Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.4, No.10, 2013

Comparative Analysis of Efficiency of Fibonacci Random Number Generator Algorithm and Gaussian Random Number Generator Algorithm in a Cryptographic System.

Dr. Ing. Edward Opoku-Mensah[1], Abilimi A. Christopher[2], Francis Ohene Boateng[3]

1. Department of Information Technology Education, University of Education Winneba, P.O.Box 1277, Kumasi, Ghana, Email : ingeopm@gmail.com
2. Department of Computer Science, Christian Service University College, P.O. Box 3110, Kumasi, Ghana, Email: cabilimia@gmail.com
3. Department of Information Technology Education, University of Education Winneba, P.O.Box 1277, Kumasi, Ghana. Email : fanbotgh@yahoo.com

Abstract
Random Numbers determine the security level of Cryptographic Applications as they are used to generate padding schemes in the encryption and decryption process as well as used to generate cryptographic keys. The more randomness in the numbers a generator generates the more effective the cryptographic algorithm, and the more secured it is to be used for protecting confidential data. Sometimes developers find it difficult to determine which Random Number–Generators (RNGs) can provide a much secured Cryptographic System for secured enterprise application implementations. Two of such random number generators include the Fibonacci Random Number Generator and the Gaussian Random Generator. The researchers sought to determine, between these two, the better to be used for improving data security in cryptographic software systems. The researchers employed statistical tests like Frequency test, Chi-Square test, Kolmogorov-Smirnov test on the first 100 random numbers between 0 and 1000 generated using the above generators. The research concluded that Fibonacci Random Number Generator is more efficient than the Gaussian Random Number Generator and therefore recommended the choice of Fibonacci Random Number Generator when choosing between the two for use in a cryptographic system for better data security.
Keywords: Cryptographic Algorithms, Random Number Generators, encryption, decryption

1. Introduction

According to Kessler (2012), in the information security world, it is repeatedly examining statements like 'protected via 128-bit Advance Encryption Standard' or 'sheltered by 2048 bit validation'. Invariably people would want to know the data security or safety power of various cryptographic algorithms. Some cryptographic algorithms like Rivest, Shamir Adleman, Elliptic curve cryptography and Advance Encryption Standard contain a confirmation evidence of not being easy to crack. These cryptographic algorithms are mostly used in protocols to take care of data that need to be protected for their confidentially, integrity and identity. As an illustration think about using Secure Socket Layer (SSL) or Transport Layer Security (TLS) while a book is bought from an online market like the Amazon, or when a sum of money will have to be transferred to your bank account by a bond (Hasani, 2011). Another example can be seen in how Internet Protocol Security (IPsec) and Internet Key Exchange (IKE) are used while a computer is connected to a network in order to access information on the World Wide Web.

AuthenTec Embedded Security Solutions (2010) stated that some of the things that are not often seen are random number generator strengths and their use by security applications. The most important thing software designers are concerned with is the controlled use of bits and their creation rate, but not much of the real randomness associated with the bits produced.

The security level of a cryptographic application is dependent on the importance of the random numbers used in the application. Furthermore, how intricate it is to crack a cryptographic system is dependent on the quality of random numbers used in the cryptography (Biege, 2006).

High security cryptographic system can be achieved if researchers can understand cryptographic systems are developed based on the principle of cryptographic algorithms design called Kerckhoff Principle which states that "*The security of the system must depend solely on the key material, and not on the design of the system"*(AuthenTec Embedded Security Solutions, 2010*).* This is due to the fact that before an attacker can break modern cryptographic systems, the attacker needs to deduce the keys used in developing the algorithms as well as the protocol used. The strength of a cryptographic application is expressed with the assumption that the attacker has no idea of the bits and its pattern used in the key formation and used in the cryptographic systems. An enhanced attack against cryptographic system exposes extra key bits that can be computed by looking at the (inadequate sum of) output data, and that diminishes the 'valuable strength' of an algorithm.

A manual prepared by the Internet Engineering Task Force (IETF) and named a 'Best Practices' document is used to describe the significance of true randomness in cryptographic applications, and these serve as guidelines on how random numbers are created (Eastlake, Schiller & Crocker, 2005). The National Institute of Standard and Technology provides a subdivision on Random Number Creation in their Cryptographic Toolbox pages, and a number of standards organizations like International Standard Organisation (ISO), Institute of Electrical and Electronic Engineers (IEEE), IETF, and American National Standards Institute (ANSI), are working on, standards associated to random number generation. This goes to show the importance of proper random number generation. This research therefore compares the effectiveness of Pseudo Random Number Generator so as to get a secured cryptographic system, when implemented in cryptography.

2. Random numbers in Cryptography

A high-quality Cryptographic System needs good-quality random numbers (Mike, Paul, & Mark, 2012). This paper evaluates software security based on Pseudo Random Number Generator Algorithms (PRNGA) needed for Cryptographic Applications.

Nearly each and every cryptographic protocol needs the exploit and creation of furtive ethics that attackers should not know. Algorithms for the generation of random numbers are requisites for generation of private keys or public key pairs for symmetry and hybrid cryptosystems as well as asymmetric encryption algorithms like Diffie-Hellman, Rivest Shamir Adleman and Digital Signature Algorithm. Random Number Generators are considered necessary to generate blinding values, challenges, padding bytes, and nonces (salts) in cryptographic applications. The randomness of keys used in cryptographic system is the pre-requisite for security protocol; RNGs used in cryptographic systems ought to convene rigid parameters. Hiding the important patterns concerning the output of the Random Number-Generator from the attackers, as well as folks who know how the Random Number Generator Algorithm is designed, is very important in every cryptographic application. For illustration, the clear entropy of the output of Random Numbers Generator must be as close up as probable to the length of the bit use. The paper compares the efficiency of two different random number generators, Fibonacci Random Number Generator Algorithm and Gaussian Random Number Generator Algorithm, in Cryptographic Algorithm to see the better one to be used in cryptography.

The ideal quality of high-quality pseudo random number generator is that the series should not be easy to predict by the attacker of a cryptosystem. This is because sometimes the adversary may know the cryptographic algorithm used, and this is what is called the Kerckhoff's principle (Yehuda, 2006).

According to Andrew et al. (2010), an additional reason for statistical randomness of a pseudo random number generator is the basis for good pseudo random number generators. All correlations and trends in the generators should be hidden. Statistically these can be proven. The statistical component of a pseudo random number generator shows that every value or result over a given range has equal chance of incidence. Every number has a probability of ½ in the range of [0, 1]. Every couple of numbers used will have an occurrence of ¼ and a frequency of 1/8 for every triplet. That is when the basis for the numbers generated is considered not biased and consecutive output is considered not correlated. The mean and variance computation must also be included in the statistical randomness test. If the output for the Mean is closer to 0.5 and that of the variance is closer to 0.08, then the pseudo random numbers are homogeneous.

According to Andrew et al. (2010), another test for the efficacy of pseudo random numbers generators is the Frequency Test also called the Monobits. The Frequency Test concentrates on the percentage of ones and zeroes in the entire series of numbers produced by the generators. This test is used to establish that the series of numbers generated or created produces the same number of ones and zeroes as should be anticipated for any true random series. That is the assessment of how close the fraction of ones is to ½, which means that, the amount of zeroes and ones in series must be almost the same when compared to each other. The Monobit Test in a Block concentrates on the percentage of ones and zeroes in the Blocks of M-bits. This assessment ascertains whether M/2 is roughly the number of ones in the block of M-bit.

According to Justin (2012), the effectiveness of Pseudo Random Number Generator algorithms also depends on the period. Every algorithm has input parameters that produce an equivalent period. What happens is that any time an algorithm for generating the random numbers creates a repeated value then it will start the generation process all over again in order to ensure that the results are not correlated. Therefore it should be recognized that the seed or the value or the parameter determines the period of the algorithm used. This means that the more the repetition, the shorter the period used and vice versa.

Lastly, uniformity also determines the period of the pseudo random number generator. That is every perfect generator algorithm will produce records that are consistently distributed within a certain interval. This makes it very difficult for algorithms that have smaller periods, if even they have huge intervals, to pass this kind of statistical test for randomness (Christophe & Diethelm, 2003). They also explained that if there exist correlations within the numbers generated, then the algorithms with larger period will experience massive

problems.

Christophe and Diethelm (2003), stated that the ideal thing to do to any cryptographic application systems is to make sure that the generator used in the generation of the random numbers produces values that correlate serially. When smaller values of numbers are produced via random number generators, then algorithms that have larger periods and high-uniformity may be highly inconsistent. The security of the generator then depends on the correlation of values produced, and hence it can be explained that the more the system is secured the lesser the correlations of numbers in the generator and vice versa.

3. Methodology

The researchers employed statistical tests for testing the randomness of two pseudo random numbers generators, namely Fibonacci Random Number Generator and Gaussian random number generator algorithms, in order to determine whether a set of data has a recognizable pattern to it or not. These tests are Chi-Square Test and Kolmogorov-Smirnov test (KS-test).

3.1 Chi-square Test

The Chi-Square Test has shown to conform to the errors of pseudo random series generators sensitivity. The distribution of the Chi-Square is a non-negative value and a proportion which show how often real random series should surpass a computed number or value determined as a result for collections of zeroes and ones in a certain data file. The procedures used are as follows:

i. Java Programming codes are written for each of Fibonacci Random Number Generator and Gaussian Random Number Generator.

ii. This test involved producing sequences of 100 random integers between 0 and 1000 using the Java codes for each of Fibonacci Random Number Generator and Gaussian Random Number Generator.

iii. The generated random numbers in (ii) above is coded in Statistical Package for Social Science (SPSS) software to test for the randomness of the numbers in the generators. The procedures include:
   a. Go to and click on the ***Analyse Menu*** on the SPSS bar after the data has been coded.
   b. Choose ***Nonparametric Test*** on the submenu.
   c. Go to and then Click on ***Chi-Square Test***.
   d. Select the two random numbers generators and move them to the ***variable list.***
   e. The results generated for the Chi-Square Test.

iv. For uniform distributions of random numbers from the generators, then the expectations is that there should be 1 appearance or occurrence for the number 1 or 2 or 3 and so on to the end of the dimension of the numbers, 100. Then how often each of the numbers appears will be expected to have a frequency of 1.0 or 1.1 averagely for all the numbers from 1 to 100 ranges. The observed frequency is the real frequency for every one of the numbers generated by the generator with the ranges of numbers specified. The Chi-Square value or statistics is calculated from the difference between the observed frequency of the test and that of the expected frequency of every one of the numbers generated as follows:

$$\chi^2 = \sum_{i=1}^{R} \frac{(O_i - E_i)^2}{E_i}$$

R is the distinct individual random numbers likely ($R = 100$), the number of the observed frequencies of happenings or occurrences for the random numbers (i) is O_i and where the frequency of expectation or the expected frequency is E_i for the random integer or number i. Since the expectation is that the integer distribution must uniformly spread, then the occurrences for every random number or integer must have equal expected frequencies. E_i can then be computed with N as the total number of observations, using the equation above.

There is one way to explain the above percentages namely; it explains the level of non-randomness suspicions, that is, the degree to which the series of random numbers generated is assumed of being non-random. If the percentages are between 90% and 95%, 10% and 5% shows the series is “almost suspect”. If the percentage is in the range of 99% and 95% or in the range of 1% and 5%, then the series is suspect. Finally, if the percentage is greater than 99% or less than 1%, then the series is almost certainly not random.

3.2 Kolmogorov-Smirnov Test (KS-test)

The Kolmogorov-Smirnov (K-S) test is based on the Empirical Distribution Function (ECDF). Given N ordered data points $\boldsymbol{Y_1, Y_2, ..., Y_N}$, the ECDF is defined as

$$E_N = n(i)/N$$

where $\boldsymbol{n(i)}$ is the number of points less than $\boldsymbol{Y_i}$ and the $\boldsymbol{Y_i}$ are ordered from smallest to largest value. This is a step function that increases by $1/N$ at the value of each ordered data point.

The procedures used to perform the Kolmogorov-Smirnov test (KS-test) for the two random numbers generators namely: Fibonacci Random Number Generator and Gaussian Random Number Generator are as

follows:

i. Java Programming codes are written for each of Fibonacci Random Number Generator and Gaussian Random Number Generator.

ii. This test involved produces sequences of 100 random integers between 0 and 1000 using the Java codes for each of Fibonacci Random Number Generator and Gaussian Random Numbers Generator.

iii. The generated random numbers in (ii) above is coded in Statistical Package for Social Science (SPSS) software to test for randomness of the numbers in the generators. The procedures include:

a. Go to and click on the ***Analyse Menu*** on the SPSS bar after the data has been coded.
b. Choose ***Nonparametric Test*** on the submenu.
c. Go to and then Click on ***One Sample KS-Test***.
d. Select the two random numbers generators and move them to the ***variable list*** and check ***Normal.***
e. Click on ***Exact*** tap and select ***asymptotic*** option and then click OK.

4. Results

4.1 Gaussian Random Number Generator

This generator produces smaller random numbers at the beginning and at the end of the range used, while producing bigger numbers at the middle of the range of the numbers generated as shown in ***Figure 1***. However abnormal uniformity occurs at the range of -100 to 50. The generator produces random numbers that are much skewed to the center, hence produces a substantially good standard deviation of 68.515. This standard deviation value shows that the generator obeys the normal distribution curve.

Table 1
The descriptive statistics of the Random Numbers Generators

Generators	N	Mean	Std. Deviation	Minimum	Maximum
Gaussian Random Number Generator	100	5.08	68.51	-121	126
Fibonacci Random Number Generator	100	-90142675.73	975930540	-2092787285	2.E9

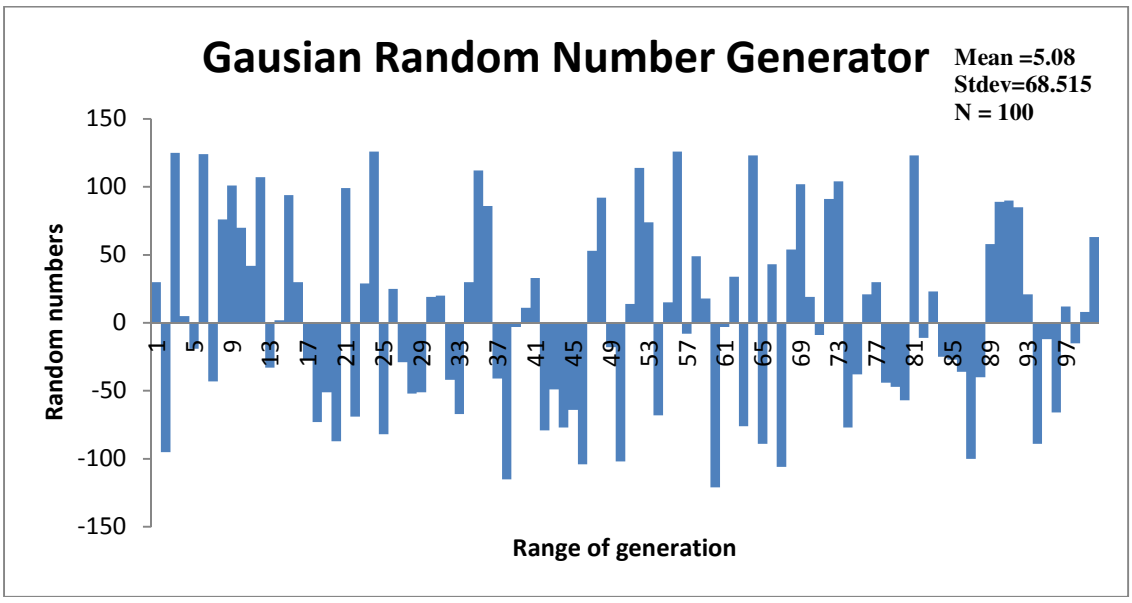

Figure 1: The trend of randomness Gaussian Random number Generator

4.2 Fibonacci Random Number Generator

From the results generated, it is found out that the Fibonacci Random Number Generator started with smaller random numbers at the beginning, elevates gradually, gets to its peak at a value of 0.0E0 and declines gradually to the end of the range as in ***Figure 2.*** This result tends to obey the normal distribution curve more than all the other generators studied, with standard deviation of 9.759E8.This represents the greater deviation of individual numbers produced by the generator from each other and hence not normally distributed.

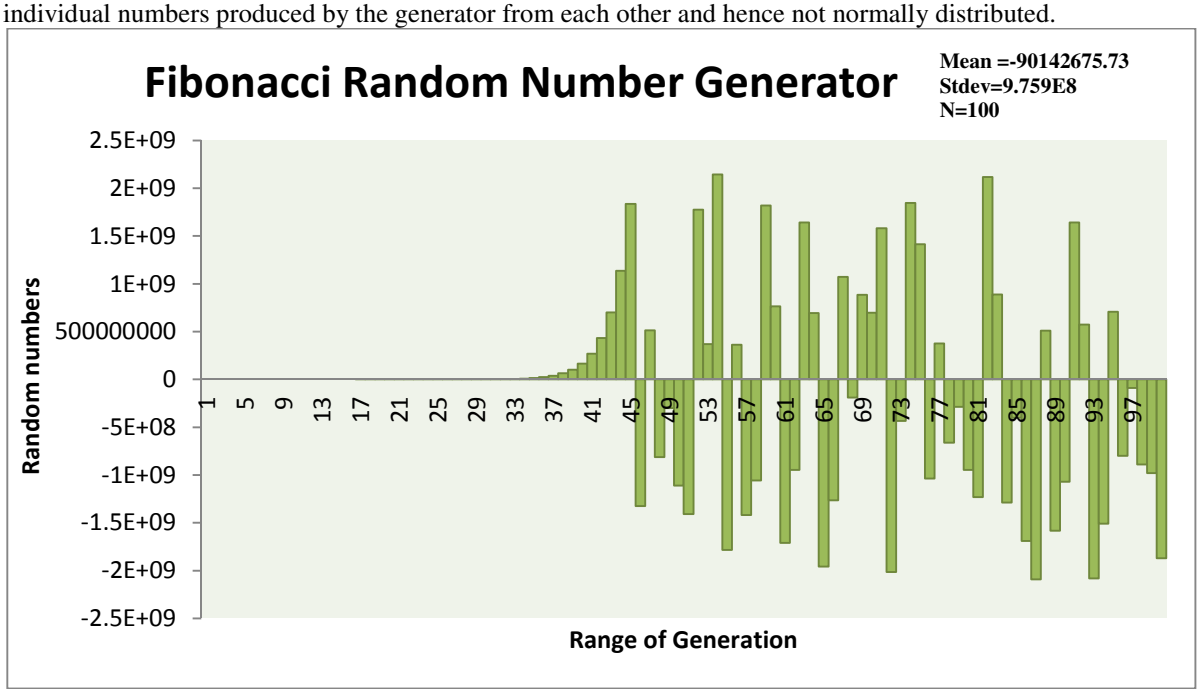

Figure 2: The trend of randomness in Fibonacci Random Number Generator

The researchers compared the two Pseudo Random Number Generators (PRNGs) studied namely, Fibonacci Random Number Generator (FRNG) and Gaussian Random Number Generator (GRNG). The results of the repetition of the two generators were used to draw a line graph as shown in ***Figure 3.*** The figure shows a line graph of frequencies for the two random numbers generators. It was found out that the random number generator with the highest number of repeated numbers is Gaussian Random Number Generator, while Fibonacci Random Number Generator had no repeated numbers in the random numbers. This means that numbers were more likely to be random with Fibonacci Random Number Generator compared with the Gaussian Random Number Generator algorithm as shown in ***Figure 3.***

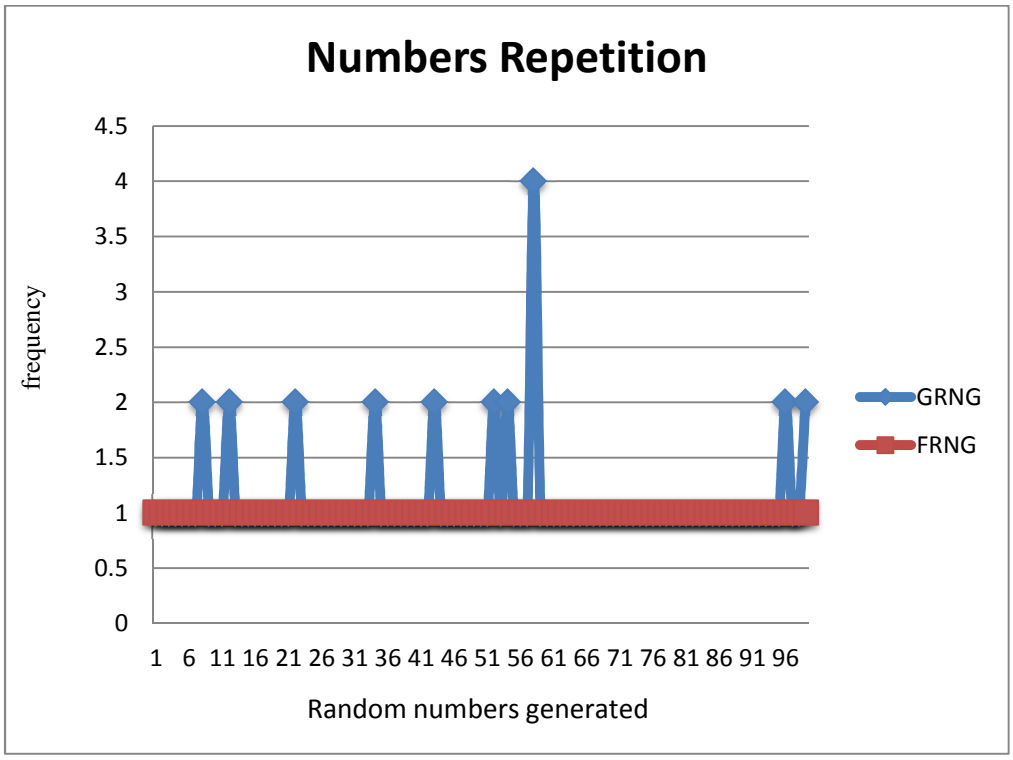

Figure 3: The number of repetitions in Pseudo Random Number Generators

4.3 The Chi-Square Test for Independence of Random Numbers

The analysis of the factors responsible for randomness in a random number generator–showed that factors tend to give many considerations to Fibonacci Random Number Generator than the Gaussian Random Number Generators compared with. This is because the Chi-Square analysis for the independence of numbers in the generators showed that numbers were more independent to each other in the Fibonacci Random Number Generator (Chi-Square Value = 0.000) than the Gaussian Random Number Generators under studied. This is because independence of numbers increases with the decrease in Chi-Square Value and vice versa.

However, the worse generator in terms of independence is Gaussian Random Number Generator (Chi-Square value = 14.400) as shown in ***Table 2***. This is because it produces a higher Chi-Square Value from the test. This also means that numbers in the GRNG were more likely to depend on each other than the Fibonacci Random Number Generator. ***Table 2*** represents the descriptive statistics of the analysis (Standard Deviation, Minimum, and Maximum & Mean). Standard deviation of numbers increases with decreasing normality and vice versa. This means the higher the standard deviation value, the higher the deviation from the normal and vice versa. Therefore Fibonacci Random Number Generator is more deviated from the normal distribution, while Gaussian Random Number Generator produces numbers that obeys the normal distribution far more than the Fibonacci Random Generator as shown in ***Table 2***.

Table 2
The Chi-Square Test Result for Pseudo-Random Number Generator

Test Statistics	**Gaussian Random Number Generator**	**Fibonacci Random Number Generator**
Chi-Square	14.400[a]	.000[b]
Df	87	99
Asymp.Sig.	1.000	1.000
Monte Carlo Sig.	1. 000[a]	1.000[b]
99% Confidence Interval		
Lower Bound	1.000	1.000
Upper Bound	1.000	1.000

a. 88 cells (100.0%) have expected frequencies less than 5. The minimum expected cell frequency is 1.1.
b. . 100 cells (100.0%) have expected frequencies less than 5. The minimum expected cell frequency is 1.0.

4.4 The Kolmogorov-Smirnov Test for Uniformity of random numbers

In Kolmogorov-Smirnov, the more the z-value falls in between the lower and upper bound of the Monte

carlo Significance.(2-tailed) Value, the more uniform the numbers generated by the random number generator is and vice versa. From ***Table 3***, the Gaussian Random Number Generator produces more uniform numbers than its counterpart Fibonacci Random Number Generator. This is because Gaussian Random Number Generator produces z-value that fall closer between the lower and upper bounds of z-value, while Fibonacci Random Number Generator produces z-value that fall outside the range of lower and upper bounds z-value as shown in ***Table 3***. The factors responsible for decision also showed that factors tend to give more considerations to Fibonacci Random Numbers Generator (K-S value = 2.020) than the Gaussian Random Number Generator compared with in this paper. This therefore makes Fibonacci Random Number Generator less uniform than the Gaussian Random Number Generator (K-S value = 0.725) as shown in ***Table 3***.

However, the lesser the uniformity of numbers produced, the more independent numbers are and vice versa. This accounts for why Fibonacci Random Number Generator is more independent of the two generators**.**

Table 3
The Kolmogorov-Smirnov Test for Uniformity of random numbers

Statistics			**Gaussian Random Number Generator**	**Fibonacci Random Number Generator**
N			100	100
Uniform Parameters	Minimum		-121	-2092787285
	Maximum		126	2144908973
Most Extreme Differences	Absolute		0.072	0.202
	Positive		0.072	0.202
	Negative		-0.060	-0.164
Kolmogorov-Smirnov Z			0.725	2.020
Asymp.Sig. (2-tailed)			0.670	0.001
Monte carlo.Sig.(2-tailed)	Sig.		0.640	0.000
	90%Confidence Interval	Lower Bound	0.561	0.000
		Upper Bound	0.719	0.23

5. Discussions

The analysis of factors for uniformity also shows that factors tend to give more considerations to Gaussian Random Number Generator than the Fibonacci generator. This therefore makes Fibonacci random Number Generator less uniform and more secure than Gaussian Random Numbers Generator.

A good PRNG will produce a sequence of numbers that cannot be easily guessed or determined by an adversary. The general assumption is that the opponent knows the algorithm being used. This is usually referred to as Kerckhoff's principle (Yehuda, 2006). This assertion is evidenced in the findings relating to Fibonacci Random Number Generator and the Gaussian Random Number Generator. The Chi-Square analysis for the independence of numbers in the generators showed that numbers were more independent to each other in the Fibonacci Random Number Generator than the other generator under study and therefore less likely to be guessed by an adversary.

According to Andrew et al. (2010), another factor to be considered for a good PRNG is that the series generated should be statistically random. It should be able to hide all patterns and correlations. This can be tested by some statistical tests. A PRNG should have the statistical property that each value over the interval has an equally likely occurrence. In the case of the interval [0,1] each number would appear with the frequency 1/2 in a long run. Each pair of numbers would appear with frequency 1/4, and each triplet with frequency 1/8. This would form a basis of observation that the numbers being generated are unbiased and successive outcomes are uncorrelated (James et al., 2003). Mean should be close to 0.5 and variance 1/12 or 0.08 for uniformly distributed pseudorandom numbers. This is also evidenced more in Gaussian Random Number Generator with lower Variance (square root of standard deviation) compared with that of Fibonacci Random Number Generator.

Again, Andrew et al. (2010) stated that Frequency (Monobits) Test is also another test for efficiency**.** The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether that number of ones and zeros in a sequence are approximately the same as would be

expected for a truly random sequence. The test assesses the closeness of the fraction of ones to ½, that is, the number of ones and zeroes in a sequence should be about the same. Test for Frequency within a Block: the focus of the test is the proportion of zeroes and ones within M-bit blocks. This test also determines whether the frequency of ones in an M-bit block is approximately M/2. In the analysis of the factors responsible for decision to be taken on the two random number generators revealed that Fibonacci Random Number Generator produces random numbers that have less number of ones and zeroes to be about the same than the Gaussian Random Number Generator and hence makes numbers generated by former more independent of each other.

6. Conclusions

The discussions of the findings indicate that Gaussian Random Number Generator Algorithm is more uniform than the Fibonacci Random Number Generator Algorithm. Consequently, Gaussian Random Number Generator algorithm is less secure than the Fibonacci Random Number Generator Algorithm, and therefore makes the Fibonacci Random Number Generator more efficient than the Gaussian Random Number Generator.

It is therefore recommended that the Fibonacci Random Number Generator Algorithm, instead of the Gaussian Random Number Generator, is used for better data security of enterprise software applications in a cryptographic system.

References

Andrew, R., Juan, S., James, N., Miles, S., Elaine, B., & Stefan, L. (2010). *A statistical test suite for random and pseudorandom number generators for cryptographic application*, Revision 1a, USA. Special Publication 800-22 (http://csrc.nist.gov/publications/nistpubs/800-22- rev1a/SP800-22rev1a.pdf. Accessed 2012, September 1).

AuthenTec Embedded Security Solutions (2010). *The importance of true randomness in cryptography*. (http://www.authentec.com/Portals/5/Documents/TRNG%20Whitepaper_91411.pdf. Accessed 2012, January 14).

Biege, T. (2006). A*nalysis of a strong pseudo random number generator*. (http://www.suse.de/~thomas/papers/random-analysis.pdf. Accessed 2012, February).

Christophe, D., & Diethelm, W. (2003). *A note on random number generation*. (http://cran.r project.org/web/packages/ randtoolbox/ vignettes/fullpres.pdf. Accessed 2012, September 7).

Eastlake, D., Schiller, J. & Crocker, S. (2005). *RFC 4086.* (http://www.ietf.org/ rfc/rfc4086.txt. Accessed 2012, June 10).

Fisnik, H. (2011). *Safe internet banking.* (http://www.itknowledge24.com /blog/safe-internet-banking/ Accessed 2012, June 10).

Gary, C. K. (2012). *An overview of cryptography*. (http://www.garykeSecure Socket Layerer. net/ library/crypto.html. Accessed 2012, August 10).

Justin, A. (2012). *Progressing past poor pseudo-randomness*, Cosmos Cluster 4. (http://cosmos.ucdavis.edu/archives/2012/cluster4/Adsuara,%20Justin. pdf. Accessed 2012, August 14).

Mike, H., Paul, K., & Mark, E. M.(2012). *Analysis of Intel's Ivy Bridge digital random number generator.* (http://www.cryptography.com/public /pdf/Intel_TRNG_Report_20120312.pdf. Accessed 2012, August 20.)

Mohammed, A., & Annapurna, P., P. (2012.). *Implementing a secure key issuing scheme for communication in p2p networks.* M.S.Ramiah Institute of Technology,Department of Computer Science and Engineering, Bangalore- 560054. India.

Yehuda, L. (2006). *Introduction to cryptography*, 89-656. (http://u.cs.biu.ac.il/~lindell/89-656/Intro-to-crypto-89-656.pdf. Accessed 2012, August 1).

Yuval, I. (2011). *Theory of cryptography*. 8th Theory of Cryptography Conference, TCC 2011 Providence, RI, USA.