Fruit: Ultra-Lightweight Stream Cipher with Shorter Internal State

Vahid Amin Ghafari, Honggang Hu, and Chengxin Xie
Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences,
University of Science and Technology of China, Hefei, China, 230027
vahidaming@mail.ustc.edu.cn, hghu2005@ustc.edu.cn, cxxie@mail.ustc.edu.cn

Abstract

In eSTREAM project, a few lightweight stream ciphers for hardware were introduced (2008). In FSE 2015, Sprout was proposed. Sprout introduced a new idea, the design of stream ciphers with shorter internal state by using secret key not only in initialization but also in keystream generation. Unfortunately, it is insecure. Grain-v1 is the lightest secure cipher in the portfolio of eSTREAM project. Fruit is the successor of Grain family and Sprout. We show that Fruit is secure and ultra-lightweight. The size of LFSR and NFSR in Fruit is only 80 bits (for 80-bit security), while for resistance against time-memory-data trade-off attack, the internal state should be at least twice of the security level. In order to compensate this, we use some new ideas in the design.

Keywords: Stream Cipher, Ultra-lightweight, Lightweight, Grain, Sprout, Cryptographic Primitive, NFSR, LFSR, Hardware Implementation

Introduction

Nowadays the need to secure lightweight symmetric cipher is obviously more than that of eSTREAM project time-(this is provable by a lot of papers in design and cryptanalysis of lightweight ciphers). WSN, RFID and mobile phones are instances which lead us to accept the importance of designing new and secure lightweight ciphers.

Three stream ciphers (Trivium [7], MICKEY 2.0 [8] and Grain-v1 [9]) have been introduced in the hardware profile of the portfolio of eSTREAM project. Grain-v1 uses NFSR and LFSR together. The linear section guarantees good statistical properties and large period, while the nonlinear section protects against attacks that can be mounted against a linear cryptosystem. An attack based on the weakness in the initialization step of Grain-v1 proposed [11]. Grain-128 was introduced in 2006 [10], and some attacks were proposed [11, 12, 13, 14, 15, 16, 17]. Indeed Grain-128 was not secure as expected (such as Grain-v1). Grain-128a [18] was proposed in 2011. Although some attacks have been applied to Grain-128a [19, 21], it is still stand-up from practicality point of view.

Sprout is a stream cipher with shorter internal state that was introduced in FSE 2015 [1]. A short while after Sprout was introduced, many attacks were published against it [3, 6, 4, 23, 2, 5]. Although it has been found that Sprout is insecure, but it had a new idea for designing stream cipher with smaller area size. The new idea is to effect secret key not only in the initialization step but also during keystream generation. Actually this idea helps to extend internal state to secret key. Due to save key for reuse by different IVs in most applications, the idea helps us have bigger internal state (therefore we can design stronger ciphers). On the other hand, we need to save key in a fixed memory in most applications (in these cases one fixed key is sufficient for ever) and it is known that saving fixed bits needs less area size in comparison to saving bits in temporary memory (e.g. bourn fixed key in a fuse). Thus we can design stream ciphers with shorter internal state [1].

Fixed key was not used in a suitable way in the design of Sprout, and it is too hard to design stream cipher using the new idea, which mentioned in the some papers about cryptanalysis of Spout [3, 5], and in other paper, authors told it is fascinating [23] and other authors predicted that secure cipher will be proposed by this new idea very soon [2].

The necessary condition for stream ciphers to be resistance against time-memory-data trade-off attack is internal state size should be at least twice of its security (while secret key only use in the initialization step), this can be seen in Trivium, MICKEY 2.0 and Grain-v1. However, stream cipher with minimal internal state will be a better choice in some applications with less available resource (such as RFID and WSN). We define stream ciphers with less than 950 GE (gate equivalents) in area size of hardware implementation, ultra-lightweight stream ciphers (stream cipher with less than 1K GE is called ultra-lightweight cipher in [2]) and we think that this is a new generation in the design of stream ciphers. Here we propose another reduced internal state stream cipher, Fruit. It is successor of Grain family and Sprout. We show that Fruit is secure and ultra-lightweight cipher and it is resistant against related-key attack.

We present our hardware implementation results and compare area size of Fruit and Grain-v1. In our implementation, Grain-v1 requires 1269 GE and Fruit requires 904 GE (without consider to initialization step). This show that area size of Grain-v1 is about 40% more than Fruit.

Really, after a while grain will sprout in the nature and in finally we have fruit. We think that Grain-v1 and Sprout were two new generations in the design of stream ciphers and now Fruit is mature in stream ciphers. We summarize our new ideas in design of Fruit as follows:

1- New round key function (most weaknesses of sprout is related to round key function)
2- New nonlinear feedback for NFSR, feedback for LFSR and output function (in Sprout they were similar to Grain-128a, they were different only in positions)
3- Increase the size of LFSR to achieve longer keystream in each loading (in comparison to Sprout)
4- New way to load IV in the initialization (IV loads in LFSR and NFSR in Grain family and Sprout)

The rest of the paper is organized as follows. We explain about design of Fruit and the design criteria. Then we show Fruit is resistance to known attacks. Finally we discuss about the hardware implementation of Fruit.

Design Details of Fruit

Internal state consists of 43-bit LFSR ($l_t, \dots, l_{t+42}$), 37-bit NFSR ($n_t, \dots, n_{t+36}$), 7-bit counter (Cr: $(c_t^0, \dots, c_t^6)$) and 8-bit counter (Cc: $(c_t^7, \dots, c_t^{14})$). A general view of Fruit stream cipher is presented in Fig. 1. Inputs of Fruit are 80-bit secret key (K: $(k_0, \dots, k_{79})$) and 70 bits public Initial Value (IV: $(v_0, \dots, v_{69})$). Note that maximum number of stream bits that can be produced from one key and IV is 2^{43} bits and each key should be used less than 2^{15} times with different IVs. It is not acceptable to reuse IV, i.e. use identical IV with different keys. IV should be produced in a random way.

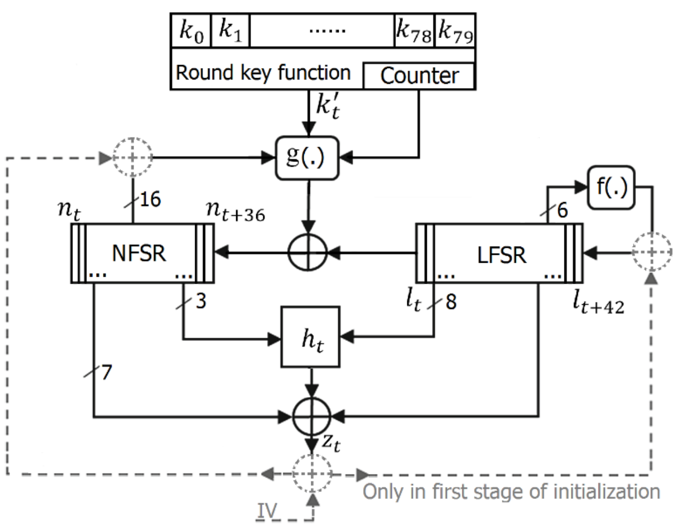

Fig. 1: The Block Diagram of Fruit

Now we explain each part of the cipher in details:

-Counter: the first 7 bits of counter (Cr) are allocated to round key function and the last 8 bits (Cc) are allocated to initialization and keystream generation. These two counters work (count) independently, i.e. first counter ($c_t^0, \dots, c_t^6$) is increased one by one in each clock, and also second counter ($c_t^7, \dots, c_t^{14}$) count from zero independently. These two counters increase at each clock, and work continually, i.e. after first and second parts become all ones, counting from zeros to all ones again. Note that c_t^6 and c_t^{14} are LSB of two counters, i.e. before first clock our counter is (000000000000000) and then after first clock is (000000100000001).

-**Round key function**: we define $s = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$, $y = (c_t^3 c_t^4 c_t^5)$, $z = (c_t^4 c_t^5 c_t^6)$, $p = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4)$, $q = (c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$, and $r = (c_t^3 c_t^4 c_t^5 c_t^6)$. We combine 6 bits of the key to obtain bits of round key as follow in each clock.

$$k_t' = k_s . k_{y+64} \oplus k_{z+72} . k_p \oplus k_{q+32} \oplus k_{r+64}$$

-g function: we use 1 bit of the LFSR, 1 bit of the counter, k_t' and 16 bits of the NFSR as variables of **g** function for clocking of NFSR.

$$n_{t+37} = k_t' \oplus l_t \oplus c_t^{10} \oplus n_t \oplus n_{t+10} \oplus n_{t+20} \oplus n_{t+12} . n_{t+3} \oplus n_{t+14} . n_{t+25} \oplus n_{t+8} . n_{t+18} \oplus n_{t+5} . n_{t+23} . n_{t+31} \oplus n_{t+28} . n_{t+30} . n_{t+32} . n_{t+34}$$

-f function: the feedback function in LFSR is primitive. Thus, it can produce string with maximum period.

$$l_{t+43} = l_t \oplus l_{t+8} \oplus l_{t+18} \oplus l_{t+23} \oplus l_{t+28} \oplus l_{t+37}$$

-h function: this function produces pre-output stream from LFSR and NFSR states.

$$h_t = n_{t+1} . l_{t+15} \oplus l_{t+1} . l_{t+22} \oplus n_{t+35} . l_{t+27} \oplus n_{t+33} . l_{t+11} \oplus l_{t+6} . l_{t+33} . l_{t+42}$$

-keystream generation: keystream will be produced by 7 bits from NFSR, 1 bit from LFSR and output of h function.

$$z_t = h_t \oplus n_t \oplus n_{t+7} \oplus n_{t+13} \oplus n_{t+19} \oplus n_{t+24} \oplus n_{t+29} \oplus n_{t+36} \oplus l_{t+38}$$

-Initialization of the cipher: we extend IV to 80 bits by concatenating 9 bits zero and 1 bits one to the first of IV, as follow.

$$IV' = v'_0 v'_1 \dots v'_{78} v'_{79} = 1000000000 v_0 v_1 v_2 \dots v_{67} v_{68} v_{69}$$

In initialization step, key bits are loaded to NFSR and LFSR respectively from LSB to MSB (k_0 to n_0, k_1 to n_1, …, k_{36} to n_{36}, k_{37} to l_0, k_{38} to l_1, …, k_{79} to l_{42}). $c_0^0 c_0^1 \dots c_0^{13} c_0^{14}$ are all set to 0 in the first stage of initialization. We clock the cipher 80 times and before each clock, XOR output with IV' bits and also feedback output stream to NFSR and LFSR, i.e. $z_i \oplus v'_i$, $0 \le i \le 79$ (as shown in Fig. 1). Then in the second stage of initialization, we set all bits of Cr equal to LSB of NFSR except the last bit of Cr that is equal to LSB of LFSR ($c_{80}^0 = n_{80}, c_{80}^1 = n_{81}, \dots, c_{80}^4 = n_{84}, c_{80}^5 = n_{85}, c_{80}^6 = l_{80}$), and also l_{80} is set to 1 for preventing all zeros in LFSR.

Then the cipher should be clock 80 times without feedback output in LFSR and NFSR (i.e. during last 80 clocks we disconnect feedback of z_t to LFSR and NFSR). Thus, the cipher doesn't produce any keystream in the 160 initial clocks, i.e. z_0 to z_{159} are discarded. Now the cipher is ready to produce first bit of keystream, i.e. z_{160}.

Design criteria

-Limitation for the producing keystream: the maximum length of keystream is 2^{43} bits in each initialization, because of the LFSR length (period of NFSR is a multiple of $2^{43} - 1$). We think that 1 terabyte is almost sufficient for all application because our cipher is special for hardware application (e.g. WSN and RFID).

-**Round key function**: we produce 2^7 different keys from original key. Attacker can (with guessing internal states and known output keystream) obtain some bits of k'_t, but due to the unknown counter (unknown index of key in round key function), it is not easy to solve equations system. Round key function in Fruit involves bits of key independently in the **g** function, while in Sprout cipher, none of key bits involves in **g** function in some clocks.

-g function: The function that produces n_{t+37}, is chosen in only 16 variables of NFSR with regard to light implementation in hardware in comparison to Grain-v1 and Sprout. If we suppose $k'_t \oplus c_t^{10} \oplus l_t = 0$, the nonlinearity of **g** function will be $2^3 \times 3760$ and resiliency 2. Variables for high degree term is chosen from n_t with $t > 27$ that cause the degree of variables reaches the maximum possible degree in NFSR very soon.

-f function: the period of produced string by LFSR with non-zero initial is maximum because feedback polynomial is primitive. Due to l_{80} is set to 1 after disconnecting feedback of output bit to LFSR, we are sure that the period of LFSR and NFSR is at least $2^{43}-1$. Some attacks was proposed to Grain and Sprout from this weakness (i.e. it is possible that LFSR becomes all zeros in the during of initialization) [22, 23].

-Output function: The nonlinearity of h function is 976. We add 8 linear terms in order to increase the nonlinearity to 249856 and also to make function with 7 resiliency. The best linear approximation of output function has 8 terms with $2^{-5.415}$ bias.

Note that n_{t+36} and n_t are used in output function for preventing produce keystream in the next and previous clock with unknown k'_t.

Resistance against known attacks

Security level of Fruit is 80 bits, thus we discuss about the feasibility of applying some main attacks to it.

-Linear Approximation Attack: this attack was applied to Grain-v0 [24]. In [24] discussed that if NFSR and output function are chosen with high nonlinearity and suitable resiliency, it will be resistant to linear approximations attack. We choose NFSR and output function with high nonlinearity and good resiliency and also a nonlinear function of key is involved on NFSR. If an attacker obtains linear approximation of output and also linear approximation of **g** function, and then eliminates NFSR bits between the two relations, he can obtain follow relation with $2^{-43.86}$ bias.

$$z_t \oplus z_{t+10} \oplus z_{t+20} \oplus z_{t+80} = l_t \oplus k'_t \oplus c_t^{10} \oplus l_{t+7} \oplus k'_{t+7} \oplus c_{t+7}^{10} \oplus l_{t+13} \oplus k'_{t+13} \oplus c_{t+13}^{10} \oplus l_{t+19} \oplus k'_{t+19} \oplus c_{t+19}^{10} \oplus l_{t+24} \oplus k'_{t+24} \oplus c_{t+24}^{10} \oplus l_{t+29} \oplus k'_{t+29} \oplus c_{t+29}^{10} \oplus l_{t+36} \oplus k'_{t+36} \oplus c_{t+36}^{10} \oplus l_{t+38}$$

Now, if the attacker supposes that round key bits and counter bits are equal to zero, and if he tries to obtain a relation only based on output bits (by using feedback polynomial of LFSR), the bias of the relation is $2^{-301.02}$. This bias is too small and therefore Fruit is resistant to this attack.

-Guess and Determine Attack: due to shorter LFSR and NFSR in Fruit and the weakness of Sprout against this attack [23], this attack is very important. If an attacker guesses all bits of the internal state in Sprout, he can clock 2 times forward and one time backward (with unknown key), and in each clock he can decrease the wrong candidates of the internal state in Sprout. In the next clocks, the attacker obtains one bit of the key or decreases the wrong candidates of the internal state. We strengthen the round key function and use n_{t+36} and n_t in output function for preventing produce keystream in the next and previous clock with unknown key. If the attacker can obtain some bits of k'_t, with regard to unknown number of Cr (unknown index of key in round key function), it is too hard for the attacker to solve equations and obtain key bits.

If an attacker guesses all bits of Cr, LFSR and NFSR, i.e. 87 bits, he can obtain round key function bits. In this situation, due to each bit of round key function is dependent on some bits of the key, it is impossible for the attacker to identify wrong candidates of internal states before 80 clocks (except in the first clock he can identify half of wrong candidates). The computational complexity of this attack is (at least) 80.2^{86}, which is more than complexity of exhaustive attack, i.e. 2^{80}. Thus, Fruit is resistant against this attack.

-Time-Memory-Data Trade-off Attack: it is well known that cipher is weak to this attack if the size of internal state is not at least twice of security level. We use fixed key as an internal state in Fruit, and also some bits of the counter in round key function, and also we independently use k'_t to preventing bypass of key in **g** function (such as attack to Sprout [2]), therefore there is no problem from this view.

-Related-key Attack: There exist weakness in the initialization step of all member of Grain family [11, 21] and Sprout [3]. Designers of Sprout ruled out related key attack. They believed this attack is not workable on Sprout because key is fix in ultra-lightweight ciphers [1]. Nevertheless we propose a new scheme in the initialization step to strengthen against this attack. We do not load IV bits directly in the internal state and do not combine IV and key bits straightforward together. Also we use asymmetric padding with IV, so there is no weakness related to this attack.

-Cube Attack: due to the suitable clock number in the initialization step, it is too hard to find any low degree multiplicative expression (of some bits of IV) based on key in Boolean function of the output. In Fruit, the length of LFSR and NFSR are shorter than Grain-v1, so with equal clock number in initial step of Grain-v1 and Fruit, Fruit cipher is stronger against this attack. Therefore our design is more resistant against to all types of Cube attack.

-Algebraic Attack: this attack has not been applied to Grain family but combination of this attack was applied to Sprout [4]. Short internal state (or we tell weak round key function) in Sprout caused this weakness. We strengthen round key function and involve bits of key independently in **g** function, so we think Fruit is secure against this attack. Due to fast growth of degree in the internal state of Fruit, it is impossible for an attacker to apply pure algebraic attack. But here we discuss that a combine of guess and determine attack and algebraic attack is not applicable to Fruit.

If an attacker guesses bits of NFSR, bits of counter and bits of round key function, then he can obtain two equations in each clock (one from output keystream and one from round key function). These equations are degree 2 and it is not easy to solve, but we suppose that the attacker can solve equations of output keystream and obtain 1 bits of LFSR in each clock. In this scenario the attacker should guess at least 40 bits of round key function. Totally the attacker should guess $37 + 7 + 40$ bits that is more than computational complexity of exhaustive attack.

-Fault Attack: this attack is applied successfully to all members of Grain family [20] and also to Sprout [4, 25], but we think that it is not applicable on the real world. Fault attack is based on some impractical supposes. An attacker should be able to induce fault on the cipher in a special time and supposes that the induced fault effect on special section of cipher, e.g. the attacker should be free for injecting a single bit fault in NFSR just after initialization [25]. Other unreal suppose is that the attacker can reset the cipher and obtains correct keystream. Therefore we do not consider this attack.

Hardware implementation cost

Design of lightweight cipher is very important in industries, while we need to light ciphers in many fields such as communication, WSN, RFID and etc. Thus, our goal was to design a strong cipher with less than 950 GE (i.e. Ultra-lightweight stream cipher). In order to get area size in hardware implementation for Fruit and Grain-v1, we designed the circuit of them described in VHDL and chose TSMC 0.18 µm technology process to do the synthesis. There was no optimization in our hardware implementation, for fair comparison with Grain-v1.

In Table 1, we compare the area size of hardware implementation of eSTREAM finalist (in hardware profile) and Fruit. Different results for hardware implementation based on GE, is normal, because hardware implementation cost is dependent on many factors such as hardware used in implementation, techniques of implementation and etc. The area size of Fruit is significantly less than other ciphers, as expected with regard to length of the internal state of them (except Sprout). Area size (GE) of Fruit is about 71%, 65%, 30%, 22%, 29% less than Mickey [26], Trivium [26], Grain-v1 [26] and Grain-v1 [1] respectively. Fruit is much more secure than Sprout, but requires a little more GE.

Table 1. Area size for eSTREAM finalists and Fruit in hardware implementation

Stream ciphers	Area size(GE)	Throughput (Kb/s)#	Platform	Source
Mickey [8]	3188	100	0.13 μm CMOS	[26]
Trivium [7]	2580	100	0.13 μm CMOS	[26]
Grain-v1 [9]	1294	100	0.13 μm CMOS	[26]
Grain-v1 [9]	1162	100	0.18 μm CMOS	[1]
Sprout	813	100	0.18 μm CMOS	[1]
Grain-v1 [9]	1269	100	0.18 μm CMOS	Our work
Fruit	904	100	0.18 μm CMOS	Our work

#The throughput is for the clock with 100 KHz frequency

Note that we don't dedicate any GE to key bits, because in every cipher should save key for reuse with different IVs.

Conclusion

Fruit stream cipher in comparison with Grain-v1 is very lightweight in hardware implementation, and with regard to Grain-v1 is the lightest candidate in the eSTREAM finalist of hardware profile, it is obvious that design of secure stream ciphers such as Fruit is very interesting. We discussed that Fruit unlike Sprout cipher is secure with some new ideas. In most applications of symmetric cipher, secret key (for reuse with other IVs) should be saved (in a memory), and we showed how we can exploit it in the design. We showed that suitable use of fixed secret key as an internal state can save significantly the area size in the design of secure stream cipher. Due to Fruit initialization step is more complicated than initialization step of Grain-v1, we implement the ciphers in two types, with and without consider any GE for initialization step .Table 2 shows that the area size of Grain-v1 is about 40% more than Fruit without consider to initialization step. If we consider to the initialization step of both ciphers, we still have about 18% save in area size. We hope the design of Fruit (as a first secure ultra-lightweight stream cipher) will be a beginning of the design of ultra-lightweight stream ciphers.

Table 2. Synthesize of Fruit and Grain-v1 on TSMC 0.18 µm technology process

Cipher	Implementation type	Area size(GE)
Grain-v1 [9]	with initialization	1573
	without initialization	1269
Fruit	with initialization	1331
	without initialization	904

Acknowledgement

This work has been supported by CAS-TWAS President's Fellowship for International PhD program.

References

[1] Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 451–470. Springer, Heidelberg (2015)
[2] Esgin, M.F., Kara, O.: Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. Selected Areas in Cryptography - SAC 2015. LNCS, vol. 9566 pp 67-85. Springer, Heidelberg (2015)
[3] Hao, Y.: A Related-Key Chosen-IV Distinguishing Attack on Full Sprout Stream Cipher. http://eprint.iacr.org/2015/231.pdf
[4] Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key Recovery from State Information of Sprout: Application to Cryptanalysis and Fault Attack. http://eprint.iacr.org/2015/236.pdf
[5] Zhang, B. ,Gong, X. : Another Tradeoff Attack on Sprout-Like Stream Ciphers, ASIACRYPT 2015, Part II, LNCS 9453, pp. 561–585, 2015
[6] Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of full sprout. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology – CRYPTO 2015. LNCS, vol. 9215, pp. 663–682. Springer, Heidelberg (2015)
[7] D e Can n i`ere, C., Preneel, B.: Trivium. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)
[8] Babbage, S., Dodd, M. "The Stream Cipher MICKEY 2.0" ECRYPT Stream Cipher Project Report. http://www.ecrypt.eu.org/stream/ p3ciphers/mickey/mickey_p3.
[9] Hell, M., Johansson, T., Meier, W. "Grain—A Stream Cipher for constrained environments" New Stream Cipher Designs, 2008 Lecture Notes in Computer Science, vol. 4986, pp. 179–190, available at http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf.
[10] Hell, M., Johansson, T., Maximov, A., Meier, W. "A Stream Cipher Proposal: Grain-128," in International Symposium on Information Theory—ISIT 2006. IEEE, 2006
[11] Lee, Y., Jeong, K., Sung, J., & Hong, "Related-key chosen IV attacks on Grain-v1 and Grain-128." Information Security and Privacy. Springer Berlin Heidelberg, pp. 321-335. Springer Berlin Heidelberg, 2008.
[12] Dinur, I., Shamir, A. "Breaking grain-128 with dynamic cube attacks," FSE 2011, Lyngby, Denmark, 2011, pp. 167–187.
[13] Dinur, I. ,Güneysu, T., Paar, C., Shamir, A., Zimmerman, R. "An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware," Proc. ASIACRYPT 2011, Seoul, South Korea, 2011, pp. 327–343.
[14] J. M. Miodrag, G. Sugata, P. Goutam, and I. Hideki, "Generic cryptographic weakness of K-normal Boolean functions in certain stream ciphers and cryptanalysis of grain-128," Periodica Mathematica Hungarica, vol. 65, no. 2, pp. 205–227, Dec. 2012.

[15] J. P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir, “Efficient FPGA implementations of high-dimensional cube testers on the stream cipher grain-128,” IACR Cryptology ePrint Archive, pp.218–218, 2009
[16] P. Stankovski, “Greedy distinguishers and nonrandomness detectors,” INDOCRYPT 2007, Hyderabad, India, 2010, pp. 210–226.
[17] S. Knellwolf, W. Meier, and M. Naya-Plasencia, “Conditional differential cryptanalysis of NLFSR-based cryptosystems,” in Proc. ASIACRYPT 2010, Singapore, 2010, pp. 130–145.
[18] M. Ågren, M. Hell, T. Johansson, and W. Meier, “A new version of Grain-128 with authentication,” in Proc. Symmetric Key Encryption Workshop, Lyngby, Denmark, Feb. 2011 [Online]. Available: http://skew2011.mat.dtu.dk/
[19] Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on Grain-128a using MACs. SPACE 2012. LNCS, vol. 7644, pp. 111–125. Springer, Heidelberg (2012)
[20] Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the Grain family of stream ciphers CHES 2012. LNCS, vol. 7428, pp. 122–139. Springer, Heidelberg (2012).
[21] L. Ding and J. Guan, "Related Key Chosen IV Attack on Grain-128a Stream Cipher", IEEE Trans. Inform. Forensic Secur., vol. 8, no. 5, pp. 803-809, 2013.
[22] H. Zhang and X. Wang, “Cryptanalysis of stream cipher Grain family,” Cryptology ePrint Archive, Report 2009/109, 2009, http://eprint.iacr.org/.
[23] Banik Subhadeep., Some Results on Sprout. INDOCRYPT 2015, LNCS 9462, pp. 124–139, 2015. DOI: 10.1007/978-3-319-26617-6 7
[24] A. Maximov, “Cryptanalysis of the “Grain” family of stream ciphers”, in ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS’06), 2006, pp. 283–288.
[25] Roy, D., & Mukhopadhyay, S. Fault analysis and weak key-IV attack on Sprout. https://eprint.iacr.org/2016/207.pdf
[26] Good, T., Benaissa, M.: Hardware performance of estream phase-III stream cipher candidates. In: Proceedings of Workshop on the State of the Art of Stream Ciphers (SACS 2008) (2008)

Appendix A: Test vector

Due to in each clock one bit will be produced, we use hexadecimal format for presenting the keystream, key and IV. First bit of output keystream (i.e. z_{160}) is most significant bit in the first hexadecimal as follow.

$$K = \{k_0k_1k_2k_3, k_4k_5k_6k_7, k_8k_9k_{10}k_{11}, \dots, k_{72}k_{73}k_{74}k_{75}, k_{76}k_{77}k_{78}k_{79}\}$$

$$IV = \{00v_0v_1, v_2v_3v_4v_5, v_6v_7v_8v_9, \dots, v_{62}v_{63}v_{64}v_{65}, v_{66}v_{67}v_{68}v_{69}\}$$

$$Z = \{z_{160}z_{161}z_{162}z_{163}, z_{164}z_{165}z_{166}z_{167}, z_{168}z_{169}z_{170}z_{171}, \dots\}$$

First test vector:

$$k = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$$

$$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$$

$$Z = \{9,3,c,7,8,e,3,b,c,6,e,4,6,2,d,8,1,c,b,2\}$$

Second test vector:

$k = \{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$

$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$

$Z = \{5,5,3,5,8,6,4,1,1,4,\mathrm{f},1,1,\mathrm{a},\mathrm{c},6,\mathrm{e},\mathrm{f},\mathrm{f},8\}$

Third test vector:

$k = \{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$

$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$

$Z = \{8,5,3,2,9,4,7,1,e,4,0,0,3,5,4,8,1,e,b,1\}$

Fourth test vector:

$k = \{1,2,3,4,0,0,0,0,\mathrm{a},\mathrm{b},\mathrm{c},\mathrm{d},0,0,0,0,0,1,2,3,4\}$

$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$

$Z = \{3,0,d,a,4,0,0,6,4,d,d,7,2,c,8,7,0,8,5,6\}$