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

Abstract. In the eSTREAM project, a few lightweight stream ciphers for hardware were introduced (2008). In FSE 2015, while presenting a new idea (i.e. the design of stream ciphers with shorter internal state by using secret key not only in initialization but also in keystream generation), Sprout was proposed. Unfortunately, Sprout is insecure. Due to Grain-v1 is the lightest secure cipher in the portfolio of eSTREAM project, we introduce Fruit as a successor of the 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 level), 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 · NFSR · LFSR · Hardware Implementation

Introduction

Nowadays the need of secure lightweight symmetric cipher is obviously more than eSTREAM project time-(this is provable by a lot of papers in design and cryptanalysis of lightweight ciphers [AM15, EK15, Hao15, MSBD15, ZG15, LN15, SIH+11, LLW16]). 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 [Can06], MICKEY 2.0 [BD06] and Grain-v1 [HJM07]) have been introduced in the hardware profile of the portfolio of eSTREAM project. Grain-v1 uses both NFSR and LFSR. 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. A related-key attack based on the weakness in the initialization procedure of Grain-v1 was proposed in [LJSH08]. Grain-128 was introduced in 2006 [HJMM06], and some attacks were proposed to it [LJSH08, DS11, DGP+11, MGPI12, ADH+09, Sta10, KMN10]. Indeed Grain-128 is not secure as expected. Grain-128a was proposed in 2011 [ÅHJM11]. Although some attacks have been applied to Grain-128a [BMS12a, DG13], it is still good from the practicality point of view.

In FSE 2015 a new idea, the design of stream ciphers with shorter internal state, was introduced. Sprout stream cipher was proposed as an instance based on the new idea [AM15]. A short while after Sprout was introduced, many attacks were published against it [Hao15, LN15, MSBD15, Ban15, EK15, ZG15]. Although there has been found that Sprout is insecure, it has a new idea to design stream cipher with smaller area size. Its new idea is to use the secret key not only in the initialization procedure, but also in the keystream generation. Actually this idea helps to extend the internal state to the key. Because the storing of a key for reuse by different IVs is essential in most applications, and also it is need to store the key in a fixed memory in some applications (in these cases,

one fixed key is sufficient, e.g. in a SIM card of mobile phone) the idea helps us to have a bigger internal state (therefore we can design stronger ciphers).

On the other hand, it is known that storing fixed bits needs less area size in comparison to storing bits in a temporary memory (e.g. burn fixed key in a fuse). Thus, it is possible to design stream ciphers with shorter internal state [AM15].

The key is not used in a suitable way in the design of Sprout, and it is hard to design stream cipher using the new idea, which has been mentioned in some papers about cryptanalysis of Sprout [Hao15, ZG15]. In other research, authors stated that it is fascinating [Ban15] and another paper predicted that secure cipher will be proposed by this new idea very soon [EK15]. The design of secure cipher with shorter internal sate has been introduced as a question from the year 2015.

The necessary condition for stream ciphers to be resistant against time-memory-data trade-off attack is that internal state size should be at least twice of its security level (while the secret key is only used in the initialization procedure), such as Trivium, MICKEY 2.0 and Grain-v1. We show how we can exploit key in a design to achieve shorter internal state.

Stream cipher with minimal internal state is a better choice for most applications with less available resource (such as RFID and WSN). We call stream ciphers with less than 950 GE (gate equivalents) in area size of hardware implementation as ultra-lightweight stream ciphers (stream cipher with less than 1K GE is called ultra-lightweight cipher in [EK15]). We think that it is a new generation in the design of stream ciphers. In this paper, we propose another reduced internal state stream cipher: Fruit. It is a successor of Grain family and Sprout. As we show, Fruit is secure and ultra-lightweight and is more resistant than Grain family against some attacks such as related-key attack.

We also present the hardware implementation results, and compare the area size of Fruit, Sprout and Grain-v1. In our implementation, Grain-v1 requires 1258 GE and Fruit requires 918 GE. This shows that the area size of Grain-v1 is about 37% bigger than that of Fruit.

We summarize our new ideas in the design of Fruit as follows:

1- New round key function (most weaknesses of sprout are related to the round key function)

2- New nonlinear feedback for NFSR, feedback for LFSR and output function (in Sprout they are similar to Grain-128a, and the difference is only in the positions)

3- Increase the size of LFSR to achieve longer keystream in each loading

4- New way to load IV in the initialization

5- New idea for preventing that LFSR becomes all zeros after the initialization (unlike Grain family and Sprout)

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

1 The design of Fruit

The internal state consists of 43-bit LFSR $(l_t, ..., l_{(t+42)})$, 37-bit NFSR $(n_t, ..., n_{(t+36)})$, 7-bit counter $(Cr : (c_t^0, ..., c_t^6))$ and 8-bit counter $(Cc : (c_t^7, ..., c_t^{14}))$. A general view of Fruit is presented in Figure 1. Inputs of Fruit are 80-bit secret key (K: $(k_i, 0 \leq i \leq 79)$) and 70-bit public Initial Value, (IV: $(v_i, 0 \leq i \leq 69)$). Note that the maximum number of keystream 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.

Now we explain each part of the cipher in details:

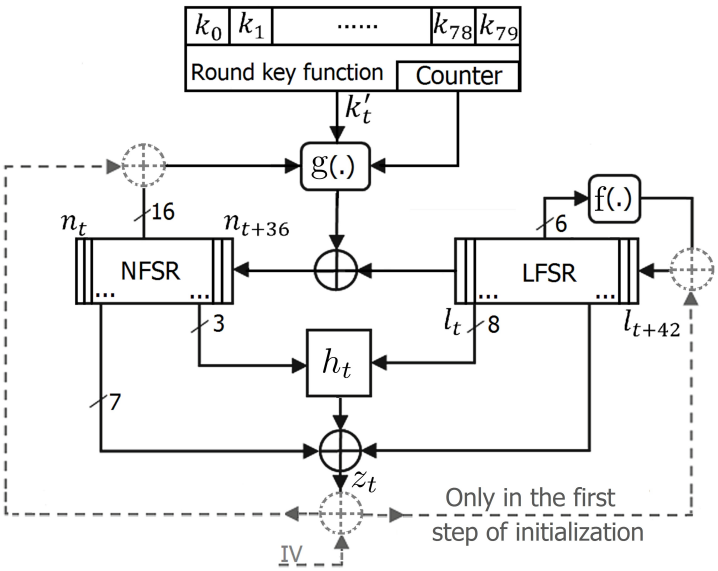

Figure 1: The Block Diagram of Fruit

Counters. the first 7 bits of the counter (Cr) are allocated to the round key function and the last 8 bits (Cc) are allocated to the initialization and keystream generation. These two counters work (count) independently, i.e. the first counter $c_t^0, ..., c_t^6$ is increased one by one in each clock, and also the second counter $(c_t^7, ..., c_t^{14})$ 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 the two counters, i.e. before first clock our counter is (000000000000000) and then after first clock is (000000100000001).

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

$$k'_t = k_s \cdot k_{(y+64)} \oplus k_{(u+72)} \cdot k_p \oplus k_{(q+32)} \oplus k_{(r+64)} \tag{1}$$

g function. we use 1 bit of the counter, k'_t and 16 bits of the NFSR as variables of **g** function for clocking of NFSR. The feedback function of NFSR is as follow.

$$\begin{aligned} n_{(t+37)} = {} & k'_t \oplus l_t \oplus c_t^{10} \oplus n_t \oplus n_{(t+10)} \oplus n_{(t+20)} \oplus n_{(t+12)} \cdot n_{(t+3)} \\ & \oplus n_{(t+14)} \cdot n_{(t+25)} \oplus n_{(t+5)} \cdot n_{(t+23)} \cdot n_{(t+31)} \\ & \oplus n_{(t+8)} \cdot n_{(t+18)} \oplus n_{(t+28)} \cdot n_{(t+30)} \cdot n_{(t+32)} \cdot n_{(t+34)} \end{aligned} \tag{2}$$

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

$$l_{(t+43)} = l_t \oplus l_{(t+8)} \oplus l_{(t+18)} \oplus l_{(t+23)} \oplus l_{(t+28)} \oplus l_{(t+37)} \tag{3}$$

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

$$\begin{aligned} h_t = & n_{(t+1)} \cdot l_{(t+15)} \oplus l_{(t+1)} \cdot l_{(t+22)} \oplus n_{(t+35)} \cdot l_{(t+27)} \\ & \oplus n_{(t+33)} \cdot l_{(t+11)} \oplus l_{(t+6)} \cdot l_{(t+33)} \cdot l_{(t+42)} \end{aligned} \tag{4}$$

Output function. the output stream will be produced by 7 bits from NFSR, 1 bit from LFSR and output of ***h*** function.

$$\begin{aligned} z_t = & h_t \oplus n_t \oplus n_{(t+7)} \oplus n_{(t+13)} \oplus n_{(t+19)} \oplus n_{(t+24)} \\ & \oplus n_{(t+29)} \oplus n_{(t+36)} \oplus l_{(t+38)} \end{aligned} \tag{5}$$

Initialization of the cipher. we extend IV bits to the 130 bits by concatenating 1 bit one and 9 bit zeros to the first of IV and 50 bit zeros to the end of IV, as follow.

$$IV' = 1000000000 v_0 v_1 v_2 ... v_{67} v_{68} v_{69} 000...000 \tag{6}$$

In the initialization procedure, key bits are loaded to the NFSR and LFSR 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}). $c_0^0 c_0^1 ... c_0^{13} c_0^{14}$ are set to 0 in the first step of the initialization. The cipher is clocked 130 times, but before each clock, the XOR of the output bits and IV' bits is fed to the NFSR and LFSR (i.e. $z_i \oplus v'_i$, $0 \leq i \leq 129$ (as show in Figure. 1).

Then, in the second step of the initialization, we set all bits of Cr equal to LSB of the NFSR except the last bit of Cr that is equal to LSB of the LFSR ($c_{130}^0 = n_{130}$, $c_{130}^1 = n_{131}$, ..., $c_{130}^4 = n_{134}$, $c_{130}^5 = n_{135}$, $c_{130}^6 = l_{130}$) and also l_{130} is set to 1 for preventing that LFSR becomes all zeros after initialization.

Then, the cipher should be clocked 80 times without the feedback in the LFSR and NFSR (i.e. during the last 80 clocks the feedback of $z_i \oplus v'_i$ is disconnected to the LFSR and NFSR). The cipher does not produce any keystream in the 210 initial clocks, i.e. z_0 to z_{209} are discarded. Now the cipher is ready to produce the first bit of the keystream, i.e. z_{210}.

2 The design criteria

Limitation for the producing keystream. the maximum length of produced 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 sufficient for most applications because our cipher is special for hardware applications with limited resource (e.g. WSN and RFID).

Round key function. we produce 2^7 different keys from the 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 the round key function), it is not easy to solve equations system. Round key function in Fruit involves bits of the key independently in **g** function, while in Sprout cipher, none of the 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 the NFSR with regard to light implementation in hardware in comparison to Grain-v1 and Sprout. If we suppose $k'_t \oplus c_t^{10} = 0$, 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 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 the feedback polynomial is primitive. Due to l_{130} is set to 1 after disconnecting feedback of output bit to the LFSR, we are sure that the period of the 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 the LFSR becomes all zeros in during the initialization) [ZW09, Ban15].

number of clock in the initialization. it is very important to generate maximum degree monomial based on the key and IV in the initialization procedure and distribute it in the all bits of the LFSR and NFSR. In this situation, the random distribution of low degrees monomial is likely suitable. The numbers of clocks in the initialization procedure of Grain-v1 and Grain-128a are 160 and 256 respectively and also the length of NFSRs (or LFSR) are 80 and 128 respectively. It seems that 160 initial clock for Fruit is sufficient, but due to IV bits will be injected in the LFSR and NFSR one-by-one after 10 clocks and also feedback of output will be disconnected in the second step of the initialization; 210 initial clock is suitable. Note that 320 initial clocks could not provide enough security for Sprout because of weakness in the other sections.

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

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

3 The resistance against known attacks

The security level of Fruit is 80 bits. We discuss the feasibility of applying some main attacks on it.

Linear Approximation Attack. this attack was applied to Grain-v0 [Max06]. In [Max06] discussed that if the NFSR and the output function are chosen with high nonlinearity and suitable resiliency, it will be resistant to linear approximations attack. We choose the NFSR and output function with high nonlinearity and good resiliency and also a nonlinear function of key is involved on the NFSR. The best linear approximation of the output has $2^{(-5.415)}$ bias as follow.

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

If we suppose $k'_t \oplus c_t^{10} = 0$, the best linear approximation of NFSR feedback function has $2^{(-4.6)}$ bias as follow.

$$n_{(t+37)} = n_t \oplus n_{(t+10)} \oplus n_{(t+20)} \oplus l_t \tag{8}$$

If an attacker eliminates the NFSR bits between these two relations (by shifting and XORing of linear approximation of the output), he can obtain the following relation with $2^{(-43.86)}$ bias (by Piling-up Lemma).

$$\begin{aligned} z_t \oplus z_{(t+10)} \oplus z_{(t+20)} \oplus z_{(t+37)} = {} & l_t \oplus l_{(t+7)} \oplus l_{(t+13)} \oplus l_{(t+19)} \oplus l_{(t+24)} \oplus l_{(t+29)} \\ & \oplus l_{(t+36)} \oplus l_{(t+38)} \oplus l_{(t+48)} \oplus l_{(t+58)} \oplus l_{(t+75)} \end{aligned} \tag{9}$$

Now, if the attacker tries to obtain a relation only based on the output bits (by using feedback function of LFSR), the bias of the relation 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 [Ban15], 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 to prevent producing the keystream in the next and previous clock with unknown key.

If an attacker guesses the required bits of LFSR and NFSR for producing the first 8 bits (i.e. 80 bits), he can obtain some bits of k'_t by known keystream. It is too hard for him to solve equations and obtain key bits with regard to unknown number of Cr (unknown index of key in round key function).

If an attacker guesses all bits of Cr and LFSR and NFSR, i.e. 87 bits, he can obtain round key function bits. In this situation, due to each bit of the 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 higher 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 the cipher is weak to this attack if the size of internal state is not at least twice of security level. Key and some bits of the counter are used as an internal state in Fruit. Therefore, the efficient internal state is 167 bits.

In the design of stream cipher (similar to Sprout and Fruit) is very important how to benefit secret key as an internal state. It is obvious that all of internal sate (including key for Sprout and Fruit) should affect on the internal state proportionally and continually. It is necessary that secret key affects on the internal state freely and independently in the during of keystream production. Whatever this effect is more in the early clock, is better, e.g. 6 bits key effect on the internal state (such as Fruit) is secure than only 1 bit effect (such as Sprout). Due to the key do not effect independently on the internal state of Sprout, the time-memory-date trade-off attack is applied successfully to Sprout [EK15, ZG15]. In [EK15], attacker suppose that for some consecutive clocks, key has no effect on the internal state. He saves correct guesses for internal state according to the assumption and keystream. In the online phase of the attack, attacker tries to retrieve the internal state and find key according to the keystream. If the attacker has access to the keystream that satisfy the assumption, he can refer to the storage and obtain internal state and key [EK15].

In Fruit, k'_t is independently used to prevent bypass of the key in $\mathbf{g}$ function, therefore there is no problem from this view.

Related-key Attack. There exist a weakness in the initialization procedure of all members of Grain family [LJSH08, DG13] and Sprout [Hao15]. 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 [AM15]. Nevertheless we propose a new scheme in the initialization procedure to strengthen Fruit against this attack. We did not load the IV bits directly in the internal state and did not combine the IV and key bits straightforward together (e.g., First bit of IV (i.e. v_0) is XORed with 11th bit of the output). We use key bits in the round key function and also we load in the LFSR and NFSR. Thus, the new idea increases the resistance against related-key attack. In Grain-v1, key bits are loaded in the NFSR and IV bits in the LFSR, and also key and IV bits use only one time and combine together directly [HJM07]. In Sprout, IV bits are loaded (in the LFSR and

NFSR) and it is possible that key bits used only one time in the during of the initialization [Hao15]. Thus, Fruit is resistant against this attack by exploiting the new idea and asymmetric padding in the loading IV bits.

Cube Attack. according to the suitable clock number of Fruit in the initialization procedure, it is too hard to find any low degree multiplicative expression (of some bits of the IV) based on the key in the Boolean function of the output. In Fruit, the length of LFSR and NFSR are shorter than Grain-v1, so with more number of the clocks in the initialization of Fruit, Fruit is stronger than Grain-v1against this attack. Therefore our design is resistant against to all types of Cube attack.

Algebraic Attack. this attack has not been applied to the Grain family, but a combination of this attack was applied to Sprout [MSBD15]. Short internal state (or actually weak round key function in Sprout) has caused to make this weakness. The round key function is strengthened and the key bits is independently involved in the **g** function, so Fruit is secure against this attack. Due to fast growth of the degree of polynomials in the internal state of Fruit, it is impossible for an attacker to apply pure algebraic attack. But here we discuss that a combination of guess and determine attack with 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 the keystream and one from the 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 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 higher than computational complexity of exhaustive attack.

Fault Attack. this attack is applied successfully to all members of Grain family [BMS12b, BMS12a] and also to Sprout [MSBD15, RM16], but we think that it is not applicable in the real world. Fault attack is based on some impractical hypotheses. An attacker should be able to induce fault in the cipher in a special time and supposes that the induced fault affect only on special section of the cipher, e.g. the attacker should be free for injecting a single bit fault in the NFSR just after initialization [RM16]. Another unrealistic hypothesis is that the attacker can reset the cipher and obtains the correct keystream (i.e. output with same key and IV, but without injecting of the fault).

We think this attack is not applicable in the real world, nevertheless it is possible to protect both LFSR and NFSR from injecting fault or use mirror or mask in the hardware implementation [BCC+09].

Weak key-IV. there are weak key-IVs in all members of Grain family and Sprout [ZW09, Ban15, RM16]. It is possible that all bits of the LFSR become zeros after initialization. In this situation, LFSR remains all zeros for all clocks and NFSR statistical properties will become non-random. As a result, the period of the cipher is unknown and the keystream is only dependent on NFSR bits and the cipher is too much vulnerable. This is very important with regard to shorter LFSRs in Fruit and Sprout. Fortunately, because we set l_{130} to 1 in the second step of the initialization and after that LFSR works independently, it is impossible that all bits of the LFSR become zeros in Fruit. Thus, there is no weak key-IV in Fruit.

4 Hardware implementation

The design of lightweight cipher is very important in industries, while we need 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, Sprout and Grain-v1, we implemented them 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.

In Table 1, we compare the area size of hardware implementation of Grain-v1, Sprout and Fruit. The area size of Fruit is significantly less than Grain-v1, as expected with regard to the length of internal state (the internal state of Grain-v1 is 160 bits for the FSR and 8 bits for the counter in initialization; but for Fruit is 80 bits for the FSR and 15 bits for the counter).

Due to Fruit initialization procedure is more complicated than initialization procedure of Grain-v1, we implemented the ciphers in two types, with and without considering any GE for initialization procedure. Table 1 shows that the area size of Grain-v1 is about 37% bigger than Fruit without considering to the initialization procedure. If we consider to the initialization procedure of both ciphers, we still have about 25% save in area size. The area size of Fruit is about 4.6% and 19.8% bigger than Sprout without considering to the initialization procedure and with considering to the initialization procedure respectively. We achieve the secure cipher in comparison with Sprout by the bigger area size.

Table 1: The synthesize of Fruit, Sprout and Grain-v1 on TSMC 0.18 μm technology process

Cipher	Implementation type	Area size(GE)
Grain-v1 [HJM07]	with initialization	1573
	without initialization	1258
Sprout [AM15]	with initialization	1048
	without initialization	877
Fruit	with initialization	1256
	without initialization	918

We did not dedicate any GE to the key bits, because key (for reuse with different IVs) should be stored in most applications, and note that IV bits are loaded into LFSR in Grain-v1 at once; but in Fruit, IV bits are XORed bit by bit with the output in the initialization step. This new idea for loading IV bits has some advantages. All bits of IV should be in access to start loading in Grain-v1 (and in many stream ciphers such as Sprout [AM15], Trivium [Can06] and WG-8 [FMG15]), but in Fruit (and in some stream ciphers such as A5/1 [Gol97] and MICKEY 2.0 [BD06]) initialization can be started as soon as the first bit of the IV is available. Thus, the speed of the initialization for Fruit is higher than Grain-v1 while IV is not available at once. This is important because in some cases IV bits are produced one by one.

We did not consider any memory for storing the IV bits in the case that all bits of IV are available at once. In this case, we suppose that there is a memory for storing of IV bits in the where that IV bits are produced in the out of Fruit.

Another advantage of Fruit's IV loading is more security against related-key attack (as we discussed before).

5 Conclusion

The storing of key for reuse by different IVs and the storing the key in a fixed memory (e.g. in a SIM card of mobile phone) is essential in most applications, therefor the use of

a secret key not only in the initialization, but also in the keystream generation [AM15]; is useful. This idea helps us to have a bigger internal state, thus we can design stronger ciphers without any extra cost in GE. We showed how we can exploit this the design by introducing Fruit. We presented a suitable way of using a fixed secret key as an internal state.

Fruit 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 is secure, but requires 41 more GE and also it is secure than Grain-v1 in some cases such as related-key attack and weak key-IV.

Table 2: The synthesize of Fruit, Sprout and Grain-v1 on TSMC 0.18 μm technology process

Cipher	Area size(GE)	Throughput $(Kb/s)^{\Omega}$	Platform	Source
Mickey [BD06]	3188	100	0.13 μm CMOS	[GB08]
Trivium [Can06]	2580	100	0.13 μm CMOS	[GB08]
Grain-v1 [HJM07]	1294	100	0.13 μm CMOS	[GB08]
Sprout [AM15]	813	100	0.18 μm CMOS	[AM15]
Sprout [AM15]	877	100	0.18 μm CMOS	Our work
Grain-v1 [HJM07]	1258	100	0.18 μm CMOS	Our work
Fruit	918	100	0.18 μm CMOS	Our work

$^{\Omega}$The throughput is for the clock with 100 KHz frequency

Table 2 shows that the area size of Fruit is 2270, 1662 and 340 GE less than Mickey , Trivium and Grain-v1 respectively.

Acknowledgement

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

References

[ADH+09] Jean-Philippe Aumasson, Itai Dinur, Luca Henzen, Willi Meier, and Adi Shamir. Efficient FPGA implementations of high-dimensional cube testers on the stream cipher grain-128. *IACR Cryptology ePrint Archive*, 2009:218, 2009.

[ÅHJM11] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of grain-128 with optional authentication. *IJWMC*, 5(1):48–59, 2011.

[AM15] Frederik Armknecht and Vasily Mikhalev. On lightweight stream ciphers with shorter internal states. In *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 451–470. Springer, 2015.

[Ban15] Subhadeep Banik. Some results on sprout. In *INDOCRYPT*, volume 9462 of *Lecture Notes in Computer Science*, pages 124–139. Springer, 2015.

[BCC+09] Alexandre Berzati, Cecile Canovas, Guilhem Castagnos, Blandine Debraize, Louis Goubin, Aline Gouget, Pascal Paillier, and Stephanie Salgado. Fault analysis of grain-128. In *Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on*, pages 7–14. IEEE, 2009.

[BD06] Steve Babbage and Matthew Dodd. The stream cipher mickey 2.0. *ECRYPT Stream Cipher, Available at http://www. ecrypt. eu. org/stream/p3ciphers/mickey/mickey p3. pdf*, 2006.

[BMS12a] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A differential fault attack on grain-128a using macs. In *SPACE*, volume 7644 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2012.

[BMS12b] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A differential fault attack on the grain family of stream ciphers. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 122–139. Springer, 2012.

[Can06] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *ISC*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2006.

[DG13] Lin Ding and Jie Guan. Related key chosen IV attack on grain-128a stream cipher. *IEEE Trans. Information Forensics and Security*, 8(5):803–809, 2013.

[DGP+11] Itai Dinur, Tim Güneysu, Christof Paar, Adi Shamir, and Ralf Zimmermann. An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 327–343. Springer, 2011.

[DS11] Itai Dinur and Adi Shamir. Breaking grain-128 with dynamic cube attacks. In *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 167–187. Springer, 2011.

[EK15] Muhammed F. Esgin and Orhun Kara. Practical cryptanalysis of full sprout with TMD tradeoff attacks. In *SAC*, volume 9566 of *Lecture Notes in Computer Science*, pages 67–85. Springer, 2015.

[FMG15] Xinxin Fan, Kalikinkar Mandal, and Guang Gong. WG-8: A lightweight stream cipher for resource-constrained smart devices. *ICST Trans. Security Safety*, 2(3):e4, 2015.

[GB08] Tim Good and Mohammed Benaissa. Hardware performance of estream phase-iii stream cipher candidates. In *Proc. of Workshop on the State of the Art of Stream Ciphers (SACSŠ08)*, 2008.

[Gol97] Jovan Dj. Golic. Cryptanalysis of alleged A5 stream cipher. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 1997.

[Hao15] Yonglin Hao. A related-key chosen-iv distinguishing attack on full sprout stream cipher. *IACR Cryptology ePrint Archive*, 2015:231, 2015.

[HJM07] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *IJWMC*, 2(1):86–93, 2007.

[HJMM06] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *IEEE International Symposium on Information Theory (ISIT 2006)*. Citeseer, 2006.

[KMN10] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of nlfsr-based cryptosystems. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.

[LJSH08] Yuseop Lee, Kitae Jeong, Jaechul Sung, and Seokhie Hong. Related-key chosen IV attacks on grain-v1 and grain-128. In *ACISP*, volume 5107 of *Lecture Notes in Computer Science*, pages 321–335. Springer, 2008.

[LLW16] Lang Li, Botao Liu, and Hui Wang. Qtl: A new ultra-lightweight block cipher. *Microprocessors and Microsystems*, 2016.

[LN15] Virginie Lallemand and María Naya-Plasencia. Cryptanalysis of full sprout. In *CRYPTO (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 663–682. Springer, 2015.

[Max06] Alexander Maximov. Cryptanalysis of the "grain" family of stream ciphers. In *ASIACCS*, pages 283–288. ACM, 2006.

[MGPI12] Miodrag J. Mihaljevic, Sugata Gangopadhyay, Goutam Paul, and Hideki Imai. Generic cryptographic weakness of k-normal boolean functions in certain stream ciphers and cryptanalysis of grain-128. *Periodica Mathematica Hungarica*, 65(2):205–227, 2012.

[MSBD15] Subhamoy Maitra, Santanu Sarkar, Anubhab Baksi, and Pramit Dey. Key recovery from state information of sprout: Application to cryptanalysis and fault attack. *IACR Cryptology ePrint Archive*, 2015:236, 2015.

[RM16] Dibyendu Roy and Sourav Mukhopadhyay. Fault analysis and weak key-iv attack on sprout. *IACR Cryptology ePrint Archive*, 2016:207, 2016.

[SIH+11] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357. Springer, 2011.

[Sta10] Paul Stankovski. Greedy distinguishers and nonrandomness detectors. In *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 210–226. Springer, 2010.

[ZG15] Bin Zhang and Xinxin Gong. Another tradeoff attack on sprout-like stream ciphers. In *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 561–585. Springer, 2015.

[ZW09] Haina Zhang and Xiaoyun Wang. Cryptanalysis of stream cipher grain family. *IACR Cryptology ePrint Archive*, 2009:109, 2009.

Appendix A: Test vector of Fruit

Due to in each clock one bit will be produced, we use hexadecimal format for presenting the keystream, key and IV. First bit of the keystream (i.e. z_{210})is most significant bit as follow.

$K = \{k_0k_1k_2k_3, k_4k_5k_6k_7, k_8k_9k_{10}k_{11}, \ldots, k_{72}k_{73}k_{74}k_{75}, k_{76}k_{77}k_{78}k_{79}\}$
$IV = \{00v_0v_1, v_2v_3v_4v_5, v_6v_7v_8v_9, \ldots, v_{62}v_{63}v_{64}v_{65}, v_{66}v_{67}v_{68}v_{69}\}$
$Z = \{z_{210}z_{211}z_{212}z_{213}, z_{214}z_{215}z_{216}z_{217}, z_{218}z_{219}z_{220}z_{221}, \ldots\}$

First test vector:

$k = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$Z = \{d,c,2,8,b,9,5,6,7,1,7,9,f,b,e,7,7,d,0,b\}$

Second test vector:

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

$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$Z = \{c,1,d,e,6,2,c,2,d,1,c,0,3,a,6,a,0,0,3,0\}$

Third test vector:

$k = \{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$
$Z = \{6,2,c,e,4,a,5,c,5,8,9,2,9,e,7,0,3,1,c,4\}$

Fourth test vector:

$k = \{1,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4\}$
$IV = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$
$Z = \{3,5,a,5,d,e,f,b,6,f,c,8,7,e,0,8,d,4,9,6\}$