Image Encryption based on Floating-Point Representation

Ali Hussein Fadel

University of Diyala- Diyala- Iraq

Received 19 April 2016        Accepted 8 June 2016

Abstract

In this paper we have presented a new design random numbers generator based on single precision floating point(RNG-SFP). Randomness of RNG-SFR is used for encryption the images. The new technique has advantage of bigger key space, smaller iteration times and high security analysis such as key space analysis. The experimented result show that the proposed technique is efficient and has high security feature

Keywords: Entropy , Histogram, Correlation Coffiencal Horizontal , Correlation Coffiencal Vertical requency test, Serial test, Poker test, runs test, chi-square
Introduction

Because the Internet has become a very big, Digital photos and videos security, it has become a necessary issue to whole Internet users. Therefore, the cryptography styles can be used to conserve the information before transmission. Transform the important information into garbage data so that no hackers can read the data called the encryption; the researchers suggested a lot of algorithms to cryptography the information such as DES, IDES and RSA. On the other hand, specific styles and specific rules need to be considered to secure the images and multimedia application. Image cryptography systems random distribution rhymester uses. Chaos is one of the most important notions that are utilized to generate a random chain because of rising suspicion of the cryptography process, which first used in the computer in 1963 by Edward Lorenz. It was used in the chaos cryptography system due to its advantages, such as sensitivity to prime stipulations and the inability predict the sequence of chaos. Many roads in the attempt to design algorithms to encrypt the image using the chaos, such as [1] uses multiple chaotic maps to encrypt images by splitting the system in the first place in two stages. In the first stage by using a Arnold Cat map pixels are permuted and then in the second stage the permuted pixels are encrypt using multi-chaotic maps. In [2] where used one-dimensional detached Chebyshev chaotic series for column and row jostle for every pixel on the main image. [3] Used Rossler chaotic system to augmentation the suspicion in the cipher images by performing changes in the pixels value and their postures. [4] To cryptography the image and increment the size of the encrypted keys in cipher the one time pads are used together with the logistic map (as a chaotic function). In [5] to cryptography the image without using any chaotic functions; it was used a knight’s tour with slips cryptography filter. However, analyzed security results, hurdles and the power of the chaotic systems [6, 7, 8]. In this sheet, we used a double precision floating point format with three different initial stipulations to establishment three different double precision floating point format series with two pixels mapping tables to increment the suspicion in the encrypted image without shuffling the original image and change the pixels value. This way reduce the implementation time of the algorithm and raise the worthiness and performance of the system.
Floating-Point Representation

A non-negative real number can be represented in decimal form with an integer fraction and denary point and it is the standard way such as in example, 33.20829, 0.000457 1128 and 70 00519.44059. We can use another standard way to represent this number by shifting the denary point and supplying appropriate powers of 10 and this method known as normalized scientific notation. So, the previous numbers have Substitute representations as

\[
12.26837827 = 0.1226837827 \times 10^2
\]

\[
0.002271828 = 0.2271828 \times 10^{-1}
\]

\[
30.0052711059 = 0.300052711059 \times 10^7
\]

The number is represented in normalized scientific notation by a fraction multiplied and the pioneer digit in the portion is not zero "except when the number involved is zero" so we write 79325 as 0.79325 \times 10^5, not as 0.079325 \times 10^6 or 7.9325 \times 10^4 or some other way.

The word length in numerous binary computers is 32 bits (binary digits) we shall characterize contrivance of this kind whose imitative numerous work stations and personal computers in widespread use. This collection is a limited subset of the real numbers. It includes \( \pm 0; \pm \infty \) the normal and sub normal single-accuracy floating-point numbers, but not the values of the number. It is noteworthy that because of the real numbers have infinite decimal or binary extensions they cannot be represented precisely as floating-point numbers for example \( \pi; e; \frac{1}{3}; 0.1 \ and \ son \ on \).

The standard single-precision floating-point representation

\[
(-1)^s \times 2^{e-127} \times (1.f)_2
\]

for sign of mantissa we use the most significant bit for this purpose where \( s=1 \) coincide with – s, \( s=0 \) coincide with + and the number c in the exponent represented by using the next eight bits.
Image Encryption based on Floating-Point Representation

Ali Hussein Fadel

![Diagram](image)

Figure 1 Partitioned floating-point single-precision computer word

of $2^{c-127}$, It is interpret as a surplus-127 code. At the end, the final 23 bits from the fractional section of the mantissa in the 1-plus form represent $(1.f)_2$: Each floating-point single-accuracy word is partitioned as in Figure 1.1.

In the example below of how we can find the single-precision machine representation of the denary number $2654.42045133441$. Converting the integer portion to binary, we have $(2654.)_{10} = (5136.)_8 = (10100101110.)_2$. Next, converting the fractional portion, we have $(.42045133441)_{10} = (.471205351201)_8 = (.100111001000101101101001000001)_2$. Now

$$(2654.42045133441)_{10} = (10100101110.100111001010000101011101001010000001)_2$$

$$= (1.0100101111101001110010000101011101001010000001)_2 \times 2^{11}$$

is the corresponding one-plus form in base 2, and $(.10100101110)_2$ is the stored mantissa. Next the exponent is $(11)_{10}$, and since $c - 127 = 11$, we immediately see that $(138)_{10} = (212)_{8} = (10001010)_2$ is the stored exponent. Thus, the single-precision representation of 2654.42045133441 is

$$[110001010100100000101011101001001010000001]_2 =$$

$$[1100010100100111010110010010000101011101001010000001]_2$$

In the table below shows the Floating-Point Representation group of random numbers
Proposed system mode

The propose image encryption algorithm consists of two stage iteration (multilevel) block permeation and nonlinear key stream cipher.

1. proposed key generation

In this section of the pseudo-random number generation and the structure is based on floating point linear congruential generator and representation, and such a demand generator natural source of randomness (non- deterministic)
Image Encryption based on Floating-Point Representation

Ali Hussein Fadel

Algorithm (1,1)

Goal: the generation of the key Depending on height and Wight image

Input: Wid-Image , Hgt-Image

Output: Key Generation

Step 1: Set parameter from key generation

- \( \text{Len of key} \leftarrow \text{Wid-Image} \times \text{Hgt-Image} \times 24 \)
- Get \( a, b, s_i \) where \( 1 \leq a, b \leq \text{len of key} - 1 \) and \( 0 \leq s_0 \leq \text{len of key} - 1 \)

(Linear Congruential Generator)

Step 2: Generation bit key

- Key bit = null
- For all \( i \) Do { where 0 To Len of key }
  - \( s_i = (as_{i-1} + b) \mod \text{Len of key} \)
  - \( x = e^{s_i} \)
  - Stram_bit= call Function Floating - Point Representation(\( x \))
  - Key bit= Key bit + Stram_bit
- Exit For

Figure 1 algorithms key generation
2. Encryption and Decryption process

Then proposed image encryption and decryption algorithm can be summarized in the following algorithm:

---

**Algorithm (1,1) Encryption / Decryption**

<table>
<thead>
<tr>
<th>Goal</th>
<th>the generation of the key Depending on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Wid-Image, Hgt-Image</td>
</tr>
<tr>
<td>Output</td>
<td>Key Generation</td>
</tr>
</tbody>
</table>

**Encryption of image**

Offset ← 1
Countk ← 0

For all X, Y Do {where Offset to Wid - Offset, Offset To Hgt- Offset}

For all I Do {Where Offset To Offset -1}

For all J Do {Where Offset To Offset -1}

ReGrBl ← Convert To Bin(GetPixel(j + fi, i + fj))

xin ← ""

For all K Do {Where 1 To 24}

If Countk = Length Bits Key THEN

Countk ← 0
End If

Countk+ ← + 1

xin += Key[Countk]

End For

xReGrBl1 ← ReGrBl1 XOR Bin2Dec(xbin)

Put ReGrBl1 (x, y)

End For

Exit For

---

**Figure 2 Encryption / Decryption image**

**Figure 3 Algorithm Encryption / Decryption**
Result and Analysis

In this paragraph will be tested on Statistics generated by the algorithm mentioned above, where the key was conducted four tests which tests, 10000 key length and the results were as shown in the table below (frequency test, serial test, poker test, runs test)

Table 2 Statistical tests

<table>
<thead>
<tr>
<th>Key size</th>
<th>Statistical tests</th>
<th>X2 (chi-square) distribution</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name Test</td>
<td>Value (x)</td>
<td>α</td>
</tr>
<tr>
<td>500</td>
<td>Frequency test</td>
<td>3.752</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Serial test</td>
<td>5.665</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Poker test</td>
<td>10.552</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Runs test</td>
<td>12.055</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Frequency test</td>
<td>3.830</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Serial test</td>
<td>5.027</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Poker test</td>
<td>38.247</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Runs test</td>
<td>9.318</td>
<td></td>
</tr>
</tbody>
</table>

Where the key that is configured of the proposed algorithm has been applied in the encrypted color image has been holding a series of measurements or tests (Entropy, Histogram, Correlation Coffiencal Horizontal, Correlation Coffiencal Vertical) As shown in the chart below
Image Encryption based on Floating-Point Representation

Ali Hussein Fadel

<table>
<thead>
<tr>
<th>Test</th>
<th>Original image</th>
<th>Encryption image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Correlation Coefficient Vertical (X,Y) (X, Y+1)</td>
<td>0.985662091165786</td>
<td>0.61052688914019</td>
</tr>
<tr>
<td>2 Correlation Coefficient Vertical (X,Y) (X+1, Y)</td>
<td>0.989217276188664</td>
<td>0.595872112765076</td>
</tr>
<tr>
<td>3 Entropy</td>
<td>7.27129320337589</td>
<td>7.99651685627914</td>
</tr>
</tbody>
</table>
Conclusions

In this research was to provide a new random number generator depends on (Floating-Point Representation) Where it was generating an initial value through a function numbers \( x = e^x \) (Floating-Point). The result was characterized by sequential access to statistical characteristics of a good where succeeded in statistical tests as in the Table FIGURE 7. After that has been adopted on a row in the encrypted image of colorful Bmp type where the results were as shown in Table FIGURE 9, It was chosen as the resulting image in the encryption key based on the proposed test methods (Correlation Coefficient Vertical, Correlation Coefficient Vertical and Entropy) The results were so good that hold up against the statistical analysis and differential analysis.

References

