Multimedia Tools and Applications (2022) 81:34027–34046
https://doi.org/10.1007/s11042-022-12368-3

An 8-bit precision cipher for fast image encryption

J. S. Armand Eyebe Fouda[1,2] · Wolfram Koepf[2]

Received: 3 January 2021 / Revised: 26 April 2021 / Accepted: 18 January 2022 /
Published online: 21 April 2022

Abstract
Implementing chaos based ciphers usually involves 32-bit floating-point arithmetics that is hardware resources costly. The limitation of the computational precision is hardware imposed and transforms chaotic orbits into limit cycles with short periods, hence alters their randomness. In cryptographic applications, short period dynamics and weak randomness result in security issues. In order to address this concern, we propose an 8-bit precision cipher that can be implemented with low-end microprocessors running 8-bit integer arithmetics. The cipher includes a quantized pseudo-random number generator (QPRNG) based on a 16-dimensional quantized Arnold's cat map (QACM). We used entropy measure, statistical, sensitivity and key space analyses to evaluate its security level under limited computational precision. Simulation results attest that it is as highly secure as those involving real-number arithmetics, even for only 8-bit precision. We also showed that the period of the proposed QACM can be chosen such that $T_x > 10^{27}$, which is very large as compared to existing QACM. Such a large period implies a high randomness of the derived QPRNG that is confirmed by statistical NIST tests. Contrary to existing ciphers that include other chaotic systems than the QACM for strengthening the security level, ours is exclusively based on the QACM and is fast, despite the included high-dimensional QACM.

Keywords Chaos · Multimedia encryption · Information security

1 Introduction

Chaos based ciphers mostly include permutation and diffusion operations [16, 22, 25, 36]. The permutation operation allows to shuffle the plaintext characters while the diffusion process changes the character values. For the generation of permutation and diffusion keys that are necessary for these two operations, a random number generator is required. Mostly,

✉ J. S. Armand Eyebe Fouda
fouda@mathematik.uni-kassel.de

Wolfram Koepf
koepf@mathematik.uni-kassel.de

[1] Department of Physics, Faculty of Science, University of Yaoundé I, P.O. Box 812, Yaoundé, Cameroon

[2] Institute of Mathematics, University of Kassel, Heinrich-Plett Str. 40, 34132 Kassel, Germany

random numbers are derived from chaotic systems due to their ergodicity and sensitivity to initial conditions [16, 30]. These chaotic systems usually involve real-number arithmetic, while data to be encrypted usually are integer encoded characters (image pixels, text characters...). Moreover, in some special cases, the target hardware implementing the chaotic system is precision limited [7, 20]. Although there is no need for converting real-numbers into integers in some chaos based ciphers to generated permutation keys [15, 16], the generation of the diffusion keys inevitably imposes the chaotic numbers to fit the phase space and the precision of the plaintext. For ASCII symbols for example, the values used for the diffusion operation need to be 8-bit encoded for the ciphertext to remain an ASCII symbol. In the case of a gray-level image, the diffusion values should be 8-bit encoded for the encrypted image to preserve its initial format.

The orbits generated from a chaotic system using finite precision are no longer chaotic, but limit cycles with a finite period length [2, 4, 5, 13, 14, 20, 27, 34]. Thus, the randomness of the discrete chaotic sequence is altered by the limited computational precision [12, 19, 20, 28], which seriously affects the security level of the cipher. Some investigations have been made to evaluate the impact of the data precision on the randomness of some well-known chaotic systems [32]. Indeed, using finite computational precision transforms chaotic sequences into periodic orbits with short period lengths, which does not meet requirements of cryptography. It is well known that longer periods and flat period distribution allow to overcome the limited range in the number representation of digital systems, which is very important in constructing high quality PRNGs [39]. Therefore, most of the algorithms proposed in the literature have been implemented under more than 8-bit precision condition. Wafaa et al. in [37] presented a fixed-point hardware realization of the logistic map experiencing a trade-off between computational efficiency and accuracy. They showed that the minimum bus size for the pseudo-random number generator (PRNG) to pass all the NIST tests is 45. Nagaraj et al. in [32] proposed a PRNG in which the average period length is increased by switching between robust chaotic maps. There are several works in the literature that have been carried out in order to increase the period of digital chaotic systems realized under limited precision conditions. Chunlei et al. recently investigated the effects of limited computational precision on discrete chaotic sequences [14]. They proposed a new PRNG that exhibits random sequences with period lengths longer than those of the logistic and tent maps under the same precision, but they didn't give an estimate of this period. As it is the case for many digital PRNG, the period of the exhibited orbits is usually much smaller than the number of non-trivial points of the system.

Arnold's cat map (ACM) is known to be chaotic, area-preserving, ergodic and mixing, and invertible [9, 23, 28]. It has a unique hyperbolic fixed point and the linear transformation defining it is hyperbolic. Its quantized version also forms short limit cycles whose lengths do not exceed $3m$, m being the modulo value. In the case of n-bit precision ($m = 2^n$) which is convenient for digital applications, the period of the quantized Arnold cat map (QACM) is only equal to $3 \cdot 2^{n-2}$, $n \geq 2$, which is effectively smaller than $3m$. Therefore, for the security level of ciphers including the QACM to be enhanced, its period needs to be increased [3, 13, 39]. The other properties of this interesting map can be found in the literature [3, 9, 13, 23, 28]. The ACM is used in cryptography, in digital tattoo applications, in watermarking and for random number generation to cite a few [9, 10, 29]. The QACM is particularly used for image scrambling due to its periodic nature [8]. In such a case, it is combined with another chaotic map to increase the security level of the cipher [6, 31, 31], which involves floating-point arithmethics. The QACM has been also used for the implementation of public key ciphers [24], but the latter are not secure when dealing with QACM with weak periods [26]. When the QACM is not combined with another chaotic map, to overcome the security issue caused by its weak period, most ciphers are based on its continuous phase space version that corresponds to $m = 1$ and which also involves floating-point arithmetics [18, 41]. As floating-point arithmetics is hardware resource costly, another alternative is to increase the dimensionality of the QACM, as high dimensional maps provide more complicated dynamics than lower ones for some appropriate parameter setting. In addition, highly complex dynamics enhance the confusion and diffusion properties in cryptographic applications [38]. In [39], Ta et al. proposed an approach to extend the dimension of the basic 2-dimensional (2D) QACM by using the fast pseudo-Hadamard transform. The resulted Cat-Hadamard map presents a period that is not so large for enhancing diffusion and confusion properties in cryptographic applications.

In this paper, we propose an 8-bit precision cipher that can be implemented with low-end microprocessors running 8-bit integer arithmetics. The cipher includes exclusively a quantized pseudo-random number generator (QPRNG) based on a 16-dimensional (16D) QACM that exhibits a large period. In order to considerably increase both the period and the complexity of the proposed QPRNG, we suggest to switch between different 2D QACM by defining coupling methods that allow to easily extend the dimension of the system as the number of switches increases. As the period of the set of switches is the least common multiple (LCM) of the periods of the individual switches, the switching instants are chosen as distinct prime numbers for the period of the switches set to be the product of all the individual periods. For the QPRNG to include both initial conditions and control parameters, we suggest to control an 8D time switching based QACM by another 8D time switching based QACM with amplitude-controlled switching instants. Thus, the first 8D QACM is time-controlled using prime numbers, while the second one is space-controlled. We verify that the period and the complexity of the proposed PRNG is strongly related to the number of 2D QACM and switches under interplay. Considering the large period and the complexity of the resulting QPRNG, the proposed 8-bit precision cipher involves exclusively integer arithmetics, and combines the confusion and diffusion operations in a single loop.

The rest of the paper is organized as follows: Section 2 is devoted to the generation of random integers, Section 3 presents the new cipher, Section 4 is devoted to the performance analysis of the proposed cipher, while Section 5 summarizes the paper.

2 Generation of random integers

Our purpose is to design a PRNG exhibiting as much as possible complex dynamics for multimedia encryption. In practice, chaotic systems are complex as their dynamics is nearby brownian. The particularity of the system we are going to propose is that it generates integers, instead of real numbers as it is the case for many chaotic systems.

2.1 System modeling

We consider the basic model of the QACM, which is known to be chaotic and reversible. It is also known to be periodic, according to its finite state space, and that its period depends on the initial conditions [3, 11]. The basic 2D QACM is modeled by

$$\begin{cases} x_1(t+1) = x_1(t) + x_2(t) \\ x_2(t+1) = x_1(t+1) + x_2(t) \end{cases} \mod m, \tag{1}$$

where $m \in \mathbb{N}_{>1}$, $x_i \in \mathbb{N}$, $i = 1, 2$, and $t \in \mathbb{N}$. While taking $m = 2^p$, $p \in \mathbb{N}$, the minimal period of the corresponding QACM is

$$\Pi(p) = 3 \times 2^{p-2},\ p > 2, \tag{2}$$

with $\Pi(1) = \Pi(2) = 3$, while its upper bound is $3m$ [3, 9, 13]. For relatively small values of m, such a short period needs to be increased to improve the performance of QACM based ciphers [17]. Working in this direction, we propose to couple four two-dimensional (2D) QACM to obtain an 8D QACM. The proposed 8D QACM is supposed to provide a large key space for data encryption .

We assume that variables x_1 and x_2 in (1) describe respectively the momentum and the position of a particle. For extending this assumption to the 8D system, let us assume that x_i, $1 \le i \le 4$, are the momenta of four particles and x_i, $5 \le i \le 8$ the corresponding positions. Then the behavior of the first particle (or first QACM) is described by (x_1, x_5), the second one by (x_2, x_6), the third particle by (x_3, x_7) and the fourth one by (x_4, x_8). The state of the system is described by $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)^T$, where $(\cdot)^T$ is the transpose of $(\cdot)$. In order to increase the basic period of the 8D QACM, one can switch between different configurations, i.e., systems with different initial conditions. Such a switching process can be seen as shock occurrences between particles. We assume that there are two shock occurrences that can suddenly change the behavior of each variable x_i (four shocks per particle) and that these shocks periodically occur in time. Therefore, there are 16 shock occurrences or shock instants d_i that influence the behavior of the whole system. The corresponding shock instants vector is noted as $\mathbf{d} = (d_1, d_2, \cdots, d_{16})^T$. We can also consider more than four shocks per particle without modifying the dimension of the phase space of the system: the dimension of the shock space is independently chosen of that of the system state space. Now considering a linear coupling between particles for describing interactions and including shocks between particles, we define the following general coupling term:

$$\begin{aligned} x_i(t+1) &= x_i(t) + a_i \cdot x_j(t+\tau_j) + a_{i+8}(1-a_i) \cdot x_k(t+\tau_k) \\ &\quad + (1 - a_i - a_{i+8} + a_i a_{i+8}) \cdot x_l(t+\tau_l) \mod 2^p \end{aligned} \tag{3}$$

where $1 \le i \ne j \ne k \ne l \le 8$, $a_i = \delta\,(t \mod d_i)$, $\tau_j = 0$ (resp. $\tau_k = 0$, $\tau_l = 0$) if $i < j$, (resp. $i < k, i < l$), and $\tau_j = 1$ (resp. $\tau_k = 1$, $\tau_l = 1$) if $i > j$ (resp. $i > k, i > l$). $\delta(t)$ is the Dirac function and the coefficients $a_i(t)$ are defined such that

$$a_i(t) = \begin{cases} 1, & \text{if } 0 \equiv t \mod d_i; \\ 0, & \text{otherwise.} \end{cases}. \tag{4}$$

There are many coupling possibilities and we set the following

$$\begin{cases} x_1(t{+}1){=}x_1(t){+}a_1x_5(t){+}(1-a_1)a_9x_8(t){+}(1-a_1-a_9{+}a_1a_9)x_7(t) \\ x_2(t{+}1){=}x_2(t){+}a_2x_6(t){+}(1-a_2)a_{10}x_7(t){+}(1-a_2-a_{10}{+}a_2a_{10})x_5(t) \\ x_3(t{+}1){=}x_3(t){+}a_3x_7(t){+}(1-a_3)a_{11}x_6(t){+}(1-a_3-a_{11}{+}a_3a_{11})x_8(t) \\ x_4(t{+}1){=}x_4(t){+}a_4x_8(t){+}(1-a_4)a_{12}x_5(t){+}(1-a_4-a_{12}{+}a_4a_{12})x_6(t) \\ x_5(t{+}1){=}x_5(t){+}a_5x_1(t{+}1){+}(1-a_5)a_{13}x_3(t{+}1){+}(1-a_5-a_{13}{+}a_5a_{13})x_2(t{+}1) \\ x_6(t{+}1){=}x_6(t){+}a_6x_4(t{+}1){+}(1-a_6)a_{14}x_2(t{+}1){+}(1-a_6-a_{14}{+}a_6a_{14})x_3(t{+}1) \\ x_7(t{+}1){=}x_7(t){+}a_7x_2(t{+}1){+}(1-a_7)a_{15}x_1(t{+}1){+}(1-a_7-a_{15}{+}a_7a_{15})x_4(t{+}1) \\ x_8(t{+}1){=}x_8(t){+}a_8x_3(t{+}1){+}(1-a_8)a_{16}x_4(t{+}1) + (1-a_8-a_{16}+a_8a_{16})x_1(t+1) \end{cases} \mod 2^p, \tag{5}$$

which can be put into matrix form as

$$\mathbf{x}(t+1) = A(t)\mathbf{x}(t) \mod 2^p, \tag{6}$$

where the matrix $A(t)$ is defined as

$$A(t) = \begin{pmatrix} 1 & 0 & 0 & 0 & m_{15} & 0 & m_{17} & m_{18} \\ 0 & 1 & 0 & 0 & m_{25} & m_{26} & m_{27} & 0 \\ 0 & 0 & 1 & 0 & 0 & m_{36} & m_{37} & m_{38} \\ 0 & 0 & 0 & 1 & m_{45} & m_{46} & 0 & m_{48} \\ m_{51} & m_{52} & m_{53} & 0 & m_{55} & m_{56} & m_{57} & m_{58} \\ 0 & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} & m_{67} & m_{68} \\ m_{71} & m_{72} & 0 & m_{74} & m_{75} & m_{76} & m_{77} & m_{78} \\ m_{81} & 0 & m_{83} & m_{84} & m_{85} & m_{86} & m_{87} & m_{88} \end{pmatrix}, \tag{7}$$

with $m_{15} = a_1; m_{17} = 1 - a_1 - a_9 + a_1a_9; m_{18} = (1 - a_1)a_9;$
$m_{25} = 1 - a_2 - a_{10} + a_2a_{10}; m_{26} = a_2; m_{27} = (1 - a_2)a_{10};$
$m_{36} = (1 - a_3)a_{11}; m_{37} = a_3; m_{38} = 1 - a_3 - a_{11} + a_3a_{11};$
$m_{45} = (1 - a_4)a_{12}; m_{46} = 1 - a_4 - a_{12} + a_4a_{12}; m_{48} = a_4;$
$m_{51} = a_5; m_{52} = 1 - a_5 - a_{13} + a_5a_{13}; m_{53} = (1 - a_5)a_{13};$
$m_{55} = 1 + a_5a_1 + (1 - a_5 - a_{13} + a_5a_{13})(1 - a_2 - a_{10} + a_2a_{10});$
$m_{56} = (1 - a_5 - a_{13} + a_5a_{13})a_2 + (1 - a_5)a_{13}(1 - a_3)a_{11};$
$m_{57} = (1 - a_5 - a_{13} + a_5a_{13})(1 - a_2)a_{10} + a_5(1 - a_1 - a_9 + a_1a_9) + (1 - a_5)a_{13}a_3;$
$m_{58} = a_5(1 - a_1)a_9 + (1 - a_5)a_{13}(1 - a_3 - a_{11} + a_3a_{11});$
$m_{62} = (1 - a_6)a_{14}; m_{63} = 1 - a_6 - a_{14} + a_6a_{14}; m_{64} = a_6;$
$m_{65} = a_6(1 - a_4)a_{12} + (1 - a_6)a_{14}(1 - a_2 - a_{10} + a_2a_{10})$
$m_{66} = 1 + a_6(1 - a_4 - a_{12} + a_4a_{12}) + (1 - a_6)a_{14}a_2 + (1 - a_6 - a_{14} + a_6a_{14})(1 - a_3)a_{11};$
$m_{67} = (1 - a_6)a_{14}(1 - a_2)a_{10} + (1 - a_6 - a_{14} + a_6a_{14})a_3;$
$m_{68} = a_6a_4 + (1 - a_6 - a_{14} + a_6a_{14})(1 - a_3 - a_{11} + a_3a_{11});$
$m_{71} = (1 - a_7)a_{15}; m_{72} = a_7; m_{74} = 1 - a_7 - a_{15} + a_7a_{15};$
$m_{75} = a_7(1 - a_2 - a_{10} + a_2a_{10}) + (1 - a_7)a_{15}a_1 + (1 - a_7 - a_{15} + a_7a_{15})(1 - a_4)a_{12};$
$m_{76} = a_7a_2 + (1 - a_7 - a_{15} + a_7a_{15})(1 - a_4 - a_{12} + a_4a_{12});$
$m_{77} = 1 + a_7(1 - a_2)a_{10} + a_{15}(1 - a_7)(1 - a_1 - a_9 + a_1a_9);$
$m_{78} = (1 - a_7 - a_{15} + a_7a_{15})a_4 + (1 - a_7)a_{15}(1 - a_1)a_9;$
$m_{81} = 1 - a_8 - a_{16} + a_8a_{16}; m_{83} = a_8; m_{84} = (1 - a_8)a_{16};$
$m_{85} = (1 - a_8)a_{16}(1 - a_4)a_{12} + (1 - a_8 - a_{16} + a_8a_{16})a_1$
$m_{86} = a_8(1 - a_3)a_{11} + (1 - a_8)a_{16}(1 - a_4 - a_{12} + a_4a_{12});$
$m_{87} = a_8a_3 + (1 - a_8 - a_{16} + a_8a_{16})(1 - a_1 - a_9 + a_1a_9);$
$m_{88} = 1 + a_8(1 - a_3 - a_{11} + a_3a_{11}) + (1 - a_8)a_{16}a_4 + (1 - a_8 - a_{16} + a_8a_{16})(1 - a_1)a_9.$

The matrix of the system at iteration t is equal to the product of the first t matrices of occurring shocks. As each coefficient a_k, $1 \leq k \leq 16$, can either be 0 or 1, the maximum number of distinct matrices is $N_A = 2^{16}$. The distribution of the matrices is periodic and its period T_A is equal to the least common multiple (LCM) of $\{d_k\}_{k=1,2,\dots,16}$. While setting d_k as distinct prime numbers, T_A takes its maximum value, that is

$$T_A = \prod_{k=1}^{16} d_k. \tag{8}$$

The behavior of the whole system is thus the modulation of the behaviors of the individual 2D QACM. Therefore, the period $T_{\mathbf{x}}$ of the system is then

$$T_{\mathbf{x}} = \Pi(p)T_A, \tag{9}$$

This period depends on the choice of d_k once p has been fixed. We choose $\mathbf{d} = (5, 7, 11, 13, 17, 19, 23, 29, 211, 223, 227, 229, 233, 239, 241, 251)^T$, which corresponds to the minimal period $T_{\mathbf{x}} = 8.8844 \times 10^{27} \cdot \Pi(p)$. Such a period is sufficiently large, as compared to the basic period of the QACMs. The particular case $p = 0$, hence $m = 1$, corresponds to the ACM that exhibits chaotic behaviors in a continuous phase space.

Although the QACM period $\Pi(p)$ is multiplied by the T_A factor in the proposed system, the orbit length still depends on the initial conditions, and it could be too small for some initial condition values. For the system to be used as a pseudo-random number generator, it is better to get full length orbits. Such a requirement is satisfied by considering an external force temporarily acting on the system as

$$\mathbf{x}(t+1) = A(t)\mathbf{x}(t) + \mathbf{u}_x(t) \mod 2^p, \tag{10}$$

where

$$\mathbf{u}_x(t) = (a_1(t), 0, 0, 0, 0, 0, 0, 0)^T \tag{11}$$

for example. By this approach, the system does no longer present a steady state within the interval $[0, 2^p - 1]$, and the number of non-trivial points that an orbit may contain is $N_p = 2^{8p}$. For $p = 2$ and $\mathbf{x}(0) = (0, 0, 0, 0, 2, 0, 0, 2)^T$ for example, the orbit lengths are respectively 252 for the unforced system and 65536 for the forced one, which clearly corresponds to $N_p = 2^{16}$. Forcing the system thus considerably increases the orbit length, hence acts as a pseudo-random number generator. Figure 1 shows the corresponding first return maps of the system state $x = 2^{7p}x_8 + 2^{6p}x_7 + 2^{5p}x_6 + 2^{4p}x_5 + 2^{3p}x_4 + 2^{2p}x_3 + 2^p x_2 + x_1$. One can observe that the unforced system presents a fractal aspect while the forced one is behaving like brownian motion.

2.2 Key space extension: inclusion of control parameters

The period of the system typically depends on the number of particles and the choice of the shock instants. The shock instants, while chosen as prime numbers, need to be all distinct, otherwise the period factor does no longer follow the rule in (8). Indeed, a redundant shock instant appears only once in the computation of T_A as it corresponds to the LCM of $\{d_k\}$, which contributes to reduce the predicted period. In order to extend the key space, we modify the architecture of the system by considering amplitude-dependent shock instants, that could be used as control parameters for the generation of random numbers. Therefore, we adopt a piece-wise coupling principle interacting two distinct systems, the controlling system with time-dependent shock instants and the controlled one with phase space-related shock instants. Indeed, particular phase space values of the controlling system are used as shock instants for another system of the same type (controlled system). We agree that the

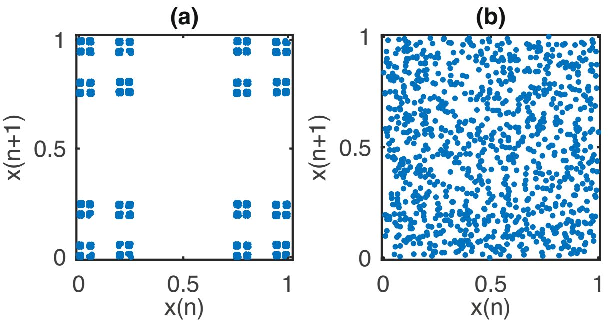

Fig. 1 (Color online) Normalized first return map of the (a) unforced and (b) forced system, $p = 2$, $\mathbf{x}(0) = (0, 0, 0, 0, 2, 0, 0, 2)^T$

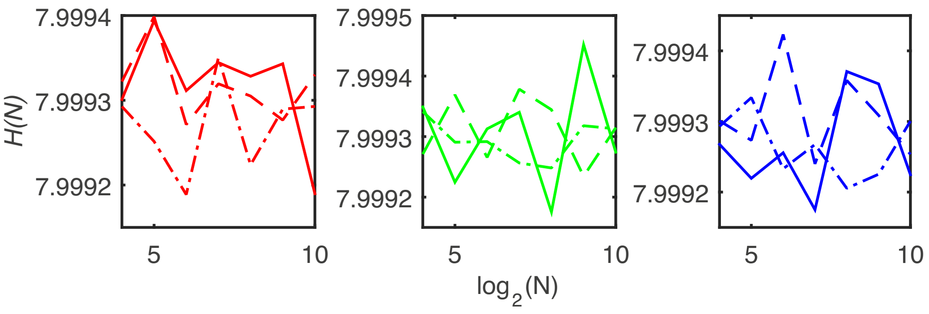

Fig. 2 (Color online) Entropy values H in terms of the number of rounds R and the block length N. From left to right are presented, respectively, the entropy values of the Red, Green and Blue colors. $R = 1$ corresponds to the solid line, $R = 2$ the dashed line and $R = 3$ the dash-dotted line

second system is amplitude-controlled by the first one, which itself is time-controlled (due to time-dependent shock instants).

Similarly to the time-controlled system, we define $\{s_k\}_{k=1,2,\ldots,16}$ the set of control parameters or shock amplitudes. The dynamics of the amplitude-controlled system thus depends on the values of control parameters s_k. The general coupling term of such a system can thus be written as

$$\begin{aligned} y_i(t+1) &= y_i(t) + b_i \cdot y_j(t+\tau_j) + b_{i+8}(1-b_i) \cdot y_k(t+\tau_k) \\ &\quad + (1 - b_i - b_{i+8} + b_i b_{i+8}) \cdot y_l(t+\tau_l) \mod 2^q, \end{aligned} \tag{12}$$

where $i \neq j \neq k \neq l$, $\tau_j = 0$ (resp. $\tau_k = 0$, $\tau_l = 0$) if $i < j$, (resp. $i < k$, $i < l$), and $\tau_j = 1$ (resp. $\tau_k = 1$, $\tau_l = 1$) if $i > j$ (resp. $i > k$, $i > l$). The coefficients b_i are defined as

$$b_i(t) = \begin{cases} 1, & \text{if } x_i < s_i; \\ 0, & \text{otherwise,} \end{cases}. \tag{13}$$

and

$$b_{i+8}(t) = \begin{cases} 1, & \text{if } x_i < s_{i+8}; \\ 0, & \text{otherwise,} \end{cases}. \tag{14}$$

with $1 \leq i \leq 8$, $q \in \mathbb{N}_{\geq 1}$ and $s_i < s_{i+8}$.

For the system to exhibit full range orbits, we also consider a forcing term $\mathbf{u}_y$. The corresponding amplitude-controlled system can thus be put into the following form

$$\mathbf{y}(t+1) = B(t)\mathbf{y}(t) + \mathbf{u}_y(t) \mod 2^q. \tag{15}$$

The elements of the matrix $B(t)$ are similar to those of matrix $A(t)$, except that the coefficients a_i are replaced by b_i. The main advantage of this approach is that the two systems can be run in parallel, which can easily allow to speed up the generation of random numbers. Moreover, the precision of the two systems are completely independent, which also means that the amplitude-controlled system can be seen as a converter of the time-controlled system. Indeed, in the case the phase space of the time-controlled system is continuous, that of the amplitude-controlled one can be seen as its digitized version: it acts like an analogue-to-digital converter. In the case $p > 1$ and $q > 1$, the complete system is a 16-dimensional time varying QACM, and can be written as

$$\mathbf{z}(t+1) = \begin{pmatrix} A(t) & 0 \\ 0 & B(t) \end{pmatrix} \mathbf{z}(t) + \mathbf{u}(t) \mod 2^r, \tag{16}$$

where $r = (p, q)^T$, p being the precision of the controlling system and q that of the controlled system. This system thus contains 48 key parameters, namely 16 initial conditions and 16 control parameters, 8 forcing parameters for the forcing system, and 8 other forcing parameters for the controlled system. Such a key length is large enough for designing secure ciphers. As the controlling system is periodic, the controlled system also is periodic, knowing that it is quantized.

2.3 Evaluation of the randomness of the system

The NIST-800-22 test suite is useful for evaluating statistical properties and conclude on the randomness of our system. Such an evaluation is required for the system to be used as PRNG for data encryption. Therefore, we applied the NIST test to our time varying QACM for various initial conditions, control parameters and precisions r. For simplification purposes, we set $s_i = \left\lfloor \frac{2^p}{3} \right\rfloor$ and $s_{i+8} = 2\left\lfloor \frac{2^p}{3} \right\rfloor$, $1 \leq i \leq 8$, $\mathbf{u}_x$, $\mathbf{u}_y$ as in (11).

Table 1 shows the results obtained with 21 different initial conditions ($x = 0$ to $x = 20$, $y = x$), where

$$x = \sum_{k=1}^{8} 2^{8(k-1)} x_k, \tag{17}$$

and

$$y = \sum_{k=1}^{8} 2^{8(k-1)} y_k. \tag{18}$$

The sequence length is set as $N = 10^6$ for each initial condition. The results are presented for $p = 8$ and various values of q. It then appears that the controlling system passes all the

Table 1 P-values [35] of the NIST-800-22 suite test in terms of the number of encoding bits $p = 8$

	x	y			
Statistical test	$p = 8$	$q = 1$	$q = 2$	$q = 4$	$q = 8$
Frequency	0.7887	0.3925	0.1866	0.5852	0.1866
Block Frequency	0.9411	0.0000	0.5852	0.2430	0.6890
Cumulative Sums	0.0151	0.4846	0.6890	0.2430	0.5852
Runs	0.9411	0.0000	0.2236	0.9411	0.4846
Longest Run	0.1866	0.6890	0.7887	0.0571	0.7887
Rank	0.6890	0.4846	0.3115	0.3115	0.0414
FFT	0.6890	0.0781	0.2430	0.9411	0.4846
Non-Overlapping Template	0.5852	0.0000	0.5852	0.0298	0.6890
Overlapping Template	0.6890	0.1056	0.3925	0.4846	0.3115
Universal	0.4846	0.6890	0.5852	0.4846	0.2430
Approximate Entropy	0.2430	0.0000	0.3115	0.0414	0.7887
Random Excursions	0.4373	0.9114	0.9763	0.0127	0.5341
Random Excursions Variant	0.1626	0.7399	0.3799	0.0909	0.7399
Serial-1	0.6890	0.0000	0.3115	0.8755	0.9809
Serial-2	0.3115	0.4846	0.3115	0.2430	0.4846
Linear Complexity	0.6890	0.3115	0.5852	0.6890	0.0781

statistical tests. Five tests (Block Frequency Test, Runs Test, the Non-overlapping Template Test, Approximate Entropy Test, and the Serial Test-1) were not successful for the controlled system in the case $q = 1$, while it successfully passes all the statistical tests for $q > 1$.

We recall that in our case, a given test is successful as the corresponding P-value is greater than 0.01. The other tests that are not successful fail for some initial values, but not for all. Such results were observed only for $q = 1$. For the rest of the paper, we are going to consider both systems (controlling and controlled) in the proposed cipher, with $p = q = 8$.

3 Proposed encryption algorithm

The algorithm we are proposing includes the QACM above presented as PRNG. It combines the confusion and diffusion steps in a single loop. Both permutation and diffusion keys are image dependent, which contribute to reinforce the security level of the cipher. We implemented it for color images for a more general use. The Algorithmic steps of the proposed cipher are given below in Algorithm 1.

Algorithm 1

```
1.  Choose the image component $R$, $G$ or $B$ to be ciphered and reshape it into a 1-D
    image vector of length $L$ ($L$ is the size of the underlying component).
2.  Choose an external 256-bit key, derive initial conditions and control parameters for
    the QACM, and set $p = q = 8$.
3.  Iterate the QACM $N$ times (after $t = 100$ iterates transient die out) and consider $\mathbf{X} =
    (x_1(t+1), x_1(t+2), \dots, x_1(t+N))^T$, and $\mathbf{Y} = (y_1(t+1), y_1(t+2), \dots, y_1(t+N))^T$
    as initial pseudo-random sequences.
4.  Sort sequence $\mathbf{X}$ into ascending order and collect the corresponding sequence of time
    index $\mathbf{I}_x$ as the initial permutation key; meanwhile, consider $\mathbf{D}_y = \mathbf{Y}$ as the initial
    diffusion key.
5.  Perform a right circular shift of the image component by $\delta = N - 1$ positions.
6.  Split the 1-D image into $n_w$ sub-images of length $N$.
7.  Consider a sub-image and use $\mathbf{I}_x$ to shuffle its pixel positions.
8.  Diffuse by XOR-ing the shuffled sub-image with $\mathbf{D}_y$.
9.  Update $\mathbf{I}_x$ and $\mathbf{D}_y$ as described in Sect. 3.2.
10. Next the sub-image and go to step 7.
11. Set $\mathbf{I}_x$ and $\mathbf{D}_y$ to their initial values in step 4.
12. Next round and go to step 5.
13. Next the image component and go to step 5.
14. Reshape each ciphered component and recombine all of them to obtain the ciphered
    image.
```

3.1 Generating permutation and diffusion keys

The permutation and diffusion keys are directly derived from the QACM using the external key. In this paper, we used a 256-bit key S, hence a set of 32 ASCII symbols $S = S_1 S_2 \dots S_{32}$ to derive initial conditions and control parameters. The corresponding decimal values are set as $\mathbf{K} = (K_1, K_2, \dots K_{32})$. There are sixteen initial conditions to be derived from the external key. For the time-controlled sub-system, these initial conditions are determined as

$$x_i(0) = \sum_{k=i}^{i+24} k \cdot K_k \mod 2^p, \tag{19}$$

while those of the amplitude-controlled sub-system are determined as

$$y_i(0) = \sum_{k=4(i-1)+1}^{4i} k \cdot K_k \mod 2^q, \tag{20}$$

where $1 \leq i \leq 8$. K_k is the decimal value of the k-th ASCII symbol S_k of the external key. Similarly to the initial conditions, the control parameters also are set from the external key. For this purpose, we first sort into ascending order values K_{17} to K_{32} and obtain a sorted vector $\mathbf{Q}$ of sixteen values ranged from 0 to 255. Thereafter, the control parameters are set as

$$\begin{cases} s_j = 6 + \left\lfloor \frac{\mathbf{Q}(j)}{3} \right\rfloor, & \text{if } 1 \leq j \leq 8; \\ s_j = 6 + 2\left\lfloor \frac{\mathbf{Q}(j)}{3} \right\rfloor, & \text{if } 9 \leq j \leq 16. \end{cases} \tag{21}$$

The above initial conditions and control parameters are then included in the QACM to generate the permutation and diffusion keys. For this purpose, we remove the first $t = 100$ iterates for transient die out and consider the following N ones to form N-length random sequences. Sequence $\mathbf{X} = (x_1(t+1), x_1(t+2), \ldots, x_1(t+N))^T$ is sorted into ascending order and the corresponding time index sequence is considered as our initial permutation key $\mathbf{I}_x$. Similarly, sequence $\mathbf{Y} = (y_1(t+1), y_1(t+2), \ldots, y_1(t+N))^T$ is used as the initial diffusion key. The generation of the permutation and diffusion keys includes steps 2 to 4 of our algorithm, and combines only integer operations. The confusion and diffusion processes are respectively realized by applying the permutation key to the plaintext sequence $\mathbf{U}$ as

$$\mathbf{U}_s = \mathbf{U}(\mathbf{I_x}), \tag{22}$$

and XOR-ing the diffusion key with the shuffled sequence $\mathbf{U}_s$ as

$$\mathbf{U}_c = \mathbf{U}_s \oplus \mathbf{D}_y. \tag{23}$$

$\mathbf{U}_c$ is a one-round encrypted sub-image and $\oplus$ is the bitwise XOR operation. Once a sub-image has been confused and diffused, the permutation and diffusion keys need to be updated before encrypting the following sub-image.

3.2 Updating permutation and diffusion keys

From a sub-image to another one are used different permutation and diffusion keys. However, all of them are related and the process to move from the previous key to the new one is called updating. For the updating of the permutation key, eight random integers are generated; in the previous sequence $\mathbf{X}'$, the first eight values are discarded, then the sequence is eight steps left shifted while the eight newly generated integers are placed at the end of the sequence. Indeed, let $\mathbf{X}' = (X'(1), X'(2), \ldots, X'(N))^T$ be the previous sequence, then the updated sequence is $\mathbf{X} = \big(X'(9), X'(10), \ldots, X'(N), x_1(1), x_2(1), \ldots, x_8(1)\big)^T$, where $x_i(1)$, $1 \leq i \leq 8$ are the newly generated integers. Note that only one iteration of the PRNG is necessary for generating the 8 integers. The updated sequence $\mathbf{X}$ is thereafter sorted into ascending order and the corresponding time index sequence is considered as the updated permutation key.

For the generation of the eight new random integers, updated initial conditions also are required. These initial conditions are image dependent. We set the first fifteen updated initial conditions as

$$\begin{cases} x_i(0) = \mathbf{K}(1 + (\mathbf{U}_c(i) \mod 32)), & \text{if } 1 \leq i \leq 8; \\ y_{i-8}(0) = \mathbf{K}(1 + (\mathbf{U}_c(i) \mod 32)), & \text{if } 9 \leq i \leq 15. \end{cases} \tag{24}$$

The last initial condition completely depends on the image and is set as

$$y_8(0) = \sum_{j=1}^{N} \mathbf{U}_c(j) \mod 2^q. \tag{25}$$

According to this updating process, the initial conditions change with the sub-image.

The diffusion key also needs to be updated for the cipher to be secure. Thus, we consider the previous diffusion key $\mathbf{D}'_y$ and set $\mathbf{Y}' = \mathbf{D}'_y$; then update eight values in $\mathbf{Y}'$ as it was the case for $\mathbf{X}'$. The updated sequence is then $\mathbf{Y} = (Y'(9), Y'(10), \ldots, Y'(N), y_1(1), y_2(1), \ldots, y_8(1))^T$, and the updated diffusion key is obtained as

$$\mathbf{D}_y = \mathbf{D}'_y + \mathbf{Y}(\mathbf{I}_x) \mod 256, \tag{26}$$

where $\mathbf{I}_x$ is the updated permutation key.

4 Results and security analysis

The performance of the algorithm is evaluated with RGB test images of size 512×512 and 256 gray levels to show the color image encryption ability of the algorithm. We also consider as sub-image length $N = 2^n$, $n \in \{4, 5, 6, 7, 8, 9, 10\}$. The encryption scheme should resist all kinds of known attacks: known-plaintext, ciphertext-only, statistical, differential and brute-force attacks. We present in this section some security analysis results for the proposed cipher, including: key-space analysis, statistical analysis, differential analysis, number of pixel change rate ($NPCR$) and unified average changing intensity ($UACI$) for one pixel difference in the plain-text image. The 256-bit external encryption key used for our simulation is set as $S =$ *azertyuiopqsdfgjazertyuiopqsdfg0*.

4.1 Statistical analysis

The statistical analysis concerns the histogram, the correlation of adjacent pixels and the information entropy of the ciphered image. The statistical analysis of several 256 gray-scale color images having different contents were evaluated and we present here the results obtained for the image of Lena (Fig. 7(a)). We evaluate the statistical parameters for different values of N. We first evaluate the entropy of image encryption using our algorithm. The entropy is determined as

$$H = -\sum_{i=0}^{255} p(v_i) \log_2(p(v_i)), \tag{27}$$

where $0 \leq v_i \leq 255$ are pixel values and $p(v_i)$ the probability of v_i. Figure 2 shows the behavior of the entropy H in terms of N and the number of rounds R. It is observed from this figure that the entropy does not depend on N and R. The entropy values of the ciphered image remain satisfactory for all the simulated block lengths as $H > 7.9992$, $\forall N \geq 16$. For $N = 16$ for example, the entropy of the red component of the image passes from

$H^{p}_{Red} = 7.253$ for the plain-image to $H_{Red} = 7.9993$ for the ciphered image with $R = 1, 2$ or 3.

Similarly, the correlation of horizontally, vertically and diagonally adjacent pixels is evaluated. For this purpose, we used Pearson's correlation coefficient defined as

$$\rho_{A,B} = \frac{E((A - \mu_A)(B - \mu_B))}{\sigma_A \cdot \sigma_B}, \tag{28}$$

where $E(\cdot)$ is the expectation value; μ and σ are mean value and standard deviation, respectively; A and B are images to be compared.

Figure 3 shows the behavior of the correlation coefficients of horizontally adjacent pixels as a function of N for various values of R. This figure also shows that the correlation coefficients of adjacent pixels do not depend on N and R. The correlation coefficients of adjacent pixels in the plain-image are, respectively, $\rho_{Red} = 0.9798$, $\rho_{Green} = 0.9691$ and $\rho_{Blue} = 0.9327$, while the corresponding values for one round ciphered image are $\rho_{Red} = -0.0045$, $\rho_{Green} = 0.0008$ and $\rho_{Blue} = -0.0004$ with $N = 16$; and $\rho_{Red} = 0.0002$, $\rho_{Green} = 0.0030$ and $\rho_{Blue} = 0.0019$ with $N = 1024$. Similar results were obtained with vertically and diagonally adjacent pixels. This result proves that the proposed cipher satisfies the zero co-correlation property that is necessary to resist statistical attacks even for $N = 16$ only.

Figure 4 shows the histograms of one round ciphered image of Lena with $N = 16$. It appears that the histogram of each component of the ciphered image is fairly uniform and significantly different from that of the corresponding plain-image component. According to this result, deducing the secret key from the cipher-text during the known/chosen plaintext attacks is a hard task.

4.2 Differential attack

The sensitivity of the cipher to small changes in the plain-image (single pixel change) is required for the cipher to resist differential attacks. The metrics commonly used to

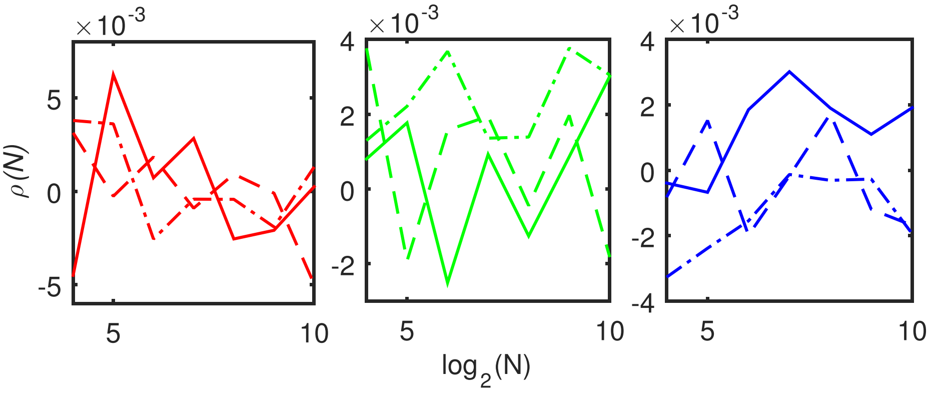

Fig. 3 (Color online) Correlation coefficient ρ of horizontally adjacent pixels in terms of the number of rounds R and the block length N. From left to right are presented, respectively, the ρ values of the Red, Green and Blue colors. $R = 1$ corresponds to the solid line, $R = 2$ the dashed line and $R = 3$ the dash-dotted line

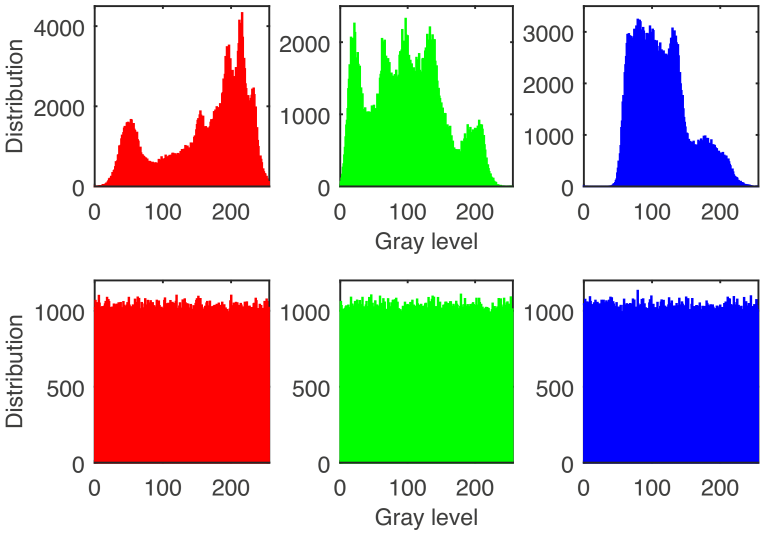

Fig. 4 (Color online) Histograms of the image of Lena. In the first line, from left to right, are shown the original histograms of the red, green and blue components, respectively; while the second line is showing the histograms of the corresponding ciphered image components

evaluate the robustness against the differential attacks are the $NPCR$ and $UACI$. The $NPCR$ between two ciphered images A and B of size $m \times n$ is defined by:

$$NPCR_{A,B} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} D(i,j)}{m \times n} \times 100 \tag{29}$$

where

$$D(i,j) = \begin{cases} 1, & \text{if } A(i,j) \neq B(i,j); \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Similarly, the $UACI$ is defined as:

$$UACI_{A,B} = \frac{100}{255} \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} |A(i,j) - B(i,j)|}{m \times n} \tag{31}$$

The result in Fig. 5 shows that the cipher is sensitive to one pixel change for $R > 1$. Indeed, the cipher is secure as $NPCR > 99.5810$ and $33.3445 \leq UACI \leq 33.5826$ ($\alpha = 0.01$ significance level) for gray images of size 512×512 [21]. In the case $R = 1$, the maximal values of NPCR and UACI were obtained for $N = 16$. We found $NPCR_{Red} = 97.8458$, $NPCR_{Green} = 97.8394$ and $NPCR_{Blue} = 97.8325$; and $UACI_{Red} = 32.8434$, $UACI_{Green} = 32.8836$ and $UACI_{Blue} = 32.8228$, thus attesting that the cipher is not secure for $R = 1$. All these values are far less than the target values that are necessary for the cipher to be secure. When $R > 1$, the cipher becomes much more secure as observed in Fig. 5. The number of rounds that are necessary for the cipher to be secure increases with N. We observed that the minimal number of rounds required is $R_{\min} = 2$ for $N = 2^4$ and that the system is secure with $R_{\min} = 3$ for all the values of N chosen on simulation. The dependency of the security level on N and R is justified by the fact that during the first round, the impact of the single changed pixel value does not affect the overall image;

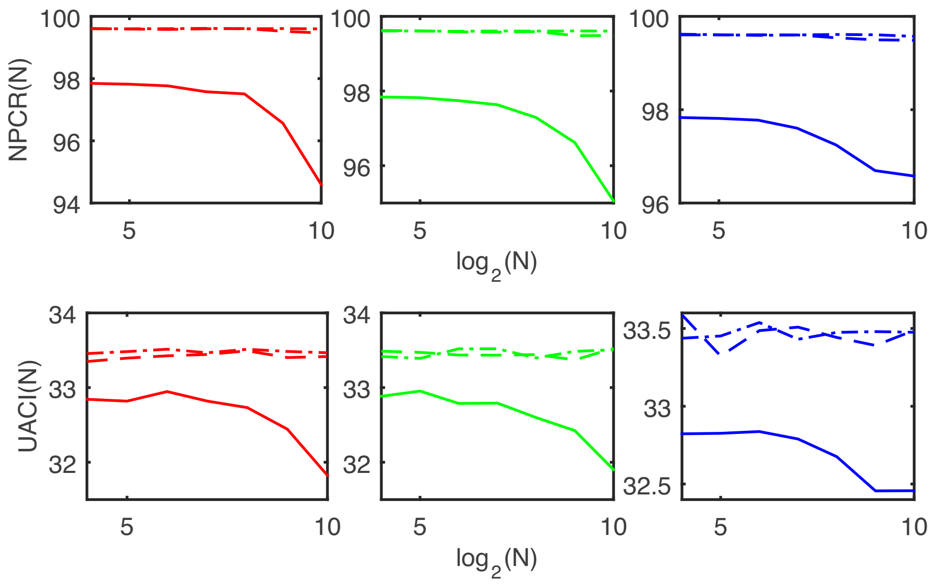

Fig. 5 (Color online) NPCR and UACI of one pixel change ciphered image of Lena in terms of N. In the first line, from left to right, are shown the NPCR of the red, green and blue components, respectively, for $R = 1$ (solid line), $R = 2$ (dashed line) and $R = 3$ (dash-dotted line). The corresponding UACI values are shown in the second line

starting from the second round, the influence of the pixel change propagates in the other sub-images, depending on N, which contributes to enhance the $NPCR$ and $UACI$. The values obtained for $R = 3$ and $N = 2^4$ are respectively $NPCR_{Red} = 99.6010$, $NPCR_{Green} = 99.6086$ and $NPCR_{Blue} = 99.6174$; and $UACI_{Red} = 33.4559$, $UACI_{Green} = 33.4164$ and $UACI_{Blue} = 33.4373$. This ability of the cipher to encrypt small block sizes with high security level is advantageous when implementing it with low-end processors under limited memory space constraints.

We also evaluated the impact of the step size δ (used to perform a right circular shift) on the security level. Figure 6 shows the behavior of the NPCR and UACI of the red component of the image of Lena in terms of δ. We set $N = 128$ and $R = 3$ for this experiment. It appears that the NPCR approaches its reference value as $\delta \to N$.

4.3 Key space analysis

4.3.1 The key space

We designed the cipher with a 256-bit key corresponding to 32 ASCII symbols, as such a key length is known to be sufficiently large for resisting all presently known kinds of brute-force attacks. The key space is the number of effective combinations of 32 symbols that can be built from the set of ASCII symbols, i.e 2^{256} while using the whole set of ASCII symbols. This key space can easily be extended to 2^{384} (48 ASCII symbols) by also considering the forcing terms $\mathbf{u}_x$ and $\mathbf{u}_y$ as parameters of the 16D QACM. The key space can also be extended by increasing the dimension of the PRNG.

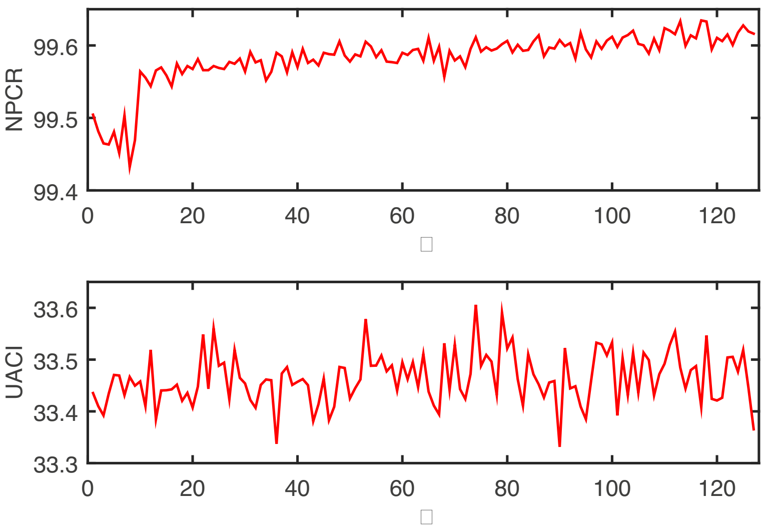

Fig. 6 (Color online) Impact of δ on the sensitivity of the cipher with respect to the plain-image. From top to bottom are shown, respectively, the behaviors the NPCR and UACI of one pixel change ciphered red component of the image Lena, with $R = 3$ and $N = 128$

4.3.2 Sensitivity of the key

A high key sensitivity allows to prevent adaptive chosen-plaintext attacks and linear cryptanalysis. In order to evaluate the sensitivity of our cipher to the external key, we considered two slightly different keys $S1$ = *azertyuiopqsdfghazertyuiopqsdfg0* and $S2$ = *azertyuiopqsdfghazertyuiopqsdfg1* to encrypt the same image. Table 2 summarizes the sensitivity of the key of the proposed cipher, for various test images. Values of $NPCR$ and $UACI$ confirm the high sensitivity of the proposed scheme to one bit change in the external key.

In Fig. 7 is presented an example of ciphering/deciphering. The ciphered image is successfully deciphered when using the same key for both the encryption and decryption processes, whilst the decryption fails for a different key.

4.4 Speed performance analysis

The running speed of the algorithm is evaluated using Matlab 14b. The algorithm was not optimized and its performances were measured on a computer with Windows 10 operating system, Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz, and 8GB RAM. The average running time, for $R = 3$ and $N = 1024$, is about 146 ms for 512×512 gray-scale images. The corresponding average security parameters evaluated with 512×512 gray-scale images of Lena, Baboon, Airplane, and Peppers are, respectively, $NPCR = 99.6023$, $UACI = 33.4690$ $\rho_h = -0.0003$, $\rho_v = -0.0009$ and $\rho_d = -0.0007$, thus attesting that the algorithm is secure for the chosen parameter setting. ρ_h, ρ_v and ρ_d are, respectively, the horizontal, vertical and diagonal correlation coefficients. Table 3 shows the average running time and security parameters for $3 \leq R \leq 8$ and $N = 1024$.

Table 2 Detailed statistical properties of images encrypted with two slightly different keys

Image	Color	NPCR	UACI	Entropy	Correlation
Lena	Red	99.5895	33.5155	7.9993	– 0.0032
	Green	99.6208	33.4007	7.9992	0.0054
	Blue	99.5983	33.3896	7.9993	0.0024
Baboon	Red	99.6136	33.4619	7.9993	– 0.0003
	Green	99.6090	33.4378	7.9993	0.0005
	Blue	99.6120	33.5040	7.9993	– 0.0007
Airplane	Red	99.6223	33.5334	7.9993	– 0.0015
	Green	99.5975	33.4402	7.9993	– 0.0001
	Blue	99.5674	33.4904	7.9994	– 0.0011
Peppers	Red	99.6029	33.4945	7.9993	– 0.0012
	Green	99.6094	33.4660	7.9993	0.0003
	Blue	99.6258	33.5040	7.9993	– 0.0010

$NPCR(\%)$, $UACI(\%)$ and entropy are to be compared with reference values 99.6093%, 33.4621% and 8, respectively

4.5 Comparison with existing ciphers

Table 4 shows comparison results with existing algorithms. We used the color image of Lena for the comparison of the average $NPCR$, $UACI$ and correlation coefficients. It appears

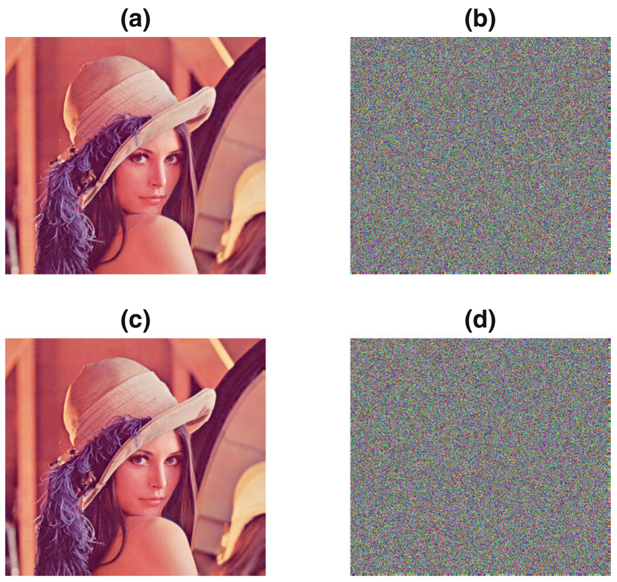

Fig. 7 (Color online) Sensitivity of the key to one-bit change: (a) Original image, (b) ciphered image with $S1 = azertyuiopqsdfgjazertyuiopqsdfg0$, (c) successfully deciphered image with $S1$ and (d) unsuccessfully deciphered image with $S2 = azertyuiopqsdfgjazertyuiopqsdfg1$, $N = 32$, $R = 2$

Table 3 Average running speed and corresponding security parameters for $3 \leq R \leq 8$, $N = 1024$ and $S1$ as encryption key

R	3	4	5	6	7	8
$t(s)$	0.1461	0.1909	0.2394	0.2833	0.3310	0.3774
$NPCR$	99.6023	99.6132	99.6018	99.6133	99.6136	99.6020
$UACI$	33.4690	33.4267	33.4488	33.4571	33.4567	33.4555
ρ_h	− 0.0003	− 0.0005	− 0.0018	− 0.0005	− 0.0008	0.0009
ρ_v	− 0.0009	− 0.0005	0.0003	0.0009	0.0007	− 0.0007
ρ_d	− 0.0007	− 0.0001	0.0009	− 0.0020	0.0009	0.0013

The $NPCR(\%)$ and $UACI(\%)$ are to be compared with their corresponding reference values 99.6093% and 33.4621%, respectively

that the proposed cipher and Ref. [16] are those presenting both a large $NPCR$ and a $UACI$ close to the ideal value. In the proposed cipher, all the permutation and diffusion keys used are N-length sequences, thus easy to implement, which is not the case for the cipher in Ref. [16]. The circular shift of the image is also performed by sequentially shifting N-length blocks, which allows all the shifting, permutation and diffusion operations to be combined in a single loop. This architecture also allows to reduce the memory space that is necessary for encrypting the entire image.

Table 5 compares the running speed of the proposed algorithm with other chaos based ciphers. We used the gray-scale images of cameraman (256×256) and Lena (512×512) for this experiment. It appears that the running speed of the proposed algorithm can allow real-time data encryption.

From the overall comparison, it appears that the proposed cipher is faster compared to those in Ref. [1] and Ref. [40]. The one in Ref. [16] is 2.5 times faster than the proposed one, but it requires floating point arithmetics, which is more constraining than using 8-bit integer arithmetics. The proposed algorithm offers the advantage to combine only 8-bit integer operations, which is much better for its implementation with low-end microprocessors, without loss of security properties.

Table 4 Comparison of the proposed cipher with existing chaos based image ciphers

Tests	Proposed cipher*	Ref. [33]	Ref. [1]	Ref. [16]*	Ref. [40]
$NPCR$	99.6081	99.6200	99.6000	99.6126	99.5900
$UACI$	33.4864	33.5300	33.5000	33.4483	33.4700
ρ	− 0.0002	0.0040	0.0030	0.0002	0.0040
H	7.9998	7.9994	7.9994	7.9998	7.9993

The * symbol indicates references compared in the same environment. The average security parameters of the proposed cipher are obtained with $R = 3$, $N = 1024$ and $S1$ as encryption key

Table 5 Comparison of encryption time of different algorithms

File name	Image size	Proposed cipher*	Ref. [16]*	Ref. [1]	Ref. [40]
cameraman.tif	256×256	0.0570	0.0202	0.1789	0.1950
Lena.png	512×512	0.1461	0.0708	0.6639	0.6500

The * symbol is used for algorithms compared in the same environment. The average running speeds of the proposed cipher are obtained with $R = 3$, $N = 1024$ and $S1$ as encryption key

5 Conclusion

We presented in this paper an 8-bit precision cipher involving exclusively integer arithmetics and that can be implemented with low-end microprocessors. Our cipher includes a PRNG that was obtained by coupling an 8D time varying with an 8D amplitude varying QACM to achieve a 16D system. Eight dimensional QACM themselves were obtained by considering a linear coupling between 2D QACM with shock occurrences to model interactions. Such a coupling method allowed us to considerably increase both the period and the complexity of the resulting system, thus achieving a minimal period $T_{\mathbf{x}} > 10^{27}$, which is sufficiently large to predict the behavior of the QPRNG. Although it is a 16D system, the proposed PRNG runs fast as it combines only 8-bit integer operations. Its randomness was evaluated using the NIST suite tests. We particularly set the precision of the PRNG to 8 bits for the generated sequences of integers to be directly used for image encryption, without need of data conversion. We therefore evaluated the performance of the proposed cipher under 8-bit precision condition and verified that it runs fast and presents a high security level as compared to existing 32-bit precision chaos based ciphers. However, we need to consider a 16D QACM to achieve periods greater than 10^{27}. Our intent in prospect is to reduce the dimensionality of the system while increasing its period. Such a reduction of the dimensionality will allow to gain computation time while reducing the hardware requirements. We also intend to develop a new confusion approach that does not imply the data sorting and which is much faster than the sorting process.

Acknowledgements This work was supported by the Alexander von Humboldt Foundation under Ref 3.4-CMR/1133622.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Ahmed A, El-Latif A, Li L, Niu X (2014) A new image encryption scheme based on cyclic elliptic curve and chaotic system. Multimed Tools Appl 70:1559–1584

2. Bao J, Yang Q (2012) Period of the discrete arnold cat map and general cat map. Nonlinear Dyn 70:1365–1375
3. Bao J, Yang Q (2012) Period of the discrete Arnold cat map and general cat map. Nonlinear Dyn 70:1365–1375
4. Berry MV (1987) Quantum chaology. Proc R Soc A 413:183–198
5. Binder PM, Jensen RV (1986) Simulating chaotic behavior with finite-state machines. Phys Rev A 34:4460–4463
6. Bisht A, Dua M, Dua S, Jaroli P (2020) A color image encryption technique based on bit-level permutation and alternate logistic maps. J Intell Syst 29:1246–1260
7. Cai GQ, Zheng XD (2000) Performance analysis of the chaotic spread spectrum sequences with finite precision. J Inform Eng Univ 1:19–22
8. Chen G, Mao Y, Chui C (2004) A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos Solitons Fractals 21:749–761
9. Chen F, Wong KW, Liao X, Xiang T (2012) Period distribution of generalized discrete Arnold cat map for $n = p^e$. IEEE Trans Inf Theory 58:445–452
10. Chen F, Wong KW, Liao X, Xiang T (2013) Period distribution of generalized discrete Arnold cat map for $n = 2^e$. IEEE Trans Inf Theory 59:3249–3255
11. Chen F, Wong KW, Liao X, Xiang T (2014) Period distribution of generalized discrete Arnold cat map. Theor Comput Sci 552:13–25
12. Deng Y, Hu H, Xiong N, Xiong W, Liu L (2015) A general hybrid model for chaos robust synchronization and degradation reduction. Inf Sci 305:146–164
13. Dyson FF, Falk H (1992) Period of a discrete cat mapping. Am Math Mon 99:603–614
14. Fan C, Ding Q, 2019 Effects of limited computational precision on the discrete chaotic sequences and the design of related solutions. Complexity p. https://doi.org/10.1155/2019/3510985
15. Fouda JSAE, Effa JY, Ali M (2014) Highly secured chaotic block cipher for fast image encryption. Appl Soft Comput 25:435–444
16. Fouda JSAE, Effa JY, Sabat SL, Ali M (2014) A fast chaotic block cipher for image encryption. Commun Nonlinear Sci Numer Simul 19:578–588
17. Ganesan K, Murali K (2014) Image encryption using eight dimensional chaotic cat map. Eur Phys J Spec Top 223:1611–1622
18. Gu G, Linga J (2014) A fast image encryption method by using chaotic 3d cat maps. Optik 125:4700–4705
19. Hu H, Deng Y, Liu L (2014) Counteracting the dynamical degradation of digital chaos via hybrid control. Commun Nonlinear Sci Numer Simul 19:1970–1984
20. Hu H, Xu Y, Zhu Z (2008) A method of improving the properties of digital chaotic system. Chaos, Solitons & Fractals 38:439–446
21. Kang S, Liang Y, Wang Y, VI M (2019) Color image encryption method based on 2d-variational mode decomposition. Multimed Tools Appl 78:17,719–17,738
22. Kassem A, Hassan HAH, Harkouss Y, Assaf R (2014) Efficient neural chaotic generator for image encryption. Digit Signal Process 25:266–274
23. Keating JP, Mezzadri F (2000) Pseudo-symmetries of Anosov map and spectral statistics. Nonlinearity 13:747–775
24. Kocarev L, Sterjev M, Fekete A, Vattay G (2004) Public-key encryption with chaos. Chaos: Interdisciplinary J Nonlinear Sci 14:1078–1082
25. Kumar A, Ghose M (2011) Extended substitution and diffusion based image cipher using chaotic standard map. Commun Nonlinear Sci Numer Simul 16:372–382
26. Li C, Tan K, Feng B, Lu J (2017) The graph structure of the generalized discrete Arnold's cat map. https://arxiv.org/pdf/1712.07905.pdf, pp 1–15
27. Li S, Chen G, Mou X (2005) On the dynamical degradation of digital piecewise linear chaotic maps. Int J Bifurcat Chaos 15:3119–3151
28. Liu L, Miao S (2017) Delay-introducing method to improve the dynamical degradation of a digital chaotic map. Inf Sci 396:1–13
29. Lou D, Sung C (2004) A steganographic scheme for secure communications based on the chaos and euler theorem. IEEE Trans Multimedia 6:501–509
30. Matthew R (1989) On the derivation of a chaotic encryption algorithm. Cryptologia 13:29–42
31. Musanna F, Kumar S (2019) A novel fractional order chaos-based image encryption using fisher yates algorithm and 3-d cat map. Multimed Tools Appl 78:14,867–14,895
32. Nagaraj N, Shastry MC, Vaidya PG (2008) Increasing average period lengths by switching of robust chaos maps in finite precision. Eur Phys J Spec Top 165:73–83

33. Nkandeu YPK, Tiedeu A (2019) An image encryption algorithm based on substitution technique and chaos mixing. Multimed Tools Appl 78:10,013–10,034
34. Persohn K, Povinelli R (2012) Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floatingpoint representation. Chaos Solitons & Fractals 45:238–245
35. Press W, Teukolsky S, Vetterling W (1993) Numerical recipes in C : the art of scientific computing, 2nd edn. Cambridge University Press, Cambridge
36. Sangeetha Y, Meenakshi S, Sundaram CS (2014) A simple, sensitive and secure image encryption algorithm based on hyper-chaotic system with only one round diffusion process. Multimed Tools Appl 71:1469–1497
37. Sayed WS, Radwan AG, Rezk AA, Fahmy HAH (2017) Finite precision logistic map between computational efficiency and accuracy with encryption applications. Complexity p. https://doi.org/10.1155/2017/8692046
38. Sze K (2007) High-dimensional chaotic map: formulation, nature and applications. Phd dissertation, City University of Hong Kong
39. Zareai D, Balafar M, Feizi Derakhshi M (2017) Complexity and properties of a multidimensional cat-hadamard map for pseudo random number generation. Eur Phys J Special Topics 226:2263–2280
40. Zhu ZL, Zhang W, Wong KW, Yu H (2011) A chaos-based symmetric image encryption scheme using a bitlevel permutation. Inf Sci 181:1171–1186
41. Zhua H, Zhao C, Zhanga X, Yanga L (2014) An image encryption scheme using generalized arnold map and affine cipher. Optik 125:6672–6677