

Cryptanalysis of Fruit-F: Exploiting Key-Derivation Weaknesses and Initialization Vulnerabilities

Subhadeep Banik¹ and Hailun Yan^{2,3*}

¹ Università della Svizzera Italiana, Lugano, Switzerland
`subhadeep.banik@usi.ch`

² University of Chinese Academy of Sciences, Beijing, China
`hailun.yan@ucas.ac.cn`

³ State Key Laboratory of Cryptology, P. O. Box 5159, Beijing 100878, China

Abstract. Fruit-F is a lightweight short-state stream cipher designed by Ghafari et al. The authors designed this version of the cipher, after earlier versions of the cipher viz. **Fruit 80/v2** succumbed to correlation attacks. The primary motivation behind this design seemed to be preventing correlation attacks. **Fruit-F** has a Grain-like structure with two state registers of size 50 bits each. In addition, the cipher uses an 80-bit secret key and an 80-bit IV. The authors use a complex key-derivation function to update the non-linear register which prevents the same key-bit alignment across fixed-length window of keystream bits, which is essentially what stops the correlation attacks.

In this paper, we first present two attacks against **Fruit-F**. The first attack stems from the fact that the key-derivation can be rewritten as the Boolean xor of two key-dependent terms one of which is the Boolean OR of two bits of the key. Using this we show that the cipher does not offer 80-bit security: the effective key space of **Fruit-F** is slightly less than 2^{80} , i.e. a simple brute force attack costs around $2^{80} - 2^{49}$ time. The second is a differential attack using the cipher’s complex initialization process. We show that under some given conditions, it is possible to have two initial vectors V_1 and V_2 that produce identical keystream vectors with any given key. Using this as a distinguisher, it is possible to collect enough linear and quadratic equations of the secret key to find it in practical time with very few keystream bits.

Keywords: Stream cipher · Fruit-F · Key-derivation weakness · Differential attack

1 Introduction

Stream ciphers are an essential class of symmetric cryptographic algorithms widely used in secure communications, particularly in resource-constrained environments. They face various cryptanalytic attacks targeting keystream generation, including

* Corresponding author

cube attacks [16, 25, 34], correlation attacks [14, 29, 33], algebraic attacks [1, 2, 15, 31, 32, 35], time-memory tradeoff attacks [21], etc.

Ever since the stream cipher **Sprout** was proposed in FSE 2015 [5], cryptographic literature has witnessed the designs of several stream ciphers with short internal states. A short-state stream cipher is one whose internal state in bits is less than one and a half times the size of its key. This was counter-intuitive since after [13], it was widely accepted that to be secure against generic time-memory-data tradeoff (TMDTO) attacks, the internal state of a stream cipher needed to be at least twice the size of the secret key. However **Sprout** was secure against generic TMDTO attacks like the one in [13]. The state update function of **Sprout** required additional input from the secret key, so the effective internal state was still double the size of the key. Stream ciphers with short states make for hardware implementations with low gate count, and hence important in the discipline of lightweight cryptography. Although **Sprout** was broken in subsequent papers [6, 17, 24, 26, 27, 36], there have been lots of research into designing secure stream ciphers with short internal states: **Lizard** [20], **Plantlet** [30], **Atom** [9], **Draco** [23] are some of the constructions that have been recently designed.

While these ciphers are attractive for lightweight cryptographic applications and various efforts have been made to enhance their security, they continue to present significant challenges in resisting various cryptanalytic attacks [4, 7, 8, 10, 11, 12, 28, 37]. For example, **Lizard** has been analyzed extensively by Maitre et al. [28] and Banik et al. [11], who demonstrated its vulnerability to distinguishing and key recovery attacks. Similarly, the work of Beighton et al. [12] analyzed the key injection mechanism in Grain-like ciphers and its implications for security.

Fruit Family of Stream Ciphers. The first design proposed in the Fruit family was **Fruit v2** [18]. This design was cryptanalyzed in [22] in which the attackers provided generic distinguishers for complexity significantly smaller than that of an exhaustive key search. They also showed that there are at least 2^{64} weak keys, each of which did not provide 80-bit security as claimed by designers. The designers addressed these shortcomings and followed it up with the updated construction **Fruit 80** [3]. In [19], Ghafari et al. proposed **Fruit-F**, which is a modified version of the Fruit stream cipher, claimed to be lighter than other lightweight stream ciphers. The authors claimed 80-bit security under the condition that the total number of keystream bits produced for each Key-IV pair was limited to 2^{22} bits.

Organization of the Paper. The rest of this paper is structured as follows: In Section 2, we begin by briefly introducing the Fruit-F stream cipher. Section 3 and Section 4 present two attacks against Fruit-F. Finally, we summarize our results in Section 5.

2 Preliminaries

Fruit-F uses an 80-bit key ($K : (k_0, \dots, k_{79})$) and 80-bit Initial Value ($IV : (v_0, \dots, v_{79})$). The internal state consists of a 50-bit LFSR ($l_t, \dots, l_{t+49}$), and 50-bit NFSR ($n_t, \dots, n_{t+49}$). The cipher uses the following procedures as components.

- **Round key function.** Set:

$$r = (l_{t+49}l_{t+33}n_t n_{t+44});$$

$$p = (l_{t+41}l_{t+7}n_{t+49}n_{t+5}n_{t+20});$$

$$q = (l_{t+25}l_t n_{t+37}n_{t+13}n_{t+26});$$

Combine five bits of the key to obtain the round key bits in each clock as follows:

$$\text{kup} = k_r \oplus k_{p+16} \oplus k_{q+48} \oplus k_{p+16} \cdot k_{q+48}.$$

- **nup function.** The feedback function of the NFSR is as follows:

$$\begin{aligned} n_{t+50} = & \text{kup} \oplus l_t \oplus n_t \oplus n_{t+11} \oplus n_{t+30} \oplus n_{t+16} \cdot n_{t+32} \oplus n_{t+25} \cdot n_{t+42} \\ & \oplus n_{t+4} \cdot n_{t+45} \oplus n_{t+7} \cdot n_{t+20} \cdot n_{t+35} \oplus n_{t+40} \cdot n_{t+44} \cdot n_{t+47} \cdot n_{t+48}. \end{aligned}$$

- **lup function.** The feedback function of the LFSR is as follows:

$$l_{t+50} = l_t \oplus l_{t+8} \oplus l_{t+16} \oplus l_{t+24} \oplus l_{t+34} \oplus l_{t+43}.$$

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

$$h_t = l_{t+11} \cdot l_{t+37} \oplus l_{t+1} \cdot l_{t+19} \oplus n_{t+24} \cdot l_{t+28} \oplus n_{t+9} \cdot n_{t+49} \oplus n_{t+33} \cdot n_t \cdot l_{t+49}.$$

- **Output function.** The output stream is produced by 7 bits from the NFSR, 1 bit from the LFSR, and the output of h function, as follows:

$$z_t = h_t \oplus n_{t+1} \oplus n_{t+17} \oplus n_{t+28} \oplus n_{t+41} \oplus n_{t+48} \oplus l_{t+45}.$$

- **Initialization of the cipher.** In the initialization procedure, IV bits are loaded into the NFSR and LFSR from LSB to MSB (v_0 to n_0 , v_1 to n_1 , $\dots$, v_{49} to n_{49} , v_{50} to l_0 , v_{51} to l_1 , $\dots$, v_{79} to l_{29}). The other bits of the LFSR are set to ones and one zero ($l_{30} = l_{31} = \dots = l_{48} = 1, l_{49} = 0$). The cipher is clocked 128 times, but before each clock, the output bits are fed to the NFSR and LFSR.

Then, the output feedback in the LFSR and NFSR is disconnected, and if $l_{128} = 0$, the feedback function of the LFSR is XORed by one. If $l_{128} = 1$, the feedback function of the LFSR is not changed. The cipher can produce the first keystream bit in the next clock, i.e., z_{128} .

A block diagram of Fruit-F is presented in Fig. 1.

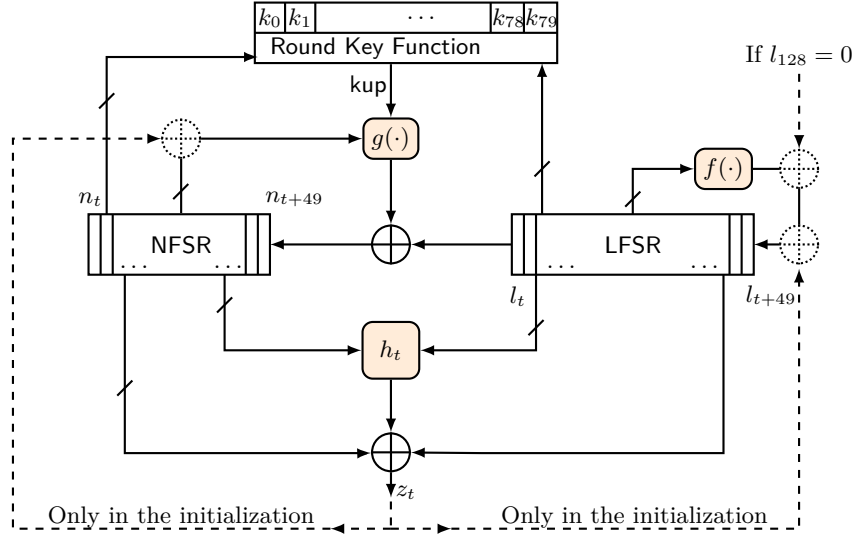


Fig. 1: The block diagram of Fruit-F

3 Attack I

In this section, we show that the cipher does not offer 80-bit security. In fact the actual key space that needs to be searched in case of a brute force attack is slightly less than 2^{80} . We begin with an observation which we state in the following property (which follows from the Boolean identity: $A \oplus B \oplus AB = A \vee B$).

Property 1. $k_r \oplus k_{p+16} \oplus k_{q+48} \oplus k_{p+16} \cdot k_{q+48} = k_r \oplus (k_{p+16} \vee k_{q+48})$.

With this property, we can reduce the key space. For any 16-bit vector $w = [w_0, w_1, \dots, w_{15}] \in \{0, 1\}^{16}$, consider two key sets:

$$S_w = \{k = (\underbrace{w_0, w_1, \dots, w_{15}}_{16 \text{ bits}}, \underbrace{1, 1, \dots, 1}_{32 \text{ bits}}, \underbrace{\star, \star, \dots, \star}_{32 \text{ bits}})\},$$

$$S'_w = \{k = (\underbrace{w_0, w_1, \dots, w_{15}}_{16 \text{ bits}}, \underbrace{\star, \star, \dots, \star}_{32 \text{ bits}}, \underbrace{1, 1, \dots, 1}_{32 \text{ bits}})\},$$

where the first 16 bits in the two sets are the same, the (17–48)-th bits are all one in set S , the (49–80)-th bits are arbitray in set S (denoted by $\star$), the (49–80)-th bits are all one in set S' , and the (17–48)-th bits are arbitray in set S' (denoted by $\star$). It is easy to see that both $|S_w| = |S'_w| = 2^{32}$ for all w . Given any w , the keys $K = (k_0, k_1, \dots, k_{79}) \in S_w$ and $K' = (k_0, k'_1, \dots, k'_{79}) \in S'_w$ generate identical round key bits in each clock. This is because:

$$\begin{cases} 0 \leq r \leq 16, 16 \leq p+16 < 48, 48 \leq q+48 < 80, \\ k_r = k'_r = w_r, \text{ and} \\ k_{p+16} \vee k_{q+48} = 1 \text{ (since } k_{p+16} = 1), \\ k'_{p+16} \vee k'_{q+48} = 1 \text{ (since } k'_{q+48} = 1). \end{cases}$$

For example, $K = 0x \text{ ABCD FFFFFFFF } 12345678$ and $K' = 0x \text{ ABCD } 87654321 \text{ FFFFFFFF}$ will generate the same round key bits. Since they generate identical round key bits, they will of course generate identical keystream bits too, since the key only affects the state update through the round key function. Therefore, given any w , we have a collection of $2^{33} = |S_w| + |S'_w|$ keys which generate identical key stream with any constant IV . In the case of a brute force attack, testing one representative of each of 2^{33} keys is enough. Also, note that there are 2^{16} such key equivalence (one for each value of w), the time complexity of the brute force attack reduces to $2^{80} - 2^{49} + 2^{16}$. This is strictly less than 2^{80} , although the difference is rather small.

4 Attack II

In this section, we present the second attack, which is a differential attack using the cipher's complex initialization process. We show that under some given conditions, two initial vectors V_1 and V_2 can produce identical keystream vectors with any given key.

In [19, Section 5, "Resynchronization Attack"], the designers had correctly observed that a single bit difference introduced in the 0^{th} and 50^{th} IV bit i.e. v_0 and v_{50} , then the difference so introduced in the state during initialization, can not be tracked with certainty after a few rounds, and hence the two IV pairs thus do not produce related keystreams that can be used to obtain a distinguisher. However, we choose 2 different IVs with the difference in v_1 and v_{51} . In this case, the differences are naturally introduced in the second bit of NFSR (namely $\Delta n_1 = 1$) and the second bit of LFSR (namely $\Delta l_1 = 1$), respectively during the initialization process. Further, we let the IV bits $v_{34} = v_{69} = 1$ which sets the following state bits to 1 during initialization.

$$l_{19} = 1, \quad (1)$$

$$n_{34} = 1. \quad (2)$$

Finally we choose v_{61} (which translates to the state bit l_{11}) so that it equals the following expression:

$$l_{11} = l_1 \oplus n_{24} \cdot l_{28} \oplus n_9 \cdot n_{49} \oplus n_1 \oplus n_{17} \oplus n_{28} \oplus n_{41} \oplus n_{48} \oplus l_0 \oplus l_8 \oplus l_{16} \oplus l_{24}. \quad (3)$$

Note that all the bits on the right side of the above expression are also initialized by IV bits, and so the attacker can set the value of the left side easily. With this pair of the chosen IVs (i.e. with differences in bit 1, 51 and which additionally satisfy the above expressions), we will see that,

- **if** we get equal keystreams, we must have $\Delta \text{kup} (t = 1) = 1$,
- **else if** we get unequal keystreams, we must have $\Delta \text{kup} (t = 1) = 0$.

Here $\Delta \text{kup} (t = t_0)$ denotes the difference of values of kup for the two IVs, at the time $t = t_0$ of the initialization process. Let us follow the passage of the differential so introduced at two subsequent clock intervals.

Clock $t = 0$. The differential trail at $t = 0$ is shown pictorially in Figure 2, where the red blocks represent the locations with differences. We will see that

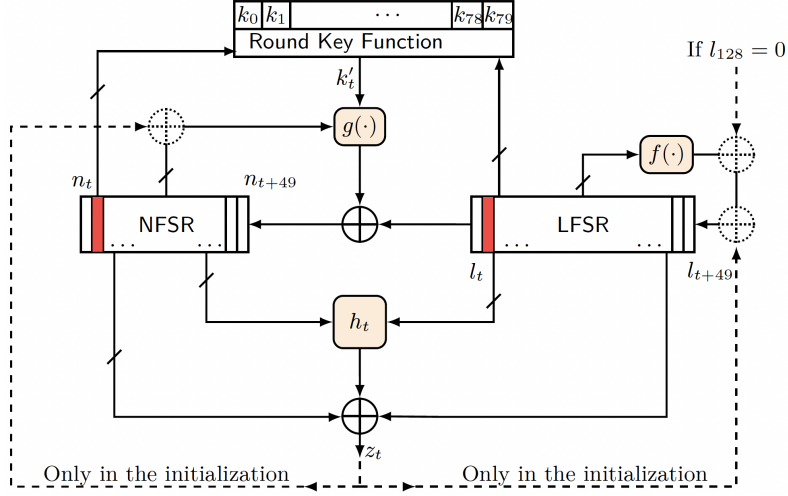


Fig. 2: Clock $t = 0$

there is no extra difference introduced in this clock, namely $\Delta l_{50} = 0, \Delta n_{50} = 0$. To compute $\Delta l_{50} = 0$, we compute the following bits at this clock:

$$h_0 = l_{11} \cdot l_{37} \oplus \textcolor{red}{l}_1 \cdot l_{19} \oplus n_{24} \cdot l_{28} \oplus n_9 \cdot n_{49} \oplus n_{33} \cdot n_0 \cdot l_{49},$$

$$z_0 = h_0 \oplus \textcolor{red}{n}_1 \oplus n_{17} \oplus n_{28} \oplus n_{41} \oplus n_{48} \oplus l_{45},$$

$$\Delta z_0 = \Delta h_0 \oplus \Delta n_1 = \Delta l_1 \oplus \Delta n_1 = 1 \oplus 1 = 0.$$

$$\text{lup} = l_0 \oplus l_8 \oplus l_{16} \oplus l_{24} \text{ (Since } l_{34} = l_{43} = 1 \text{ is part of pad),}$$

$$\Delta \text{lup} = 0,$$

$$l_{50} = \text{lup} \oplus z_0 = 1, \text{ (because of Eq 3),}$$

$$\Delta l_{50} = \Delta \text{lup} \oplus \Delta z_0 = 0.$$

To compute $\Delta n_{50} = 0$, we compute the following bits at this clock:

$$\text{kup} = k_r \oplus (k_{p+16} \vee k_{q+48}), \text{ where}$$

$$r = (l_{49} l_{33} n_0 n_{44}),$$

$$p = (l_{41} l_7 n_{49} n_5 n_{20}),$$

$$q = (l_{25} l_0 n_{37} n_{13} n_{26}).$$

$$\Delta \text{kup} = 0.$$

$$\begin{aligned}
\text{nup} &= n_0 \oplus n_{11} \oplus n_{30} \oplus n_{16} \cdot n_{32} \oplus n_{25} \cdot n_{42} \oplus n_4 \cdot n_{45} \oplus n_7 \cdot n_{20} \cdot n_{35} \oplus n_{40} \cdot n_{44} \cdot n_{47} \cdot n_{48}, \\
\Delta \text{nup} &= 0, \\
n_{50} &= \text{kup} \oplus l_0 \oplus \text{nup} \oplus z_0, \\
\Delta n_{50} &= \Delta \text{kup} \oplus \Delta l_0 \oplus \Delta \text{nup} \oplus \Delta z_0 = 0.
\end{aligned}$$

The above proves that no other difference is propagated to other state bits at $t = 0$.

Clock $t = 1$. The situation is shown graphically in Figure 3. In this clock, if there is still no extra difference introduced, namely if $\Delta l_{51} = 0, \Delta n_{51} = 0$, we will have the same key stream with the two chosen IVs. In this case, we must have $\Delta \text{kup} = 1$ as we will prove in the following. Let us consider the bits computed at this time. First,

$$h_1 = l_{12} \cdot l_{38} \oplus l_2 \cdot l_{20} \oplus n_{25} \cdot l_{29} \oplus n_{10} \cdot n_{50} \oplus n_{34} \cdot \textcolor{red}{n}_1 \cdot l_{50}.$$

Recall that $l_{50} = \text{lup} (t = 0) \oplus z_0$. With condition (3), we have $l_{50} = l_{45} = 1$. Combined with condition (2), $h_1 = n_1 \oplus \dots$, then

$$\Delta h_1 = \Delta n_1 = 1.$$

To compute Δl_{51} :

$$\begin{aligned}
z_1 &= h_1 \oplus n_2 \oplus n_{18} \oplus n_{29} \oplus n_{42} \oplus n_{49} \oplus l_{46}, \\
\Delta z_1 &= \Delta h_1 = 1; \\
\text{lup} &= l_1 \oplus l_9 \oplus l_{17} \oplus l_{25} \oplus l_{35} \oplus l_{44} = \textcolor{red}{l}_1 \oplus l_9 \oplus l_{17} \oplus l_{25} \quad (\text{Since } l_{35} = l_{44} = 1), \\
\Delta \text{lup} &= \Delta l_1 = 1, \\
l_{51} &= \text{lup} \oplus z_1, \\
\Delta l_{51} &= \Delta \text{lup} \oplus \Delta z_1 = 0.
\end{aligned}$$

To compute Δn_{51} :

$$\begin{aligned}
\text{nup} &= \textcolor{red}{n}_1 \oplus n_{12} \oplus n_{31} \oplus n_{17} \cdot n_{33} \oplus n_{26} \cdot n_{43} \oplus n_5 \cdot n_{46} \oplus n_8 \cdot n_{21} \cdot n_{36} \oplus n_{41} \cdot n_{45} \cdot n_{48} \cdot n_{49}, \\
\Delta \text{nup} &= \Delta n_1 = 1, \\
n_{51} &= \text{kup} \oplus l_1 \oplus \text{nup} \oplus z_1, \\
\Delta n_{51} &= \Delta \text{kup} \oplus \Delta l_1 \oplus \Delta \text{nup} \oplus \Delta z_1 = \Delta \text{kup} \oplus 1.
\end{aligned}$$

Fig 3 clearly shows that after this clock, both the differences which reside at the 0th bits of the NFSR and LFSR leave the state in the next cycle. Thus if the value of $\Delta n_{51} = 0$ (added with the fact that $\Delta l_{51} = 0$) this means that both the IVs result in identical state values from this point going forward and hence both the IVs produce identical keystream bits!! Since $\Delta n_{51} = \Delta \text{kup} \oplus 1$, if at clock $t = 1$, $\Delta n_{51} = 0$, then $\Delta \text{kup} = 1$. However, if $\Delta \text{kup} = 0$ at this clock, the difference is reintroduced in the NFSR via Δn_{51} and then it is highly unlikely that the IV pairs produce identical keystream. Thus the only condition needed to satisfy the identical keystream criterion is as follows:

$$\Delta \text{kup}(t = 1) = 1. \quad (4)$$

that n_{11} is a term that only linearly affects n_{50} ⁴. So, let's say we prepare 2 sets of related IVs (therefore 4 in total) which satisfy conditions (1)(2)(3), one in which $v_{11} = n_{11} = 1$ and the other in which $v_{11} = n_{11} = 0$, and check if the keystream in difference in the 2 sets of IVs is zero or not. Since n_{11} is a linear term, the value of y in one of the IV sets is A and in the other set $A + 4$. However it is not possible to say conclusively at this point in which set y takes which value.

Therefore the idea is that, for the chosen IVs with differences $\Delta n_1 = 1$ and $\Delta l_1 = 1$ and with conditions (1)(2)(3), we check if the key streams are equal or not for both $n_{11} = 0$ and $n_{11} = 1$. Then the value of y must be different in the two instances. We discuss three cases:

- The keystreams are equal for both the IV sets $n_{11} = 0$ and $n_{11} = 1$: Then as per Equation 5 we have

$$\begin{cases} k_x \oplus k_{x+2} \oplus (1 + k_A) \cdot (k_z \oplus k_{z+8}) = 1, \\ k_x \oplus k_{x+2} \oplus (1 + k_{A+4}) \cdot (k_z \oplus k_{z+8}) = 1. \end{cases} \quad (6)$$

This gives us 2 known quadratic equations in the key bit variables. Note that x , A , and z depend only on the IV and are hence known to the attacker.

- The keystreams are unequal for both $n_{11} = 0$ and $n_{11} = 1$:

$$\begin{cases} k_x \oplus k_{x+2} \oplus (1 + k_A) \cdot (k_z \oplus k_{z+8}) = 0, \\ k_x \oplus k_{x+2} \oplus (1 + k_{A+4}) \cdot (k_z \oplus k_{z+8}) = 0. \end{cases} \quad (7)$$

This gives us two more quadratic equations in the key bit variables.

- Otherwise the keystreams are equal for only one of the two IV sets: $\text{kup}(t = 1)$ (for $n_{11} = 1$) $\oplus$ $\text{kup}(t = 1)$ (for $n_{11} = 0$) = 1, which simplifies to $(k_A \oplus k_{A+4}) * (k_z \oplus k_{z+8}) = 1$, which equals to

$$\begin{cases} k_A \oplus k_{A+4} = 1, \\ k_z \oplus k_{z+8} = 1. \end{cases} \quad (8)$$

This gives us two linear equations in the key bit variables. Thus by testing only four IVs we get either a pair of linear or a pair of quadratic equations on the key bits!!

4.2 Attack steps

Let us recall the expressions for x , A , z :

$$x = n_{45} + 12,$$

$$A = 8 \cdot l_8 + 2 \cdot n_6 + n_{21} + 32,$$

$$z = 16 \cdot l_{26} + 4 \cdot n_{38} + 2 \cdot n_{14} + n_{27} + 48.$$

We have 8 free variables in these three expressions (each initialized by one IV bit): which can be varied to get $2^8 = 256$ sets of four IVs each as described in the previous subsections.

⁴ n_{11} is used as an example, but it is not the only option.

Note that by varying the free variables: x can take only 2 values 12, 13 and so $x+2$ takes two values 14, 15. A can take 8 values 32, 33, 34, 35, 40, 41, 42, 43, and $A+4$ takes the values 36, 37, 38, 39, 44, 45, 46, 47. And z can take a total of 16 values $[48, 55] \cup [64, 71]$ and $z+8$ takes values from the range $[56, 63] \cup [72, 79]$. Thus we have a total of 52 key variables involved and after the procedure we get $256 \times 2 = 512$ equations in these 52 variables (since each set gives us 2 equations).

The number of unique equations is not exactly 512, since for two different sets of free variables it is possible that we end up with the same pair of linear equations. The steps are formally stated in Algorithm 1 after which we get a list of equations L .

4.3 Next Steps

We note that in all the equations in L , (i) the k_z term always appears as xored with k_{z+8} , and (ii) the k_x term always appears as xored with k_{x+2} . As such it will not be possible to get a unique solution for k_x, k_z . The best we can hope for is a unique solution for the linear expressions $k_x \oplus k_{x+2}$ and $k_z \oplus k_{z+8}$. Accordingly define $\alpha_x = k_x \oplus k_{x+2}$ and $\beta_z = k_z \oplus k_{z+8}$. From the previous discussion we know that there are 2 expressions of the type α_x and 16 expressions of the type β_z , i.e.

$$\alpha_{12}, \alpha_{13}, \text{ AND } \beta_{48}, \beta_{49}, \dots, \beta_{55}, \text{ AND } \beta_{64}, \beta_{65}, \dots, \beta_{71}.$$

We can thus rewrite all the equations obtained so far as (with $A \in [32, 35] \cup [40, 43]$)

$$\begin{aligned} \alpha_x \oplus (1 + k_A) \cdot \beta_z &= 1, \\ \alpha_x \oplus (1 + k_{A+4}) \cdot \beta_z &= 1, \\ \alpha_x \oplus (1 + k_A) \cdot \beta_z &= 0, \\ \alpha_x \oplus (1 + k_{A+4}) \cdot \beta_z &= 0, \\ k_A \oplus k_{A+4} &= 1, \\ \beta_z &= 1. \end{aligned}$$

If we were to guess the values of these 18 expressions each equation in L becomes a linear equation in the 16 keybit variables k_{32} to k_{47} . This can be written in matrix notation $M \cdot \mathbf{k} = \mathbf{b}$, where M is a 512×16 matrix over $\text{GF}(2)$ and $\mathbf{k} = [k_{32}, k_{33}, \dots, k_{47}]^T$.

We ran experiments with around 10000 random values of the key. We found that whenever the values of the expressions α_x, β_z are incorrectly guessed, the above linear equation system is unsatisfiable, i.e. $\mathbf{b}$ is not in the column-space of M . On the other hand when the values are correctly guessed, the system is seldom full rank, i.e we do not get a unique solution for $\mathbf{k}$. By experimenting over 10000 random keys, we found that the dimension of the solution space on average is around $d \approx 4.65$, which means that the system has $T = 2^d$ solutions on average. Also note that if we guess the α_x, β_z values correctly, we have a total of 2^{18} assignments of the individual k_x, k_z values (for example if $\alpha_x = 0$ is correct, then both $k_x = k_{x+2} = 0$ and $k_x = k_{x+2} = 1$ need to be considered as

Algorithm 1: Populating Equation Bank

1 Finding a set of equations for Fruit-F

```

2 Randomly select initial vector  $v \in \{0, 1\}^{80}$ 
3 Let  $K$  be the secret key
4 for  $n_{45}l_8n_6n_{21}l_{26}n_{38}n_{14}n_{27} \in \{0, 1\}^8$  do
5    $l_{19} \leftarrow 1, n_{34} = 1$ 
6    $l_{11} \leftarrow l_1 \oplus n_{24} \cdot l_{28} \oplus n_9 \cdot n_{49} \oplus n_1 \oplus n_{17} \oplus n_{28} \oplus n_{41} \oplus n_{48} \oplus l_0 \oplus l_8 \oplus l_{16} \oplus l_{24}$ 
7   Set  $n_{11} \leftarrow 0$ 
8   Form an IV  $V_1$  using the above values
9   Let  $V_2 = V_1 \oplus$  difference in bit 1,51
10  if  $(K, V_1)$  and  $(K, V_2)$  generate an identical keystream then
11     $u_1 \leftarrow 1$ 
12  end
13  else
14     $u_1 \leftarrow 0$ 
15  end
16  Set  $n_{11} \leftarrow 1$ 
17  Form an IV  $V_1$  using the above values
18  Let  $V_2 = V_1 \oplus$  difference in bit 1,51
19  if  $(K, V_1)$  and  $(K, V_2)$  generate an identical keystream then
20     $u_2 \leftarrow 1$ 
21  end
22  else
23     $u_2 \leftarrow 0$ 
24  end
25  Let  $x = n_{45} + 12, A = 8 \cdot l_8 + 2 \cdot n_6 + n_{21} + 32$ 
26  Let  $z = 16 \cdot l_{26} + 4 \cdot n_{38} + 2 \cdot n_{14} + n_{27} + 48$ 
27  if  $u_1 = u_2 = 1$  then
28    Add  $k_x \oplus k_{x+2} \oplus (1 + k_A) \cdot (k_z \oplus k_{z+8}) = 1$  to list  $L$ 
29    Add  $k_x \oplus k_{x+2} \oplus (1 + k_{A+4}) \cdot (k_z \oplus k_{z+8}) = 1$  to list  $L$ 
30  end
31  if  $u_1 = u_2 = 0$  then
32    Add  $k_x \oplus k_{x+2} \oplus (1 + k_A) \cdot (k_z \oplus k_{z+8}) = 0$  to list  $L$ 
33    Add  $k_x \oplus k_{x+2} \oplus (1 + k_{A+4}) \cdot (k_z \oplus k_{z+8}) = 0$  to list  $L$ 
34  end
35  if  $u_1 \oplus u_2 = 1$  then
36    Add  $k_A \oplus k_{A+4} = 1$  to list  $L$ 
37    Add  $k_z \oplus k_{z+8} = 1$  to list  $L$ 
38  end
39 end

```

Algorithm 2: Finding the Key

```

40 Finding the secret key
   Input: The list of Equations  $L$ 
   Output: The secret key  $K$ 


---


41 for  $\chi = \alpha_{12}\alpha_{13}\beta_{48}\beta_{49} \cdots \beta_{55}\beta_{64}\beta_{65} \cdots \beta_{71} \in \{0, 1\}^{18}$  do
42   Assign the values of  $\chi$  to  $L$  to obtain linear system  $M \cdot \mathbf{k} = \mathbf{b}$ 
43   Solve the system using Gaussian elimination
44   if System is unsatisfiable then
45     End current iteration of For loop and try another value of the guess
46   end
47   else
48     Store all solutions in the set  $S$  along with the  $2^{18}$  values of the keybits
       that result in  $\chi$ 
49   end
50 end
51 /*Final Exhaustive Search*/
52 for All solutions  $s \in S$  do
53   for All the  $2^{28}$  guesses  $g$  of keybits not in  $L$  do
54     Construct the candidate key  $K$  using  $g$  and  $s$ 
55     if  $K$  produces the required keystream then
56       Return  $K$ 
57     end
58   end
59 end


---



```

candidate solutions). This results in the fact that after the end of this exercise we have a total of $T \cdot 2^{18} \approx 2^{22.65}$ solutions for the 52 keybits listed above. The final step that remains is to perform an exhaustive search with the 2^{28} guesses of the keybit variables not in L along with the $2^{22.65}$ solutions obtained above. This implies that the complexity of the final exhaustive search is $2^{28+22.65} \approx 2^{50.65}$. The process is stated formally in Algorithm 2.

5 Conclusion

In this paper, we present flaws in the lightweight cipher Fruit-F. We present two weaknesses: the first shows that a brute force attack in complexity slightly less than an exhaustive search may be mounted. The second shows that for a set of four judiciously chosen IVs it is possible to get two linear or quadratic equations in the key variables. Solving them the security of Fruit-F essentially reduces to less than 51 bits. We believe the problem in the design arose due to the simplistic nature of the key-selection function which can be expressed by using only an OR and an XOR gate. Designing good key-selection function is an open problem, and the designers ought to be careful that any Boolean function so selected does not leave a non-trivial subspace over which the function is greatly simplified.

It is noteworthy that the stream cipher Sprout [5] first opted for a non-linear key-select function and this was pointed out as a weakness in the attack mounted in [17]. Since then all short state cipher have opted for a strictly linear key-select function that prevents the above attack. Although it can not be concluded that all non-linear functions are weak, this requires more in-depth study and therefore left as an open problem.

Acknowledgements

We thank the reviewers of ACISP 2025 for insightful comments to improve the quality of this paper. Hailun Yan has been supported by the National Key Research and Development Program of China (2022YFB2701900), the National Natural Science Foundation of China (62202444), and the National Cryptologic Science Foundation of China.

References

1. Al-Hinai, S., Batten, L.M., Colbert, B.D., Wong, K.K.: Algebraic attacks on clock-controlled stream ciphers. In: Batten, L.M., Safavi-Naini, R. (eds.) *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings. Lecture Notes in Computer Science*, vol. 4058, pp. 1–16. Springer (2006). https://doi.org/10.1007/11780656_1, https://doi.org/10.1007/11780656_1
2. Al-Hinai, S., Dawson, E., Henriksen, M., Simpson, L.R.: On the security of the LILI family of stream ciphers against algebraic attacks. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings. Lecture Notes in Computer Science*, vol. 4586, pp. 11–28. Springer (2007). https://doi.org/10.1007/978-3-540-73458-1_2, https://doi.org/10.1007/978-3-540-73458-1_2
3. Amin Ghafari, V., Hu, H.: Fruit-80: A secure ultra-lightweight stream cipher for constrained environments. *Entropy* **20**(3) (2018). <https://doi.org/10.3390/e20030180>, <https://www.mdpi.com/1099-4300/20/3/180>
4. Anand, R., Nandi, R., Isobe, T.: Analysis of atom against quantum attacks. *IET Quantum Commun.* **5**(1), 88–102 (2024). <https://doi.org/10.1049/qtc2.12076>, <https://doi.org/10.1049/qtc2.12076>
5. Armknecht, F., Mikhalev, V.: On Lightweight Stream Ciphers with Shorter Internal States. In: Leander, G. (ed.) *FSE. Lecture Notes in Computer Science*, vol. 9054, pp. 451–470. Springer (2015), <http://dblp.uni-trier.de/db/conf/fse/fse2015.html#ArmknechtM15>; http://dx.doi.org/10.1007/978-3-662-48116-5_22
6. Banik, S.: Some Results on Sprout. In: Biryukov, A., Goyal, V. (eds.) *INDOCRYPT. Lecture Notes in Computer Science*, vol. 9462, pp. 124–139. Springer (2015), <http://dblp.uni-trier.de/db/conf/indocrypt/indocrypt2015.html#Banik15>; http://dx.doi.org/10.1007/978-3-319-26617-6_7
7. Banik, S.: Cryptanalysis of draco. *IACR Trans. Symmetric Cryptol.* **2022**(4), 92–104 (2022). <https://doi.org/10.46586/TOSC.V2022.I4.92-104>, <https://doi.org/10.46586/tosc.v2022.i4.92-104>

8. Banik, S., Barooti, K., Isobe, T.: Cryptanalysis of plantlet. *IACR Trans. Symmetric Cryptol.* **2019**(3), 103–120 (2019). <https://doi.org/10.13154/TOSC.V2019.I3.103-120>, <https://doi.org/10.13154/tosc.v2019.i3.103-120>
9. Banik, S., Caforio, A., Isobe, T., Liu, F., Meier, W., Sakamoto, K., Sarkar, S.: Atom: A stream cipher with double key filter. *IACR Trans. Symmetric Cryptol.* **2021**(1), 5–36 (2021). <https://doi.org/10.46586/tosc.v2021.i1.5-36>, <https://doi.org/10.46586/tosc.v2021.i1.5-36>
10. Banik, S., Collins, D., Meier, W.: Near collision attack against grain V1. In: Tibouchi, M., Wang, X. (eds.) *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part I. Lecture Notes in Computer Science*, vol. 13905, pp. 178–207. Springer (2023). https://doi.org/10.1007/978-3-031-33488-7_7, https://doi.org/10.1007/978-3-031-33488-7_7
11. Banik, S., Isobe, T., Cui, T., Guo, J.: Some cryptanalytic results on lizard. *IACR Trans. Symmetric Cryptol.* **2017**(4), 82–98 (2017). <https://doi.org/10.13154/TOSC.V2017.I4.82-98>, <https://doi.org/10.13154/tosc.v2017.i4.82-98>
12. Beighton, M., Bartlett, H., Simpson, L., Wong, K.K.: Key recovery attacks on grain-like keystream generators with key injection. In: Simpson, L., Bae, M.A.R. (eds.) *Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings. Lecture Notes in Computer Science*, vol. 13915, pp. 89–108. Springer (2023). https://doi.org/10.1007/978-3-031-35486-1_5, https://doi.org/10.1007/978-3-031-35486-1_5
13. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Trade-offs for Stream Ciphers. In: Okamoto, T. (ed.) *ASIACRYPT. Lecture Notes in Computer Science*, vol. 1976, pp. 1–13. Springer (2000), <http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2000.html#BiryukovS00>; http://dx.doi.org/10.1007/3-540-44448-3_1
14. Chepyzhov, V.V., Johansson, T., Smeets, B.J.M.: A simple algorithm for fast correlation attacks on stream ciphers. In: Schneier, B. (ed.) *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings. Lecture Notes in Computer Science*, vol. 1978, pp. 181–195. Springer (2000). https://doi.org/10.1007/3-540-44706-7_13, https://doi.org/10.1007/3-540-44706-7_13
15. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science*, vol. 2656, pp. 345–359. Springer (2003). https://doi.org/10.1007/3-540-39200-9_21, https://doi.org/10.1007/3-540-39200-9_21
16. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009, Proceedings. Lecture Notes in Computer Science*, vol. 5479, pp. 278–299. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_16, https://doi.org/10.1007/978-3-642-01001-9_16
17. Esgin, M.F., Kara, O.: Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. In: Dunkelman, O., Keliher, L. (eds.) *SAC. Lecture Notes in Computer Science*, vol. 9566, pp. 67–85. Springer (2015), <http://dblp.uni-trier.de/db/conf/sacrypt/sacrypt2015.html#EsginK15>; http://dx.doi.org/10.1007/978-3-319-31301-6_4

18. Ghafari, V.A., Hu, H., Chen, Y.: Fruit-v2: Ultra-lightweight stream cipher with shorter internal state. Cryptology ePrint Archive, Paper 2016/355 (2016), <https://eprint.iacr.org/2016/355>
19. Ghafari, V.A., Lin, F., Zhou, Z.: A new idea in response to fast correlation attacks on small-state stream ciphers. *Microprocess. Microsystems* **96**, 104720 (2023)
20. Hamann, M., Krause, M., Meier, W.: LIZARD - A Lightweight Stream Cipher for Power-constrained Devices. *IACR Trans. Symmetric Cryptol.* **2017**(1), 45–79 (2017), <http://dblp.uni-trier.de/db/journals/tosc/tosc2017.html#HamannKM17>; <http://tosc.iacr.org/index.php/ToSC/article/view/584>
21. Hamann, M., Krause, M., Meier, W., Zhang, B.: Time-memory-data tradeoff attacks against small-state stream ciphers. *IACR Cryptol. ePrint Arch.* p. 384 (2017), <http://eprint.iacr.org/2017/384>
22. Hamann, M., Krause, M., Meier, W., Zhang, B.: Design and analysis of small-state grain-like stream ciphers. *Cryptography and Communications* **10**(5), 803–834 (Sep 2018). <https://doi.org/10.1007/s12095-017-0261-6>, <https://doi.org/10.1007/s12095-017-0261-6>
23. Hamann, M., Moch, A., Krause, M., Mikhalev, V.: The draco stream cipher: A power-efficient small-state stream cipher with full provable security against tmdto attacks. *IACR Transactions on Symmetric Cryptology* **2022**(2), 1–42 (Jun 2022). <https://doi.org/10.46586/tosc.v2022.i2.1-42>, <https://tosc.iacr.org/index.php/ToSC/article/view/9712>
24. Hao, Y.: A related-key chosen-iv distinguishing attack on full sprout stream cipher. *IACR Cryptol. ePrint Arch.* p. 231 (2015), <http://eprint.iacr.org/2015/231>
25. He, J., Hu, K., Lei, H., Wang, M.: Massive superpoly recovery with a meet-in-the-middle framework - improved cube attacks on trivium and kreyvium. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part I. *Lecture Notes in Computer Science*, vol. 14651, pp. 368–397. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_13, https://doi.org/10.1007/978-3-031-58716-0_13
26. Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of Full Sprout. In: Gennaro, R., Robshaw, M. (eds.) *CRYPTO (1)*. *Lecture Notes in Computer Science*, vol. 9215, pp. 663–682. Springer (2015), <http://dblp.uni-trier.de/db/conf/crypto/crypto2015-1.html#LallemandN15>; http://dx.doi.org/10.1007/978-3-662-47989-6_32
27. Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key recovery from state information of sprout: Application to cryptanalysis and fault attack. *IACR Cryptol. ePrint Arch.* p. 236 (2015), <http://eprint.iacr.org/2015/236>
28. Maitra, S., Sinha, N., Siddhanti, A., Anand, R., Gangopadhyay, S.: A tmdto attack against lizard. *IEEE Transactions on Computers* **67**(5), 733–739 (2018). <https://doi.org/10.1109/TC.2017.2773062>
29. Meier, W., Staffelbach, O.: Fast correlation attacks on stream ciphers (extended abstract). In: Günther, C.G. (ed.) *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of Cryptographic Techniques*, Davos, Switzerland, May 25–27, 1988, Proceedings. *Lecture Notes in Computer Science*, vol. 330, pp. 301–314. Springer (1988). https://doi.org/10.1007/3-540-45961-8_28, https://doi.org/10.1007/3-540-45961-8_28
30. Mikhalev, V., Armknecht, F., Müller, C.: On Ciphers that Continuously Access the Non-Volatile Key. *IACR Trans. Symmetric Cryptol.* **2016**(2), 52–79 (2016), <http://dblp.uni-trier.de/db/journals/tosc/tosc2016.html#MikhalevAM16>; <http://tosc.iacr.org/index.php/ToSC/article/view/565>

31. Pillai, N.R., Bedi, S.S.: Algebraic attacks on a class of stream ciphers with unknown output function. *Des. Codes Cryptogr.* **69**(3), 317–330 (2013). <https://doi.org/10.1007/S10623-012-9665-7>, <https://doi.org/10.1007/s10623-012-9665-7>
32. Rønjom, S.: Improving algebraic attacks on stream ciphers based on linear feedback shift register over $\mathbb{F}_{2^k}$. *Des. Codes Cryptogr.* **82**(1-2), 27–41 (2017). <https://doi.org/10.1007/S10623-016-0212-9>, <https://doi.org/10.1007/s10623-016-0212-9>
33. Siegenthaler, T.: Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers* **34**(1), 81–85 (1985). <https://doi.org/10.1109/TC.1985.1676518>, <https://doi.org/10.1109/TC.1985.1676518>
34. Sun, Y.: Automatic search of cubes for attacking stream ciphers. *IACR Trans. Symmetric Cryptol.* **2021**(4), 100–123 (2021). <https://doi.org/10.46586/TOSC.V2021.I4.100-123>, <https://doi.org/10.46586/tosc.v2021.i4.100-123>
35. Wenxiao, Q., Siwei, S., Ying, C., Lei, H.: New coefficient grouping for complex affine layers. *Chinese Journal of Electronics* **34**(2), 1–13 (2025). <https://doi.org/10.23919/cje.2023.00.373>, <https://cje.ejournal.org.cn/en/article/doi/10.23919/cje.2023.00.373>
36. Zhang, B., Gong, X.: Another Tradeoff Attack on Sprout-Like Stream Ciphers. In: Iwata, T., Cheon, J.H. (eds.) *ASIACRYPT* (2). *Lecture Notes in Computer Science*, vol. 9453, pp. 561–585. Springer (2015), <http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2015-2.html#ZhangG15>; http://dx.doi.org/10.1007/978-3-662-48800-3_23
37. Zhou, Z., Ghafari, V.A., Liu, H.: Multiple sampling fast correlation attack on small state stream ciphers with limited round key period. *IACR Cryptol. ePrint Arch.* p. 877 (2024), <https://eprint.iacr.org/2024/877>