Fruit: Ultra-Lightweight Stream Cipher with Shorter Internal State

Vahid Amin Ghafari, Honggang Hu and Ying Chen

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. A few lightweight stream ciphers were introduced for hardware applications in the eSTREAM project. In FSE 2015, while presenting a new idea (i.e. the design of stream ciphers with the shorter internal state by using a secret key, not only in the initialization but also in the keystream generation), Sprout was proposed. Unfortunately, Sprout is insecure. Because Grain-v1 is the lightest cipher in the portfolio of the eSTREAM project, we introduce Fruit as a successor of the Grain-v1 and Sprout. It is demonstrated that Fruit is safe and ultra-lightweight. The size of LFSR and NFSR in Fruit is only 80 bits (for 80-bit security level), while for resistance to the classical time-memory-data trade-off attack, the internal state size should be at least twice of the security level. To satisfy this rule and to design a concrete cipher, we used some new design ideas. The discussions are presented that Fruit can be more resistant than Grain-v1 to some attacks such as classical time-memory-data trade-off. The main objective of this work is to show how it is possible to exploit a secret key in the design to achieve a smaller area size.

Keywords: Stream Cipher · Ultra-lightweight · Lightweight · Grain · Sprout · NFSR · LFSR · Hardware Implementation

Introduction

Nowadays the need for secure lightweight symmetric ciphers is obviously more than at 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 and RFID are instances which lead us to accept the importance of designing new and secure lightweight ciphers. Stream ciphers with a minimal internal state are the best choice for the most applications with the less available resource.

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 a linear feedback shift register (LFSR) and a non-linear feedback shift register (NFSR). The linear section guarantees good statistical properties and large period, while the nonlinear section protects against the 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 suggested 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 practical point of view.

In FSE 2015 a new idea, the design of stream ciphers with the shorter internal state, was introduced [AM15]. Sprout stream cipher was proposed as an instance based on the new idea. A short while after Sprout was introduced, many attacks were published against

it [Hao15, LN15, MSBD15, Ban15, EK15, ZG15]. Although it 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 key not only in the initialization procedure but also in the keystream generation.

The storing of a key for reuse by different IVs is essential for the most applications, and also it is necessary to store a key in a fixed memory in some applications (in these cases, one fixed key is sufficient forever, e.g. in a RFID system or a SIM card of a mobile phone). It is a valuable idea that the stored key is also used in the design as a part of the internal state. This idea helps designers to extend the internal state to the key bits. Thus, it is possible to design ciphers with significantly smaller area size (i.e. ultra-lightweight ciphers), or, the idea helps us to achieve a bigger internal state and stronger ciphers.

On the other hand, there is still a benefit in the cryptosystems that they do not need to store the key. The storing fixed bits (e.g. burn the key in a fuse) needs less area size in comparison to the storing bits in a volatile memory [AM15]. Because the storing a fixed key needs less area, it is possible to design stream ciphers with smaller area size. Thus, there is the advantage in the cases that it does not need to store the key in the cryptosystems.

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 would be proposed by this new idea very soon [EK15]. The design of secure cipher with the shorter internal state has been introduced as a challenging question since 2015.

The necessary condition for a stream cipher to be resistant against time-memory-data trade-off (TMDTO) attack is that the internal state size should be at least twice of its security level, such as Trivium, MICKEY 2.0, and Grain-v1. We show how we can exploit key in a design to achieve shorter internal state. We think that it is a new generation in the design of stream ciphers. In this paper, another reduced internal state stream cipher called Fruit is presented. Fruit is a successor of Grain family and Sprout. As we show, Fruit is secure and ultra-lightweight and can be more resistant than Grain-v1 to some attacks such as classical TMDTO attack, Cube attack, and related-key attack. Unlike the Grain family and Sprout, there is no weak key-IV in Fruit. The TMDTO attacks to Sprout are not directly related to the main design idea of Sprout. We present some discussions that the main design idea of Sprout (and Fruit) is not potentially weak to the classical TMDTO attack.

Because Grain-v1 is lightweight and there is no practical attack to it, we introduce Grain-v1 as a basis for identifying the ultra-lightweight stream ciphers. We call a stream cipher with the less than 80% GE (gate equivalents) of the hardware implementation of Grain-v1 (in the same condition such as same technology process and same compiler) as an ultra-lightweight stream cipher. A stream cipher with less than 1K GE is called ultra-lightweight cipher in [EK15]. As the GE is dependent on many factors, (and different implementations of a cipher have the different GEs), our definition is better.

We present the hardware implementation results and compare the area size of Fruit, Sprout, and Grain-v1. In our implementation, Grain-v1 requires 1269 GE and Fruit requires 990 GE. These results show that the area size of Grain-v1 is about 28.1% bigger than that of Fruit.

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

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

2. New scheme in the initialization procedure to strengthen against related-key attack (There are weaknesses in the initialization procedure of all members of Grain family [LJSH08, DG13] and Sprout [Hao15])

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

4. Increase the size of LFSR to achieve longer keystream in each loading

5. New lighter feedback function for NFSR and output function (in comparison with Grain-v1 and Sprout).

The rest of the paper is organized as follows. The design of Fruit and the design criteria are presented. Then, we show that 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 $(K\colon\ (k_i, 0 \leq i \leq 79))$ and 70-bit public Initial Value, $(IV\colon\ (v_i, 0 \leq i \leq 69))$. The maximum number of the keystream bits that can be produced from one key and IV is 2^{43} bits. Each key is recommended to be used less than 2^{15} times with different IVs. It is not acceptable to reuse IVs (i.e. the use an identical IV with different keys is prohibited). IVs should be produced in a random way.

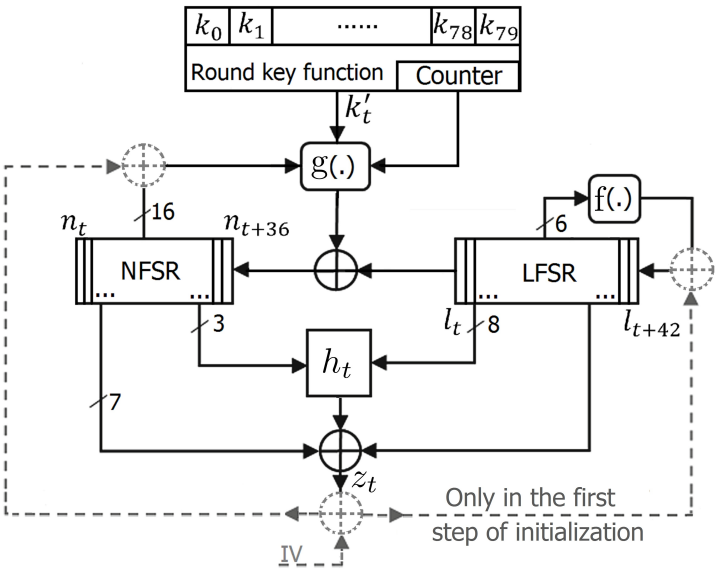

Figure 1: The Block Diagram of Fruit

Now we explain each part of the cipher in details:

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 the 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 the first clock, our counter is (000000000000000). Then, after the first clock, our counter is (000000100000001).

Round key function. we define some indexes of the key for exploiting in the round key function. The indexes are dependent on Cr counter and they change in the each clock. We introduce $s = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$, $y = (c_t^3 c_t^4 c_t^5)$, $u = (c_t^4 c_t^5 c_t^6)$, $p = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4)$, $q = (c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$, 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 the round key in each clock as follows..

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

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 the NFSR. The feedback function of the NFSR is as follows.

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

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

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

***h* function.** this function produces a pre-output stream from the LFSR and NFSR states as follows.

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

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

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

Initialization of the cipher. we extend IV bits to the 130 bits by concatenating 10 bits to the first and 50 bits to end of it. 1 bit one and 9 bit zeros are concatenated to the first of IV , and 50 bit zeros are concatenated to the end of IV, as follows.

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

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 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 the produced keystream is 2^{43} bits in each initialization because the period of the NFSR is a multiple of $2^{43} - 1$ (the period of the LFSR). We think that 1 terabyte is sufficient for the most applications because Fruit is special for hardware applications with the limited resource (e.g. WSN and RFID).

Round key function. the key bits should participate in the internal stats updating by considering two important criteria. First, the round key function should be lightweight in hardware. Second, it should be able to provide the appropriate participation of all bits of the key (as a part of the internal state) in the internal stats updating. It produces 2^7 different keys by involving 6 bits of the key. The 6 bits are changed in every clock uniformly. The most weaknesses of sprout are related to unsuitable round key function. If an attacker can (with guessing the internal state and known keystream) obtain some bits of k'_t, it is not easy to solve equation system due to the unknown counter (unknown index of the key in the function). 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.

Note that we copy the 6 bits of Cr from NFSR values and we clock 80 times after the copy. If the values of Cr had been copied from LFSR, there would have been a weakness (the LFSR works independently from other sections of the cipher in the second step of the initialization).

g function. the function that produces n_{t+37} , has been chosen in only 16 variables of the NFSR with regard to the light implementation in the 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 $2^3.3760$ and resiliency 2. The variables for the highest degree term have been 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 the produced string by LFSR with non-zero initial is maximum because the feedback polynomial is primitive. Since l_{130} is set to 1 after disconnecting feedback of output bit to the LFSR (in the end of the first step of initialization), we are sure that the period of the LFSR and NFSR is at least $2^{43} - 1$. Some attacks were proposed to the Grain family and Sprout from this weakness (i.e. it is possible that the LFSR becomes all zeros just before producing the first bit of the keystream in the Grain family and Sprout) [ZW09, Ban15].

number of clock in the initialization. it is very important to generate the monomial with maximal degree based on the key and IV variables in the initialization procedure, and it is very important distribution of it in all bits of the LFSR and NFSR. In this situation, the distribution of low degrees monomial is likely random. The numbers of clocks in the initialization procedure of Grain-v1 and Grain-128a are 160 and 256 respectively, and also the length of the NFSRs (or LFSR) are 80 and 128 respectively. It seems that 160

initial clocks are sufficient for Fruit, but 210 initial clocks are suitable (because IV bits is injected in the LFSR and NFSR one-by-one after 10 clocks, and also feedback of the output is disconnected in the second step of the initialization). 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 $2^8.976 = 249856$, and also to make a 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

In Sprout, the designers have not properly affected the key in the internal state updating (and consequently in the keystream generation). A coefficient (that its value is XORing of some bits of the LFSR and the NFSR) determines whether the key can affect the internal state or not in the clock. In the cases that, the coefficient is non-zero, only one bit of the key is effective at each clock. It is obvious that if a section of the internal state does not sufficiently participate in the internal state updating and the keystream generation, there will be a weakness in the stream cipher, especially with the minimal internal state. Thus, the key bits are not properly used as an internal state in Sprout. In Fruit, the 6 bits of the key are directly used in the internal state updating. The balance of participation of the key (as an internal state) is suitable in the internal state updating of Fruit. Here, the feasibility of applying some main attacks on Fruit is discussed.

Time-Memory-Data Trade-off Attack. it is well known that the cipher is weak to this attack if the size of its internal state is not at least twice of the security level. It means that the number of the possible internal states (i.e. space size) should be at least 2^{160} for Fruit after initialization (or after so many clocks) to resist against classical TMDTO attack. The key, LFSR, NFSR, and counter are used as an internal state in Fruit. It is obvious that the period of an LFSR is maximum in a non-zero state. As l_{130} is set to one in Fruit and after that, it works independently, we are sure that the space size of LFSR is $2^{43} - 1$. The space size of Cr is 2^7 because it is a counter. The space size of the key is 2^{80} because it can take any value. The period of the NFSR is multiple of the period of the LFSR in the structure similar to Grain-v1 and Fruit [HG11]. Thus, the space size of the NFSR is 2^{37}. Therefore, the effective internal states are 167 bits and it is suitable.

Note that the keys and IVs can provide a better cover for the space of the internal states just before producing the first bit of the keystream (i.e. the space of the internal states which produces the first bit of the keystream) in Fruit than that of Grain-v1. The size of the LFSR, NFSR, and Cr in Fruit (i.e. 87 bits register) is smaller than the size of the LFSR and NFSR in Grain-v1 (i.e. 160 bits register). Thus, the same size of the key and IV can better cover a space with smaller size. It is not provable that how much of the internal states of Fruit or Grain-v1 is covered with different keys and IVs, but this discussion shows that Fruit potentially is likely better than Grain-v1 against TMDTO attack from this point of view.

Another necessary condition to be resistant a stream cipher to TMDTO attack is the enough number of distinct classes of the internal states (that an initialization procedure can produce) [AM15]. The internal states that can be produced by a state are in the same class. All member of a class produce the keystreams that they are shifted together. There is only one class in an LFSR with non-zero initial (one state can produce all possible states by clocks). The number of the distinct classes of NFSR is unknown in the structure similar to Grain-v1. It is known that the NFSR is periodic (i.e. it is not ultimately periodic) and

also the period of the NFSR is multiple of the period of the LFSR in Grain-v1 [HG11]. In Fruit, because the key affects in all procedure (i.e. even in the keystream generation) and there are 2^{80} different keys, there are at least 2^{80} distinct classes in the internal states. It is an open problem that how many distinct classes there are in Grain-v1 (and Fruit), but If we divide the number of all possible internal states of Grain-v1 to the number of states in the smallest class of the internal state, we can obtain the maximum number of the distinct classes in the internal stats. The period of NFSR is dependent on the initial states of the LFSR and NFSR, for example, the period of NFSR for some initial states is $\alpha.(2^{80}-1)$, and for some other initial states is $\beta.(2^{80}-1)$ ... $(\alpha, \beta, ... \geqslant 1)$. One class contains all states that they are produced with an initial state for LFSR and NFSR. The smallest class is a class with the minimal period for the NFSR (and the minimal period is $2^{80}-1$ for the NFSR). There are $2^{80}-1$ states for LFSR and $2^{80}-1$ states for NFSR in the smallest class. The number of states in the smallest class is $2.(2^{80}-1)$. The number of all possible internal states of Grain-v1 is $2^{80}.(2^{80}-1)$. Thus, the maximum number of the distinct classes is 2^{79} (while there are at least 2^{80} distinct classes in the internal states).

Another condition to be resistant a stream cipher to TMDTO attack is the good sampling resistance [BSW00]. It means that an attacker cannot easily identify the internal states for producing the keystreams with special pattern (e.g. the keystreams that start with 10 zeros). An attacker should fix at least 5 variables of $\boldsymbol{h}$ function to linearize the output function and to easily identify the special internal states (it is at least 3 variables for Grain-v1). Therefore, it is suitable.

In the design of a stream cipher similar to Sprout and Fruit is very important how to benefit a key as an internal state in the internal state updating (after initialization procedure). It is obvious that all bits of the internal states (including the key for Sprout and Fruit) should affect the other internal states in a suitable ratio. The internal states excluding the key bits participate in the internal state updating proportionally and continually. Because the key does not affect independently (and proportionally) on the internal state of Sprout, the TMDTO attack was applied successfully to Sprout [EK15, ZG15]. In [EK15], the attacker suppose that for some consecutive clocks, a key has no effect on the internal states. He saves correct guesses for the internal states according to the assumption and keystreams. In the online phase of the attack, he tries to retrieve the internal states and find the key according to the keystreams. If the attacker has access to the keystreams that satisfy the assumption, he can refer to the storage and obtain the internal state and key [EK15]. The TMDTO attacks to the Sprout are not directly related to the main design idea of Sprout (i.e. using a key not only in initialization but also in keystream generation). It is related to the unsuitable round key function. In Fruit, k'_t is independently used to prevent bypassing of the key in $\mathbf{g}$ function, therefore there is no problem from this point of view.

All of the discussions was already discussed show that the main design idea of Sprout is not potentially weak to classical TMDTO attack, and also, Fruit can be more resistant than Grain-v1 to classical TMDTO attack.

Guess and Determine Attack. due to the 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. We strengthened the round key function, and use $n_{(t+36)}$ and n_t in the output function to prevent producing the keystream in the next and previous clock with an unknown key.

If an attacker wants to produce only the first 8 bits of the keystream, he needs to

guess all bits of the LFSR and NFSR. He can clock one time and identify the half of wrong candidates by known keystream and also, he can obtain some bits of k'_t in the next clocks. It is too hard for him to solve equations and obtain key bits with regard to the unknown number of Cr (unknown index of the key in the round key function). There are 128 different k'_t bits. Thus, the attacker should guess the bits of Cr.

Note that the 6 bits of the NFSR are copied to Cr during the initialization and after that, the cipher is clocked 80 times. It means that 13 times the value of the 6 bits are entirely changed based on the key, LFSR, and other bits of the NFSR. On the other hand, the one bit of the LFSR (i.e.l_{130}) is changed after the copy with probability $2^{(-1)}$. Therefore, it is too hard that attacker can obtain valuable information about Cr from the NFSR and LFSR.

If an attacker guesses all bits of the LFSR, NFSR, and Cr i.e. 86 bits (because l_{130} is set to 1 in the second step of the initialization, we suppose that the attacker needs to guess 42 bits of the LFSR), he can obtain the round key function bits. The attacker can obtain an equation in each clock and provide a nonlinear equation system based on the unknown key bits. He needs to clock at least 80 times the cipher. It is impossible for the attacker to identify wrong candidates of the internal state before 80 clocks (except in the first clock that he can identify half of wrong candidates). In this situation, the attacker should solve the nonlinear equation system, or he should try to identify wrong candidates (that both of them are not easy). Thus, the computational complexity of this attack is (at least) 80.2^{85}, which is higher than the complexity of exhaustive search attack. Thus, Fruit is resistant to this attack.

We analyzed that how an attacker can use the copied values in the second step of initialization in another scenario. If the attacker guesses all bits of the LFSR and NFSR in end of the first step of the initialization, he also has the values of Cr. If he considers the round key function bits as some unknown variables, he can obtain an equation system. In this scenario, the attacker guesses the 80 bits and he should clock 160 times (80 clocks for the second step of initialization and 80 clocks for using of all bits of the key). The degree of equations will be maximum (in the NFSR based on the unknown variables of round keys bits before the producing the first bit of output). Then, the attacker should solve another equation system for obtaining the key bits from the round key function bits. The complexity of solving these equation systems is much bigger than 2^{18} (the computational complexity of solving a linear equation system is $O(n^3)$). Finally, the time complexity of this attack is more than 2^{104}.

Linear Approximation Attack. this attack was applied to Grain-v0 [Max06]. In [Max06] has been discussed that if the NFSR and the output function are chosen with high nonlinearity and suitable resiliency, the ciphers similar to Grain-v0 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 the key is involved on the NFSR. The best linear approximation of the output has $2^{(-5.415)}$ bias as follows.

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

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

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

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).

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

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. Therefore, Fruit is resistant to this attack.

Related-key Attack. There are weaknesses in the initialization procedure of all members of the Grain family [LJSH08, DG13] and Sprout [Hao15, RM16]. Designers of Sprout ruled out the related-key attack. They believed this attack is not workable on Sprout because the key is fixed 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 to the 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 the key bits used only one time in the during of the initialization cite3. Thus, Fruit is resistant to this attack by exploiting the new idea and asymmetric padding in the loading IV bits.

Cube Attack. A type of this attack (i.e. dynamic Cube attack) was applied to Grain-128 [DS11, ADH+09] because the degree of NFSR feedback was low, i.e. 2. The degree of NFSR feedback was increased to 4 in Grain-128a to compensate this weakness [ÅHJM11]. According to the suitable clock number of Fruit in the initialization procedure and the degree of NFSR feedback, 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 that of Grain-v1, and also there are 11 variables in the $\boldsymbol{h}$ function of Fruit while there are 5 variables in the $\boldsymbol{h}$ function of Grain-v1. Thus, it is acceptable that the degree of key and IV variables in the initialization procedure of Fruit grows faster than that of Grain-v1. As the number of the clocks in the initialization of Fruit is more than that of Grain-v1, Fruit is likely stronger than Grain-v1 against this attack.

We implemented a Cube attack with Cube size of 32 bits (i.e. we XOR the first bits of the keystreams for all possible values of 32 bits of the IV, and we suppose that other bits of IV are zeros) for 100 and 80 initial clocks. The superpolys were not linear or constant over the key variables. These results show that Fruit (with 210 initial clocks) is resistant 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 (actually, weak round key function in Sprout) has caused to make this weakness. In Sprout, the key is not effective on the internal state updating (and consequently in the keystream generation) in the half of clocks. An attacker can guess some bits of the NFSR and LFSR and obtain equations for the keystream generation based on the unknown round key function bits. Then, he can solve equation system and obtain the key bits. In the clocks that the key does not affect the internal state updating, he has the chance to identify a bit of LFSR in the next clock of Sprout. Due to the direct effect of the key on the round key function of Fruit, this attack is not workable on Fruit. On the other hand, the attacker should guess

almost all bits of the LFSR and NFSR and Cr to obtain an equation system based on the key bits.

It is impossible for an attacker to apply the pure algebraic attack because the degree of polynomials in the internal state of Fruit grows very fast. Here we discuss that a combination of a guess and determine attack with an algebraic attack is not applicable to Fruit.

If an attacker guesses bits of NFSR, Cr and round key function, then he can obtain two equations in each clock (one from the keystream generation and one from the round key function). These equations are degree 2 and 3. 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, he should guess at least 42 bits of the round key function. Then, he can obtain the next bits of the round key function. Totally, the attacker should guess 37+7+42 bits and he should clock (at least) 80 times for each guess and should solve two non-linear equation system (one for LFSR and one for key). Thus, the computational complexity of the attack is more than computational complexity of exhaustive search attack.

Fault Attack. this attack has been applied successfully to all members of Grain family [BMS12b, BMS12a] and also to Sprout [MSBD15, RM16]. Fault attack is based on some impractical hypotheses. An attacker should be able to induce a fault in the cipher in a special time and supposes that the induced fault affects only a 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 in the hardware implementation from injecting fault by using mirror or mask in the hardware [BCC+09].

Weak key-IV. there are weak key-IVs in all members of Grain family [ZW09] and Sprout [Ban15, RM16]. It is possible that all bits of the LFSR become zeros before producing the first bit of the keystream. In this situation, the 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, the 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 the industries, while we need light ciphers in many fields such as WSN and RFID. Thus, our goal was to design a strong cipher with less than 80% GE of the hardware implementation of Grain-v1 (in the same condition) as an ultra-lightweight stream cipher. To get area size in hardware implementation for Fruit, Sprout, and Grain-v1, we simulated them by Modelsim SE 10.2c, and we used TSMC 90 nm technology process and the Synopsys Design Compiler 2013.03 sp2 version for synthesis and optimization. The hardware complexity is same both on the clock frequency of 100 KHz and 100 MHz.

In Table 1, we compare the area size of the 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 the initialization, but for Fruit is 80 bits for the

FSR and 15 bits for the counter). Note that one GE is equivalent to the area of a 2-way NAND gate.

Table 1 shows that the area size of Grain-v1 is about 28.1% bigger than that of Fruit in our results. The area size of Fruit is bigger than that of Sprout while Sprout is insecure (regard to the practical attack to Sprout, Grain-v1 is a suitable basis for comparison the area sizes).

Table 1: The area size for eSTREAM finalists, Sprout, and Fruit in hardware implementation

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]	839	100	90 nm CMOS	Our work
Grain-v1 [HJM07]	1269	100	90 nm CMOS	Our work
Fruit	990	100	90 nm CMOS	Our work

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

We did not dedicate any GE to the key bits (such as implementation of Sprout in [AM15]), because key (for reuse with different IVs) should be stored in the most applications and it is possible to store the fixed key bits in a non-volatile memory (e.g. burn the key in a fuse).

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 IV bits are produced one by one in some cases. 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 the 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 already mentioned).

5 Conclusion

The storing of a key for reuse by different IVs or the storing of a key in a fixed memory (e.g. in a RFID system or SIM card of a mobile phone) is essential in the most applications. Therefore, it is valuable idea that the stored key is also used in the design as a part of the internal state. This idea helps us to design ciphers with significantly smaller area size (i.e. ultra-lightweight ciphers). It is shown how to exploit a key as a part of the internal state in the design of a safe stream cipher by introducing Fruit.

Fruit in comparison with Grain-v1 is ultra-lightweight in hardware implementation, and with regard to Grain-v1 is the lightest candidate in the eSTREAM finalist of the hardware profile, it is obvious that design of secure stream ciphers such as Fruit is attractive. We demonstrated that Fruit, unlike Sprout, is secure, and we discussed that Fruit can be more resistant to some attacks (such as classical TMDTO attack and related-key attack) than Grain-v1.

6 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.

[BSW00] Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.

[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.

[HG11] Honggang Hu and Guang Gong. Periods on two kinds of nonlinear feedback shift registers with time varying feedback functions. *Int. J. Found. Comput. Sci.*, 22(6):1317–1329, 2011.

[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

We use hexadecimal format for presenting the keystream, key, and IV because in each clock one bit will be produced. The first bit of the keystream (i.e. z_{210}) is the most significant bit as follows.

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

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

$Z = \{z_{210}z_{211}z_{212}z_{213},\ z_{214}z_{215}z_{216}z_{217},\ z_{218}z_{219}z_{220}z_{221}, \ldots\}$

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:

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

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

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

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:

$k = \{1,2,3,4,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4\}$

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

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