Proposed A Permutation and Substitution Methods of Serpent Block Cipher

Intisar Abid Yousif
Department of Computer, College of Education, University of Mustansiriyah
int_abd@uomustansiriyah.edu.iq

Article history: Received 19 February 2019, Accepted 21 April 2019, Publish May 2019

Doi: 10.30526/32.2.2120

Abstract

Block cipher technique is one of cryptography techniques to encrypt data block by block. The Serpent is one of AES candidates. It encrypts a 128-bit block by using 32 rounds of a similar calculation utilizing permutations and substitutions. Since the permutations and substitutions of it are static. Then this paper proposes dynamic methods for permutation, substitution and key generation based on chaotic maps to get more security. The proposed methods are analyzed and the results showed that they were able to exceed the weakness resulting from the use of static permutations and substitutions boxes in the original algorithm and also can reduce number of rounds and time usage compared with a classical Serpent block cipher algorithm.

Keywords: Serpent block cipher, initial and final permutation, substitutions boxes, key generation, logistic chaos map, standard chaos map.

1. Introduction

Chaos theory is an awesome dynamic part in current cryptography. The principle highlights of utilizing chaos theory in numerous cryptosystems can be finished up on its affectability to starting conditions parameter and irregular conduct settings that accomplish the essential Shannon prerequisites of disarray and dissemination. Among them, a great deal of calculations in view of disorganized hypothesis endorsed its quality in many concerned angles with respect to security, execution speed, intricacy, control utilization and computational overhead, and so forth. These days, some turmoil based picture encryption cryptosystems and irregular number age calculations in view of discrete bedlam are proposed, however its security is for the most part not affirmed [1,2]. For some applications, the Data Encryption Standard (DES) algorithm endorsed its shortcoming against a great deal of cryptanalytic assault; therefore require a safe calculation has been issued by the US National Institute of Standards and Technology, to be known as the Advanced Encryption Standard (AES). The primary highlights of the proposed calculation that it must be both quicker and more secure than triple DES, it should likewise bolster a 128 bit block length and a 128, 192, and 256 bit key length [3]. The block cipher Serpent was outlined by Anderson, Biham and
Knudsen to be contender for AES [4]. while, Daemen and Rijmen outlined the block cipher Rijndael. Serpent and Rijndael cipher were in an opposition between the last 5 finalist algorithms. The block cipher Rijndael standardized by NIST as the AES algorithm [5]. The security of AES has been submitted to present [6]. since its acknowledgment as an AES, closed equation for AES has been found by Ferguson, Shroeppel and Whiting which can be considered as a promotion of proceeded divisions [7,8]. Courtois and Pieprzyck see that the byte substitution change which utilized in the AES calculation can be determined by various certain quadratic Boolean conditions. Recently, there are numerous viable extensions in the cryptanalysis of AES. The AES applies a wide-trail configuration arranging [4, 9]. This was at first proposed on account of its quality against ground-breaking cryptanalytic systems.

This paper presents a proposed modification of Serpent algorithm by replacing the static permutation and substitution with dynamical properties using logistic chaos map and standard map. The organization of this paper is chaos based cryptography in section 2,serpent block cipher in section 3,overview of logistic map and standard map in section 4, the proposed dynamic permutation and substitution methods in section 5,Security and Statistical analysis in section 6 and the rest sections are conclusion.

2. Chaos and cryptography

It is a modern study that many researchers towards it [10-12]. The cryptography techniques are stream cipher and block cipher. Stream cipher algorithms are developed with using chaos maps as (Pseudo Random Bit Generator) PRBG to generate keystream,[13-17]. S-box of block cipher algorithms are developed with using chaos maps [18-22].This paper study and developed permutation box (IP & FP), S-box of serpent block cipher using chaos map, and key round generation.

3. Serpent block cipher

Serpent is a symmetric block cipher and has Substitution - Permutation network (SPN) design. It has 32 rounds and each round requires its special 128 bit round key which is generated by a key schedule. The encrypting and decrypting phase have the same level of complexity. The decryption operations are exactly the inverted transformations used to encrypt the message but in opposite order Serpent uses different mathematical substitutions "S-boxes" with a 4 bit entrance and a 4 bit exit. Every encryption phase uses an S-box that work collaterally for the 32 times [23]. Figure 1, shows the encryption and decryption algorithms for Serpent block cipher.
At first, it accepts a 128 bit of plaintext and permutes these bits by fixed initial permutation (IP) which is shown in Table 1. The resulted bits input to 32 rounds, each round has three processes: - XOR with round key, substitution process by one of eight S-boxes as shown in Table 2, depend on round number (i) mod 8, linear transformation process as shown in Figure 2, with its algorithm. After last round there is XOR process with round key $k_{32}$ and do permutation of the result 128 bit by fixed final permutation (FP) as shown in Table 3, to get a ciphertext [24]. In decryption algorithm reverses an encryption block diagram processes to recover a plaintext [25].
Table 1. Initial permutation (IP).

<table>
<thead>
<tr>
<th>0</th>
<th>32</th>
<th>64</th>
<th>96</th>
<th>1</th>
<th>33</th>
<th>65</th>
<th>97</th>
<th>2</th>
<th>34</th>
<th>66</th>
<th>98</th>
<th>3</th>
<th>35</th>
<th>67</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>36</td>
<td>68</td>
<td>100</td>
<td>5</td>
<td>37</td>
<td>69</td>
<td>101</td>
<td>6</td>
<td>38</td>
<td>70</td>
<td>102</td>
<td>7</td>
<td>39</td>
<td>71</td>
<td>103</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>72</td>
<td>104</td>
<td>9</td>
<td>41</td>
<td>73</td>
<td>105</td>
<td>10</td>
<td>42</td>
<td>74</td>
<td>106</td>
<td>11</td>
<td>43</td>
<td>75</td>
<td>107</td>
</tr>
<tr>
<td>12</td>
<td>44</td>
<td>76</td>
<td>108</td>
<td>13</td>
<td>45</td>
<td>77</td>
<td>109</td>
<td>14</td>
<td>46</td>
<td>78</td>
<td>110</td>
<td>15</td>
<td>47</td>
<td>79</td>
<td>111</td>
</tr>
<tr>
<td>16</td>
<td>48</td>
<td>80</td>
<td>112</td>
<td>17</td>
<td>49</td>
<td>81</td>
<td>113</td>
<td>18</td>
<td>50</td>
<td>82</td>
<td>114</td>
<td>19</td>
<td>51</td>
<td>83</td>
<td>115</td>
</tr>
<tr>
<td>20</td>
<td>52</td>
<td>84</td>
<td>116</td>
<td>21</td>
<td>53</td>
<td>85</td>
<td>117</td>
<td>22</td>
<td>54</td>
<td>86</td>
<td>118</td>
<td>23</td>
<td>55</td>
<td>87</td>
<td>119</td>
</tr>
<tr>
<td>24</td>
<td>56</td>
<td>88</td>
<td>120</td>
<td>25</td>
<td>57</td>
<td>89</td>
<td>121</td>
<td>26</td>
<td>58</td>
<td>90</td>
<td>122</td>
<td>27</td>
<td>59</td>
<td>91</td>
<td>123</td>
</tr>
<tr>
<td>28</td>
<td>60</td>
<td>92</td>
<td>124</td>
<td>29</td>
<td>61</td>
<td>93</td>
<td>125</td>
<td>30</td>
<td>62</td>
<td>94</td>
<td>126</td>
<td>31</td>
<td>63</td>
<td>95</td>
<td>127</td>
</tr>
</tbody>
</table>

Table 2. S-box.p.

<table>
<thead>
<tr>
<th>S0:</th>
<th>3</th>
<th>8</th>
<th>15</th>
<th>1</th>
<th>10</th>
<th>6</th>
<th>5</th>
<th>11</th>
<th>14</th>
<th>13</th>
<th>4</th>
<th>2</th>
<th>7</th>
<th>0</th>
<th>9</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1:</td>
<td>15</td>
<td>12</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>14</td>
<td>8</td>
<td>6</td>
<td>13</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S2:</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>3</td>
<td>12</td>
<td>10</td>
<td>15</td>
<td>13</td>
<td>1</td>
<td>14</td>
<td>4</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>S3:</td>
<td>0</td>
<td>15</td>
<td>11</td>
<td>8</td>
<td>12</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>13</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>S4:</td>
<td>1</td>
<td>15</td>
<td>8</td>
<td>3</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>10</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>S5:</td>
<td>15</td>
<td>5</td>
<td>2</td>
<td>11</td>
<td>4</td>
<td>10</td>
<td>9</td>
<td>12</td>
<td>0</td>
<td>3</td>
<td>14</td>
<td>8</td>
<td>13</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>S6:</td>
<td>7</td>
<td>2</td>
<td>12</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>11</td>
<td>14</td>
<td>9</td>
<td>1</td>
<td>15</td>
<td>13</td>
<td>3</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>S7:</td>
<td>1</td>
<td>13</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>8</td>
<td>2</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>12</td>
<td>10</td>
<td>9</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 2. Linear transform (LT).
Table 3. Final permutation (FP).

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
<th>40</th>
<th>44</th>
<th>48</th>
<th>52</th>
<th>56</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>68</td>
<td>72</td>
<td>76</td>
<td>80</td>
<td>84</td>
<td>88</td>
<td>92</td>
<td>96</td>
<td>100</td>
<td>104</td>
<td>108</td>
<td>112</td>
<td>116</td>
<td>120</td>
<td>124</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>21</td>
<td>25</td>
<td>29</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>45</td>
<td>49</td>
<td>53</td>
<td>57</td>
<td>61</td>
</tr>
<tr>
<td>65</td>
<td>69</td>
<td>73</td>
<td>77</td>
<td>81</td>
<td>85</td>
<td>89</td>
<td>93</td>
<td>97</td>
<td>101</td>
<td>105</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>121</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>22</td>
<td>26</td>
<td>30</td>
<td>34</td>
<td>38</td>
<td>42</td>
<td>46</td>
<td>50</td>
<td>54</td>
<td>58</td>
<td>62</td>
</tr>
<tr>
<td>66</td>
<td>70</td>
<td>74</td>
<td>78</td>
<td>82</td>
<td>86</td>
<td>90</td>
<td>94</td>
<td>98</td>
<td>102</td>
<td>106</td>
<td>110</td>
<td>114</td>
<td>118</td>
<td>122</td>
<td>126</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>19</td>
<td>23</td>
<td>27</td>
<td>31</td>
<td>35</td>
<td>39</td>
<td>43</td>
<td>47</td>
<td>51</td>
<td>55</td>
<td>59</td>
<td>63</td>
</tr>
<tr>
<td>67</td>
<td>71</td>
<td>75</td>
<td>79</td>
<td>83</td>
<td>87</td>
<td>91</td>
<td>95</td>
<td>99</td>
<td>103</td>
<td>107</td>
<td>111</td>
<td>115</td>
<td>119</td>
<td>123</td>
<td>127</td>
</tr>
</tbody>
</table>

4. Overview of logistic map and standard map

- **Logistic map:** This paper used logistic chaotic map as the following equation for developed IP and FP permutation:

  \[ x_{n+1} = rx_n(1 - x_n) \]  

  Where \( r \) range is \( 3.57 < r \leq 4 \) that become chaotic. For example, using \( x_0 = 0.2 \), and \( r = 3.738 \) as **Figure 3**.

![Figure 3. Logistic map signal.](image)

- **Standard map:** This paper used Standard chaotic map as the following equation for developed S-box:

  \[ X_{n+1} = X_n + k \sin y_n \mod 2\pi \]  

  \[ y_{n+1} = y_n + X_{n+1} \mod 2\pi \]  

  Where \( X_n \) and \( y_n \) \( \in [0,2\pi] \), and \( k \geq 18.9 \).

5. The proposed dynamic permutation and substitution methods

A. The proposed Dynamic Initial Permutation (DIP):

This section shows a distribution of fixed IP serpent as **Figure 4**. This figure shows that no random in distribution and the attacker can re-permute in easy. So that this paper suggests use random chaotic map to permute IP with a special key for each encryption process.
To permute IP using logistic map by ascending or descending the generated values of map to get IP with a new random distribution as Table 4. and Figure 5.

The following algorithm shows the method of proposed Initial permutation (DIP):

**Algorithm 1**

**Input:**
- The table of Initial permutation (IP)
- The initial values of logistic map (r, n, x(1))

**Output:**
- A new random distribution of IP (DIP)

**Begin**
1. Save the table of IP in vector with length 128.
2. Generate a sequence of random values from logistic map with length 128 as index of IP vector.
3. Sort in ascending of these values to get a change of IP vector locations.

**Table 4.** Permuted IP with logistic map.

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>97</td>
<td>3</td>
<td>36</td>
<td>69</td>
<td>102</td>
<td>8</td>
<td>41</td>
<td>74</td>
<td>107</td>
<td>87</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>120</td>
<td>46</td>
<td>26</td>
<td>79</td>
<td>117</td>
<td>59</td>
<td>51</td>
<td>49</td>
<td>50</td>
<td>30</td>
<td>96</td>
</tr>
<tr>
<td>116</td>
<td>29</td>
<td>52</td>
<td>93</td>
<td>112</td>
<td>53</td>
<td>23</td>
<td>33</td>
<td>66</td>
<td>99</td>
<td>5</td>
<td>38</td>
</tr>
<tr>
<td>71</td>
<td>104</td>
<td>10</td>
<td>43</td>
<td>94</td>
<td>76</td>
<td>56</td>
<td>127</td>
<td>109</td>
<td>89</td>
<td>15</td>
<td>122</td>
</tr>
<tr>
<td>114</td>
<td>113</td>
<td>32</td>
<td>92</td>
<td>115</td>
<td>48</td>
<td>86</td>
<td>2</td>
<td>35</td>
<td>68</td>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>40</td>
<td>73</td>
<td>106</td>
<td>12</td>
<td>119</td>
<td>63</td>
<td>45</td>
<td>25</td>
<td>78</td>
<td>58</td>
<td>28</td>
<td>111</td>
</tr>
<tr>
<td>22</td>
<td>91</td>
<td>123</td>
<td>54</td>
<td>16</td>
<td>83</td>
<td>60</td>
<td>81</td>
<td>82</td>
<td>90</td>
<td>110</td>
<td>57</td>
</tr>
<tr>
<td>77</td>
<td>95</td>
<td>24</td>
<td>44</td>
<td>62</td>
<td>11</td>
<td>105</td>
<td>72</td>
<td>39</td>
<td>6</td>
<td>100</td>
<td>67</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>118</td>
<td>21</td>
<td>80</td>
<td>61</td>
<td>20</td>
<td>124</td>
<td>84</td>
<td>64</td>
<td>125</td>
<td>18</td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>27</td>
<td>85</td>
<td>47</td>
<td>121</td>
<td>14</td>
<td>88</td>
<td>108</td>
<td>126</td>
<td>55</td>
<td>75</td>
</tr>
<tr>
<td>42</td>
<td>9</td>
<td>103</td>
<td>70</td>
<td>37</td>
<td>4</td>
<td>98</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B. The proposed dynamic Final Permutation (DFP):

This section shows a distribution of fixed FP serpent as Figure 6. This figure shows that no random in distribution and the attacker can re-permute in easy. So that this paper suggests use random chaotic map to permute FP with a special key for each encryption process.

To permute FP using logistic map by ascending or descending the generated values of map to get FP with a new random distribution as Table 5, and Figure 7. The following algorithm shows the method of proposed final permutation (DFP):

**Algorithm 2**

Input:
- The table of final permutation (FP)
- The initial values of logistic map (r, n, x(1))

Output:
- A new random distribution of FP (DFP)

Begin
1. Save the table of FP in vector with length 128.
2. Generate a sequence of random values from logistic map with length 128 as index of FP vector.
3. Sort in ascending of these values to get a change of FP vector locations.

### Table 5. Permute FP with logistic map.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 28| 48| 68| 88| 108| 1 | 21| 41| 61| 122| 115|
| 81| 15| 101|35| 121|94 | 55| 54| 22| 38 | 99 | 12|
| 78| 83| 70| 91| 14| 86 |114| 20| 40 | 60 | 80 |100|
| 120|13 | 33| 53| 107|73 | 7 |127| 93 | 27 |113 | 47|
| 46| 30| 4 |75 | 62 |6 |106| 32| 52 | 72 | 92 |112|
| 5 |25 |45 |65 |126|119| 85 |19 |105| 39 | 67 |125|
| 98| 59| 63| 102|2 |58 |71 |26 |42 |43 |109 |23|
| 89|123| 3 |69 |103| 49 |29 |9 |116| 96 |76 |56|
| 36|16 |110| 82| 10 |87 |6 |79 |74 |8 |95 |34|
| 18|50 |51 |90 |117| 31 |97 |11 |77 |111|118| 57|
| 37|17 |124|104| 84 |64 |44 |24|   |   |   |   |

**Figure 7.** Distribution of proposed dynamic FP.

- **The Implementation Example**

A plaintext is “Mary had a little lamb”. A 128 bit of first 16 characters is

```
0100110110000101110011111101001000000110100001100010111001100000110000110101000010000101000011001100010000101000001101100011000010110110101100010
```

IP of these bits is

```
0000100010110010000011011100000110111111100010100000111111111111010100000110111111100100010000000110000110110111000110000001000000011010110110101000010
```

With a simple statistical test to know the features of the result bits:

- Autocorrelation test = -2.75 pass
- Poker test = 17.4878306878307 pass
- Serial test = fail
- Run test = 0.0060, h=1 (no random)
Permute the result IP with chaotic logistic map as following binary string:

```
0100000111010001110011000010100010000011111101110010000101101000000110100100
1000101100110011100101100000110111110000001101000001101100000000110
```

With a simple statistical test to know the features of the result bits compared with previous IP bits. The test shows the proposed IP has best features:

- Autocorrelation test = -0.25 pass
- Poker= 7.8709947089947 pass
- Serial = 5.51402559055117 pass
- Run test=0.5974, h=0 (random)

Example of result original FP as following binary string:

```
00101100111100001010010010101110001100011000010111000100011011000010001011001000100
0110111100001000111100100010011000001011110000101000001100000000110
```

With a simple statistical test to know the features of the result bits:

- Autocorrelation test = 0.25 pass
- Poker= 7.735497354973 pass
- Serial = 4.7581207874015 pass
- Run test=0.4463, h=0 (random)

Permute the result FP with chaotic logistic map as following binary string:

```
0010011101101110110001000110000100101000101001111000101111000010100000110100000000110
```

With a simple statistical test to know the features of the result bits compared with previous FP bits. The test shows the proposed FP has best features:

- Autocorrelation test = -0.25 pass
- Poker= 10.4444444444 pass
- Serial = 4.5691437007874 pass
- Run = 0.6962, h =0 (more random)

### C. The proposed dynamic substitution box

The following algorithm shows the method of proposed dynamic substitution box (S-box) rather than fixed table.

**Algorithm 3**

**Input:**
- Eight S-boxes table ($S_0$ to $S_7$)
- The initial values of standard map ($n$, $k$, $x(1)$, $y(1)$)

**Output:**
- A new random distribution of eight S-boxes

**Begin**

1. Save the table of each of eight S-boxes into eight vectors each with length 8.
2. Generate a sequence of random values from standard map with length 8 as index of each S-box vector. Where apply $X_{n+1}$ with even S-boxes ($S_0$, $S_2$, $S_4$, $S_6$) and $Y_{n+1}$ with odd S-boxes ($S_1$, $S_3$, $S_5$, $S_7$).
3. Sort in ascending of the even S-box values to get a change of vector locations.
4. Sort in descending of the odd S-box values to get a change of vector locations.

### D. The proposed round keys

The following algorithm shows the method of proposed dynamic two round keys for each round based on chaotic standard map.
Algorithm 4
Input:
- The initial values of standard map (n, k, x(1), y(1))
Output:
- A new two round keys for each round
Begin
1. Implement While Loop (no. round <n), where n =10 in the proposed method rather than 32 of classical serpent.
   a. Generate a sequence of random values from standard map. Where apply $X_{n+1}$ to generate 1\textsuperscript{st} round key and $Y_{n+1}$ to generate 2\textsuperscript{nd} round key.
   b. Rounding these values to the nearest integer.
   c. Convert the integer values into binary sequence.
   d. Pick 1\textsuperscript{st} 128 bits as 1\textsuperscript{st} round key
   e. Pick 2\textsuperscript{nd} 128 bits as 2\textsuperscript{nd} round key
   f. Change the value of initial values of chaotic map to generate new sequence for next round (i.e.) $X(n+1)=x(n)+0.1$ and $y(n+1)=y(n)+0.1$
End of While Loop

Flowchart of the proposed modification Serpent

Figure 8, shows the proposed of encryption method that describe by previous sections of proposed algorithms.

Figure 8. Flowchart of proposed encryption method.

6. Security and Statistical Analysis
- Brute Force Attack
  Key space size is the total number of different keys that can be used for the encryption algorithm. It should be large enough to resist the brute force attack. The precision of
The initial values and parameter of the proposed method is \(10^{16}\). The key space size of secret keys is \(10^{16}K\). The number of initial values and parameters is \(k = 3\) of logistic map \((r, n, x(1))\) and \(k = 4\) of standard map \((n, k, x(1), y(1))\). So, the total \(k\) value is 7. Then the key space size is \(10^{112}\) so that the at least key space size of our proposed is \(10^{112}\). It is large enough against brute force attack.

- **Key Sensitivity**
  An ideal encryption algorithm should be sensitive to secret key that means the encrypted method cannot be decrypted correctly, if there is only a slight change in the secret key. The proposed method has a key sensitivity when changing only one parameter of the secret keys and keeping all other keys unchanged. The secret keys are \((r, n, x(1), y(1), k)\).

- **NIST Tests**
  The average of 100 NIST tests between a classical serpent and the proposed modified as following Table 6, of NIST statistical tests. This table shows that the proposed method has best randomness compared with a classical serpent. And also can reduce the number of round to 10 rounds rather than 32 so the result is the best randomness and decrease the time average.

<table>
<thead>
<tr>
<th>No.</th>
<th>NIST test</th>
<th>Classical Serpent with 32 round</th>
<th>Modified Serpent with 10 round</th>
<th>Modified Serpent with 32 round</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frequency</td>
<td>0.5022</td>
<td>0.8025</td>
<td>0.8814</td>
</tr>
<tr>
<td>2</td>
<td>Block frequency M=128</td>
<td>0.8818</td>
<td>0.9997</td>
<td>0.9335</td>
</tr>
<tr>
<td>3</td>
<td>Runs</td>
<td>0.0155</td>
<td>0.0169</td>
<td>0.0161</td>
</tr>
<tr>
<td>4</td>
<td>Longest Runs of Ones</td>
<td>0.0247</td>
<td>0.0437</td>
<td>0.0446</td>
</tr>
<tr>
<td>5</td>
<td>Rank</td>
<td>0.1576</td>
<td>0.1559</td>
<td>0.1916</td>
</tr>
<tr>
<td>6</td>
<td>FFT</td>
<td>0.2629</td>
<td>0.3018</td>
<td>0.5153</td>
</tr>
<tr>
<td>7</td>
<td>NonOverlapping Templates m=9</td>
<td>0.3260</td>
<td>0.4872</td>
<td>0.4301</td>
</tr>
<tr>
<td>8</td>
<td>Overlapping Templates all ones B='111111111'</td>
<td>0.0223</td>
<td>0.0351</td>
<td>0.0373</td>
</tr>
<tr>
<td>9</td>
<td>Universal statistical L=6 &amp; Q=1000</td>
<td>0.6864</td>
<td>0.6532</td>
<td>0.6973</td>
</tr>
<tr>
<td>10</td>
<td>Linear complexity M=500</td>
<td>0.1842</td>
<td>0.4087</td>
<td>0.7816</td>
</tr>
<tr>
<td>11</td>
<td>Serial m=12</td>
<td>0.3310</td>
<td>0.3656</td>
<td>0.5969</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5001</td>
<td>0.6851</td>
<td>0.6142</td>
</tr>
<tr>
<td>12</td>
<td>Approximate Entropy m=8</td>
<td>0.1365</td>
<td>0.1515</td>
<td>0.1732</td>
</tr>
<tr>
<td>13</td>
<td>Cumulative Sums (Forward)</td>
<td>0.5219</td>
<td>0.6784</td>
<td>0.9161</td>
</tr>
</tbody>
</table>

- **Histogram Analysis and Other Tests**
  Figure 9, shows the implement of proposed encryption for baboon image. Note that a good result of the flat histogram for encryption image. The result of other analysis tests is good such as (entropy =7.9989). The tests against differential attack (NPCR=99.8022, and UACI= 33.3382).
7. Conclusion
The proposed dynamic initial permutation (IP), final permutation (FP), Substitution box (S-box), and the generated round keys give the best randomness compared with classical serpent algorithm and also can reduce the number of round and time usage. The proposed method has sensitivity to any change in the key since it uses chaotic map in the key round generation. And also, it more robust than classical method since it used another chaotic map to create dynamic permutation and substitution.

References
2. Xiaojun, T.; Minggen, C.A. Novel Image Encryption Scheme Based on Feedback and 3D Baker. 4th International Conference on Wireless Communications. *2008*


