ENGINEERING TRANSACTIONS • *Engng. Trans.* • **57**, 2, 89–104, 2009
Polish Academy of Sciences • Institute of Fundamental Technological Research

ON IMPROVED IMAGE ENCRYPTION SCHEME BASED ON CHAOTIC MAP LATTICES

K. Jastrzębski [1)], Z. Kotulski [2)]

[1)] **Warsaw University of Technology**
Department of Electronics and Information Technology
Nowowiejska 15/19, 00-665 Warszawa, Poland

[2)] **Institute of Fundamental Technological Research**
Polish Academy of Sciences
Pawińskiego 5B, 02-106 Warszawa, Poland

In this paper we examine one of the recently proposed chaotic image encryption algorithms, based on chaotic map lattices (CML). We show certain problems with the chaotic map, as well as errors in the designed algorithm. Then we propose a way to improve it and present a new version of algorithm and its implementation. At the end, we show the results of a security analysis and a comparison of both schemes. These results were obtained in the MSc Thesis [25].

Key words: Discret Chaotic Cryptography (DCC), image encryption, chaotic dynamical systems.

Chaotic dynamical systems were independently identified by Edward Lorenz and Yoshisuke Ueda in 1961. They started to be involved into secure communication in the 9th decade of the 20th century. First, the chaos-based security research was concentrated on continuous (in time) dynamical systems, governed by nonlinear differential equations, and two techniques: chaos control and chaos synchronization. Such techniques are useful for secure streaming, analogous to stream ciphers known in traditional cryptography. Then, discrete dynamical systems were applied in a way analogous to block cipher encryption. The second technique started to be extensively studied when it was applied to image encryption, which needs very efficient algorithms (due to large sets of bits representing multicolor pixels). In this paper, we propose an algorithm that could satisfy expectations of efficiency and security in a case of image encryption, both black and white and color.

1. Introduction

Discrete chaotic dynamical systems are nonlinear, exponentially sensitive to changes of initial conditions, the points of their trajectories take values from

uncountable sets and their subsets of phase space, while the systems are ergodic or mixing, are equiprobable and asymptotically statistically independent. Moreover, chaos may occur even in simple recursive equations. These properties are very good from cryptographic point of view, which was intuitively described even by SHANNON [1], when he was introducing the concept of mixing in the information theory field. Analogies between features of chaotic systems and properties of good cryptosystems are widely known [2–4]. The main problem, still unsolved is how to transfer chaos into finite-state space-valued digital systems, see [5]. Despite this, scientists are still trying to find a cryptographic application of mathematical tools of the chaos theory. Many ideas concentrate on ciphers based on discrete (in time) chaotic maps.

The first cipher of this class was proposed in 1991 by HABUTSU *et al.* [6]. The transformation used was a tent map, a plain text was an initial condition and a key was the map control parameter, which placed the top of a tent. This cipher was easily broken [7]. The second idea was to insert the key into the initial condition of a dynamical system, proposed by KOTULSKI and SZCZEPAŃSKI [8]. Afterwards, BAPTISTA [9] suggested a cryptosystem, in which both the initial condition and the control parameters played a role of a secret key. In the next cryptosystem, [10], the inspiration was a thermodynamical model of a gas particle, closed in a container, which is subject to a chaotic reflection law. This time the map was two-dimensional and the key was one of the two initial conditions. d-dimensional dynamical systems (previously agreed between communication sides) were a base of the cryptosystem introduced by ALVAREZ *et al.* [11]. In all of these systems the number of iterations of a chaotic map was limited on the one hand by the condition of obtaining a statistically good ("random") ciphertext, on the other hand by the time of computation. These conditions, the way of using the map and described forms of the key became typical for most of the proposed ciphers. A more detailed review of propositions may be found [12].

There are attempts to apply discrete in time, chaotic systems also in image encryption. Image encryption is connected with some specific problems [13], such as: huge redundancy and size of data, strong correlation between pixels, compression, different significance of bits (to human eye), speed of computation (sometimes more important than high security level), etc. In many applications, conventional encryption schemes are not suitable [14]. Consequently, many scientists turned their attention to algorithms specifically designed for image and video encryption and to be fast, including many chaotic encryption propositions (e.g. SPIHT algorithm [15], CKBA [16], BRIE [17], cipher based on 3D Baker's transformation [18], RRS-CVES [19]). There exist two approaches: the first one is to use chaos to permute the order of the pixels; the second one is based on changing the numerical values responsible for the color of each pixel. There appeared also some ideas to join these two operations, calling them the confusion

phase and the diffusion phase [20]. The level of sophistication among the propositions on a secure chaotic cipher constantly grows, although most algorithms turned out to be insecure (a detailed review [12]). One of the latest propositions is the image encryption algorithm based on chaotic map lattices (CML) [21]. This cipher is the starting point of our paper.

The scheme of the paper is the following. Section 2 presents the original CML algorithm. In Sec. 3 we point out some problems in this proposition. Next (in Sec. 4), we introduce modifications, which improve the CML scheme. In Sec. 5 we present a security analysis of the modified algorithm. In the last Sec. 6 we draw conclusions and show the direction of future works.

2. CML image encryption scheme

As it was mentioned, image encryption hides some specific problems, such as huge redundancy and size of data or required effectiveness. Some scientists believe that chaotic ciphers will turn out to be a good solution to these problems. An interesting proposition is the CML scheme [21]. This algorithm has features similar to *cipher-block-chaining* (CBC) mode, known from the block ciphers theory [22]; here blocks are pixels. It means that the encrypted value of a pixel depends not only on the value of the pixel itself but also on the values of cryptograms of its neighbors. Such a property is especially important for image encryption because any ECB mode (*electronic code book* mode, where identical pixels are identically encrypted) does not hide contours of elements of a picture.

As a map, CML uses the logistic equation:

$$x_{n+1} = a x_n (1 - x_n), \tag{2.1}$$

where $a \in (3.57, 4.00)$ is the control parameter. For all values of a there exist uniquely determined values $x_{\min}$ and $x_{\max}$, such that in successive iterations the values of the variable x_n don't leave the range of $[x_{\min}, x_{\max}]$. Let us also define $\delta x = x_{\max} - x_{\min}$.

Input data of the cipher are pixel matrices, i.e. three matrices of color components in RGB coding, where each element of the matrix takes a value from 0 to 255. If the image is monochromatic, there is obviously one matrix of numbers (grayscale values). The value of each color component of the pixel is used as initial value for the logistic equation. Conversion to floating-point numbers and back to colors are realized by formulas:

$$x_c = x_{\min} + \delta x \left(\frac{C_c}{255} \right), \tag{2.2}$$

$$C_c = round \left((x_c - x_{\min}) \frac{255}{\delta x} \right), \tag{2.3}$$

where $c = \{R,\ G,\ B\}$, C_c is the c-th component of the color C, and x_c is its floating-point value. The Eq. (2.3) should guarantee returning to the $\{0,\ 1,\ \ldots,\ 255\}$ set, thus colors of pixels.

Encryption procedure

Let the plain image contain $N \times M = m$ pixels, and let $i = 1,\ 2,\ \ldots,\ m$ be a pixel index. To make the description simpler, let us assume that the image is black and white, represented by one matrix of numbers.

At the beginning, all pixels are changed to floating-point numbers, according to Eq. (2.2). The value of m-th pixel is taken as an initial condition for the map, i.e. $x_0^1 = x_c^m$. The value obtained from the map after n iterations is added to the value of the first pixel x_c^1, i.e. $x_c^1 + x_n^1 = x_c^1 + M^n\left(x_0^1\right)$. If the sum exceeds $[x_{\min},\ x_{\max}]$, it has to be normalized by subtracting δx. The obtained result is overwritten as a value of the first pixel. This value becomes a new initial condition for the map, which is iterated n times, and the obtained result is added to the next pixel. The sum, if necessary, is normalized and overwritten on the next pixel's place and so on. The described procedure is carried through all pixels and repeated j times for the entire image. Using Eq. (2.3) we convert numbers to grayscale shades. If we have a color image, the algorithm is done subsequently to all three RGB matrices.

The algorithm proposed in [21], in a more formal way, may be presented as:

Algorithm 1. CML cipher encryption procedure.

Input:
- j – number of cycles, part of a cipher key
- m – number of pixels, part of a cipher key
- IMG – stream of image pixels
- n – number of iterations, part of a cipher key
- M – chaotic map
- p – control parameter, part of a cipher key
- $x_{\min},\ x_{\max}$ – borders of the attractor, dependent on p
- $\delta x = x_{\max} - x_{\min}$

Output: stream of encrypted pixels

1. **for** $pixel \leftarrow 1$ **to** m **do** $IMG(pixel) \leftarrow x_{\min} + \delta x \dfrac{IMG(pixel)}{255}$
2. **for** $cycle \leftarrow 1$ **to** j **do** steps from 3 to 7
3. **for** $pixel \leftarrow 1$ **to** m **do** steps from 4 to 7
4. $x_0 \leftarrow IMG(pixel\ - 1)$
5. **for** $i \leftarrow 1$ **to** n **do** $x_i \leftarrow M(x_{i-1}, p)$
6. $IMG(pixel) \leftarrow IMG(pixel) + x_n$
7. **if** $IMG(pixel) > x_{\max}$ **do** $IMG(pixel) \leftarrow IMG(pixel\ ) - \delta x$
8. **for** $pixel\ \leftarrow 1$ **to** m **do** $IMG(pixel) \leftarrow round\left((IMG(pixel) - x_{\min})\,\dfrac{255}{\delta x}\right)$

As a pixel preceding the first one, the last image pixel is taken.

The secret key of the algorithm consists of four numbers: the control parameter a, the number of iterations n, the number of cycles j and the size of the image $m = N \times M$, i.e. the tuple $\{a,\ n,\ j,\ m\}$.

Decryption procedure

Again, let us assume for simplicity, that the encrypted image is black and white.

Firstly, to recover the original image, pixels of ciphertext should be converted to a matrix of floating-point numbers. Then, from the value of m-th pixel we subtract the value of the previous pixel, which was earlier iterated n times on the chaotic map. If the result is less than 0, it should be normalized by adding δx. These operations are repeated for all pixels of the image, remembering about the assumed condition for the last pixel. The whole image is processed j times. If we have a color image, we expand the algorithm in a way analogous to the encryption procedure.

The decryption algorithm may be presented more formally as:

Algorithm 2. CML cipher decryption procedure.

Input: j – number of cycles, part of a cipher key
m – number of pixels, part of a cipher key
IMG – stream of encrypted image pixels
n – number of iterations, part of a cipher key
M – chaotic map
p – control parameter, part of a cipher key
$x_{\min},\ x_{\max}$ – borders of the attractor, dependent on p
$\delta x = x_{\max} - x_{\min}$

Output: stream of decrypted pixels

1. **for** $pixel \leftarrow m$ **to** 1 **do** $IMG(pixel) \leftarrow x_{\min} + \delta x \dfrac{IMG(pixel)}{255}$
2. **for** $cycle \leftarrow j$ **to** j **do** steps from 3 to 7
3. **for** $pixel \leftarrow m$ **to** 1 **do** steps from 4 to 7
4. $x_0 \leftarrow IMG(pixel - 1)$
5. **for** $i \leftarrow 1$ **to** n **do** $x_i \leftarrow M(x_{i-1}, p)$
6. $IMG(pixel) \leftarrow IMG(pixel) - x_n$
7. **if** $IMG(pixel) < 0$ **do** $IMG(pixel) \leftarrow IMG(pixel) + \delta x$
8. **for** $pixel \leftarrow 1$ **to** m **do** $IMG(pixel) \leftarrow round\Big((IMG(pixel) - x_{\min})\,\dfrac{255}{\delta x}\Big)$

Again, as a pixel preceding the first one, the last image pixel is taken.

To achieve a reasonable security level, authors [21] recommend using the key values not smaller than: $n = 75$ and $j = 3$. Obviously, along with the growth of the both parameters, the computation time of the algorithm also grows. As

a remedy, the authors claim that it is possible to increase the security level, by simply using different values of a and n for each pixel. It does not increase the computation time significantly.

3. Certain problems with CML encryption scheme

The version of CML algorithm, presented in [21], unfortunately contains several mistakes and inaccuracies, which make the algorithm impossible for practical use. In this section we point out the observed inaccuracies of the algorithm.

There exists a problem with conversion from floating-point numbers to pixel colors in the algorithm. Using $round(\cdot)$ operator results in small information leakage and initial conditions for the map in the encryption device, and the decryption device are slightly different. Because of exponential sensitivity of chaotic maps, it makes a proper decryption of the original image impossible. Table 1 illustrates quantitatively this fact.

Table 1. Values of exemplary pixels in encryption device and decryption device as initial conditions, values obtained from them by using the logistic map with $a = 3.9$ and $n = 75$ and the error between these values. Slight differences grow exponentially and are unpredictable. Therefore, a proper decryption is impossible.

initial condition in encryption	initial condition in decryption	value from the map (encryption)	value from the map (decryption)	error	error percent [%]
0.457858	0.457389	0.957239	0.964287	0.007048	0.736
0.632534	0.629926	0.869645	0.904273	0.034628	3.982
0.694364	0.692040	0.103459	0.645242	0.541783	523.669
0.781274	0.778308	0.612869	0.319281	0.293588	47.904
0.811925	0.809365	0.155187	0.908149	0.752962	485.197
0.299123	0.298655	0.335547	0.910547	0.575000	171.362
0.258364	0.257247	0.732454	0.900507	0.168053	22.944
0.973357	0.971549	0.146878	0.974617	0.827739	563.556
0.375212	0.374572	0.098140	0.343083	0.244943	249.585
0.169265	0.167527	0.432184	0.097871	0.334313	77.354

It may be considered, whether the algorithm in authors' intention would not have worked in a different way: it were the floating-point numbers which should be transmitted, not the colors. Unfortunately, this form of algorithm means that the size of transmitted data grows (using floating-point variables) and that

the attack on the key-space is possible (we can easily extract extreme values from a ciphertext, which would be close to $x_{\min}$, $x_{\max}$, uniquely determining the parameter a).

There is also another problem. In order to have the floating-point values not exceeding the borders of an attractor, i.e. in the range of $[x_{\min}, x_{\max}]$, the normalization process was introduced (adding and subtracting δx). Unfortunately, the normalization process does not work properly, what is illustrated in Tables 2 and 3 (it was assumed that the encryption procedure encounters values close to the extreme).

Table 2. Result of executing the algorithm on the extreme values $x_{\max} = 0.9$ and $x_{\min} = 0.1$, with $\delta x = 0.9 - 0.1 = 0.8$. In the right down quarter we can see leaving the range of $[x_{\min}, x_{\max}]$, to which normalization ought to prevent.

$+$	**0.1**	**0.9**
0.1	0.2	$1.0 - \delta x = 0.2$
0.9	$1.0 - \delta x = 0.2$	$1.8 - \delta x = 1.0$

Table 3. Result of executing the algorithm on the extreme values $x_{\max} = 0.9$ and $x_{\min} = 0.1$, with $\delta x = 0.9 - 0.1 = 0.8$. In three cases we see 0, which means leaving the range of $[x_{\min}, x_{\max}]$, which normalization ought to prevent.

$-$	**0.1**	**0.9**
0.1	0.0	$-0.8 + \delta x = 0.0$
0.9	0.8	0.0

As we can see, with the values close to extreme, it occurs that normalization causes leaving outside the attractor.

Moreover, the algorithm brings some questions about the map. A dynamical system is chaotic, when it is mixing and has at least one positive Lyapunov exponent [23]. The logistic map is chaotic for control parameters $a > s_\infty$, where $s_\infty = 3.5699\ldots$ is the Feigenbaum point. Unfortunately, for the logistic map, even for many isolated values $a > s_\infty$, the Lyapunov exponents λ take negative values (a diagram of numerical calculations may be seen e.g. in [24]). To those a, chaos does not appear in the system and as we know, security of the cipher was based on chaos. What is more, using the chaotic map in finite precision is associated with a so-called dynamical degradation. It emerges as a result of quantization of state space of the mathematical system, which operates in the set equivalent with $\mathbf{R}$. This phenomenon causes appearing of short cycles and general worsening of statistical properties of a system [5, 12]. It may also be used to attack a chaotic cryptosystem [12].

In the CML algorithm, the size of the key-space depends on types of variables used in the implementation, which was not precisely defined by the authors. What is more, the range of parameters n and j is obviously limited and makes a small contribution to overall size of the key-space. Using different values of n and a for each pixel, which was suggested in [21], is not a good idea because of the large length of such a key.

4. An improved CML algorithm

Because of the problems described above, we try to show the direction of necessary changes in the CML algorithm. As a chaotic map we chose the 1D *piece-wise linear chaotic map* (1D PWLCM) defined by the formula:

$$F(x,p) = \begin{cases} x/p, & x \in [0,\ p), \\ (x-p)/(0.5-p), & x \in [p,\ 0.5), \\ (1-x-p)/(0.5-p), & x \in [0.5,\ 1-p), \\ (1-x)/p, & x \in [1-p,\ 1], \end{cases} \tag{4.1}$$

where $p \in (0,\ 0.5)$ (see Fig. 1). The map is simple (easy to compute) and is chaotic in the whole range of parameter p changes [12].

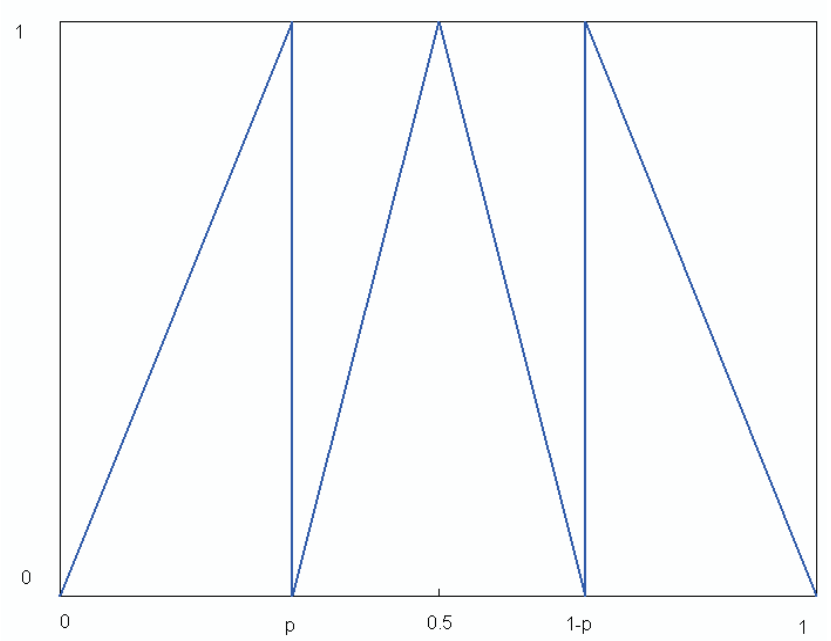

Fig. 1. 1D piece-wise linear chaotic map, defined by Eq. (4.1).

In order to avoid effects of dynamical degradation, we use a perturbation algorithm [12], defined by the formula:

$$x_{i+1} = rem\left(\mathbf{F}\left(x_i\right) + r_i,\ 1\right), \tag{4.2}$$

where $\mathbf{F}\colon [0,\ 1] \to [0,\ 1]$ is a given chaotic map, $r_i \in (0,\ 1)$ – i-th pseudorandom number from the generator with uniform distribution (PRNG), which perturbs the map, and $rem(\cdot,\ 1)$ means taking rest from division by 1 (taking the fractional part). The state of generator is determined by a number s (generator seed). With this map, conversion to floating-point numbers is done by simple division by 255, while returning to colors is done by multiplication by 255 and rounding to the nearest integer. Conversion does not make problems with decryption, because it is done always before and after making n iterations on the map. Therefore, it costs more computations. All the other operations are performed in group $\mathbf{Z}_{256}$ and the normalization process has an obvious nature of adding or subtracting 256. The key of the cipher is a pair $(p,\ s)$, i.e., the map control parameter p and the seed s of the pseudorandom number generator.

Encryption and decryption procedures are described below.

Algorithm 3. Improved CML cipher encryption procedure.	
Input:	s – seed, part of a cipher key
	j – number of cycles
	m – number of pixels
	IMG – stream of image pixels
	n – number of iterations
	M – chaotic map
	p – control parameter, part of a cipher key
	$RAND$ – stream of pseudorandom numbers from PRNG
Output:	stream of encrypted pixels
1.	initialize pseudorandom number generator with seed s
2.	**for** $i \leftarrow 1$ **to** $j \cdot m$ **do** $RAND(i) \leftarrow$ *next number from PRNG*
3.	**for** $cycle \leftarrow 1$ **to** j **do** steps from 4 to 10
4.	**for** $pixel \leftarrow 1$ **to** m **do** steps from 5 to 10
5.	$x_0 \leftarrow \dfrac{IMG(pixel\ -1)}{255}$
6.	**for** $i \leftarrow 1$ **to** n **do** $x_i \leftarrow M(x_{i-1}, p)$
7.	$k = m \cdot (cycle\ -1) +\ pixel$
8.	$x_n \leftarrow (x_n +\ RAND(k)) \bmod 1$
9.	$IMG(pixel) \leftarrow IMG(pixel) +\ round(x_n \cdot 255)$
10.	**if** $IMG(pixel) > 255$ **do** $IMG(pixel) \leftarrow IMG(pixel) - 256$

Algorithm 4. Improved CML cipher decryption procedure.

Input: s – seed, part of a cipher key
j – number of cycles
m – number of pixels
IMG – stream of encrypted image pixels
n – number of iterations
M – chaotic map
p – control parameter, part of a cipher key
$RAND$ – stream of pseudorandom numbers from PRNG

Output: stream of decrypted pixels

1. initialize pseudorandom number generator with seed s
2. **for** $i \leftarrow 1$ **to** $j \cdot m$ **do** $RAND(i) \leftarrow$ *next number from PRNG*
3. **for** $cycle \leftarrow j$ **to** 1 **do** steps from 4 to 10
4. **for** $pixel \leftarrow m$ **to** 1 **do** steps from 5 to 10
5. $x_0 \leftarrow \dfrac{IMG(pixel\ -1)}{255}$
6. **for** $i \leftarrow 1$ **to** n **do** $x_i \leftarrow M(x_{i-1}, p)$
7. $k = m \cdot (cycle\ -1) +\ pixel$
8. $x_n \leftarrow (x_n +\ RAND(k)) \bmod 1$
9. $IMG(pixel) \leftarrow IMG(pixel) -\ round(x_n \cdot 255)$
10. **if** $IMG(pixel) < 0$ **do** $IMG(pixel) \leftarrow IMG(pixel) + 256$

In both algorithms, as a pixel preceding the first one, the last image pixel is taken. Parameter k is used for indexing the stream from PRNG. As we can see, in the decryption procedure the numbers from generator are used in reverse order. Other operations were described above.

5. Implementation and security analysis

The improved version of the CML algorithm, presented above, was implemented in Matlab language to study its practical functioning. In our implementation p is a double precision floating-point number and s is an unsigned integer, both 64-bits. Entropy measurement and symmetry of the map [24] cause that the bit count of p is effectively equal to 53. Thus the size of the key-space is about $2^{53} \times 2^{64} = 2^{117}$, which is quite good for some applications.

The tests were done for an image of the size 256×256. The ciphertext turned out to be illegible even after a small number of iterations (e.g. $n = 5$), although it was checked that the ciphertext is not sensitive to small changes in the key. Thus, the accepted values were $n = 25$ and $j = 5$. The used key was (0.12345, 123). The plain image, the encrypted image and their grayscale histograms are shown in Fig. 2.

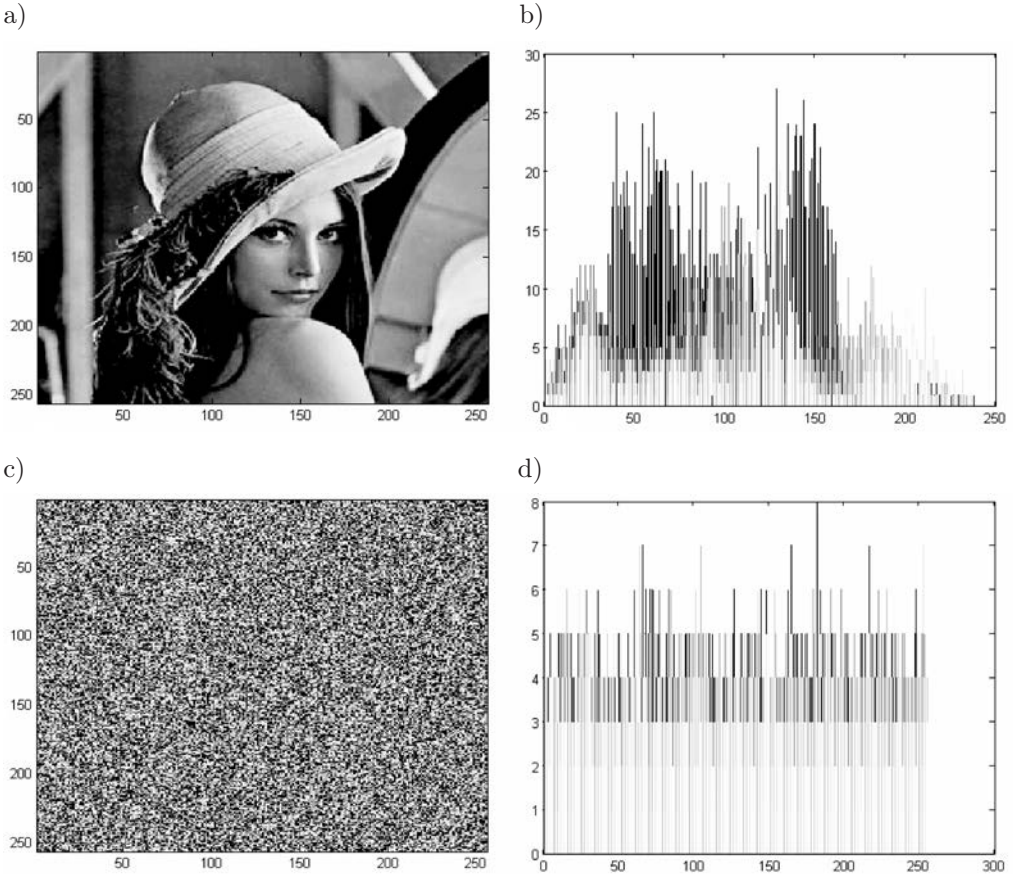

FIG. 2. Plain image (a), encrypted image (c) and their histograms (b and d).

As we can see, the encryption process is quite good with making histogram of grayscale values flat. It was checked that increasing iteration and cycle numbers do not provide significant improvement. In fact, histogram d) is not better than histogram made after only a few iterations and cycles. A situation with pixel's correlation is similar, see Fig. 3.

Tests of the sensitivity on a key change was carried out with parameters $n = 25$, $j = 5$. Firstly, the $(0.222,\ 2)$ key was used, then $(0.222 - 2^{-53},\ 2)$. Next, we took the first part of the key as constant and changed the other. First, the key was $(0.222,\ 123456789)$, then $(0.222,\ 123456788)$. Results of the subtraction of the encrypted image pairs are shown in Fig. 4.

The obtained ciphertexts in the a) case are identical only in 0.36%, whereas in case b) only in 0.41%. In the next test we encrypted the image with the key $(0.222,\ 2)$ and decrypted with the key $(0.222 - 2^{-53},\ 2)$. The other pair of keys was investigated analogously. The results are shown in Fig. 5.

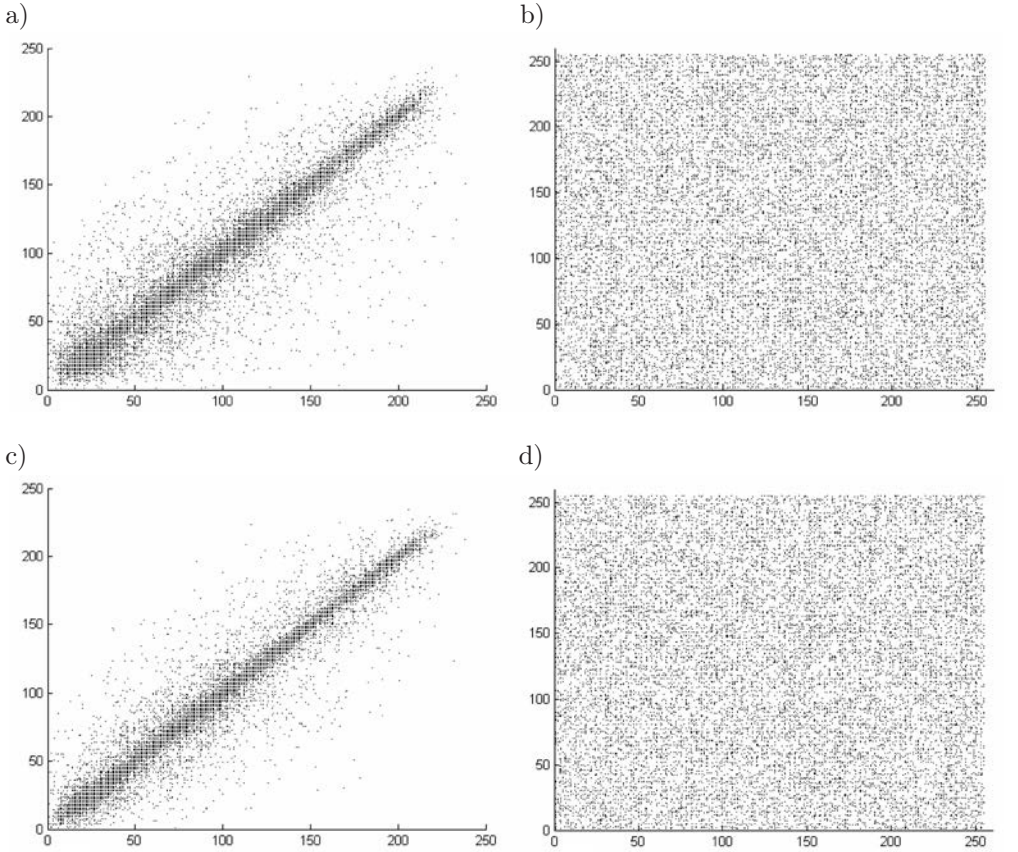

FIG. 3. Correlation between the neighboring pixels of plain image and encrypted image: horizontally (a and b), vertically (c and d).

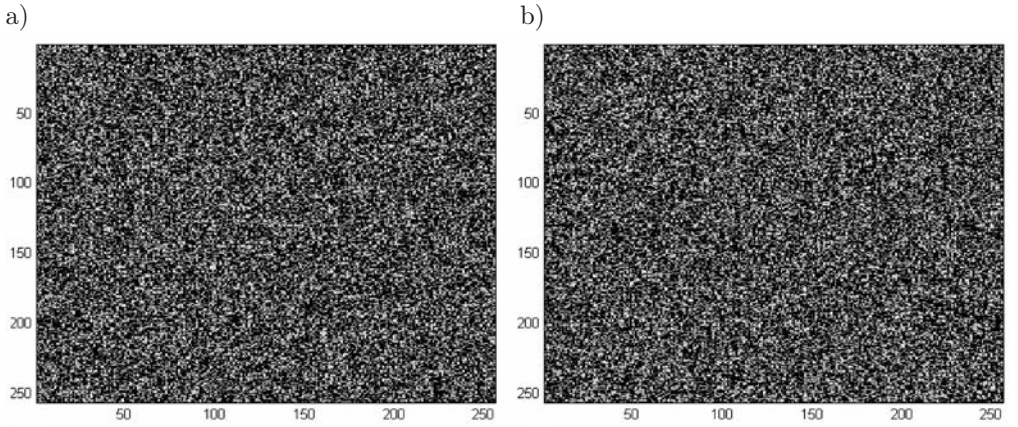

FIG. 4. Result of subtracting the images encrypted with the keys: a) $(0.222,\ 2)$ and $\big(0.222 - 2^{-53},\ 2\big)$; b) $(0.222,\ 123456789)$ and $(0.222,\ 123456788)$.

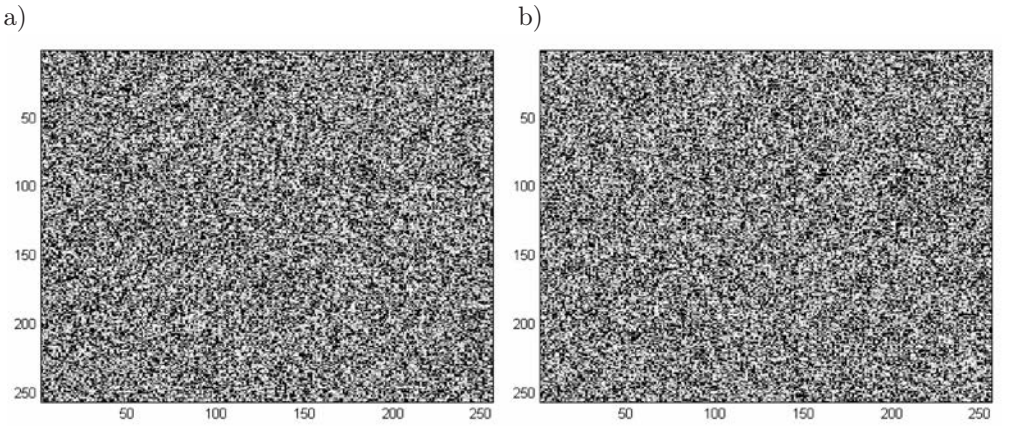

FIG. 5. Result of decryption an encrypted image using another key: a) encryption using $(0.222,\ 2)$, decryption using $(0.222 - 2^{-53},\ 2)$; b) encryption using $(0.222,\ 123456789)$, decryption using $(0.222,\ 123456788)$.

For the tests of sensitivity on small plain image changes, we used an image created by making black a square of 2×2 pixels of the original image. We used the key equal to $(0.222,\ 2)$, $n = 25$ iterations, $j = 5$ cycles. The result of subtracting of both images is presented in Fig. 6.

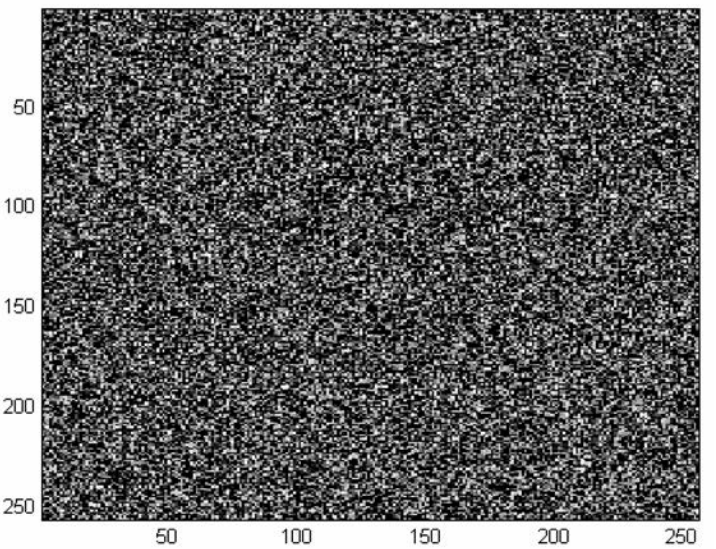

FIG. 6. Result of subtracting of two ciphertexts, obtained form very similar images and the same key.

The obtained ciphertexts are identical only in 0.39%. Thus, the cipher is sensitive to small changes in the plain image.

As it was mentioned, the proposed cipher's nature is similar to CBC encryption mode. We think that this property makes the chosen-plaintext attack difficult. Using chaos should prevent the cipher from the known-plaintext and ciphertext-only attacks.

6. Conclusions and future work

In this paper, we have presented several improvements of the errors found in the CML algorithm [21]. Our improved version of the cryptosystem is defined precisely, works properly and is ready for implementation. To work correctly, it does not have to make the data bigger by using floating point numbers to transmit the ciphertext. A comparison of both versions of the algorithm is presented in Table 4.

Table 4. Comparison of the two versions of the CML algorithm.

Feature	CML algorithm	New CML algorithm
key format	$\{a,\ n,\ j,\ m\}$ ={map control parameter, number of iterations, number of cycles, image size}	$\{p,\ s\}$ ={map control parameter, PRNG seed}
chaotic map used	logistic map	1D PWLCM Eq. (4.1)
dynamical degradation prevention	–	perturbation algorithm, defined by Eq. (4.2)
conversion to/from colors	twice: at the beginning and at the end; does not work properly	always before iterating map n times and after perturbation algorithm; works properly
normalization	by Eq. (2.2) and (2.3); does not work properly	by simple subtracting and adding 256; works properly
key length	using double precision numbers, 256×256 image and few more assumptions, about $2^{53} \times 2^5 \times 2^2 \times 2^8 \times 2^8 = 2^{76}$	about 2^{117}
ciphertext/ plaintext ratio	using floating point numbers as a ciphertext, $\frac{m \cdot 64 \text{ bits}}{m \cdot 8 \text{ bits}} = 8$	$\frac{m \cdot 8 \text{ bits}}{m \cdot 8 \text{ bits}} = 1$

Implementation made in Matlab language intensively increases the randomness of the image, even with small number of iterations. The pattern of the image is disappearing, the image looks like noise, histogram of grayscale values is flat and the correlation vanishes. The influence of the key and plain image to ciphertext is large, but the required number of iterations and cycles is bigger than the one required to obtain a ciphertext with a flat histogram. The size of the key-space in this implementation is about 2^{117}.

The proposed algorithm was tested with respect to its correctness, reversibility (in decryption device) and being ready for implementation. To test its effectiveness in professional applications, additional work should be carried out to write the adequate computer code in an optimal way. The required studies should include a choice of an appropriate PRNG in Algorithms 3 and 4.

References

1. C. E. Shannon, *Communication Theory of Secrecy Systems*, Bell System Technical Journal, **28**, 656–715, 1949.
2. Z. Kotulski, *Building block-ciphers: new possibilities*, Matematyka Stosowana, **4** (**45**), 1–24, 2003.
3. N. Masuda, G. Jakimoski, K. Aihara, L. Kocarev, *Chaotic block ciphers: from theory to practical algorithms*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, **53**, 6, 1341–1352, 2006.
4. G. Alvarez, S. Li, *Some basic cryptographic requirements for chaos-based cryptosystems*, International Journal of Bifurcation and Chaos, **16**, (8), 2129–2151, 2006.
5. D.-I. Curiac, D. Iercan, O. Dranga, F. Dragan, O. Banias, *Chaos-Based Cryptography: End of the Road?*, Proc. IEEE Int. Conf. Emerging Security Information, Systems and Technologies, 71–76, 2007.
6. T. Habutsu, Y. Nishio, I. Sasase, S. Mori, *A secret key cryptosystem by iterating chaotic map*, Proc. EUROCRYPT'91, LNCS **547**, 127–140, Springer, Berlin 1991.
7. E. Biham, *Cryptanalysis of the Chaotic-Map Cryptosystem Suggested at EUROCRYPT'91*, LNCS, **547**, 532, Springer, Berlin 1991.
8. Z. Kotulski, J. Szczepański, *Discrete chaotic cryptography*, Annalen der Physik, **509**, (5), 381–394, 1997.
9. M. S. Baptista, *Cryptography with chaos*, Physics Letter A, **240**, 1, 50–54, 1998.
10. Z. Kotulski, J. Szczepański, K. Górski, A. Paszkiewicz, A. Zugaj, *Application of discrete chaotic dynamical systems in cryptography – DCC method*, International Journal of Bifurcation and Chaos, **9**, 6, 1121–1135, 1999.
11. E. Alvarez, A. Fernandez, P. Garcia, J. Jimenez, and A. Marcano, *New approach to chaotic encryption*, Physics Letter A, **263**, 4–6, 373–375, 1999.
12. S. Li, *Analyses and New Designs of Digital Chaotic Ciphers*, PhD thesis, http://www.hooklee.com/Thesis/ethesis.zip, 2005.
13. S. Li, G. Chen, X. Zheng, *Chaos-Based Encryption for Digital Images and Videos*, [in:] Multimedia Security Handbook, [Eds.] B. Furht and D. Kirovski 133–167, CRC Press, Boca Raton 2004.
14. Y. Mao, G. Chen, *Chaos Based Image Encryption*, [in:] Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neural Computing and Robotics, edited by E. Bayro-Corrochano, Springer, New York 2003.

15. A. Said, W. A. Pearlman, *A new fast and efficient image codec based on set partitioning in hierarchical Trees*, IEEE Trans Circuits and Systems for Video Technology, **6**, 6, 243–250, 1996.

16. J.-C. Yen, J.-I. Guo, *A new chaotic key-based design for image encryption and decryption*, Proc. IEEE Int. Conf. Circuits and Systems, **4**, 49–52, 2000.

17. J.-C. Yen, J.-I. Guo, *A new image encryption algorithm and its VLSI architecture*, [in:] Proc. IEEE Workshop Signal Processing Systems, 430–437, 1999.

18. Y. Mao, G. Chen, S. Lian, *A novel fast image encryption scheme based on 3D chaotic baker maps*, International Journal of Bifurcation and Chaos, **14**, 10, 3613–3624, 2004.

19. S. Li, X. Zheng, X. Mou, Y. Cai, *Chaotic encryption scheme for real-time digital video*, Proc SPIE on Electronic Imaging, San Jose CA USA, Real-Time Imaging VI, **4666**, 149–160, 2002.

20. J. Fridrich, *Symmetric ciphers based on two-dimensional chaotic maps*, International Journal of Bifurcation and Chaos, **8**, 1259–1284, 1998.

21. P. Pisarchik, N. J. Flores–Carmona, M. Carpio–Valadez, *Encryption and decryption of images with chaotic map lattices*, Chaos, **16**, 033118, 2006.

22. M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, NIST Special Publication 800-38 A, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf, 2001.

23. E. Ott, *Chaos in dynamical systems*, Cambridge University Press, London 1993.

24. A. Skrobek, P. Sukiennik, *Cryptanalysis of Chaotic Product Cipher*, [in:] Advances in Information Processing and Protection, [Eds.] J. Pejaś and K. Saeed 281–290 Springer, New York 2007.

25. K. Jastrzębski, *Cryptanalysis of some chaos-based algorithm of an image encryption*, MSc Thesis, supervisor: Z. Kotulski, Warsaw University of Technology, September 2007.

Received February 16, 2009; revised version September 17, 2009.