An encryption algorithm must be invertible. Otherwise, it becomes impossible to uniquely recover the concealed messages. Moreover, the algorithm needs to operate correctly for all defined inputs and in machines working with finite precision arithmetic. In this Comment, we demonstrate that the algorithm is not invertible. We suggest simple modifications that can remedy some of the problems we identified. © 2008 American Institute of Physics. [DOI: 10.1063/1.2966114]

I. INTRODUCTION

In the image encryption system proposed in Ref. 1, the chaotic logistic map

\[ x(k + 1) = f(x(k)) = ax(k)(1 - x(k)) \]  

(1)

is used in an algorithm to encrypt an image.

Let \( T \) denote the set \( \{0, 1, \ldots, 255\} \) and let the vector \( c \in \mathbb{T}^m \) represent an image as a vector. For an \( N \times M \) image, \( m \) is the total number of pixels, i.e., \( m = NM \). The encryption algorithm takes the vector \( c \) as the plaintext input, and it generates another vector \( d \in \mathbb{T}^m \) as the ciphertext output. The algorithm transforms plaintext vector \( c \) in three steps: D/A conversion, chained chaotic iteration for a number of cycles, and finally A/D conversion.

In the D/A conversion step, each integer pixel value \( c_i \) is mapped to one of 256 distinct real values in the chaotic attractor \((x_{\text{min}}, x_{\text{max}})\) using

\[ x_i = x_{\text{min}} + \frac{(x_{\text{max}} - x_{\text{min}}) c_i}{255}, \quad 1 \leq i \leq m, \]  

(2)

where \( x_{\text{min}} = (4a^2 - a^3)/16 \) and \( x_{\text{max}} = a/4 \).

In the chained chaotic iteration step, the real values \( x_i \) are transformed using repeated chaotic iteration as follows. We first initialize cycle 0 values as \( y_i(0) = x_i, 1 \leq i \leq m \). The transformation for the \( j^{th} \) cycle, \( j \geq 1 \), is given as

\[ y_i(j) = A(f^n(y_m(j - 1)) + y_1(j - 1)), \quad i \geq 2, \quad 1 \leq j \leq r, \]  

(3)

where the function \( A : \mathbb{R} \rightarrow \mathbb{R} \) is defined as

\[ A(u) = \begin{cases} u, & u \leq x_{\text{max}}, \\ u - (x_{\text{max}} - x_{\text{min}}), & u > x_{\text{max}}. \end{cases} \]  

(4)

and \( r \) denotes the number of cycles in the encryption. In Eq. (3), the logistic map \( f \) is iterated \( n \) times starting with the initial value \( y_{i-1}(j) \) for \( i \geq 2 \) and with \( y_m(j - 1) \) for \( i = 1 \).

In the proposal, Eq. (3) is expressed slightly differently without the function \( A \). Here, we introduced the function \( A \) to emphasize that Eq. (3) tries to make sure that the transformed value \( y_i(j) \) remains within the attractor. However, as we show in the sequel, this choice is incorrect.

In the final A/D conversion step, \( y_i(r) \) is mapped back to an integer \( d_i \) in \( T \) using

\[ d_i = \text{round} \left( \frac{y_i(r) - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \right) \cdot \frac{255}{y_m(j - 1)} \]  

(5)

In decryption, the algorithm has the vector \( d \) as its input and generates the original plaintext vector \( c \). The decryption also has three steps: D/A conversion, chained chaotic iteration in the reverse direction, and A/D conversion.

In the D/A step, we use

\[ y_i(r) = x_{\text{min}} + \frac{(x_{\text{max}} - x_{\text{min}}) d_i}{255}, \quad 1 \leq i \leq m \]  

(6)

to map the integer values \( d_i \) to the fixed locations \( y_i(r) \) in the attractor.

In the chained chaotic iteration step, we work backwards through \( j \) cycles to get the original real values using

\[ y_i(j - 1) = B(y_i(j) - f^n(y_m(j - 1))), \quad i \geq 2, \quad 0 \leq j \leq r \]  

(7)

where the function \( B : \mathbb{R} \rightarrow \mathbb{R} \) is defined as

\[ B(u) = \begin{cases} u, & u \geq 0, \\ u + (x_{\text{max}} - x_{\text{min}}), & u < 0. \end{cases} \]  

In the proposal, Eq. (7) is expressed slightly differently without the function \( B \). Here, we introduced the function \( B \) to emphasize that Eq. (7) tries to make sure that the trans-
formed value $y_i(j-1)$ remains within the attractor.

In the A/D step, we map the real values $x_i = y_i(0)$ to corresponding integer values in $T$ using

$$c_i = \text{round} \left( \frac{y_i(0) - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \right) \cdot 255, \quad 1 \leq i \leq m. \quad (8)$$

II. ANALYSIS

The first problem with the encryption transformation is that the functions $A(\cdot)$ and $B(\cdot)$ may yield values outside the attractor ($x_{\text{min}}, x_{\text{max}}$). In such cases, the iteration of function $f$ starts with an initial condition outside the attractor and consequently A/D conversion step (5) may yield pixel values not in $T$. Indeed, using Eq. (3) with $j = 1$ and $i \geq 2$, we have

$$y_i(1) = A(f^n(y_{i-1}(1)) + y_{i-1}(0)), \quad (9)$$

$$y_{i+1}(1) = A(f^n(y_i(1)) + y_{i+1}(0)). \quad (10)$$

Note that, in Eq. (10), the initial value for the iteration of $f$ is $y_i(1)$. Also, $y_i(1)$ is obtained as the value of the function $A$ in Eq. (9). Thus, in order for the initial value $y_1(1)$ to be inside the attractor, the function $A$ must yield values inside the attractor. We demonstrate with a simple example that this is not the case in the scheme proposed in Ref. 1.

Let us choose $n=1$, $a=3.9$, $j=1$, $y_{i-1}(1)=0.5$, and $y_i(0) = x_{\text{max}} - \varepsilon$, with $i \geq 2$. Using Eqs. (9) and (1), we have $y_i(1) = A(f(0.5 + x_{\text{max}} - \varepsilon)) = A(2x_{\text{max}} - \varepsilon)$. If $\varepsilon$ is small enough, we have $2x_{\text{max}} - \varepsilon < x_{\text{max}}$, and using Eq. (4) with $u = 2x_{\text{max}} - \varepsilon$, we find $y_i(1) = x_{\text{max}} + x_{\text{min}} - \varepsilon$. Clearly, for small $\varepsilon$, $y_i(1)$ is outside the attractor ($x_{\text{min}}, x_{\text{max}}$). Therefore, using the functions $A(\cdot)$ and $B(\cdot)$ as proposed in Ref. 1 leads to incorrect operation.

In order to guarantee that the all the values fall within the attractor, we suggest the following modified functions for $A(\cdot)$ and $B(\cdot)$.

$$A(u) = \begin{cases} u, & u \leq x_{\text{max}}, \\ u - (x_{\text{max}} - x_{\text{min}}), & x_{\text{min}} < u \leq 2x_{\text{max}} - x_{\text{min}}, \\ u - 2(x_{\text{max}} - x_{\text{min}}), & 2x_{\text{max}} - x_{\text{min}} < u. \end{cases}$$

$$B(u) = \begin{cases} u, & u \geq x_{\text{min}}, \\ u + (x_{\text{max}} - x_{\text{min}}), & x_{\text{max}} < u \leq 2x_{\text{max}} - x_{\text{min}}, \\ u + 2(x_{\text{max}} - x_{\text{min}}), & u < x_{\text{max}} + 2x_{\text{min}}. \end{cases}$$

The second problem with the proposal is that the encryption transformation that is defined in Eqs. (2), (3), and (5) is not invertible because it involves many-to-one rounding function.

In the D/A step of the decryption, $y_i(r)$ values calculated by Eq. (6) is one of the 256 fixed points on the attractor. However, the real value $y_i(r)$ calculated using Eq. (3) is not necessarily one of those fixed 256 real values. Indeed, Eq. (5) maps $y_i(r)$ to the integer value $d_i$ corresponding to the closest of the fixed points by way of rounding. Thus, when decrypting, the initial value to the chaotic map $f$ in Eq. (7) is one of the points on the fixed grid. Therefore, $y_i(0)$ calculated in Eq. (7) is different from the original $x_i$ used in Eq. (3). If this difference is large enough, then the decrypted color value is different from the original color value $c_i$. Hence, the encryption as described in Ref. 1 is not an invertible function.

To illustrate the preceding argument, we pick a vectorized image with only two pixels with $c_1=0$ and $c_2=25$. Let the encryption keys be given as $a=3.9$, $n=25$, $r=1$. In this case, using Eqs. (2) and (3), we find $x_1 = 0.095 062 5$, $x_2 = 0.181 330 882 352 941$, and $y_1(1) = 0.663 955 819 836 359$, $y_2(1) = 0.875 143 546 668 635$. By Eq. (5), these values correspond to the encrypted color values $d_1=165$, $d_2=226$. In decrypting these values, we use Eq. (6) to obtain the fixed points on the grid as $y_1(1) = 0.664 433 823 529 412$, $y_2(1) = 0.874 928 676 470 588$. Using these values in Eq. (7), we obtain

$$x_1 = y_1(0) = 0.244 811 426 834 675$$

and

$$x_2 = y_2(0) = 0.769 496 460 931 825.$$
\[ y_2(2) = 0.96099263265698176006. \]

Now we start off with \( y(2) \) and work backwards to decrypt. We assume that the receiver has the exact values \( y_1(2) \) and \( y_2(2) \). Using these values in Eq. (7), we obtain

\[ y_1(1) = 0.47374838065180641111 \]

and

\[ y_2(1) = 0.60677852279589572504. \]

Note that \( y(1) \) so obtained is slightly different from the one obtained in encryption. This is due to the noninvertible nature of finite precision addition. When \( y(1) \) is subsequently used in the chaotic iteration Eq. (7) once more, the difference is amplified. Thus we obtain

\[ x_1 = y_1(0) = 0.36869120468326127549 \]

and

\[ x_2 = y_2(0) = 0.50596375159066242500. \]

These real values correspond to the color values \( c_1 = 79 \) and \( c_2 = 119 \). Clearly, the encryption algorithm is not invertible.

### III. CONCLUSIONS

In this Comment, we point out three flaws in a previous proposal for chaotic encryption. We show that the encryption function is not well-defined for some values. We also show that the rounding operation and the finite precision arithmetic render the algorithm noninvertible. We suggest remedies for two of the problems we identified.

### ACKNOWLEDGEMENTS

This work was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project No. 106E143.

\[ ^1 \text{A. N. Pisarchik, N. J. Flores-Carmona, and M. Carpio-Valadez, Chaos 16, 033118 (2006).} \]