Abstract— Today we are living in a world of digital natives. Security of data is of primary concern. There are different methods by which security can be ensured. Access control methods are one among them. Cryptographic methods are also of great importance. Randomness is a subject of great concern in the science of cryptography. True random number generators and pseudorandom number generators are there. Out of which true random number generators yield much better performance. But for effective performance true random number generators are used for generating seed values for pseudorandom number generators and thus combining the advantages of true random number generators and pseudo random number generators. In this paper a new algorithm known as Nested Squared Random Number Generator method is developed that uses true entropy for generating seed value and for nesting. Combined effect of Squared Prime and nesting offer better results.

Keywords—True Random Number Generator, Pseudo random number generator, Linear Congruential Generator, Prime number, Kolmogorov Smirnov Test, Runs Test

I. INTRODUCTION

Independent, unpredictable and uniformly distributed numbers that cannot be reliably reproduced are referred to as random numbers [1]. They play a major part in practical implementation and strength of most cryptographic systems. They may be used as keys for symmetric crypto-systems, public key parameters, session keys, etc. [2]. Failure of obtaining strong keys definitely will end up with data security compromise. Therefore, strong random number generators that exhibit high statistical quality and can withstand cryptanalysis efforts are keenly sought. Such strong random number generators constitute an important building block in the design and testing of high quality crypto-systems [3].

There are two principal methods used to generate random numbers. The first method known as true random number generators measures some physical phenomenon that is expected to be random and uses the measure to generate random number sequence. The second method known as pseudo random number generator uses computational algorithms that can produce long sequences of apparently random results, which are in fact completely determined by a shorter initial value, known as a seed value or key,[4][5]

A Linear Congruential Generator (LCG) is an algorithm that yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation. The method represents one of the oldest and best known pseudorandom number generator algorithms. The theory behind them is relatively easy to understand, and they are easily implemented and fast, especially on computer hardware which can provide modulo arithmetic by storage-bit truncation.

It will be better to mix together true random number generator and pseudorandom bit generator to get better performance.

In the proposed method true randomness is exploited both in the seed generation step and in series generation step. Based on the millisecond value of the current system clock a pixel value is selected from current picture captured by system camera selected picture. Seed value is generated based on this pixel value. Then using this seed value the series is generated but in this step along with the seed value another random factor is used which increase the randomness to a great extend.

II. RANDOM NUMBER GENERATORS

High-quality random number generation is essentially demanded for security. True random number generators capture random events in the real world to create its sequences. Each bit of the bit streams is independent from the other bits and the probabilities of 1/0 occurrences are identical. The generated sequence is such that it is impossible for anyone to predict the next number in the sequence. Random number generators can be broadly classified into true random number generators and pseudo random number generators.

A. True random number generator

A true random number generator uses entropy sources that already exist. Entropy refers to the amount of uncertainty about an outcome of an event. Real world events such as flipping coin and rolling of dice have a high degree
of entropy, because it is almost impossible to predict accurately what the final result will be. It is the source of entropy that makes a true random number generator unpredictable.

Because true random numbers cannot be predicted by computational methods, they are highly desirable for security purposes.

Some major true random generators are Random.org, Hotbits, lasers, and oscillators.[6]

But true random numbers have a lot of disadvantages:

1. True random numbers are to be extracted from natural sources which demand a lot of hardware and are expensive to set up.
2. The generation speed of random numbers may not be sufficient for the application.
3. Entropy sources can be influence by changes in the physical environment of the device like changes in the temperature, changes in the voltage or frequency of the power supply, exposure to radiation, etc.
4. An intruder can control at least some of the parameters that may affect the entropy value.

B. Pseudo Random number generators

Random number generators that do not rely on real world phenomena to produce their streams are referred to as pseudo random number generators. These generators appear to produce random sequences to anyone who does not know the secret initial value. In a minimalistic generator, the initial value will be the only time entropy is introduced into the system. Unlike true random number generators that convert entropy sources directly into sequences, a pseudo random needs to find entropy to use to keep itself unpredictable. Classic tactics for accomplishing this include taking the time of day, the location of the mouse, or the activity on the keyboard.[7]

So what is advisable is to combine the true random number generation technique and pseudo random number generation technique. True random number generated from a natural phenomenon can be used as the seed value and this seed value can be used to generate the random number series by some pseudo random number generator algorithm.

III. RELATED WORKS

A true random number generator uses entropy sources that already exist. Entropy refers to the amount of uncertainty about an outcome of an event. Real world events such as flipping a coin and rolling of dice have a high degree of entropy, because it is almost impossible to predict accurately what the final result will be. It is the source of entropy that makes a true random number generator unpredictable.

Some major true random generators are Random.org, Hotbits, lasers, and oscillators.[7]

Random.org is a website created in 1998 by Mads Haahr. The website produces "true" random numbers based on atmospheric noise captured by several radios tuned between stations. [8]

HotBits is an Internet resource that brings genuine random numbers, generated by a process fundamentally governed by the inherent uncertainty in the quantum mechanical laws of nature, directly to your computer in a variety of forms.

Random number generation scheme that uses broadband measurements of the vacuum field contained in the radio-frequency sidebands of a single-mode laser is a good advancement in TRNG.[9] Study of Semiconductor lasers show that good quality random bit sequences can be generated at very fast bit rates using physical chaos in semiconductor lasers.[10]

True random number generators based on sampling phase jitter in oscillator rings will generate provably random bits with some tolerance to adversarial manipulation and running in the megabit-per-second range.[11]

Random number generators that do not rely on real world phenomena to produce their streams are referred to as pseudo random number generators. These generators appear to produce random sequences to anyone who does not know the secret initial value.

Middle-square method, Linear Feedback Shift Registers, Xorshift generators are some of the early Pseudo Random Number Generators.

In Middle-square method to generate a sequence of 4-digit pseudorandom numbers, a 4-digit starting value is created and squared, producing an 8-digit number. If the result is less than 8 digits, leading zeroes are added to compensate. The middle 4 digits of the result would be the next number in the sequence, and returned as the result. This process is then repeated to generate more numbers.[12]

A Linear Congruential Generator (LCG) is an algorithm that yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation.[13]

A Lagged Fibonacci generator (LFG or sometimes L Fib) is an example of a pseudorandom number generator. This class of random number generator is aimed at being an improvement on the ‘standard’ linear congruential generator. A very popular and widely used example of feedback shift registers is the Mersenne Twister . [13] The Mersenne Twister can be classified as a twisted generalized feedback shift register (TGFSR), which has algorithms more tightly tied to matrices than strings. [14]

IV. PROPOSED METHOD

A. Methodology

Proposed method can be divided into two parts:

1. Getting the seed value
2. Generating the series
1. **Getting the seed value**

In this proposed method system clock value read. Based on the read clock value pixel value of the current picture captured by system camera is read. Hence in this step two true random source effects are combined. Using this pixel value two prime numbers p1 and p2 are generated. The two prime numbers are such that p1 is the greatest prime number less than the read pixel value and p2 is the smallest prime number greater than the read pixel value. Then the following calculations are done to generate the seed value.

\[ b_0 = p_1^2 \mod m \]  
\[ d_0 = p_2^2 \mod m \]  
\[ f_0 = b_0 * d_0 * l_0 \]

where \( l_0 \) is the current clock value read.

Equations (1), (2) and (3) are performed to generate seed value \( f_0 \).

\( m \) is relatively prime to \( p_1*p_2 \)

When compared with other physical sources of entropy system clock value is an entropy source that cannot be influenced by changes in the physical environment of the device. This system has an added advantage of reading a pixel value from the current image captured by system camera and using it for seed value generation.

Seed value is the product of two prime numbers and current system clock value read(millisecond part). The complexity of the seed is thus due to two factors.

i. Difficulty in factorizing product of two prime numbers
ii. True randomness introduced

This increased complexity makes the job of cryptanalyst difficult or rather unable.

2. **Generating the series**

Series is generated by implementing the following three equations (4), (5) and (6)

\[ b_n = b_{n-1}^2 \mod m; \]  
\[ d_n = d_{n-1}^2 \mod m; \]  
\[ l_n = (l_{n-1}*p1+p2) \mod m \]  
\[ f_n = (b_n^*d_n^*l_n) \mod q \]

The above three steps are repeated to generate the elements in the sequence. \( f_n \) denotes the \( n^{th} \) random number being generated. \( b_n \) and \( d_n \) are integers that helps in the generation and takes its values from \( b_{n-1} \) and \( d_{n-1} \). \( l_n \) is a random number generated using the Linear Congruential Generator(LCG) algorithm with \( l_0 \) as seed value. Here LCG algorithm is nested within another algorithm hence our proposed system is called Nested Squared Prime Generator.

**B. Algorithm**

Step 1. System clock time is read
Step 2. Based on the read system clock a pixel value is read from the current image captured using system camera.
Step 3. Based on the pixel value two prime numbers p1 and p2 are generated
Step 4. Perform equations (1), (2) and (3) to find the seed value \( f_0 \)
Step 5. The random series is generated by repeating equations (4), (5), (6) and (7)

The generated series is then using Kolmogorov Smirnov test and Runs test.

Flow chart of the algorithm is given below.

![Flowchart for the proposed method](image-url)
C. Advantages of the proposed Method

1. It combines the effect of two true random sources, clock value and pixel value, for seed value generation.
2. In the generated random number series a sequence is never repeated.
3. Random numbers of another random series generated with clock value read later as the seed value is also used in
   the generation of the random numbers.
4. Nested algorithm ensures random series generated to be totally unpredictable and random.
5. It exploits the prime factorization problem to make cryptanalysis difficult and increase security factor. [6][15]

D. Complexities involved in this method

a. Since the seed value depends on the current clock value there is an element of time complexity involved in this
   work for which cryptanalysis is thus difficult to be done.

b. For added effect based on the read clock value the pixel value corresponding to the clock value is read from the
   image captured by the system camera.

c. Product of two prime numbers forms the seed value. Hence seed value is difficult to be factorised. Cryptanalysis
   is thus practically impossible.

d. When random numbers are generated nesting of another random number generation algorithm makes the
   random number series more random.

V. IMPLEMENTATION

The algorithm was implemented in Matlab. It reads the current system clock value and then extracts a pixel
value from the image currently captured.

The value extracted is used to generate two prime numbers p1 and p2 less than and greater than extracted value.
Product of the square of p1 and p2 and the clock value read again (k0) forms the seed value.

That is following steps are followed

\[ b_0 = p1^2 \mod m \]
\[ d_0 = p2^2 \mod m \]
\[ k_0: \text{clock value read again after } b_0 \text{ and } d_0 \text{ is calculated} \]
\[ f_0 = (b_0 * d_0 * k_0) \mod q \]

Then the series is generated as follows

\[ b_n = b_{n-1}^2 \mod m; \]
\[ d_n = d_{n-1}^2 \mod m; \]
\[ k_n = (k_{n-1} * p1) + p2 \mod m \]
\[ f_n = (b_n * d_n * k_n) \mod q \]

here m is selected so that it is relatively prime to \( p1^2 \cdot p2 \) and q decides the range of random numbers generated.

The true random value read by using sources such as system clock value and selected image for generating p1
and p2 was 55.
With this value p1 was generated as 61 and p2 was 67

Now the seed value \( f_0 = p1^2 + p2^2 \cdot k_0 \mod m \)

\[ m \] is selected as 71

\[ f_0 = 195 \]

The generated 100 random numbers in the series are

![Figure 2. Generated result](image-url)
Now statistical tests such as Kolmogorov-Smirnov Test and runs test were applied on the generated sequence for testing its randomness.

Scatter diagram and Bar graph of Hundred generated random values are given below. The diagram depicts good randomness in the generated values. The diagram shows that there is no linear relationship or correlation between the generated random values.

![Figure 3. Scatter diagram of the output got](image1)

![Figure 4. Bar Graph of the output got](image2)

**VI. ANALYSIS**

**A. Kolmogorov Smirnov Test**

KS test can be used to check the randomness of the numbers generated by a RNG that is allowed to take on any value within a certain interval, leading to a continuous cdf [16].

- \( H_0 \) = Sequence being tested is random
- \( H_a \) = Sequence being tested is not random

<table>
<thead>
<tr>
<th>j</th>
<th>random number</th>
<th>normalised</th>
<th>((\frac{1}{n} - x))</th>
<th>(x-(\frac{1}{1}+1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>0.11557789</td>
<td>-0.08224</td>
<td>0.115578</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>0.12060302</td>
<td>-0.05394</td>
<td>0.08727</td>
</tr>
<tr>
<td>3</td>
<td>41</td>
<td>0.20603015</td>
<td>-0.10603</td>
<td>0.139363</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>0.2361809</td>
<td>-0.10285</td>
<td>0.136181</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>0.26633166</td>
<td>-0.09966</td>
<td>0.132998</td>
</tr>
<tr>
<td>6</td>
<td>58</td>
<td>0.29145729</td>
<td>-0.09146</td>
<td>0.124791</td>
</tr>
<tr>
<td>7</td>
<td>62</td>
<td>0.31155779</td>
<td>-0.07822</td>
<td>0.115538</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>0.32160084</td>
<td>-0.05494</td>
<td>0.088275</td>
</tr>
<tr>
<td>9</td>
<td>65</td>
<td>0.32663317</td>
<td>-0.02663</td>
<td>0.059966</td>
</tr>
<tr>
<td>10</td>
<td>65</td>
<td>0.32663317</td>
<td>-0.00067</td>
<td>0.026633</td>
</tr>
<tr>
<td>11</td>
<td>65</td>
<td>0.32663317</td>
<td>0.00034</td>
<td>0.000017</td>
</tr>
<tr>
<td>12</td>
<td>71</td>
<td>0.35678392</td>
<td>0.043216</td>
<td>0.00998</td>
</tr>
<tr>
<td>13</td>
<td>71</td>
<td>0.35678392</td>
<td>0.076549</td>
<td>0.04322</td>
</tr>
<tr>
<td>14</td>
<td>87</td>
<td>0.43718593</td>
<td>0.029481</td>
<td>0.03853</td>
</tr>
<tr>
<td>15</td>
<td>88</td>
<td>0.44221106</td>
<td>0.057789</td>
<td>0.02446</td>
</tr>
<tr>
<td>16</td>
<td>94</td>
<td>0.47236181</td>
<td>0.060972</td>
<td>0.02764</td>
</tr>
<tr>
<td>17</td>
<td>96</td>
<td>0.48241206</td>
<td>0.084255</td>
<td>0.05092</td>
</tr>
<tr>
<td>18</td>
<td>103</td>
<td>0.51578794</td>
<td>0.082412</td>
<td>0.04908</td>
</tr>
<tr>
<td>19</td>
<td>110</td>
<td>0.55276382</td>
<td>0.08057</td>
<td>0.04724</td>
</tr>
<tr>
<td>20</td>
<td>116</td>
<td>0.58291457</td>
<td>0.083752</td>
<td>0.05042</td>
</tr>
<tr>
<td>21</td>
<td>118</td>
<td>0.59894687</td>
<td>0.107035</td>
<td>0.00737</td>
</tr>
<tr>
<td>22</td>
<td>119</td>
<td>0.59798395</td>
<td>0.135343</td>
<td>0.10201</td>
</tr>
<tr>
<td>23</td>
<td>137</td>
<td>0.68844221</td>
<td>0.078224</td>
<td>0.04489</td>
</tr>
<tr>
<td>24</td>
<td>139</td>
<td>0.69849246</td>
<td>0.101508</td>
<td>0.06817</td>
</tr>
<tr>
<td>25</td>
<td>149</td>
<td>0.74874372</td>
<td>0.08459</td>
<td>0.05126</td>
</tr>
<tr>
<td>26</td>
<td>164</td>
<td>0.8241206</td>
<td>0.042546</td>
<td>0.00921</td>
</tr>
<tr>
<td>27</td>
<td>173</td>
<td>0.86934673</td>
<td>0.030653</td>
<td>0.02836</td>
</tr>
<tr>
<td>28</td>
<td>174</td>
<td>0.87437186</td>
<td>0.058961</td>
<td>0.02563</td>
</tr>
<tr>
<td>29</td>
<td>185</td>
<td>0.92964824</td>
<td>0.037018</td>
<td>0.00369</td>
</tr>
<tr>
<td>30</td>
<td>192</td>
<td>0.96482412</td>
<td>0.035176</td>
<td>0.00184</td>
</tr>
</tbody>
</table>
From the above table $K^+ = 0.135343$

$K^- = 0.139363$

From KS test table at $n=30$ and $1-\alpha=0.9$  

$K = 0.21756$

$K^+ < K$ and $K^- < K$ hence sequence is random and pass KS test

**B. Runs Test**

A run is defined as a series of increasing values or a series of decreasing values. The number of increasing, or decreasing, values is the length of the run.

First 30 samples are taken and median is found out. If the value is less than median then it is denoted by -1 otherwise +1. Now runs are counted and hypothesis testing is done.

- $H_0$ : Sequence is random
- $H_1$ : Sequence is not random

In the above sequence taking 30 samples are taken and median is calculated. Median is got as 91. Now all the values greater than 91 is denoted as +1 and values less than 91 as -1. Number of runs is got as 15. Now, $n_1$ , number of -1, is 15 and $n_2$, number of +1 is 15.

From runs table the test is passed if the number of runs is between 10 and 22. Here number of runs is 15 hence test is passed.

**VII. CONCLUSION**

A novel algorithm for computationally fast, cryptographically secure pseudo random key generator has been proposed and implemented in this work. Analysis using randomness tests have shown that the generated sequences exhibited strong random behavior.

Generation of the seed involved true entropy source (system clock value and input image) that increased the complexity of the algorithm. True Randomness is exploited again during sequence generation. Complexity inherent with prime factoring problem and nesting is also exploited in generation of seed value. In the generated sequence the sequence is never repeated. Cryptanalysis is practically not possible in this system.

**REFERENCES**


