-encryption
Lena	7.9992	7.9991	7.9991	7.9992
Goldhill	7.9992	7.9993	7.9992	7.9992
Barbara	7.9992	7.9992	7.9993	7.9993
Boat	7.9993	7.9991	7.9993	7.9993
Zelda	7.9991	7.9991	7.9992	7.9993
Mandrill	7.9992	7.9993	7.9993	7.9993
Peppers	7.9992	7.9993	7.9992	7.9993
France	7.9648	7.9994	7.9573	7.9994
Mountain	7.9962	7.9993	7.9960	7.9992
Library	7.8524	7.9983	7.9858	7.9985
Washsat	7.9982	7.9981	7.9981	7.9980
Frog	7.9921	7.9991	7.9932	7.9990
average	7.9832	7.9990	7.9937	7.9991

It should be noticed that the NPCR value should be as close to 100% as possible, and the UACI value should be as close to 33% as possible

$$D_i = \begin{cases} 0, & \text{if } C_i^1 = C_i^2 \\ 1, & \text{if } C_i^1 \neq C_i^2 \end{cases} \tag{8}$$

$$\text{NPCR} = \frac{\sum_i D_i}{N} \times 100\% \tag{9}$$

$$\text{UACI} = \frac{1}{N}\left(\sum_i \frac{|C_i^1 - C_i^2|}{255}\right) \times 100\% \tag{10}$$

(b) In addition, to test the key sensitivity of our encryption algorithm, we propose to encrypt plain Lena image with the key K and to decrypt it using a key with a difference of 10^{-14}. Fig. 15 shows the experimental results. As observed from Fig. 15, the decrypted images are totally different from the plain one even where there is only 10^{-14}

Table 6 Difference between ciphertexts obtained by using keys with slight differences

Test item	Test results between ciphertexts with tiny change in the key		
	k_1	k_2	k_3
NPCR, %	99.63	99.52	99.61
UACI, %	33.42	33.51	33.48

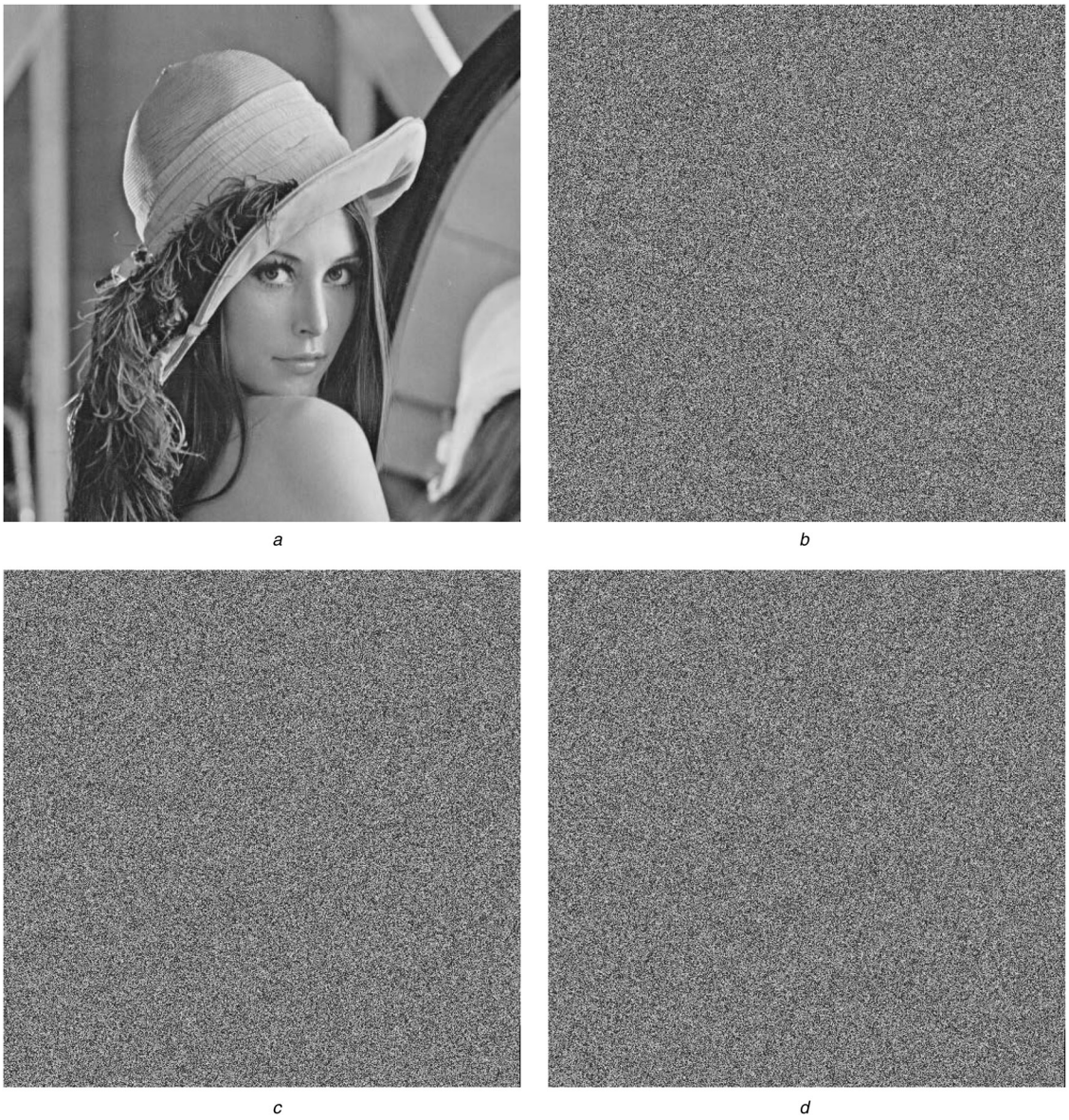


Fig. 15 *Decrypted image with*
a K
b K_1
c K_2
d K_3

difference in the decryption key. Thus, having a perfect approximation of the encryption secret key makes decryption impossible.

4.2.4 Comparison: Tables 7 and 8 present the comparison between the proposed method and the other methods in terms of encryption and compression efficiency, respectively. From Table 7, its is shown that most of the other researchers have not reported the histogram, χ^2, correlation, entropy, NPCR and UACI tests, and suffer from vulnerability to some attacks. In addition, most of the other techniques are based on BAC (each data symbol is coded bit by bit) by swapping the two mapping intervals randomly using different methods. It should be noted that the other methods are restricted to be applied only to some types of statistical models.

5 Conclusion

In this paper, we proposed an efficient and secure encryption scheme which combines AC, both on static and adaptive models, with chaos-based PRBGs to perform both lossless compression and encryption of images. In our scheme, we exploit both the efficiency of the AC in lossless image compression and the advantages of the chaos theory in data encryption to provide a scheme which can be very useful in many applications, especially for image protection. The proposed scheme presents several interesting features like: extendable to any context-based AAC approach, lossless encryption of images, the same sizes for the output bitstream and the ciphertext, for both static and adaptive models, key space is large enough, extendable to any PRBG, fast and easily implemented at a low cost.

doi: 10.1049/iet-ipr.2013.0598

Table 7 Comparison of encryption efficiency of the proposed method and the other methods

Methods	Histogram	χ^2	Correlation	Entropy	Key sensitivity		Key space	Proof of insecurity
					NPCR, %	UACI, %		
Grangetto *et al.* [19]	NA	NA	NA	NA	NA	NA	NA	[39]
Bose *et al.* [20]	NA	NA	NA	NA	NA	NA	2^{30}	
Wen *et al.* [21]	NA	NA	NA	NA	NA	NA	NA	[39, 40]
Kim *et al.* [22]	NA	NA	NA	>0.99 (binary data)	NA	NA	NA	[41, 42]
Mi *et al.* [23]	NA	NA	NA	NA	NA	NA	2^{32}	
Zhou *et al.* [24]	uniform	NA	NA	NA	NA	NA	2^{128}	
Li *et al.* [25]	uniform	NA	NA	>0.99 (binary data)	50	NA	2^{199}	
Masmoudi *et al.* [26]	passed all NIST statistical tests [43]	2^{157}						
Duan *et al.* [27]	NA	NA	NA	>0.99 (binary data)	99.60	NA	2^{256}	[28]
proposed method	uniform	<272	<0.0009	> 7.999	99.59	33.47	2^{140}	
expected value	uniform	<293	$\simeq 0$	$\simeq 8$	>99.58 [44]	33.37 < . < 33.55 [44]		

The missing values are represented by the symbol NA (not available).

Table 8 Comparison of compression efficiency of the proposed method and the other methods

Methods	Loss in, %	Data type	Statistical model
Grangetto *et al.* [19]	0	binary	MQ-coder for JPEG 2000
Bose *et al.* [20]	6	any	zeroth-order
Wen *et al.* [21]	<1	binary	static
Kim *et al.* [22]	<1	binary	static
Mi *et al.* [23]	0	binary	static
Zhou *et al.* [24]	<1	binary	any
Li *et al.* [25]	0	binary	any
Masmoudi *et al.* [26]	0	binary	any
Duan *et al.* [27]	improve RAC [19] by 33%	binary	order-1 Markov model
proposed method	0	any	any
expected value	0	any	any

The symbol 'any' in the *Data type* column corresponds to binary and non binary data. However, in the 'statistical model' column it corresponds to static, adaptive and context-based adaptive models.

6 References

1 Li, S., Mou, X.: 'Improving security of a chaotic encryption approach', *Phys. Lett. A*, 2001, **290**, (3-4), pp. 127–133
2 Yang, T.: 'A survey of chaotic secure communication systems', *J. Comput. Cogn.*, 2004, **2**, (2), pp. 81–130
3 Wu, X.G., Hu, H.P., Zhang, B.L.: 'Analyzing and improving a chaotic encryption method', *Chaos Solitons Fractals*, 2004, **22**, (2), pp. 367–373
4 Zhang, L., Liao, X., Wang, X.: 'An image encryption approach based on chaotic maps', *Chaos Solitons Fractals*, 2005, **24**, (3), pp. 759–765
5 Wong, K.W., Kwoka, B.S.H., Yuena, C.H.: 'An efficient diffusion approach for chaos-based image encryption', *Chaos Solitons Fractals*, 2008, **41**, (5), pp. 2652–2663
6 Masmoudi, A., Puech, W., Bouhlel, M.S.: 'A new image cryptosystem based on chaotic map and continued fractions'. 18th European Signal Processing Conf., 2010
7 Kanso, A., Smaoui, N.: 'Logistic chaotic maps for binary numbers generations', *Chaos Solitons Fractals*, 2009, **40**, pp. 2557–2568
8 Patidar, V., Sud, K.K.: 'A novel pseudo random bit generator based on chaotic standard map and its testing', *Electron. J. Theor. Phys.*, 2009, **6**, (20), pp. 327–344
9 Masmoudi, A., Puech, W., Bouhlel, M.S.: 'An efficient prbg based on chaotic map and engel continued fractions', *J. Softw. Eng. Appl.*, 2010, **3**, (12), pp. 1141–1147
10 Marpe, D., Schwarz, H., Wiegand, T.: 'Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard', *IEEE Trans. Circuits Syst. Video Technol.*, 2003, **13**, (7), pp. 620–636
11 Alcaraz-Corona, S., Rodriguez-Dagnino, R.M.: 'Bi-level image compression estimating the Markov order of dependencies', *IEEE J. Sel. Top. Signal Process.*, 2010, **4**, (3), pp. 605–611
12 Carpentieri, B., Weinberger, M.J., Seroussi, G.: 'Lossless compression of continuous-tone images', *Proc. IEEE*, 2000, **88**, (11), pp. 1797–1809
13 Masmoudi, A., Puech, W., Bouhlel, M.S.: 'Efficient adaptive arithmetic coding based on updated probability distribution for lossless image compression', *J. Electron. Imaging*, 2010, **19**, (2), pp. 1–6
14 Zhang, L., Wang, D., Zheng, D.: 'Segmentation of source symbols for adaptive arithmetic coding', *IEEE Trans. Broadcast.*, 2012, **58**, (2), pp. 228–235
15 Masmoudi, A., Masmoudi, A.: 'A new arithmetic coding model for a block-based lossless image compression based on exploiting inter-block correlation'. Signal, Image and Video Processing, 2013, pp. 1–7
16 Motta, G., Storer, J.A., Carpentieri, B.: 'Lossless image coding via adaptive linear prediction and classification', *Proc. IEEE*, 2000, **88**, (11), pp. 1790–1796
17 Kuroki, N., Manabe, T., Numa, M.: 'Adaptive arithmetic coding for image prediction errors'. Proc. of the 2004 Int. Symp. on Circuits and Systems, 2004 (ISCAS'04), 2004, vol. 3, pp. III–961–4
18 Matsuda, I., Umezu, Y., Ozaki, N., Maeda, J., Itoh, S.: 'A lossless coding scheme using adaptive predictors and arithmetic code optimized for each image', *Syst. Comput. Jpn*, 2007, **38**, (4), pp. 1–11
19 Grangetto, M., Magli, E., Olmo, G.: 'Multimedia selective encryption by means of randomized arithmetic coding', *IEEE Trans. Multimedia*, 2006, **8**, (5), pp. 905–917
20 Bose, R., Pathak, S.: 'A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system', *IEEE Trans. Circuits Syst.*, 2006, **53**, (4), pp. 848–857
21 Wen, J., Kim, H., Villasenor, J.D.: 'Binary arithmetic coding using key-based interval splitting', *IEEE Signal Process. Lett.*, 2006, **13**, (2), pp. 69–72
22 Kim, H., Wen, J., Villasenor, J.: 'Secure arithmetic coding', *IEEE Trans. Signal Process.*, 2007, **55**, (5), pp. 2263–2272
23 Mi, B., Liao, X., Chen, Y.: 'A novel chaotic encryption scheme based on arithmetic coding', *Chaos Solitons Fractals*, 2008, **38**, pp. 1523–1531
24 Zhou, J., Au, O.C., Fan, X., Wong, P.H.W.: 'Joint security and performance enhancement for secure arithmetic coding'. 15th IEEE Int. Conf. on Image Processing, 2008, pp. 3120–3123
25 Li, H., Zhang, J.: 'A secure and efficient entropy coding based on arithmetic coding', *Commun. Nonlinear Sci. Numer. Simul.*, 2009, **14**, (12), pp. 4304–4318
26 Masmoudi, A., Puech, W., Bouhlel, M.S.: 'A new joint lossless compression and encryption scheme combining a binary arithmetic

coding with a pseudo random bit generator', *Int. J. Comput. Sci. Inf. Sec.*, 2010, **8**, (1), pp. 170–175
27 Duan, L., Liao, X., Xiang, T.: 'A secure arithmetic coding based on Markov model', *Commun. Nonlinear Sci. Numer. Simul.*, 2011, **16**, (6), pp. 2554–2562
28 Zhao, L., Nishide, T., Adhikari, A., Rhee, K.H., Sakurai, K.: 'Cryptanalysis of randomized arithmetic codes based on markov model', in Wu, C.-K., Yung, M., Lin, D. (Eds.): 'Information security and cryptology' (Heidelberg, Springer Berlin, 2012) (*LNCS* **7537**), pp. 341–362
29 Howard, P.G., Vitter, J.S.: 'Arithmetic coding for data compression', *Proc. IEEE*, 1994, **82**, (6), pp. 857–865
30 Moffat, A., Neal, R.M., Witten, I.H.: 'Arithmetic coding revisited', *ACM Trans. Inf. Syst.*, 1998, **16**, (3), pp. 256–294
31 Witten, I.H., Neal, R.M., Cleary, J.G.: 'Arithmetic coding for data compression', *Commun. ACM*, 1987, **30**, (6), pp. 520–540
32 Salomon, D.: 'Data compression the complete reference' (Springer, 2007, 4th edn.)
33 Triantafyllidis, G.A., Strintzis, M.G.: 'A context based adaptive arithmetic coding technique for lossless image compression', *IEEE Signal Process. Lett.*, 1999, **6**, (7), pp. 168–170
34 Shannon, C.: 'Communication theory of secrecy systems', *Bell Syst. Tech. J.*, 1949, **28**, (4), pp. 656–715
35 Behina, S., Akhshani, A., Mahmodi, H., Akhavan, A.: 'A novel algorithm for image encryption on mixture of chaotic maps', *Chaos Solitions Fractals*, 2008, **35**, pp. 408–419
36 Schneier, B.: 'Applied cryptography: protocols, algorithms, and source code in C' (Wiley, New York, 1996, 2nd edn.)
37 Maniccam, S.S., Bourbakis, N.G.: 'Lossless image compression and encryption using SCAN', *Pattern Recognit.*, 2001, **34**, pp. 1229–1245
38 Chen, G., Mao, Y., Chui, C.K.: 'A symmetric image encryption scheme based on 3D chaotic cat maps', *Chaos Solitons Fractals*, 2004, **21**, pp. 749–761
39 Katti, R.S., Srinivasan, S.K., Vosoughi, A.: 'On the security of randomized arithmetic codes against ciphertext-only attacks', *IEEE Trans. Inf. Forensics Sec.*, 2011, **6**, (1), pp. 19–27
40 Jakimoski, G., Subbalakshmi, K.P.: 'Cryptanalysis of some multimedia encryption schemes', *IEEE Trans. Multimedia*, 2008, **10**, (3), pp. 330–338
41 Sun, H.M., Wang, K.H., Ting, W.C.: 'On the security of the secure arithmetic code', *IEEE Trans. Inf. Forensics Sec.*, 2009, **4**, (4), pp. 781–789
42 Zhou, J., Au, O.C., Wong, P.H.: 'Adaptive chosen-ciphertext attack on secure arithmetic coding', *IEEE Trans. Signal Process.*, 2009, **57**, (5), pp. 1825–1838
43 Rukhin, A., Soto, J., Nechvatal, J., *et al*..: 'Statistical test suite for random and pseudo random number generators for cryptographic applications'. NIST Special Publication 800–22 Revision 1, 2008
44 Wu, Y., Noonan, J.P., Agaian, S.: 'Npcr and uaci randomness tests for image encryption', *J. Sel. Areas Telecommun.*, 2011, pp. 31–38

doi: 10.1049/iet-ipr.2013.0598