

FPGA implementation of random number generator using LFSR and scrambling algorithm for lightweight cryptography

Asmita Poojari^{1*}, Nagesh H R²

¹ Department of Computer Science and Engineering, NMAMIT Nitte Karkala, Karnataka

² Head of the Department, Department of Information Science and Engineering, A J Institute of Engineering and Technology Kottara, Mangalore

ABSTRACT

The IoT (Internet of Things) is a network of devices that are interconnected and are uniquely addressable, based on common communication protocols and links to perform certain tasks. The recent developments in the wireless communications have increased the need for the IoT-connected devices. The sensors and the sensor nodes used in these networks are low-resource devices, thus increasing the vulnerability and hence becoming a possible target for hackers. The development and deployment of lightweight protection schemes for such low resource devices have also increased. The random number generation or the key generation used in the encryption process is the most important element in protecting these resource-constrained devices, as the security of the entire data depends on the key used. In this paper a novel random number generation using LFSR (Linear Feedback Shift Register) and Scrambling Algorithm for lightweight encryption algorithms is proposed using which the keys for the encryption process can be generated, thus improving the security of data transmitted in the IoT environment. The randomness of the numbers generated by this Random number generator algorithm is tested using pertinent set of statistical tests. These statistical tests analyze the cryptographic properties of the sub keys generated by the key scheduling algorithm, such as confusion, diffusion, independence, and randomness. For the purpose of simulation, the code is written in Verilog and simulated using Xilinx Vivado and the implementation is carried out using Artix-7 FPGA family for analyzing the parameters like Area, power and timing.

Keywords: Internet of things, FPGA, LFSR, Lightweight cryptography, NIST.

OPEN ACCESS


Received: May 6, 2021

Accepted: July 1, 2021

Corresponding Author:

Asmita Poojari

asmitapoojari@nitte.edu.in

 **Copyright:** The Author(s). This is an open access article distributed under the terms of the [Creative Commons Attribution License \(CC BY 4.0\)](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted distribution provided the original author and source are cited.

Publisher:

[Chaoyang University of Technology](https://www.chaoyang.edu.cn/)

ISSN: 1727-2394 (Print)

ISSN: 1727-7841 (Online)

1. INTRODUCTION

In the era of Internet of Things where the data is exchanged between any two tiny devices the power, energy and time are the major parameters the older techniques may become infeasible and cannot be engineered to fit into such resource constrained devices, thus motivating the researchers to design and implement new techniques in generation of such efficient random numbers which can be used as sub keys in the encryption process. The design of a strong key generation scheme decides the strength of the security of any encryption algorithm. The sub keys generated by the key scheduling algorithm should be so strong that the attacker should not be able to acquire any relationship between any sub keys as well as the secret key by Blumenthal and Bellovin (1996). The key scheduling algorithms mainly use linear and nonlinear operations to generate sub keys from the initial secret key and should possess good confusion and diffusion properties. A strong key scheduling algorithm makes the overall encryption process resistant against several threats like the linear-attack, differential-attack, side-channel-attack and many such attacks by Knudsen and Mathiassen (2004);

Suzaki et al. (2013); Biryukov and Nikolić (2011) and have proposed different key schedules that perform different operations such as low diffusion. Some key schedules apply simple operations such as permutation or linear operations on master keys Wu and Zhang (2011). Some use master key directly without any key schedule as proposed in Hong et al. (2006); Guo et al. (2011); May et al. (2002) explained the various desirable properties for the KSA and explained how these properties can be used to strengthen the key schedule of AES. Afzal et al. (2015) explained, statistical analysis of the sub keys generated by the KSA and showed that the avalanche effect is one of the important cryptographic property to ensure security of the entire encryption algorithm. The strength of any key schedule depends on the type of function used i.e. linear or nonlinear function and the operations used. Many key schedules have been designed in various encryption algorithm such as linear circular shift is implemented for key scheduling in the block cipher IDEA by Daemen et al. (1993). This paper presents the problem of large classes weak keys that are been identified and eliminated by slight modification of the key schedule of the IDEA algorithm. The keys are weak in the sense that their use is detected with minimum effort, whereas key schedule of PRESENT proposed by Bogdanov et al. (2007) uses linear permutation. A 64-bit plaintext block was encrypted using 80-bit key. The code was written in VHDL and synthesized using Virtual Silicon (VST) and standard cell library based on UMCL180, 18 μ 1P6M Logic. The design phase of the algorithm comprises of an S-BOX that can be used 16 times instead of having 16 different S-BOXes which eases the serialization of the design. The proposed algorithm encrypts a 64-bit plaintext block using 80-bit key in 32 clock cycles and needs an area of 1570 and consumes a power of 5 μ W. The authors Harmouch and El Kouch (2019) used the concept of chaos in the key schedule algorithm and thus a new key scheduling algorithm called CKSA based on the logic maps has been developed. The proposed algorithm is a one-way function and ensures a good diffusion and confusion and also provides a good avalanche effect. The size of the sub-keys is variable and thus can be used by many ciphers. It also has a good resistance against differential and linear attacks. A strong linear correlation between the sub-keys ensures a randomization of high degree. The authors Paje et al. (2019) used a multidimensional key algorithm for RC6. The authors proposed a modified RC6 algorithm and key sizes of different lengths like 1024/1280/1792/2048/2861 bits are used so as to provide a high degree of security. The longer key length implies that the time required to break the key would result in a longer time. Thus increasing in the length of register, resulted in improved throughput and speed. Avanzi et al. (2016) proposed some general strategies to

construct a key schedule is introduced. However, in all of these studies, the cryptographic strength of the key schedule algorithms was not evaluated using any statistical method. Any key scheduling algorithm should be tested on properties such as confusion, diffusion, randomness of the sub keys to prove the security strength of KSA and the encryption algorithm. In this paper a novel key scheduling algorithm is proposed and also its strength is evaluated based on the above properties using a required set of statistical tests using the NIST test suite. From the studies, it has been seen that the statistical tests may not be sufficient to assert the cryptographic strength of the cipher algorithms, they provide essential requirements for a strong cryptographic algorithm. The algorithm that passes all the statistical tests may not thwart the possible attacks, but the algorithm that fails the required statistical tests would not even thwart the basic attacks on the ciphers by Simion (2015). Ukrop et al. (2016) in his research paper analyzed the randomness of multiple-authenticated encryption schemes. The outputs were assessed using 168 different schemes and 3 different settings and implemented in four different tools. EACirc was defeated by all the statistical batteries of tests hence was the least suitable for given task, while the tests like NIST STS (2010), Dieharder by Robert G. Brown (2004) and TestU01 by Pierre L'Ecuyer et al. (2007) produced better results while Raviyoyla test in EACirc performed better than all other tests.

2. THE PROPOSED RANDOM NUMBER GENERATION USING LFSR AND SCRAMBLING ALGORITHM

The security of any cryptographic method depends mainly on the keys used in the encryption process and hence in turn depends on the key generation algorithm.

The proposed novel random number generation algorithm is a fusion of three different implementations, the random number generation used for SIT algorithm developed n by Usman et al. (2017) inspired from Khazad block cipher proposed by Barreto and Rijmen (2000). The key generation scheme is based on Modified Fibonacci and Scrambling Factor Amiruddin et al. (2019) and the LFSR, so as to provide more randomness, quality and lesser area. The Khazad cipher is based on wide trail strategy that comprises of linear and non-linear transformations ensuring the complexity in the dependence of output bits and input bits Daemen et al. (1995). The algorithm is said to have a linear algorithmic complexity of $O(n)$, the algorithm has a lightweight operation and hence can save the computing time making it useful in key generation function for a lightweight scheme.

In the proposed Random number generation using LFSR and Scrambling Algorithm, the round keys to be used in the various rounds of the encryption phase of the lightweight cryptographic system are generated using a novel method as shown in Fig. 1. The key length defined by the initial user is made large enough so as to provide security for an exhaustive search attack, thus it may be infeasible for an adversary to perform an exhaustive key searching attack. For this the input is 64 bits input key and output is generation of 5 keys (round keys) which will be used in each of the rounds of the encryption scheme.

The steps are as follows:

1. A 64-bit user defined initial-seed is the input to the proposed RNG scheme (key scheduling scheme).
2. The 64-bit input key is partitioned into blocks of 4-bits each. (p1, p2, p3,...,p16) .
3. The four 4-bits block are concatenated into four blocks of 16-bits each.
(say pp1 = [p1||p5||p9||p13], pp2 = [p2||p6||p10||p14], pp3 = [p3||p7||p11||p15], pp4 = [p4||p8||p12||p16]).
4. Next the 16-bit data generated from the above step is given as the input to the LFSR which outputs a 16-bit random number.
5. Next the 16-bit data generated from 3rd step and the pseudorandom number generated from the linear feedback shift register are XORed which outputs, Q1, Q2, Q3, Q4 which are fed to the Fibonacci scrambling algorithm [19]., to derive the keys for the encryption process, the steps are
 - a. The key $K(1) = \text{mod}(Q1 + Q2, n)$
 - b. Similarly, the key $K(2) = \text{mod}(Q3 + Q4, n)$
 - c. The remaining keys are determined as $(n = 4)$ for $i = 3$ to r do

$$K(i) = \text{mod}(K(i-1) + K(i-2), n)$$

end for

d. end

thus keys $K(1), \dots, K(5)$ are obtained which may be used as round keys for the encryption process of a cryptographic algorithm.

3. RESULTS AND DISCUSSION

The proposed FPGA based random number generator is simulated using Xilinx Vivado Design Suite and implemented in Nexys-4 DDR Artix-7 FPGA family. The randomness and statistical test was evaluated using the NIST800-22 statistical tests by Andrew Rukhin et al.(2010), Giga bit streams were generated from the proposed RNG with $P \geq 0.01$ (the level of significance).

3.1 Evaluation Based on NIST Statistical Test Suites

The keys generated by the key-scheduling algorithm have to be tested for its randomness. A PRNG should exhibit following characteristic

1. Uniformity: For the generated random or pseudorandom sequence of bits, the probability (P) of occurrence of a zero or one is equally likely, (i. e. $P = \frac{1}{2}$)
2. Scalability: The randomness tests applied to a sequence can be applied to the extracted subsequences. Thus, the subsequence generated should also pass all the randomness tests.
3. Consistency: the RNG must produce consistent results across initial seeds. Based on the output produced from

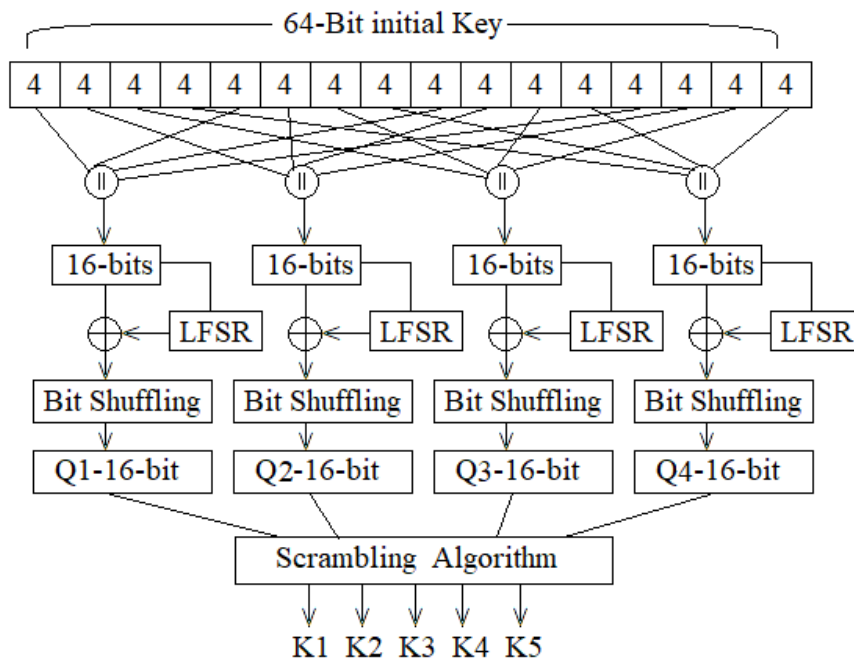


Fig. 1. Proposed Random number generation using LFSR

a given output, a single seed should be inadequate to test a random number generator.

US National Institute of Science and Technology (NIST) developed the statistical test suite validating the random number generators for cryptographic applications and is published as NIST Special Publication 800-22 Revision 1a by Rukhin et al. (2010). The tests are used to determine the quality of the random number generator. This test suite comprises of fifteen different tests and explained below.

1. Frequency (Monobit) Test: In this test the number of 0's and 1's in the given stream are compared. The proportion of ones should be same as number of zeros in the sequence.
2. Block-Frequency Test: for a given block of M-bit size this tests measures the number of ones and zeros. The frequency of ones or zeros should be approximately $M/2$ for randomness.
3. Runs-test: This test computes the occurrences of uninterrupted sequences or runs of ones or zeros for a given sequence. For a random sequence, this test figures out whether the number of runs of zeros and ones of different lengths is as required.
4. Test for the Longest-Run of Ones in a Block: This test computes the longest run of ones for a given M-bit block.
5. Binary-Matrix-Rank Test: This test determines the linear dependence of fixed length substrings from the original sequence. Here the entire sequence is divided into stream of rows and columns of matrices and the rank of disjoint sub matrices of the entire sequence is tested.
6. Discrete-Fourier-Transform (Spectral) Test: The peak heights of the sequence in the DFT (Discrete Fourier Transform) is observed. The aim is to determine the repetitive patterns in the given stream. If the number of peaks exceeds the given threshold, then the test fails.
7. Non-Overlapping-Template Matching Test: The test determines the bit stream for number of occurrences in a distinct non-periodic pattern. This test searches for a m-bit pattern in a m-bit window. The window is reset if the sequence is found to the next bit and the search starts again else window moves by one position.
8. Overlapping-Template Matching Test: The test finds the number of occurrences of the specific target strings.
9. Maurer's "Universal-Statistical" Test: In this test, in a data stream it computes the number of bits between matching patterns.
10. Linear-Complexity Test: This test determines the complexity of the generated sequence, length of a LFSR to generate the required bit-stream.
11. Serial-Test: The frequency of overlapping of m-bit sequence in a $2m$ bit-stream is calculated in this test.
12. Approximate-Entropy Test: This test measures the frequency of overlapping of an m-bit patterns across the overall sequence.
13. Cumulative-Sums (Cusum) Test: This check determines the maximal excursion from a random walk from 0 using the values $[-1, +1]$.
14. Random-Excursions Test: This test calculates the number of cycles in a cumulative sum random walk of K visits. The cumulative sum random walk is obtained from the sequence $[0, 1]$ if a "0" is traverse then $[-1]$ and if a 1 is traversed it $[+1]$. Thus, the test determines 9 states $[-4, -3, -2, -1, 0, 1, 2, 3, 4]$.
15. Random-Excursions-Variant Test: This test computes the number of occurrences of the particular state in a cumulative sum random walk and checks the deviations from number of occurrences to different states in a random walk. It uses a series of 18 tests and convulsions $\{-9, -8, -1, +1, +2, \dots, +8\}$.

Table 1 shows the results of the P value for the NIST randomness tests and it is found that the P-values obtained are greater than 0.01 hence the generated bits are random in nature.

3.2 FPGA Implementation

The proposed LFSR based RNG is implemented in ARTIX-7, Nexys-4 DDR FPGA. The optimized structure of the proposed LFSR based RNG results in lesser area and power.

The proposed key generation scheme blocks have been modelled using Verilog HDL, Xilinx Vivado is used to obtain the simulation and synthesis and verified on the Nexys 4 Artix-7 FPGA and Oasys-RTL Tool (45-nm technology). The RTL Schematic is shown in Fig. 4. The results design summary is obtained in Table 2 shows Slice LUTS, Registers and IOB's and timing/ critical Path delay (logic delay + net delay) and total on chip power (Dynamic+Static) in terms of Watts. The proposed scheme has power consumption reduced by 32% than SIMON scheme.

Fig. 2 shows the RTL schematic of LFSR Key generation scheme implemented in TEA/XTEA algorithm.

Fig. 3 shows the RTL Schematic of the Key generation scheme used in SIMON Cipher.

Table 1. NIST test results of proposed Key generation scheme

Test	P-value	Result
Frequency	0.739918	Passed
Block-Frequency	0.179120	Passed
Cumulative-Sums (forward)	0.534146	Passed
Cumulative-Sums (inverse)	0.739918	Passed
Runs	0.350485	Passed
Longest-Run	0.179120	Passed
Rank	0.035174	Passed
FFT	0.213309	Passed
Non-Overlapping-Template	0.122325	Passed
Overlapping-Template	0.430102	Passed
Universal	0.122325	Passed
Approximate-Entropy	0.350485	Passed
Random-Excursions	0.911413	Passed
Random-Excursions-Variant	0.534146	Passed
Serial	0.035174	Passed
Linear-Complexity	0.739918	Passed

Table 2. FPGA implementation of key generation schemes

Key Generation Scheme	Nexys 4 Artix-7 FPGA					Oasys RTL Tool - 45nm Technology	
	Timing/ Path delay (ns)	Area Slice Registers (15850)	Area Slice LUT's (63400)	IOB's (210)	Power (W)	Area (μm ²)	Power (μW)
32-bit LFSR	4.065	19	1	64	31.195	231	40.411373
SIMON	62.718	104	34	160	235.614	142	212.511063
Proposed RNG-LFSR	9.036	92	00	144	76.53	1213	1196.31

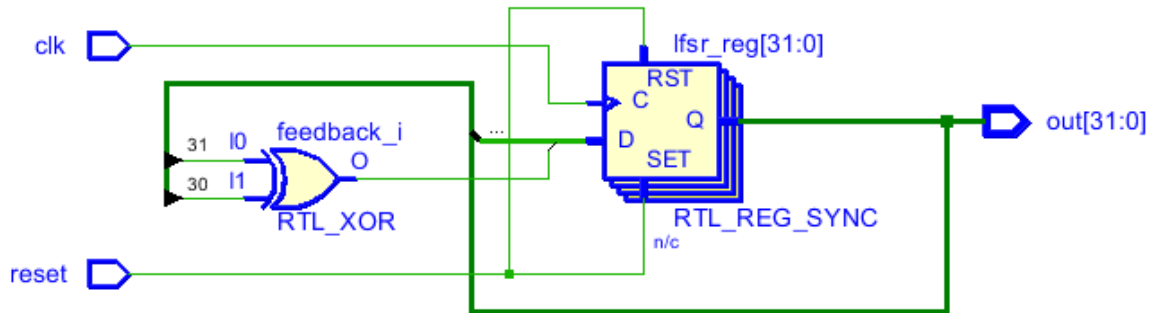


Fig. 2. RTL Schematic of Linear feedback shift register

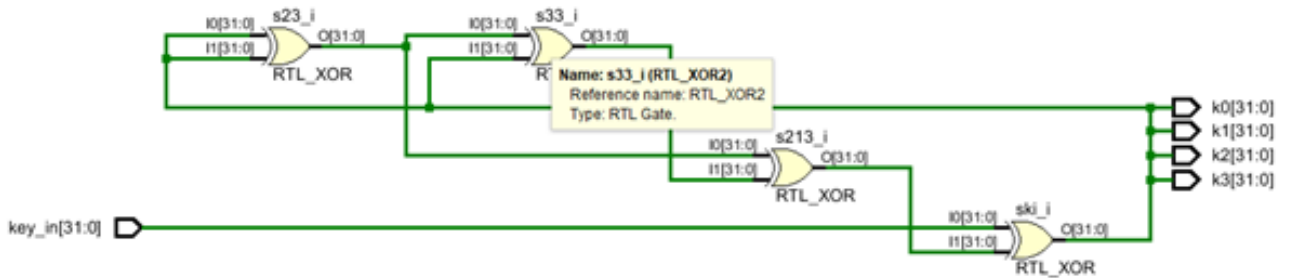


Fig. 3. RTL Schematic of SIMON Key generation scheme

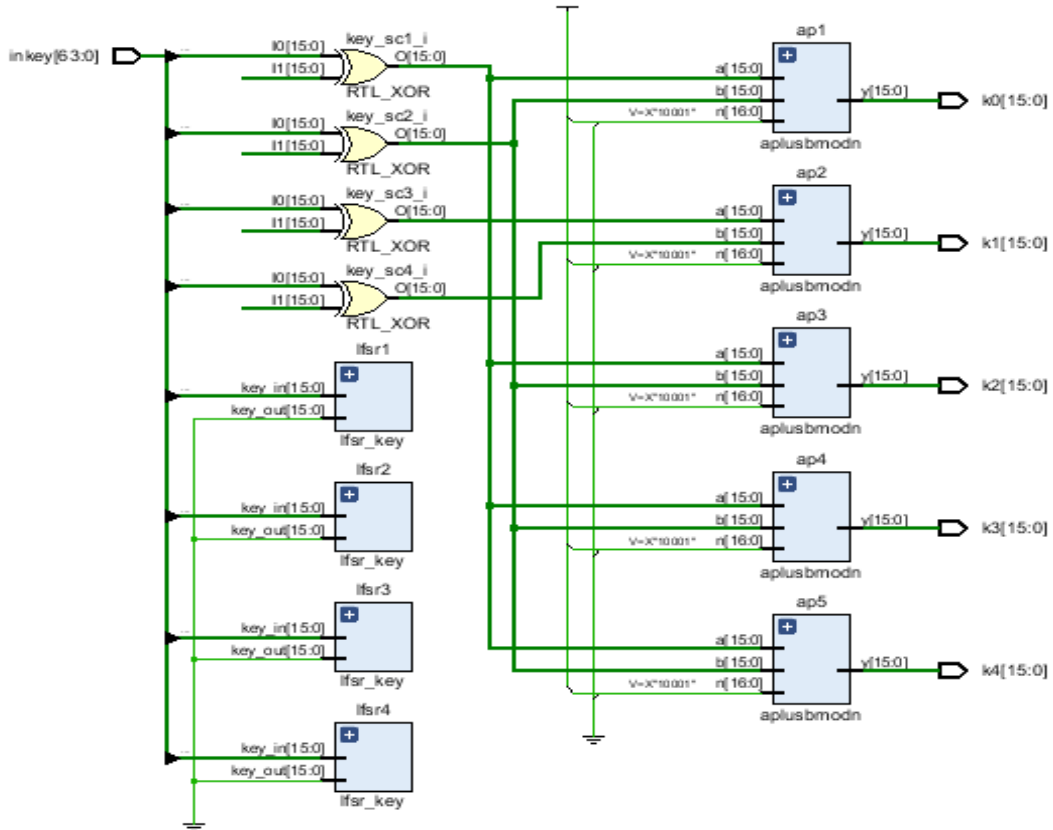


Fig. 4. RTL schematic of the key generation for proposed RNG-LFSR

Fig. 5 shows the comparison chart. It is seen that when implemented in Xilinx Vivado the LUT and slices are lesser than SIMON key generation scheme while its implementation in Oasys tool has greater area and power than the state of the art implementations which is well within the lightweight requirements as per the NISTIR report.

The comparative analysis in terms of AREA (LUT + IOBs), Bit-rate for the implemented ciphers with the state of the art implementations is shown in Table 3.

Fig. 6 shows the comparison of proposed key generation schemes with the state of the art implementations. It can be seen that the proposed RNG using LFSR encryption has better results.

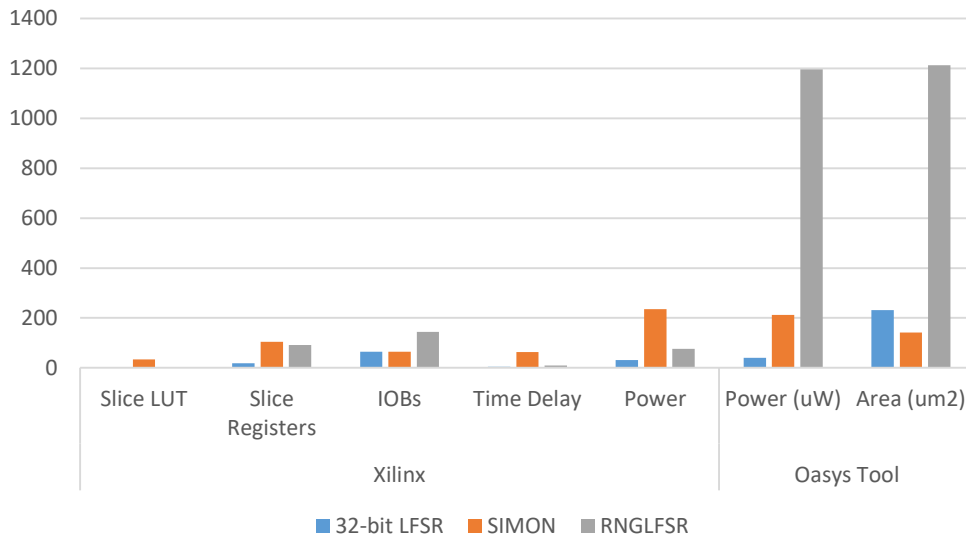


Fig. 5. Comparison of key generation schemes

Table 3. Comparative analysis of the proposed key generation scheme

RNG	Year	Area(LUT+IOBs)	Bit-rate Mb/s	BPA Mb/s/ (LUT+Reg)	FPGA
32-bit LFSR	2020	32+34	26.43	0.408	Artix--7
SIMON	2020	32+160	27.3	0.142	Artix--7
Proposed RNG-LFSR	2021	92 +144	110.668	0.4689	Artix--7
Gupta et al.	2019	581+16	1600	2.68	Artix--7
Wu and Li	2017	298	150	0.5033	Cyclone-IV
Choi et al.	2017	21+15	12.5	0.347	Cyclone-IV

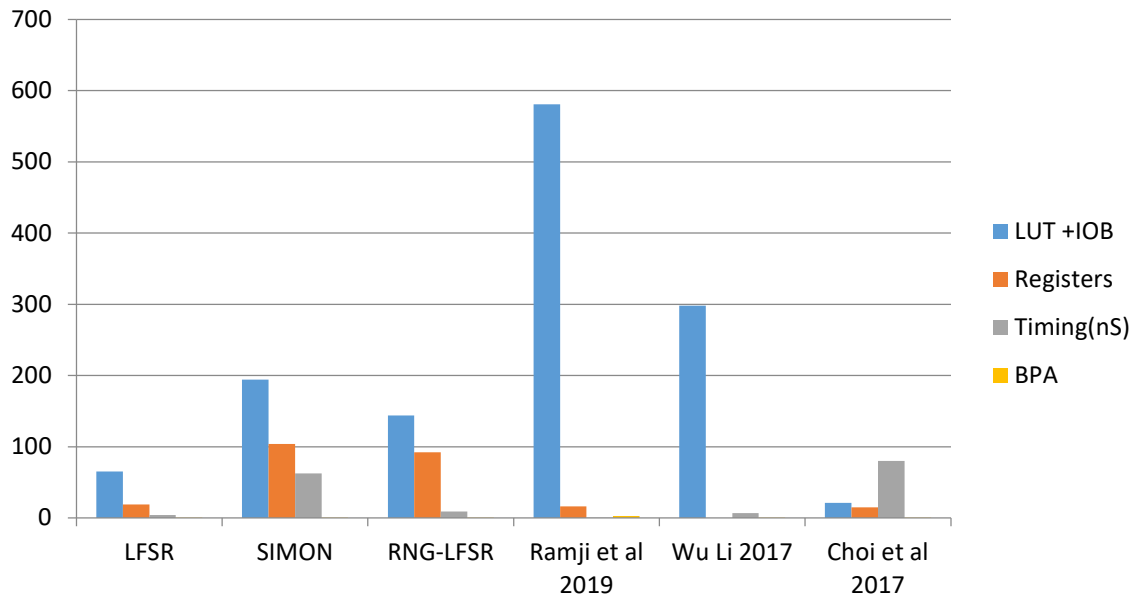


Fig. 6. Comparison of key generation scheme with the proposed key generation scheme

4. CONCLUSION

In this paper a novel RNG algorithm using LFSR and Scrambling Algorithm is proposed. The random numbers obtained can be used as round keys of an encryption process in a cryptographic algorithm especially in lightweight cryptographic platforms. The scheme generates keys which are more random in nature as seen from the implemented results. The NIST statistical randomness tests was conducted and implementation results were analyzed and found that this RNG scheme is more secure and resistance against attacks, it can be employed in an encryption process for lightweight ciphers. The proposed scheme is more efficient than the other implemented algorithms. The future work is to implement and analyze it for sensitive applications like healthcare.

ACKNOWLEDGMENT

The authors would like to thank the department of Computer Science and Engineering, N M A M Institute of Technology Karkala and Visvesvaraya Technological university (VTU-RR), Belagavi for the support for carrying out the research work.

REFERENCES

Afzal, S., Waqas, U., Mubeen, M.A., Yousaf, M. 2015. Statistical analysis of key schedule algorithms of different block ciphers, *Science International*, 27.

Amiruddin, A., Ratna, A.A.P., Sari, R. 2019. Construction and analysis of key generation algorithms based on modified Fibonacci and scrambling factors for privacy preservation. *International Journal of Network Security*, 21, 250–258.

Avanzi, R. 2016. A salad of block ciphers-the state of the art in block ciphers and their analysis (<http://eprint.iacr.org/2016/1171.pdf>, 2016).

Bakiri, M., Guyeux, C., Couchot, J.F., Oudjida, A.K. 2018. Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses. *Computer Science Review*, Elsevier, 27, 135-153. hal-02182827, 4, 5–13.

Barreto, P., Rijmen, V. 2000. The khazad legacy-level block cipher, *Primitive submitted to NESSIE*, 97.

Biryukov, A., Nikolić, I. 2011. Search for related-key differential characteristics in DES-like ciphers. In *Fast Software Encryption*, 6733, 18–34.

- Blumenthal, U., Bellovin, S.M. 1996. A better key schedule for DES-like ciphers, in Proceedings of the Pragocrypt, Prague, Czech Republic.
- Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C. 2007. PRESENT: an ultra-lightweight block cipher, in Cryptographic Hardware and Embedded Systems—CHES 2007, 450–466, Springer, Berlin, Heidelberg.
- Choi, P., Lee, M.-K., Kim, D.K. 2017. Fast compact true random number generator based on multiple sampling. *Electronics Letters*, 53, 841–843.
- Cusick, T.W., Stanica, P. 2017. Chapter 2 - Fourier analysis of Boolean functions, editor(s): Thomas W. Cusick, Pantelimon Stanica, cryptographic Boolean functions and applications (Second Edition), Academic Press, 7–29, ISBN 9780128111291, <https://doi.org/10.1016/B978-0-12-811129-1.00002-X>.
- Daemen, J. 1995. Cipher and hash function design strategies based on linear and differential cryptanalysis, Ph.D. dissertation, Doctoral Dissertation, KU Leuven.
- Daemen, J., Rene, G., Joos, V. 1993. Weak keys for IDEA, Annual International Cryptology Conference, 224–231, Springer, Berlin, Heidelberg.
- Guo, J., Peyrin, T., Poschmann, A., Robshaw, M. 2011. The LED block cipher. In CHES 2011, 6917, 326–341.
- Gupta, R., Pandey, A., Baghel, R.K. 2019. FPGA implementation of chaos-based high-speed true random number generator. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*. 32, e2604. <https://doi.org/10.1002/jnm.2604>
- Harmouch, Y., El Kouch, R. 2019. The benefit of using chaos in key schedule algorithm, *Journal of Information Security and Applications*, 45, 143–155.
- Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S. 2006. HIGHT: A new block cipher suitable for low-resource device. In CHES, 4249, 46–59. Springer.
- Jaya Sudha, K., Jaya Rani, G., Mirza Shafi Sahahsavar, 2015. Generation of uniform random numbers using look up table as shift register, *International Journal of Science, Engineering and Technology Research (IJSETR)*, 4.
- Justin, R., Mathew, B.K., Abe, S. 2016. FPGA implementation of high quality random number generator using LUT based shift registers, *Procedia Technology*, 24, 1155–1162, ISSN 2212-0173. <https://doi.org/10.1016/j.protcy.2016.05.069>.
- Kim, C.H. 2011. Improved differential fault analysis on AES key schedule, *IEEE Transactions on Information Forensics and Security*, 7, 41–50.
- Knudsen, L., Leander, G., Poschmann, A., Matthew, R.J.B. 2010. PRINTcipher: A block cipher for IC-printing. In CHES 2010, 6225, 16–32.
- Knudsen, L.R., Mathiassen, J.E. 2004. On the role of key schedules in attacks on iterated ciphers, in European Symposium on Research in Computer Security, 322–334, Springer, Berlin, Heidelberg.
- Kumar, V.G.K., Rai, C.S. 2020. FPGA implementation of simple encryption scheme for resource-constrained devices, *International Journal of Advanced Trends in Computer Science and Engineering*, 9. <https://doi.org/10.30534/ijatcse/2020/213942020>.
- Kumar, V.G.K., Rai, C.S. 2021. Efficient implementation of cryptographic arithmetic primitives using reversible logic and Vedic mathematics. *Journal of The Institution of Engineers (India): Series B* 102, 59–74. <https://doi.org/10.1007/s40031-020-00518-w>.
- Matsumoto, M., Kurita, Y. 1992. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation*, 2, 179–194. DOI: <https://doi.org/10.1145/146382.146383>.
- May, L., Henricksen, M., Millan, W., Carter, G., Dawson, E. 2002. Strengthening the key schedule of the AES, in *Information Security and Privacy*, 226–240, Springer, Berlin Heidelberg.
- McKay, K.A., Bassham, L., Turan, M.S., Mouha, N. 2016. DRAFT NISTIR 8114: Report on lightweight cryptography, National Institute of Standards and Technology Internal Report 8114.
- Mitchell, R.L., Stone, C.R. 1977. Table-lookup methods for generating arbitrary random numbers, in *IEEE Transactions on Computers*, C-26, 1006–1008, doi: 10.1109/TC.1977.1674735.
- NIST, 2010. A statistical test suite for random and pseudorandom number generators for cryptographic applications. <http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf>.
- Paje, R.E.J., Sison, A.M., Medina, R.P. 2019. Multidimensional key RC6 algorithm, in Proceedings of the 3rd International Conference on Cryptography, Security and Privacy—ICCSP'19, 33–38, Kuala Lumpur, Malaysia.
- Seok, B., Lee, C. 2019. Fast implementations of ARX-based lightweight block ciphers (SPARX, CHAM) on 32-bit processor. *International Journal of Distributed Sensor Networks*. <https://doi.org/10.1177/1550147719874180>.
- Simion, E. 2015. The relevance of statistical tests in cryptography, *IEEE Security & Privacy*, 13, 66–70.
- Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E. 2013. TWINE: A lightweight block cipher for multiple platforms. In *Selected Areas in Cryptography*, 7707, 339–354.
- Sys, M., Klinec, D., Kubicek, K., Svenda, P. 2017. BoolTest: the fast randomness testing strategy based on boolean functions with application to DES, 3-DES, MD5, MD6, and SHA-256, in *International Conference on E-Business and Telecommunications*, 123–149, Springer, Cham, Switzerland.
- Tezuka, S. 1995. Linear congruential generators. In: *Uniform Random Numbers. The Springer International Series in Engineering and Computer Science (Discrete Event Dynamic Systems)*, 315. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-2317-8_3.

- Thomas, D.B., Luk, W. 2013. The LUT-SR family of uniform random number generators for FPGA architectures, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21, 761–770.
- Ukrop, M. 2016. Randomness analysis in authenticated encryption systems, Ph.D. thesis, Masarykova univerzita, Fakultainformatiky, Brno, Czechia.
- Usman, M., Ahmed, I., Aslam, M.I., Khan, S., Shah, U.A. 2017. SIT: A lightweight encryption algorithm for secure internet of things, *International Journal of Advanced Computer Science and Applications*, 8.
- Wetzels, J., Bokslag, W. 2015. Simple SIMON: FPGA implementations of the SIMON 64/128 block cipher. *Cryptography Engineering Kerckhoffs Institute*. 1, 1–20.
- Wu, W., Zhang, L. 2011. LBlock: A lightweight block cipher. *In Applied Cryptography and Network Security*, 6715, 327–344.
- Wu, X., Li, S. 2017. A new digital true random number generator based on delay chain feedback loop, *IEEE conference 978-1-4673-6853-7/17/\$31.00*