Abstract

In this paper, we will present an encryption algorithm using the decimal expansion of irrationals where we will use the famous $\pi$ number in our examples and applications. Many papers were published in the recent years about encryption algorithms using chaotic maps that help generating 'random' sequences thanks to their properties (high sensitivity to initial conditions, ergodicity and implementation simplicity).

Unfortunately, there are some issues that can destroy the benefits of using chaotic-flow-based encryption algorithms. In fact, when simulating these maps, we are just getting some good estimations using numerical algorithms (Runge-Kutta for example), and even if the map is discrete (as the logistic map), the fact that our machines are having a finite float precision will inevitably lead to asymptotic periodic generated sequences.

We begin by an introduction of the number $\pi$ where we expose the originality, the history and also the strengths of using it against the chaotic maps, then we give a complete description of an algorithm for encrypting a text and another variant of the algorithm for encrypting an image with a set of security analysis tests that show its efficiency and its high security level.
1 Introduction

Network communications using internet are mandatory for almost every person and every organization in the world whatever their activities and their motivations are. In many cases, the exchanged pieces of information need to be secured (banks transactions, military communications ...) so the security of these pieces became a crucial issue for all the network communication tools.

Cryptography [17] has became a very important discipline in the recent decades as it offers the solutions for the communications security issues. Almost used cryptosystems today are using public-key encryption algorithms such as RSA, EL-Gamal or even the elliptic curves. The robustness of these algorithms lies in the fact that it’s not yet feasible with today’s machines to factorize a large number or to solve the discrete logarithm problem. However, this may not be the case in the coming few years, taking into account the extraordinary progress of the computers technology and, especially, the possibility of building the long-awaited quantum computer that will certainly put an end for the use of these algorithms.

Researchers have worked for the last two decades to find an alternative and they began to propose cryptosystems based on the chaotic dynamical systems [1-4] thanks to their interesting properties: high sensitivity to initial conditions, ergodicity and implementation simplicity. So, using the chaotic dynamical systems, we could design various hash functions, cryptosystems and authentication protocols [16,18,19].

Advantage of using irrationals decimal expansion against chaotic dynamical systems:
As said in the abstract, the problem with the algorithms using chaotic maps is that when trying numerical simulations we just approximate it using any numerical method (Runge-Kutta for example in the case of continuous maps) and this may lead to loose the properties of the original system and even if the system is discrete (the logistic map for example using the method of Baptista [20]), the fact that we are using a machine with a finite float precision will destroy the aperiodicity property of the chaotic map.

Instead, we are then proposing an encryption algorithm based on irrationals decimal expansions where we use as a private key an irrational number \( I \) and a sequence of bits \( K \) then using this key we encrypt a plaint data to get an encrypted data (the data can be a text or an image as we will present two variants of the algorithm). \( \pi \) for example is a welcome irrational number which the decimals represent a random and aperiodic sequence that can be used to build an efficient encryption scheme, avoiding the issues encountered when using classical chaotic maps, as will be shown in
the next section. Applications will be performed for texts and images, as they are among the most exchanged types of data in the modern network communications. So, text and image encryption algorithms are successively presented with corresponding security analysis in sections 3-6 and finally, section 7 concludes this paper.

2 the $\pi$ number

2.1 history

We begin this section by a short definition of $\pi$ taken from Wikipedia the free encyclopedia:

"$\pi$ [5] (sometimes written pi) is a mathematical constant whose value is the ratio of any circle’s circumference to its diameter, this is the same value as the ratio of a circle’s area to the square of its radius. $\pi$ is approximately equal to 3.14159 in the usual decimal positional notation. Many formulae from mathematics, science, and engineering involve $\pi$, which makes it one of the most important mathematical constants.

Many authors [6,7] have written many books about $\pi$ and many researchers have joined a historical race of computing the maximum number of $\pi$ decimals."

The oldest known approximations about $\pi$ date from around 1900 BC, they were found in the Egyptian Rhind Papyrus $\frac{256}{81} \approx 3.160$ and on Babylonian tablets $\frac{25}{8} = 3, \frac{125}{7}$. Archimedes was the first to estimate $\pi$ rigorously, he proved that $\frac{3\frac{10}{71}}{\pi} < 3 < \frac{3\frac{10}{70}}{\pi}$ and the average of these values is about 3, 14185.

Until the second millennium AD, estimations of $\pi$ were accurate to fewer than 10 decimal digits. The next major advances in the study of $\pi$ came with the development of infinite series which permit the estimation of $\pi$ to any desired accuracy by considering sufficiently many terms of a relevant series. Around 1400, Madhava of Sangamagrama found the first known such series:

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k + 1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Madhava was able to estimate $\pi$ as 3, 14159265359, which is correct to 11 decimal places. The record was beaten in 1424 by the Persian mathematician, AlKashi who gave an estimate that is correct to 16 decimal digits.

These records have continued to be broken thanks to the discovery of new formulas and approximations during all the next centuries that came after. John Walis, Isaac Newton, John Machin, Leonhard Euler, Riemann and many other great scientists have contributed in breaking the records by proposing many formulas to approximate $\pi$.

In the last century and in our current century, the computers have helped the scien-
tists to calculate a huge number of decimals of $\pi$, the last records are about some trillions of decimals.

2.2 sequences of $\pi$ decimals randomness

$\pi$ is an irrational number and this means that if we take any sequence (of any length) of its digits it will be aperiodic. This sequences have also a random behavior. To show this randomness behavior, we have performed some statistical tests where we have taken the sequences of 1, 5 and 10 digits of $\pi$ decimals and we constructed a sequence in the interval $[0,1]$ by dividing each element by $10^{\text{number of digits}}$. The table below presents the results (for every sequence, we have taken the first 100,000 elements):

<table>
<thead>
<tr>
<th>Number of digits</th>
<th>mean</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4493</td>
<td>0.0824</td>
</tr>
<tr>
<td>5</td>
<td>0.5006</td>
<td>0.0834</td>
</tr>
<tr>
<td>10</td>
<td>0.4985</td>
<td>0.0838</td>
</tr>
</tbody>
</table>

Table 1: mean and variance of the different sequences.

The same results for a uniform random variable in the interval $[0,1]$ are: mean $= \frac{1}{2} = 0.5$ and variance $= \frac{1}{12} \approx 0.0833$. We can see clearly that the sequences generated from the decimals of $\pi$ have a random behavior and are equivalent to a uniform random variable.

2.3 $\pi$ decimals can never be produced using a chaotic map!

As we mentioned earlier in the introduction, the chaotic maps represents many advantages and can be used to build very efficient algorithms of cryptography but we can have these issues:

- For the continuous chaotic maps, many of their properties may be lost after simulating them using any method as Runge-Kutta.
- The float precision in all the machines is finite, which means that if we generate any sequence using a chaotic map, we will finish by returning the an already visited point and the map becomes periodic which destroy the efficiency of the algorithms.

It’s true that the probability of having these issues is very small, but using $\pi$ decimals expansion means that the probability will be 0!
This result is guaranteed as there’s no chaotic map \((\dot{x} = f(x) \text{ or } x_{n+1} = f(x_n))\) that can generate the decimals of any irrational number and especially \(\pi\), we should note also that once a chaotic map reach an already visited point the map becomes periodic in the opposite of irrationals where any sequence of decimals of any size can be visited many times without having a periodic behavior and that’s what makes the originality and the strength of the proposed algorithm.

3 The text encryption algorithm

The text encryption algorithm is using as a private key an irrational number and a sequence of bits (for our simulations, we will use \(\pi\)).

To get the \(\pi\) decimals, we’re using the spigot algorithm of Rabinowitz and Wagon [7] that generates dynamically a sequence of \(\pi\) decimals step by step where no data is needed to be stored.

The algorithm takes as an input a plain text \(M\) and a key composed of an irrational number and 96 bits sequence \(K\) (used to generate the positions of the decimals used) then gives as an output a cipher text \(C\).

The texts are defined using the 96 usual characters (the space character, the new line character and the characters from 33 to 126 in the ASCII table).

Before presenting the algorithm, the following notations are presented:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M)</td>
<td>plain text</td>
</tr>
<tr>
<td>(K)</td>
<td>96 bits key</td>
</tr>
<tr>
<td>(K[i])</td>
<td>12 bits subsequence of (K): the bits in the range (i \times 12) to ((i + 1) \times 12 - 1)</td>
</tr>
<tr>
<td>(C)</td>
<td>cipher text</td>
</tr>
<tr>
<td>(S)</td>
<td>the sequence generated from the decimal of (\pi), each element of (S) has 2 digits</td>
</tr>
<tr>
<td>(S_i)</td>
<td>ith element of the sequence (S)</td>
</tr>
</tbody>
</table>

The encryption algorithm description can be summarized as following:

- **Begin:**
- **Step 1:** We begin by generating a vector of 5 elements \(\text{POS}\) (vector of positions) using the key \(K\). The elements of \(\text{POS}\) are generated using the following formulas:
Step 2: We will repeat the next steps (from 2 to 5) 5 times using each time the ith element of the vector POS (1 ≤ i ≤ 5). Each time we generate the sequence S of size n where n is the number of characters of the text M:

\[ S = \text{getKey}(POS[i]). \]

The function getKey is defined as below:

\[
\text{For } j \text{ in } [1, n]: \\
S_j = S_j + POS[i].
\]

Step 3: Using the generated sequence S, we will generate the text \(C'\) as follow:

\[
C'_1 = (M_1 + S_1) \mod 96. \\
\text{For } i \text{ in } [2, n]: \\
C'_i = (C'_{i-1} + M_i + S_i) \mod 96.
\]

Step 4: We reverse the data of the text \(C'\):

\[
\text{For } i \text{ in } [1, n]: \\
C''_i = C'_{n-i+1}
\]

Step 5: Using the generated sequence \(C'\), we will generate the text \(C\) as follow:

\[
C_1 = (C'_1 + S_1) \mod 96. \\
\text{For } i \text{ in } [2, n]: \\
C_i = (C_{i-1} + C'_i + S_i) \mod 96.
\]

End

The decryption algorithm is identical to the encryption algorithm, it receives as inputs the cipher text C and the 96 bits key K (the same used for the encryption) and returns as output the plain text M.

The only difference between the two algorithm are the steps 3 and 5 which are defined as below for the decryption algorithm.

Step 3:

\[
C'_1 = (C_1 - S_1) \mod 96. \\
\text{For } i \text{ in } [2, n]:
\]
Encryption algorithm

- \( C'_i = (C_i - C_{i-1} - S_i) \mod 96 \).

- Step 5:
  - \( M_1 = (C'_1 - S_1) \mod 96 \).
  - For \( i \in [2, n] \):
    - \( M_i = (C'_i - C'_{i-1} - S_i) \mod 96 \).

The table below presents some results of the presented algorithm:

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>Cipher text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network communications using internet are mandatory for almost every person and every organization and in the world whatever their activities and their motivations are.</td>
<td>zI&lt;36fj+iE-[E9]_+0B7Y+Oa&quot;38'2:1,1=qy'4'o,?FM&quot;&quot;xp7m9Meyqj&gt;</td>
</tr>
<tr>
<td>Cryptography [17] has become a very important discipline in the recent decades as it offers the solutions for the communications security issues.</td>
<td>0/DOS&quot;zLMT]p&quot;m&quot;.(PB[W)} ]FzL.cHeYIVNo</td>
</tr>
<tr>
<td>1HaoG+7!)02fg.Qg\G[L\V\e\S\V\y\s\p\w -</td>
<td>ID&lt;,d!</td>
</tr>
</tbody>
</table>

### 4 Text encryption algorithm: Security analysis

In this section we will discuss the security analysis of our algorithm such as key space analysis, sensitivity analysis (with respect to both the key and the plain text) and finally statistical analysis as any robust encryption algorithm should resist these attacks. The computation was done (for these tests and the tests relative to the encryption image algorithm) using a PC with the following characteristics: 2.67GHz Core(TM) i5 CPU, 4.00 Go RAM and 320 Go hard-disk capacity.

#### 4.1 Key space analysis

The used key for our algorithm is an irrational number and a 96 bits key which means that we have \(2^{96}\) possibilities to generate a secret key for every irrational chosen. With such large key space, the encryption algorithm can be considered secured.

#### 4.2 Sensitivity analysis

In this section, we study the effect of changing one bit in the 96 bits key (the least significant bit and the most significant bit) and also the effect of randomly changing
one character of the plaintext.
We will not study the effect of changing the irrational number as the sensitivity of the 96 bits key guarantees a high level of sensitivity to the algorithm.

4.2.1 Key sensitivity

We will consider the key "4A7859EAB452140EEA856BC8" and the two following keys "4A7859EAB452140EEA856BC9" and "CA7859EAB452140EEA856BC8" which are the result of changing the least significant bit and the most significant bit respectively.

We encrypt the same plaintext M with these three keys and we compare the results.

<table>
<thead>
<tr>
<th>Key</th>
<th>Plaintext</th>
<th>Cipher text</th>
</tr>
</thead>
<tbody>
<tr>
<td>4A7859EAB452140EEA856BC8</td>
<td>using internet is mandatory for almost every person and every organization in the world; whatever their activities and their motivations are. Cryptography has become a very important discipline in the recent decades as it offers the solutions for the communications security issues.</td>
<td>58092a1e1d70</td>
</tr>
</tbody>
</table>

We can see clearly that the fact of changing only one bit from the key, changes completely the cipher text which guarantees that the algorithm is sensitive any small
change in the key.

### 4.2.2 Plain text sensitivity analysis

Here, and using the same key "4A7859EAB452140EEA856BC8", but we will change randomly a single character from the plain text and we will compare the resulting cipher texts.

<table>
<thead>
<tr>
<th>Plain text</th>
<th>Cipher text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network communications using internet are mandatory for almost every person and every organization in the world whatever their activities and their motivations are. Cryptography [17] has become a very important discipline in the recent decades as it offers the solutions for the communications security issues.</td>
<td>Network communications using internet are mandatory for almost every person and every organization in the world whatever their activities and their motivations are. Cryptography [17] has become a very important discipline in the recent decades as it offers the solutions for the communications security issues.</td>
</tr>
</tbody>
</table>

We can see clearly that the proposed algorithm is very sensitive to a single character change in the plain text which represents a very important criteria to insure the security of the algorithm against plain text attacks.

In the next section, we will present the image encryption algorithm variant using π decimals.

### 5 The image encryption algorithm

The proposed algorithm takes as inputs a plain-image P and a private key composed of an irrational number and a 128 bits key K then generates as output the cipher-
The main idea of the algorithm was to use the irrational number ($\pi$ in this example) as a private key and to use a sequence generated from its decimals of to encrypt the plain image. The fact that the sequence is random and aperiodic will guarantee a high efficiency of the algorithm as we will see later in the numerical results. The key $K$ is used to generate the decimal start positions among $\pi$ decimals. One of the parts to be improved for the algorithm is the calculation of the decimals of $\pi$ as we will need to use millions or even billions of them to have a sequence that can be used to encrypt images of megapixels sizes. As the algorithms that calculate these decimals takes a considerable time to perform the operations (calculation of millions or billions of decimals), we have decided to do the calculation once for all and use a text file where these decimals are stored, by doing so, the encryption and decryption operations can be done in a few milliseconds.

The algorithm is almost the same as the algorithm that we have already presented in our paper named: "A Digital Image Encryption Algorithm Based On Chaotic Logistic Maps Using A Fuzzy Controller" [19]. Before presenting the algorithm, the following notations are presented:

<table>
<thead>
<tr>
<th>$P$</th>
<th>plain-image</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>128 bits key</td>
</tr>
<tr>
<td>$K[i]$</td>
<td>16 bits subsequence of $K$: the bits in the range $i \times 16$ to $(i + 1) \times 16 - 1$</td>
</tr>
<tr>
<td>$C$</td>
<td>cipher-image</td>
</tr>
<tr>
<td>$P_i$</td>
<td>$i$th pixel of $P$</td>
</tr>
<tr>
<td>$P_i(R, G, B)$</td>
<td>Red, Green or Blue value of the pixel $i$</td>
</tr>
<tr>
<td>$S$</td>
<td>the sequence generated from the decimal of $\pi$, each element of $S$ has 10 digits</td>
</tr>
<tr>
<td>$S_i$</td>
<td>$i$th element of the sequence $S$</td>
</tr>
</tbody>
</table>

The encryption algorithm description can be summarized as following:

- **Begin:**
- Step 1: We begin by generating a vector of 5 elements POS (vector of positions) using the key $K$. The elements of POS are generated using the following formulas:
Encryption algorithm


- Step 2: We will repeat the next steps (from 2 to 5) 5 times using each time the ith element of the vector POS $(1 \leq i \leq 5)$. Each time we generate the sequence $S$ of size $n$ where $n$ is the number of pixels of $P$ using $POS[i]$:
  
  $S = \text{getKey}(POS[i])$.

The function getKey is defined as bellow:

For $j$ in $[1, n]$:

- $S_j = \frac{S_j + POS[i]}{10^{10}} \times 256 \mod 256$.

- Step 3: Using the generated sequence $S$, we will generate the image $C'$ as follow:
  
  - $C'_0(R) = (P_0(R) + S_0) \mod 256$.
  - $C'_0(G) = (P_0(G) + S_0) \mod 256$.
  - $C'_0(B) = (P_0(B) + S_0) \mod 256$.

  and:

  For $i$ in $[2, n]$:

  - $C'_i(R) = (P_i(R) + S_i + C'_{i-1}(R)) \mod 256$.
  - $C'_i(G) = (P_i(G) + S_i + C'_{i-1}(G)) \mod 256$.
  - $C'_i(B) = (P_i(B) + S_i + C'_{i-1}(B)) \mod 256$.

- Step 4: We reverse the data of the image $C'$:

  For $i$ in $[1, n]$:

  - $C'_i = C'_{n-i+1}$

- Step 5: finally we construct the cipher-image $C'$ by repeating the step 3 using the image $C'$:

  - $C_0(R) = (C'_0(R) + S_0) \mod 256$.
  - $C_0(G) = (C'_0(G) + S_0) \mod 256$.
  - $C_0(B) = (C'_0(B) + S_0) \mod 256$.

  and:

  For $i$ in $[2, n]$:

  - $C_i(R) = (C'_i(R) + S_i + C'_{i-1}(R)) \mod 256$.
  - $C_i(G) = (C'_i(G) + S_i + C'_{i-1}(G)) \mod 256$. 
\[ C_i(B) = (C'_i(B) + S_i + C_{i-1}(B))) \mod 256. \]

- **End**

The decryption algorithm is identical to the encryption algorithm, it receives as inputs the cipher-image C and the 128 bits key K (the same used for the encryption) and returns as output the plain-image P.

The only difference between the two algorithm is the step 3 and step 5 which are defined as below for the decryption algorithm.

- **Step 3:**
  
  - \[ C'_0(R) = (C_0(R) - S_0) \mod 256. \]
  - \[ C'_0(G) = (C_0(G) - S_0) \mod 256. \]
  - \[ C'_0(B) = (C_0(B) - S_0) \mod 256. \]

  and:
  
  For \( i \) in \([2, n]\):
  
  - \[ C'_i(R) = (C_i(R) - S_i - C_{i-1}(R)) \mod 256. \]
  - \[ C'_i(G) = (C_i(G) - S_i - C_{i-1}(G)) \mod 256. \]
  - \[ C'_i(B) = (C_i(B) - S_i - C_{i-1}(B)) \mod 256. \]

- **Step 5:**
  
  - \[ P_0(R) = (C'_0(R) - S_0) \mod 256. \]
  - \[ P_0(G) = (C'_0(G) - S_0) \mod 256. \]
  - \[ P_0(B) = (C'_0(B) - S_0) \mod 256. \]

  and:
  
  For \( i \) in \([2, n]\):
  
  - \[ P_i(R) = (C'_i(R) - S_i - C'_{i-1}(R)) \mod 256. \]
  - \[ P_i(G) = (C'_i(G) - S_i - C'_{i-1}(G)) \mod 256. \]
  - \[ P_i(B) = (C'_i(B) - S_i - C'_{i-1}(B)) \mod 256. \]

In the next section, we will present the security analysis tests performed on our algorithm.
6 Image encryption algorithm: Security analysis

In this section and as we did for the text encryption algorithm variant, we will discuss the security analysis of our image algorithm such as key space analysis, sensitivity analysis (with respect to both the key and the plain-image, for the key analysis we will only be interested to the bits sequence K) and finally statistical analysis as any robust encryption algorithm should resist these attacks.

We will apply the same tests as we did for the algorithm presented in our paper named: "A Digital Image Encryption Algorithm Based On Chaotic Logistic Maps Using A Fuzzy Controller" (FL-CM-EA) and we will compare the results of the two algorithms.

6.1 Key space analysis

As we have discussed while studying the text encryption algorithm, the used key for our algorithm is private key composed of an irrational number and a 128 bits key which means that we have $2^{128}$ possibilities to generate a secret key for every irrational chosen. With such large key space, the encryption algorithm can be considered secured.

6.2 Sensitivity analysis

An efficient image encryption algorithm should be highly sensitive to the secret key and to the plain-image, which means that a single bit change in the encryption key will lead to a very different cipher-image from the initial cipher-image and similarly, only a pixel change in the plain-image should lead to a very different cipher-image from the initial cipher-image.

We will present in this section the results obtained by changing one bit in the encryption key and one pixel in the plain-image and we will see the effects on the cipher-image.

Before starting our analysis, we will introduce some famous statistical measures that we will use in the next sections.

The first measure that we will talk about is the statistical correlation between two vertically adjacent pixels, two horizontally adjacent pixels and two diagonally adjacent pixels.

To compute this measure, we first take randomly a set of adjacent pixels (vertically, horizontally or diagonally) from the image (let’s say 1000 pairs) then we calculate their correlation using the formulas:

$$r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{D(x)} \sqrt{D(y)}}$$
Where:

\[
E[x] = \frac{1}{N} \sum_{i=1}^{N} x_i \\
D[x] = \frac{1}{N} \sum_{i=1}^{N} (x_i - E[x])^2 \\
cov(x, y) = E[(x - E[x])(y - E[y])]
\]

We will use also to compare two images \( C_1 \) and \( C_2 \), their correlation defined as above but the used pairs of pixels are going to be this time \((C_1(i, j), C_2(i, j))\).

Other measures are going to be used to compare two images \( C_1 \) and \( C_2 \) as the Number of Pixels Change Rate (NPCR) defined as below:

\[
NPCR = \frac{\sum_{i,j} D(i,j)}{n} \times 100\%
\]

Where \( n \) is the images size (number of pixels) and: \( D(i, j) = 0 \) if \( C_1(i, j) = C_2(i, j) \) and \( D(i, j) = 1 \) otherwise.

The Unified Average Changing Intensity (UACI) will be used as well and it is defined as:

\[
UACI = \frac{1}{n} \sum_{i,j} \frac{C_1(i,j) - C_2(i,j)}{255} \times 100\%
\]

Here \( C_1(i, j) \) and \( C_1(i, j) \) are grey-scale values of the images pixels.

### 6.2.1 Key sensitivity analysis

Key sensitivity is a required property to ensure the security of any image encryption algorithm against some brute-force attacks.

To test the key sensitivity of the proposed algorithm, we have generated randomly an encryption key: "63006C1BB123046054413B5694655141" then we encrypted an original image \( P \) using this key to obtain the image \( C_1 \).

We then slightly modified the key by changing the most significant bit to obtain: "E3006C1BB123046054413B5694655141", and using this key we’ve encrypted the same original \( P \) message the obtain image \( C_2 \).

Finally, we did the same as the last operation but changing the least significant bit to obtain the key: "63006C1BB123046054413B5694655140" and using this last key we encrypted the original image \( P \) to obtain the image \( C_3 \) (see figure Fig.1).

We have calculated the correlation, the NPCR and the UACI of each two of the three cipher-images \( C_1, C_2 \) and \( C_3 \) for the current algorithm and also for the algorithm (FL-CM-EA) (Table 2, 3 and 4).

For the obtained results we can see clearly that a negligible correlation exists among the three images even if they was produced using the same original image and with
a slightly different keys. We can see also that the rate of change NPCR, the intensity of change UACI are really high, then we can conclude that our algorithm is very sensitive to encryption key change.

![Figure 1: From the left to the right: original image P, C1, C2 and C3](image)

<table>
<thead>
<tr>
<th>Image 1</th>
<th>Image 2</th>
<th>Correlation</th>
<th>Correlation (FL-CM-EA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>0.00410</td>
<td>0.00005</td>
</tr>
<tr>
<td>C1</td>
<td>C3</td>
<td>0.00300</td>
<td>-0.00400</td>
</tr>
<tr>
<td>C2</td>
<td>C3</td>
<td>-0.00080</td>
<td>0.00044</td>
</tr>
</tbody>
</table>

Table 2: Correlation between the images C1, C2 and C3 obtained by slightly changing the encryption key (one bit change)

<table>
<thead>
<tr>
<th>Image 1</th>
<th>Image 2</th>
<th>NCPR</th>
<th>NCPR (FL-CM-EA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>99.6079%</td>
<td>99.5880%</td>
</tr>
<tr>
<td>C1</td>
<td>C3</td>
<td>99.6417%</td>
<td>99.6039%</td>
</tr>
<tr>
<td>C2</td>
<td>C3</td>
<td>99.6479%</td>
<td>99.6198%</td>
</tr>
</tbody>
</table>

Table 3: NPCR of the images C1, C2 and C3 obtained by slightly changing the encryption key (one bit change)

<table>
<thead>
<tr>
<th>Image 1</th>
<th>Image 2</th>
<th>UACI</th>
<th>UACI (FL-CM-EA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>49.8726%</td>
<td>49.9184%</td>
</tr>
<tr>
<td>C1</td>
<td>C3</td>
<td>50.0696%</td>
<td>49.7313%</td>
</tr>
<tr>
<td>C2</td>
<td>C3</td>
<td>49.8321%</td>
<td>49.8719%</td>
</tr>
</tbody>
</table>

Table 4: UACI of the images C1, C2 and C3 obtained by slightly changing the encryption key (one bit change)

### 6.2.2 Plain-image sensitivity analysis

After studying the key sensitivity of the proposed image encryption algorithm, we will study now its plain-image sensitivity.
The algorithm should be also sensitive to any small change in the plaint-image which means that changing only one pixel in the plaint-image should lead to a very different cipher-image. This property will guarantee the security of the algorithm against plaint-image brute-force attacks.

To test the sensitivity to plaint-image, we will take an original image (P1) then we will encrypted it (we call the cipher-image C1), and we will randomly change a pixel in the original image then will encrypt the image again (P2) to obtain a new cipher-image C2. We repeat this a last time again to obtain a new image (P3) and a third cipher-image C3 (we have used as encryption key: ”63006C1BB123046054413B5694655141”) (see Fig.2, Fig.3 and Fig.4)

As we did for the previous section, we will calculate the correlation, the NPCR and the UACI between each two of the three cipher-images and compare the results to the results of the algorithm (FL-CM-EA) (Tables 5, 6 and 7).

Again, the obtained results show that a negligible correlation exists between the three cipher-images and we can see also that the rate of change (NPCR) and the intensity of change UACI are really high. Form the previous results, we can conclude that our algorithm is very sensitive also to plain-image change.

![Figure 2: The image P1 (left) and the image C1 (right)](image)

![Figure 3: The image P2 (left) and the image C2 (right)](image)

### 6.3 Statistical analysis

After studying the security of the proposed algorithm against some brute-force attacks (key sensitivity and plain-image sensitivity), we will study in this section the security against statistical attacks.

To perform this study, we will consider an original image P that will be encrypted to
obtain a cipher-image C (we have used also as encryption key:”63006C1BB123046054413B5694655141”). We then compare their histograms and compute for each image the values of its two vertically adjacent pixels correlation, two horizontally adjacent pixels correlation and two diagonally adjacent pixels correlation.

6.3.1 Histogram comparisons

Fig.5 and Fig.6, presents the histograms of the images P and C while Fig.7 presents the histogram of the encrypted image C’ using the algorithm (FL-CM-EA).
We can see clearly that the histogram of the image C is almost uniform and very different from the histogram of the image P. This result confirms that statistical attacks based on the histogram analysis can’t give any clue to break the algorithm as all the statistical information of the image P are lost after the encryption.

### 6.3.2 Adjacent pixels correlation comparisons

The last statistical analysis performed is the adjacent pixels correlation. We calculate for each image (P, C and C’) the three adjacent pixels correlations: vertically, horizontally and diagonally. The table 8 shows the obtained results.
From the obtained results, we can see clearly that the pixels of the plait-image P are strongly correlated while a negligible correlation exists between those of the cipher-image C. This result shows again that the proposed algorithm can be considered as secure against statistical attacks.

7 Conclusion

In this paper we presented a text and digital image encryption algorithm using random sequences generated from an irrational number decimals (π in this paper). The tests performed on the algorithms showed their robustness and their efficiency and the comparison of the image algorithm with (FL-CM-EA) which is based on the chaotic logistic map showed also that the performance of the two algorithms are almost similar which means that the proposed algorithm can be a good candidate to replace the algorithms using chaotic maps especially that it didn’t present the issues of these algorithms already highlighted in the introduction section. Future researches should be done to design new algorithms that can extract billions of π (or any other irrational) decimals dynamically in a reasonable time as we still need to store the decimals of the used irrational in a text file. Some works can be done also to design a complete cryptographic platform with all the known features (hash functions, authentication protocols,...) using irrationals decimals expansions to support the growth of the e-commerce business and to increase the security level of the confidential transactions through the internet.

References


[16] Floriane Anstett, *Les systemes dynamiques chaotiques pour le chiffrment : synthese et cryptanalyse (These)* (Universite Henri Poincare - Nancy 1)


Received: November, 2011