A New Lightweight Hybrid Cryptographic Algorithm for The Internet of Things

Mouza Bani Shemaili, Chan Yeob Yeun, Khalid Mubarak, Mohamed Jamal Zemerly
Electrical and Computer Engineering Department, Khalifa University, PO Box 573, Sharjah, UAE
mouza@kustar.ac.ae, cyeun@kustar.ac.ae, mubarak@kustar.ac.ae, jamal@kustar.ac.ae

Abstract—**The internet world is continuous revolutions from the World Wide Web and the mobile Internet to the Internet of Things (IoT). IoT is the new world for connecting the object space in the real world with the virtual space in the computer world. Radio Frequency IDentification (RFID) and Wireless Sensors (WS) are technologies that can be used to create the IoT world. This increases the needs of these technologies in our daily life. However, there is a main drawback within these technologies which is to provide the low computation devices. Such a drawback limits the capabilities of RFID and WS. These technologies can hold a very sensitive data that may be related to the physical world such as the names or places of people. Thus, exposing this data can lead to security breach issues and researchers tried to come up with different security solutions with low computation. However, adding security to such low computation devices is a great challenge as they need a suitable lightweight cipher that is able to fit their properties. This paper addresses some of the available lightweight ciphers, compares between them and comes up with a new algorithm that can fit low computation devices.**

Keywords-RFID, WSN, PNG, Linear Feedback Shift Register, Feedback with Carry Shift Register, IoT.

I. INTRODUCTION

Internet of things (IoT) is the new era that allows a connection between the virtual and the real world in such a way that it links every object in the real world with the virtual world. IoT world builds an interaction between all physical objects such as people, cars, tables, chairs, etc., in such a way that one can define each of them in the virtual world and enable connectivity between all of them anywhere and anytime. IoT can be applied in many applications such as aerospace, supply chain, telecommunications [1, 2]. There are some technologies that enable that connection such as Radio Frequency IDentification (RFID) and Wireless Sensor Network (WSN). Merging both of these technologies can make the revolutions of the IoT real world.

However, these technologies could be incapable of providing a security to their applications. Due to the low computation capabilities of the RFID and WSN there is a need to build a lightweight security cipher that can fit these devices. Designing a cipher with lightweight properties is always challenging that the designer needs to cope with the trade-off between achieving robust security with low cost and enhanced performance [3]. Achieving a cipher design that includes all these three goals is so difficult.

Cipher algorithms can be either symmetric or asymmetric. Symmetric ciphers are based on a combination of mathematics and cryptographic principles that usually use simple primitives such as permutation, substitution, rotation, bit-wise XOR, shift, etc. On the other hand, asymmetric cipher are based on hard to solve mathematics problems and usually depend on hard primitives such as modular addition, subtraction, modular multiplication, variable length rotations, and so on. Therefore, symmetric cipher is easier to implement in low computation devices [4].

Symmetric cipher is divided into two types: one is block cipher and the other is a stream cipher. The stream cipher is more suited to low computation devices since it uses small memory size and runs faster than the block cipher. Most of the available stream ciphers tried to emulate the idea of the One Time Pad (OTP) [5]. The OTP idea depends on having a random key that is the same size as the plaintext then XORs both of them to generate the ciphertext. The strength of the OTP comes from the randomness of the key. Designing such a cipher that is able to produce a real random key is considered as a difficult problem. The available stream ciphers are either about generating a True Random Generator (TRG) or Pseudo Number Generators (PNG). Implementing the TRG is not that practical solution since it depends on the physical phenomena such as thermal noise. However, the PNG depends on mathematical solutions. Thus it is a more practical solution. Stream cipher can be either software oriented (SWO) or hardware oriented (HWO) security solution. This paper will focus on the hardware oriented solution for the sake of the low computation devices. Most of the hardware oriented solutions rely on using Shift Registers (SR). SR solutions are easy to implement in hardware and can be used to create PNG that can generate a secure cryptographic key. PNG must hold some criteria as follows:

1) The length of the period. The generator idea depends on the infinite state machine model that regularly clocks the machine resulting in changing the state of the machine and producing an output with a specific state. Since, the generator depends on an infinite state then the output of the generator will be repeated at

978-1-908320-08/7/$25.00©2012 IEEE

some states. Thus, the length of the period should be long enough otherwise the key used for encryption will be used more than once that may enable the attacker to easily break the generator. The length of the period depends on the type of application and its life time.

2) The output must be independent and have equal distribution. That the output stream must have nearly equal statistics between zeros and ones with the probability of the next bit to be zero or one is half.
3) The output stream cipher needs to resist the correlation attack. Correlation attack [6] is the ability to determine the key from a given plaintext and its corresponding ciphertext by exploiting the statistical weakness coming from choosing a poor Boolean function.
4) Easy implementation.

SR is divided into two types: either LFSR solution or FCSR solution. Most of the famous solutions that are able to generate PNG could use Linear Feedback Shift Register (LFSR) and Feedback with Carry Shift Register (FCSR) [7]. However, both solutions are vulnerable and able to be analyzed by Jonckheere and Ma [8]. Thus this paper addresses the related work to the LFSR and FCSR with three Random Number Generators (RNG) algorithms that used either LFSR or FCSR which are Alternating Step (ASG), Shrinking Generator (SG), and Self-Shrinking Generator (SSG). Then choice the most suitable RNG will be identified to be used as the main building block for the cipher design.

The paper is organized as follow: Section II explains the security issues within the stream ciphers. Section III illustrates related work to the LFSR and FCSR algorithms. Section IV provides cascade family of the SG and SSG with the LFSR and FCSR. Section V shows the implementation result of 7 stream ciphers of the SG family. Section VI explains the proposed stream cipher. Section VII illustrates the security analysis of the new stream cipher. Finally, Section VIII concludes this paper.

II. STREAM CIPHERS ATTACKS TYPES

Attacks on stream ciphers usually are ciphertext only attack, known as plaintext attack, distinguishing form truly random attack and Time Memory Trade-Off (TMTO).

- Known plaintext attack: the attacker has access to a plaintext and its corresponding ciphertext. Then, the attacker tries to construct the key by finding the correlation between the plaintext and the ciphertext [9].
- Ciphertext only attack: the attacker has access to the ciphertext only without its corresponding plaintext. Then, an attacker uses the frequency analysis to break the cipher [10].
- Distinguishing form truly attacks: can be launched if there are non-random properties within the output stream. An attacker can use this flaw to find a relation between different ciphertexts and then break the cipher [11].
- TMTO attack pre-computes a large amount of data in order to lower the computation complexity of the cipher [12].

Our motivation is to provide a cipher that can overcome these types of attacks.

III. RELATED WORKS

Linear Feedback Shift Register (LFSR) is a shift register whose input bit is a linear function of its previous state. LFSR is used to generate Pseudo Random Numbers (PRN). The LFSR has two main parts which are a shift register and a feedback function. A shift register identifying function is shifting its contents into the adjacent positions within the register or, out of the register in the case of the position at the end. The position on the other end is left empty unless some new content is shifted into the register. The contents of a shift register are usually thought of as being binary, that is, ones and zeroes. For the shift register, there are three main parts. These are shift direction, output and input. Shift direction is the direction that the shift register will shift its contents. A shift register can shift its contents in either direction. The output refers to the bits that are shifted out of the register. The input refers to the bits that enter the register after the bit on the left end of the shift register and it is left empty. In the LFSR, the bits contained in selected positions in the shift register are combined in some sort of a function and the result is fed back into the register's input bit. The feedback function in an LFSR has several names: XOR, odd parity, sum modulo 2. The period of the LFSR is 2^n-1 where n is the number of stages in the shift register. Also, it is proved that LFSR can provide uniform distribution for the output stream [13, 14].

Feedback with Carry Shift Register (FCSR) was proposed by M. Goresky and A. Klapper in 1993 [15]. The idea of the FCSR is similar to the LFSR since both have a shift register and a feedback function. However, the idea is different in the way to get the output which is instead of XORing the bits in the tap sequences FCSR has a carry register that adds the bits in the shift register together with its contents. Thus the output of the FCSR is the summation of the carry register content and shift register modulo 2. The output of the FCSR is used to fill the content of the carry register also, by dividing it by 2.

Both of the LFSR and FCSR use a mathematical model that is similar in theory based analysis. LFSR is considered as rational series of ring GF (2) while FCSR is based on the rational 2-adic numbers [16]. However the output of the LFSR is linear that is easy to break while the output of the FCSR is quadratic. The quadratic output of the FCSR is more resistant to the algebraic attack and the correlation attacks than the LFSR output since it is updated nonlinearly. On the other hand, FCSR proves that it is not that secure if implemented alone [17] and implementing the FCSR costs more than the LFSR as shown in Table 1. Therefore, the implementation of the FCSR also, needs to have an additional algorithm that is able to prevent predictable output.

Table 1 shows the implementation comparison of the LFSR and FCSR with the same parameters such as feedback polynomial of degree n and hamming weight of k. FCSR assumes with full addition as shown in paper [18].

TABLE 1. COMPARISON BETWEEN LFSR AND FCSR

	LFSR	FCSR
Shift Register	N	N
Carry Register	0	k-1
Xor gate	k-1	4(k-1)
And gate	0	k-1

In order to improve the unpredictability and randomness of the LFSR a number of algorithms have been used such as using irregular clock. In the irregular clock LFSR algorithms the output sequence is not produced each time that they are clocked, however, the output is controlled by irregular clock interval to provide non-linearity to the output stream. Examples of such algorithms are Alternating Step, Shrinking Generator, Self Shrinking Generator. The Alternating Step Generator (ASG) [19] consists of three LFSRs. One of the LFSRs is called the clock controller that controls the clock of the other two LFSRs. let S denotes the clock controlled LFSR and the others by LFSR A and LFSR B. the output of the LFSR A and B is a0, a1, ...,an and b0, b1, ..., bn respectively. Where the output stream sequence is z0, z1, ..., zn which is obtained by the following steps:

- Clock S if the S output is 1 then LFSR A is clocked whereas if S output is 0 then LFSR B is clocked at step i.
- The output Z is LFSR A and LFSR B modulo 2 at step i.

The SG is a key-stream generator which was published in [20]. The SG consists of two LFSRs with a clock in order to control the output of the data. One of the LFSR is A (data) and the other one is S (sampling) LFSR, with different seeds. When they are clocked together if the output of S is 1, the output is A. But if the output of S is 0, then there is no output and another clock is applied. However, this has the disadvantage that the generator's output rate varies in a way that hints at the state of S. However, this problem can be overcome by buffering the output.

On the other hand, the Self Shrinking Generator (SSG) [21] uses the same idea of the SG however, with one LFSR with selection logic. The output stream is generated as follows:

1- Each time 2 bits of the LFSR are clocked together
2- If the resulting two bits are 10 the output is zero
3- If the resulting two bits are 11 the output is one
4- Else discard the output

SSG needs less implementation in hardware since it depends on one LFSR and provides the same output as the SG. Comparing between the ASG, SG and SSG the lighter scheme is the SSG since it uses only one LFSR where SG uses two LFSRs and ASG uses 3 LFSRs. Thus, our focus will be on the SG and SSG family of cipher and the possible ciphers that can be found within this family.

Other algorithms that can be found under the SG and SSG family are the cascade algorithms that combine the idea of the SG and SSG with the LFSR and FCSR. Next section shows three cascade ideas.

IV. CASCADE CIPHER

The cascade algorithm is a combination of the two generators ideas with the combinations of LFSR and FCSR. The cascade algorithm can enhance the period length of the stream and the statistical properties of random sequences [22]. This section explains three cascade cipher streams. The cascade ciphers stream combines both of the LFSR and FCSR with the SG and SSG idea. Figure 1 shows the cascade idea.

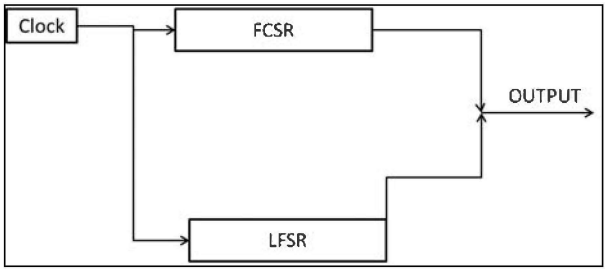

Figure 1. Cascade Cipher Structure

A. Cascade 1

This algorithm uses the SG algorithm with one LFSR and other FCSR. The idea should have the simplicity of the LFSR and the non-linearity of the FCSR to provide a random number that can be used in a cryptography key. Thus the FCSR works as the data and the LFSR works as a sampling. By clocking if the output of the FCSR is 1 then the output is the output of the LFSR at that state. Figure 1 shows cascade 1 idea.

B. Cascade 2

This algorithm uses the SG and the SG algorithm with one LFSR and one FCSR. At the start the algorithm generates the output using SSG algorithm with FCSR then it converts to the SSG with the LFSR. The output stream is generated as followed:

1. Each time 2 bits of the FCSR are clocked together.
2. If the resulting two bits are 10 the output is zero.
3. If the resulting two bits are 11 the output is one.
4. If the resulting two bits are 00 then the SSG with FCSR converts to SSG with LFSR thus the output stream will repeat the processes from 1-3.
5. If the output of the LFSR is 00, then the SSG switch back to the SSG with the FCSR.

C. Cascade 3

This algorithm also uses the SG and the SG algorithm with one LFSR and one FCSR. At the start the clock gets 4 output bits, 2 one from the FCSR and 2 bits one from the LFSR and the output stream is generated as followed:

1. Each time 2 bits of the FCSR (a_n,a_{n+1})and LFSR(b_n,b_{n+1}) are clocked together.
2. If the output of the FCSR is 10 then the output is b_n bit of the LFSR.
3. If the output of the FCSR is 11 then the output is b_{n+1} bit of the LFSR.
4. If the resulting two bits are 00 then the SSG with FCSR converts to SSG with LFSR thus the output stream will repeat the processes from 1-3.
5. If the output of the LFSR is 00, then the SSG switches back to the SSG with the FCSR again.

Next sections display the implementation results of the seven algorithms which are SG with LFSR, SSG with LFSR, SG with FCSR, SSG with the FCSR and the 3 cascade solutions.

V. IMPLEMENTATION RESULTS OF THE SG FAMILY

In order to compare between the families of the SG algorithms and find out which one can be considered as the best solution among them an implementation of the seven mentioned algorithms is done using Java language. Then, the result cipher streams are examined by three types of statistical tests to prove the cipher randomness and a comparison between them is presented. The three tests are the Frequency Test, Frequency Test within a Block, and Runs Test [23]. All of the implemented stream algorithms have the same seed value.

A. *Frequency Test (FT)*

This test examines the frequency between ones and zeros to determine if there is an approximately equal number between generated sequences. The best stream has equal probabilities between both of the zeros and ones.

B. *Frequency Test within a Block (FTWB)*

In this test the stream sequence is divided into M blocks and the number of ones is examined between each blocks. The best sequences are those with the number of ones within M block is half.

C. *Runs Test (RT)*

This test finds out the numbers of changes in the stream sequences from ones and zeros. Then the test determines if the number of ones and zeros are expected for random sequence. The best stream sequence that has more changes within the sequence between zeros and ones.

All the result of the three tests must be more than .01 in order to prove its randomness. Table 2 shows the statistical test results. From the table the cascade 1 has the best randomness sequence. Thus, this result shows that cascade 1 is the best amongst the other solutions and hence will be used as the main block for our proposed solutions next section

VI. THE PROPOSED STREAM CIPHER

Based on the statistical test analysis the proposed cipher uses the cascade 1 as the main building block. The cipher uses the following prime polynomial:

$$x^{128}+x^{126}+x^{125}+x^{124}+x^{123}+x^{122}+x^{119}+x^{117}+x^{115}+x^{111}+x^{108}+x^{106}+x^{105}+x^{104}+x^{103}+x^{102}+x^{96}+x^{94}+x^{90}+x^{87}+x^{82}+x^{81}+x^{80}+x^{79}+x^{77}+x^{74}+x^{73}+x^{72}+x^{71}+x^{70}+x^{67}+x^{66}+x^{65}+x^{61}+x^{60}+x^{58}+x^{57}++x^{56}+x^{55}+x^{53}+x^{52}+x^{51}+x^{50}+x^{49}+x^{47}+x^{44}+x^{43}+x^{43}+x^{40}+x^{39}+x^{36}+x^{35}+x^{30}+x^{29}+x^{25}+x^{23}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{11}+x^{9}+x^{8}+x^{7}+x^{6}+x^{1}+1$$

TABLE 2. STATISTICAL TEST RESULTS

Algorithm	FT	FTEB	RT
SG with LFSR	0.689157	0.487369527	0.223243461
SSG with LFSR	0.841481	0.000357729	0.686093842
SG with FCSR	0.423711	0.384039345	0.093989536
SSG with FCSR	0.841481	3.17346E-06	0.537243182
Cascade 1	1	0.237826519	0.92114228
Cascade 2	0.161513	1.53866E-05	0.154472929
Cascade 3	0.027807	0.050751875	0.177238561

The cipher stream works through two main stages which are initialization stage and output key stage.

A. *Stage 1 Initialization step*

The first step is the initiation step that uses key and Initial Vector (IV) to produce output stream. Both of the key and the IV have 128 bits length. The output stream has 256 bits and is used to feedback both of LFSR and FCSR.

B. *Stage 2 produces the secret Key*

After the Initialization step, the generator can be used to produce a stream cipher. The steps to generate the stream are as follow:

- Both of the FCSR and the LFSR are clocked together
- If the output of the FCSR is 1 then the output of the state is the output of the LFSR at that state.
- If the output of the FCSR is 0 then the output stream is discarded.
- The output stream cipher is the sum of the state and the carry value modulo 2.
- The sum value is divided by 2 to update the value of the carry value.

Figure 2 shows stage 2 process to generate the cipher key. The generator generates a stream that can be used as a key that can be XORed with the message. Figure 3 shows both of stage 1 and stage 2 processes. Next section explains the security analysis of the proposed cipher stream.

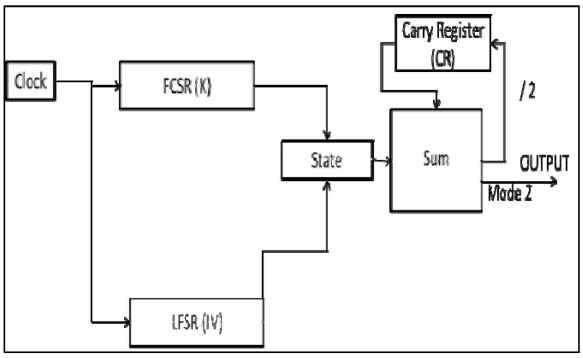

Figure 2. Main blocks of the proposed cipher

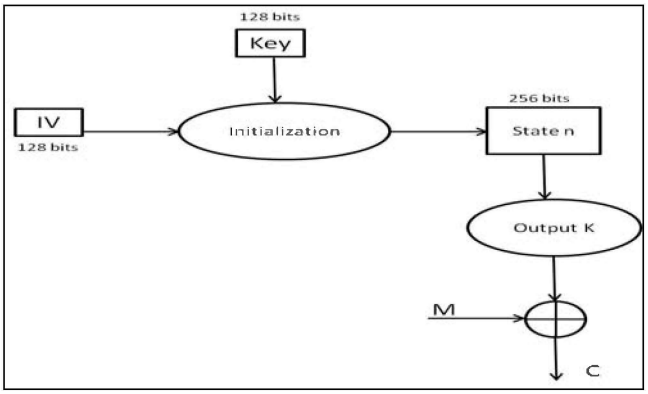

Figure 3. The two stages of the proposed cipher

VII. SECURITY ANALYSIS

The real objective of designing this stream cipher is to come up with a cipher design that can overcome the previously mentioned attacks on the stream cipher. The cipher stream is considered secure if it can be resistant to the plaintext attack [24]. The proposed cipher uses two registers with the size of 128 bits that provide a large linear span. Linear span is the size of the smallest register used to generate the key. The higher the linear span leads to higher probabilities to avoid the plaintext attack.

Also, the cipher updates its registers values which increase the difficulties of the cipher text only attack. Hence, the attacker needs to find the similarities between the different cipher streams in order to break the cipher. Also, using IV the IV can provide randomness to the encryption and synchronization between the sender and the receiver. Whereas in order to avoid the distinguishing form truly attack the proposed cipher combines both of the LFSR and FCSR to decrease the probability of finding the relation between the input and the output stream as the authors from paper [25] conclude.

Also the design cipher can avoid the TMTO attack by having the size of the state equals 256 bits which is twice the size of the key and IV (128 bits). Thus the proposed cipher can provide security and avoid most of the known attacks on the stream cipher. The stream cipher period of the proposed solution is $(2^{128}-1)$ (128 -1) where the liner complexity is at least 2^{128}. Table 3 shows the statistical test results of the proposed solution and the previously mentioned algorithms. The proposed solution is the best in terms of frequency test and the frequency test within a block. Whereas amongst the run tests cascade 1 is considered the best solution.

TABLE 3. COMPARISON OF THE STATISTICAL TESTS BETWEEN THE 7 ALGORITHMS AND THE PROPOSED ALGORITHM

Algorithms	FT	FTWB	RT
SG with LFSR	0.689157	0.487369527	0.223243461
SSG with LFSR	0.841481	0.000357729	0.686093842
SG with FCSR	0.423711	0.384039345	0.093989536
SSG with FCSR	0.841481	3.17346E-06	0.537243182
Cascade 1	**1**	**0.237826519**	**0.92114228**
Cascade 2	0.161513	1.53866E-05	0.154472929
Cascade 3	0.027807	0.050751875	0.177238561
Proposed solution	1	0.667120787	0.317310509

VIII. CONCLUSION

In summary, a new lightweight stream cipher for low computational devices such as RFID and WS is proposed. RFID and WS are combined core technologies that can be used for the IOT. The proposed solution combines both of the LFSR and FCSR. This has the advantage of having the simplicity of the LFSR and the non-linearity of the FCSR. There is a need to consider common attacks on the available stream cipher design.

Thus, in our paper we studied such attacks and tried to avoid these attacks with our design. In the future, the cipher design should also be completed to have an update state for the initial seed of the LFSR and FCSR. That addition to the cipher can strengthen its statistical randomness and prevent the correlation attack. The design will be implemented on a hardware device and the statistical tests will be applied on it to test its efficiency.

REFERENCES

[1] H. Sundmaeker,P. Guillemin, P. Friess, S. Woelfflé, "Vision and Challenges for Realising the Internet of Things", European Commission - Information Society and Media DG, March 2010.

[2] D. Uckelmann , M. Harrison, F, Michahelles, "Architecting the Internet of Things", Springer-Verlag, Berlin Heidelberg, 2011.

[3] T. Eisenbarth, C. Paar. A. Poschmann, S. Kumar, L. Usadel, "A Survey of lightweight – Cryptography Implementations", IEEE CS and the IEEE CASS, pp. 522-533, 2007.

[4] S. Charlwood, P. James-Roxby, "Evaluation of the XC6200-Series Architecure for Cryptographic Application", In FPL 98, Lecture Notes in Computer Science 1482, Springer- Verlag Berlin Heidelberg 2003 pp. 218-227, 1998.

[5] W. Stallings, "Cryptography and Network Security", Third Edition, Prentice Hall, 2002.

[6] H. Zhang , L. Li, X. Wang, "Fast Correlation Attack on Stream Cipher ABC v3", Science in China Series F: Information Sciences, pp. 936-947, 2008.

[7] B. Schneier, "Applied Cryptography, Second Edition", John Wiley & sons, Inc, 1996.

[8] E. Jonckheere and C. Ma, "A Simple Hankel Interpretation of the Berlekamp–Massey Algorithm", Linear Algebra and its Applications 125, pp. 65–76, 1989.

[9] P. C. van Oorschot and M. J. Wiener, "A Known-Plaintext Attack on Two-Key Triple Encryption", Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, pp. 318-325,1990.

[10] M. J. Aqel, Z. A. Alqadi, I. M. El Emary, "Analysis of Stream Cipher Security Algorithm", Journal of Information and Computing Science, ISSN 1746-7659, England, UK, Vol. 2, No. 4, pp. 288-298, 2007.

[11] A. Sidorenko and B. Schoenmakers, "State Recovery Attacks on Pseudorandom Generators", Western European Workshop on Research in Cryptology, Lecture Notes in Informatics (LNI) P-74, pp. 53-63, 2005.

[12] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data trades-of for stream ciphers", In T. Okamoto, editor, Proc. Asiacrypt 2000, volume 1976 of LNCS, pp. 1-13, Springer, 2000.

[13] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators", Xilinx, Tech. Rep., (Version 1.1), 1996.

[14] J. Koeter, "What's an LFSR?", Texas Instruments Incorporated, December 1996.

[15] A. Klapper and M.Goresky, "2-adic shift registers", Fast Software Encryption (FSE 1993), Lecture Notes in Computer Science 809, Springer-Verlag, pp. 174-178, 1994.

[16] N. Courtois and W. Meier. "Algebraic Attacks on Stream Ciphers with Linear Feedback", In E. Biham, editor, Advance in Cryptology-Eurocrypt 03, volume 2656 of lectures Notes in Computer Science, pp. 345-359, Springer, 2003.

[17] A. Klapper and M. Goresky, "Cryptoanalysis based on 2-adic cryptanalysis based on 2-adic rational approximation", In Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 262–273, Springer, Heidelberg, 1995.

[18] F. Arnault, T. P. Berger, C. Lanuradoux, "F-FCSR stream cipher", In H. Gilbert and H. Handschuh, editors, Fast Softwre Encryption- 2005, vol. 3557 of Lectures Notes in Computer Science, pp. 83-97, Springer, 2005.

[19] C. G. Günther, "Alternating Step Generators Controlled by De Bruijn Sequences", In EUROCRYPT, 1987: pp. 5–14, 1987.

[20] D. Coppersmith, H. Krawczyk and Y. Mansour, "The shrinking generator", In Proc. Advances in Cryptology (CRYPTO 1993), LNCS, pp. 22-39, Springer, 1994.

[21] W. Meier and O. Staffelbach, "The self-shrinking generator", In A. De Santis, editor, Advances in Cryptology - EUROCRYPT 94, volume 950 of LNCS, pp. 205-214, Berlin, Springer-Verlag 1995.

[22] R. Couture, P. L'Ecuyer, and S. Tezuka, "On the distribution of k-dimensional vectors for simple and combined Tausworthe sequences", Math-132 L'Ecuyer ematics of Computation 60(202):749-761, S11-S16,1993.

[23] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22, April 2010.

[24] T. Johansson, "Reduced Complexity Correlation Attacks on Two Clock-Controlled Generators", Advances in Cryptology, ASIACRYPT, Beijing, China, October 18-22, pp. 342-356, Lecture Notes in Computer Science 1514, 1998.

[25] S. Shun-lung, Ko-Ming Chiu, and Lih-Chyau Wuu, "The Cryptanalysis of LFSR/FCSR Based Alternating Step Generator", International Conference on Computer Engineering and Systems, pp. 228–231, Cairo, Egypt, 5-7 Nov. 2006.