Another Tradeoff Attack on Sprout-Like Stream Ciphers

Bin Zhang[1,2(✉)] and Xinxin Gong[1(✉)]

[1] TCA Laboratory, SKLCS, Institute of Software,
Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
{zhangbin,gongxinxin}@tca.iscas.ac.cn

Abstract. Sprout is a new lightweight stream cipher with shorter internal state proposed at FSE 2015, using key-dependent state updating in the keystream generation phase. Some analyses have been available on eprint so far. In this paper, we extend the design paradigm in general and study the security of Sprout-like ciphers in a unified framework. Our new penetration is to investigate the k-normality of the augmented function, a vectorial Boolean function derived from the primitive. Based on it, a dedicated time/memory/data tradeoff attack is developed for such designs. It is shown that Sprout can be broken in 2^{79-x-y} time, given $\left[c \cdot (2x + 2y - 58) \cdot 2^{71-x-y}\right]$-bit memory and 2^{9+x+y}-bit keystream, where x/y is the number of forward/backward steps and c is a small constant. Our attack is highly flexible and compares favorably to all the previous results. With carefully chosen parameters, the new attack is at least 2^{20} times faster than Lallemand/Naya-Plasencia attack at Crypto 2015, Maitra et al. attack and Banik attack, 2^{10} times faster than Esgin/Kara attack with much less memory.

Keywords: Cryptanalysis · Stream ciphers · Sprout · Tradeoff

1 Introduction

Design of secure lightweight stream ciphers for constrained hardware environments is important both in theory and practice. The most area/power consuming component in a lightweight design is the number of memory gates, which corresponds to the internal state size of the primitive. On the other hand, a common rule of thumb for stream cipher design is that the internal state size should be at least twice as long as the key size to resist against time/memory/data (TMD) tradeoff attacks [4].

This design principal indeed works, and security analysis of the eSTREAM finalists, e.g., Grain v1, Mickey v2 and Trivium [7] evolves rather slowly. At FSE

This work is supported by the program of the National Natural Science Foundation of China (Grant No. 61572482) and the National Grand Fundamental Research 973 Program of China (Grant No. 2013CB338002).

T. Iwata and J.H. Cheon (Eds.): ASIACRYPT 2015, Part II, LNCS 9453, pp. 561–585, 2015.
DOI: 10.1007/978-3-662-48800-3_23

2015, another design paradigm for stream ciphers is proposed and instantiated by a new design, called Sprout, aiming to reduce the internal state size, thus the hardware area size by using key-dependent state updating in the keystream generation phase [2]. It is expected that the immunity against TMD tradeoff attacks will not be compromised.

Surprisingly, there have been some cryptanalysis of Sprout appearing on the IACR eprint monthly after ESC 2015 and FSE 2015. In the time order of the open literature, a related key chosen IV attack on Sprout is presented in [9], but the designers have already ruled out the related key model in [2]. Then the first attack in the single key model is found in [12] by using a list merging technique with a time complexity around 2^{69} Sprout encryptions at Crypto 2015. In [13], another attack based on a SAT solver is given with a complexity of 2^{54} attempts, where each attempt takes a time equivalent to $6.6 \cdot 2^{54} \cdot 2^e$ encryptions which is more than 2^{80} if $e > 23$. Thus, it is questionable whether this work in [13] translates into a feasible attack on Sprout or not. To directly challenging the design rationale, Esgin and Kara presented a TMD tradeoff attack in [8] with an online time complexity of 2^{33} Sprout encryptions and 770 TB of memory after a pre-computation around 2^{53} basic operations. Finally in [3], a key recovery attack is launched against Sprout with a complexity of $2^{66.7}$ Sprout encryptions together with some other analysis results.

In this paper, we extend the design paradigm in general and study the security of Sprout-like ciphers in a unified framework. The model involves the secret key not only in the initialization process but also in the non-linear state updating in a Sprout-like manner during the keystream generation phase. Then based on the notion of normality first introduced by Dobbertin in [6], we investigate the k-normality of the augmented function [5], a vectorial Boolean function derived from the underlying primitive. This property is relevant for the design and analysis of cryptosystems. In [14] and [15], security implications of k-normal Boolean functions are considered when they are employed in certain stream ciphers. We make a systematic security analysis based on this property for Sprout-like stream ciphers and develop a dedicated TMD tradeoff attack framework for such designs. In particular, it is shown that Sprout can be broken in 2^{79-x-y} time, given $\left\lceil c \cdot (2x + 2y - 58) \cdot 2^{71-x-y} \right\rceil$-bit memory and 2^{9+x+y}-bit keystream, where x is the number of forward steps, y is the number of backward steps and c is a small constant. Our attack is highly flexible and compares favorably to all the previous attacks on Sprout. With carefully chosen attack parameters, our method is at least 2^{20} times faster than Lallemand/Naya-Plasencia attack at Crypto 2015, Maitra et al. attack and Banik attack, 2^{10} times faster than Esgin/Kara attack with much less memory. Practical simulations confirmed our analysis.

This paper is structured as follows. In Sect. 2, the stream cipher Sprout is described and generalized to a generic Sprout-like model. In Sect. 3, based on a natural extension of normality from Boolean functions to vectorial Boolean functions, a generic TMD cryptanalysis framework of such ciphers is formalized with complexity analysis. In Sect. 4, the framework is applied to Sprout with comparisons to other attacks. Section 5 provides the experimental results. Finally, some conclusions are given in Sect. 6.

2 Sprout-Like Stream Ciphers

In this section, a brief description of Sprout that is relevant to our work and a generic Sprout-like model that inherits the design spirit are presented. The following notations will be used throughout the paper.

- $L^t = [l_t, l_{t+1}, ..., l_{t+39}]$, the internal state of the LFSR at time t.
- $N^t = [n_t, n_{t+1}, ..., n_{t+39}]$, the internal state of the NFSR at time t.
- $[a, b] \stackrel{\Delta}{=} \{a, a+1, ..., b\}$, for two positive integers a, b $(a < b)$.
- $N^t_{[a,b]} \stackrel{\Delta}{=} \{n_{t+a}, n_{t+a+1}, ..., n_{t+b}\}$ and $L^t_{[a,b]} \stackrel{\Delta}{=} \{l_{t+a}, l_{t+a+1}, ..., l_{t+b}\}$, for two positive integers a, b $(a < b)$.
- $IV = (iv_0, iv_1, ..., iv_{69})$, the 70-bit initialization vector.
- $K = (k_0, k_1, ..., k_{79})$, the 80-bit secret key.
- k^*_t, the round key bit generated at time t.
- z_t, the keystream bit generated at time t.
- c^4_t, the round constant at time t, generated by a counter.

2.1 Description of Sprout

Sprout adopts a structure similar to the Grain family of stream ciphers [1,10,11], which consists of four parts, an 80-bit fixed key register, a 40-bit NFSR with a linked 40-bit LFSR, and a counter register, depicted in Fig. 1. Since storing a fixed key requires less area size than realizing a register of the same length, it is reported in [2] that the hardware area of Sprout is significantly less compared to the existing lightweight stream ciphers.

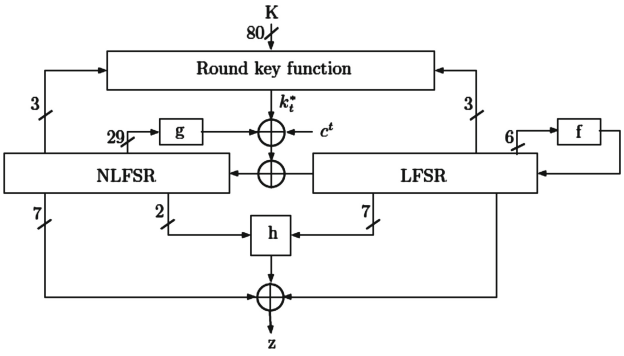

Fig. 1. Keystream generation of sprout

Denote the feedback functions of the NFSR, the LFSR and the nonlinear filter function by g, f and h respectively. There is a 9-bit counter register in Sprout, of which the lower 7 bits are a modulo 80 counter, denoted by $(c^6_t, c^5_t, c^4_t, c^3_t, c^2_t, c^1_t, c^0_t)$

at time t. The 4-th LSB c_t^4 of the counter is employed in the keystream generation. It should be noted that, c_t^4 has a cycle of length 80, i.e., in each cycle, this bit takes the values $\underbrace{0,0,...,0}_{16}\underbrace{1,1,...,1}_{16}\underbrace{0,0,...,0}_{16}\underbrace{1,1,...,1}_{16}\underbrace{0,0,...,0}_{16}$.

The 40-bit LFSR is updated recursively by f as $l_{t+40} = l_t \oplus l_{t+5} \oplus l_{t+15} \oplus l_{t+20} \oplus l_{t+25} \oplus l_{t+34}$. The NFSR is updated recursively by a non-linear feedback function g as

$$\begin{aligned} n_{t+40} &= k_t^* \oplus l_t \oplus c_t^4 \oplus g(N^t) \\ &= k_t^* \oplus l_t \oplus c_t^4 \oplus n_t \oplus n_{t+13} \oplus n_{t+19} \oplus n_{t+35} \oplus n_{t+39} \\ &\quad \oplus n_{t+2}n_{t+25} \oplus n_{t+3}n_{t+5} \oplus n_{t+7}n_{t+8} \oplus n_{t+14}n_{t+21} \oplus n_{t+16}n_{t+18} \\ &\quad \oplus n_{t+22}n_{t+24} \oplus n_{t+26}n_{t+32} \oplus n_{t+33}n_{t+36}n_{t+37}n_{t+38} \\ &\quad \oplus n_{t+10}n_{t+11}n_{t+12} \oplus n_{t+27}n_{t+30}n_{t+31}. \end{aligned}$$

Let $u_t = l_{t+4} \oplus l_{t+21} \oplus l_{t+37} \oplus n_{t+9} \oplus n_{t+20} \oplus n_{t+29}$, then

$$k_t^* = \begin{cases} k_t, \ 0 \le t \le 79 \\ k_{t(\text{mod } 80)} \cdot u_t, \ \text{otherwise.} \end{cases}$$

Given the internal state at time t, the keystream bit is generated as

$$z_t = h(n_{t+4}, l_{t+6}, l_{t+8}, l_{t+10}, l_{t+32}, l_{t+17}, l_{t+19}, l_{t+23}, n_{t+38}) \oplus l_{t+30} \oplus \Big(\bigoplus\nolimits_{i \in A} n_{t+i}\Big),$$

where $A = \{1, 6, 15, 17, 23, 28, 34\}$, and the filter function is

$$h(\cdot) = n_{t+4}l_{t+6} \oplus l_{t+8}l_{t+10} \oplus l_{t+32}l_{t+17} \oplus l_{t+19}l_{t+23} \oplus n_{t+4}l_{t+32}n_{t+38}.$$

During the key/IV setup phase, since the key is fixed, first load the IV in the following way: $n_i = iv_i, 0 \le i \le 39$; $l_i = iv_{i+40}, 0 \le i \le 29$ and $l_i = 1, 30 \le i \le 38$, $l_{39} = 0$. Then run the cipher 320 rounds as follows.

- the LFSR update function is changed to $l_{t+40} = z_t \oplus f(L^t)$.
- the NFSR update function is changed to $n_{t+40} = z_t \oplus k_t^* \oplus l_t \oplus c_t^4 \oplus g(N_t)$.
- no keystream bit is generated.

After the initialization phase, the keystream generation phase starts and there is no feedback keystream anymore.

2.2 A Model for Sprout-Like Stream Ciphers

There are three functions involved in the model: a non-linear function $G(x)$, a linear function $F(x)$ and a non-linear filter function $h(\cdot)$.

The internal state of the model consists of the non-linear state N and the linear state L. At each step, the function $G(\cdot)$ is applied to N and $F(\cdot)$ to L, respectively. Besides, there may also be some other mixing procedure that

xoring some bits of N into L, and vice versa. Further, the secret key is involved in the non-linear state updating selectively by a function $u(\cdot)$. The output of the current state is also computed as the xoring of the bits from both N and L and a non-linear filter function $h(\cdot)$, which takes some input values from both N and L, respectively. Some notations that will be used in the description are listed here.

- $L^t = [L_0^t, L_1^t, ..., L_{l_1-1}^t]$, the internal state of the linear component.
- $N^t = [N_0^t, N_1^t, ..., N_{l_2-1}^t]$, the internal state of the non-linear component.
- $rL^t = \{L_{\gamma_1}^t, L_{\gamma_2}^t, ..., L_{\gamma_{a_1}}^t\}$, a subset of L^t and the linear part of $u(\cdot)$.
- $rN^t = \{N_{\delta_1}^t, N_{\delta_2}^t, ..., N_{\delta_{a_2}}^t\}$, a subset of N^t and the non-linear part of $u(\cdot)$.
- $pL^t = \{L_{\alpha_1}^t, L_{\alpha_2}^t, ..., L_{\alpha_{n_1}}^t\}$, a subset of L^t with the variables of the filter function $h(\cdot)$ coming from the LFSR.
- $pN^t = \{N_{\beta_1}^t, N_{\beta_2}^t, ..., N_{\beta_{n_2}}^t\}$, a subset of N^t with the variables of the filter function $h(\cdot)$ coming from the NFSR.
- $qL^t = \{L_{\sigma_1}^t, L_{\sigma_2}^t, ..., L_{\sigma_{m_1}}^t\}$, a subset of L^t and the linear masking in the keystream generation function.
- $qN^t = \{N_{\tau_1}^t, N_{\tau_2}^t, ..., N_{\tau_{m_2}}^t\}$, a subset of N^t and the non-linear masking in the keystream generation function.
- $pqN^t = pN^t \cup qN^t$, the variables used in the keystream generation coming from the NFSR.

The general framework is specified by the following items (we only focus on the keystream generation phase).

1. Components

- The linear component is $L^t = [L_0^t, L_1^t, ..., L_{l_1-1}^t] \in F_2^{l_1}$, whose initial state is denoted by L^0. It is updated recursively as $L^{t+1} = F(L^t)$. Without loss of generality, we assume this process is invertible, and the inverse process is $L^{t-1} = F'(L^t)$.
- The non-linear component is $N^t = [N_0^t, N_1^t, ..., N_{l_2-1}^t] \in F_2^{l_2}$, whose initial state is denoted by N^0. It is updated recursively as

$$N^{t+1} = G(N^t \oplus L_1(L^t)) \oplus L_2(L^t) \oplus u(rL^t, rN^t) \cdot R(t, K) \oplus C_t,$$

  where $G(\cdot)$ is a (l_2, l_2)-vectorial Boolean function, C_t is a counter related vector of length l_2. Note that whether the key is involved in the state updating is dependent on the value of $u(\cdot)$. If $u(rL^t, rN^t) = 1$, the key will be involved. Similarly, we assume this non-linear process is invertible, and the inverse process is computed as

$$N^{t-1} = G'(N^t \oplus L_1'(L^{t-1})) \oplus L_2'(L^{t-1}) \oplus u(rL^{t-1}, rN^{t-1}) \cdot R(t-1, K) \oplus C_{t-1}.$$

- A filter function $h(\cdot)$ from $F_2^{n_1+n_2}$ into F_2 is used as part of the output function in the form $h(pL^t, pN^t)$, which takes n_1 input values $\{L_{\alpha_1}^t, L_{\alpha_2}^t, ..., L_{\alpha_{n_1}}^t\}$ from L^t and n_2 input values $\{N_{\beta_1}^t, N_{\beta_2}^t, ..., N_{\beta_{n_2}}^t\}$ from N^t, respectively.

– A linear Boolean function $l(\cdot)$ from $F_2^{m_1+m_2}$ into F_2 is used as part of the output function in the form $l(qL^t, qN^t)$, which takes m_1 input values $\{L^t_{\sigma_1}, L^t_{\sigma_2}, ..., L^t_{\sigma_{m_1}}\}$ from L^t and m_2 input values $\{N^t_{\tau_1}, N^t_{\tau_2}, ..., N^t_{\tau_{m_2}}\}$ from N^t, respectively.
– An output function $\phi(\cdot) = l(\cdot) \oplus h(\cdot)$, which generates the keystream $\{z_t\}_{t\geq 0}$ based on the inputs taken from both L^t and N^t, $t = 0, 1, ...$

2. Keystream Generation

The keystream $\{z_t\}_{t\geq 0}$ is recursively generated as

$$z_t = h(pL^t, pN^t) \oplus l(qL^t, qN^t),\ t = 0, 1, ...$$

Let U be the subspace of F_2^m and denote the dimension as $dim(U)$, define $\overline{U} := \{a \in F_2^m : a \notin U\} \cup \{0\}$ as the complementary space of U. Now a coset of the subspace U is represented by $U_a := a \oplus U, a \in \overline{U}$, also called a *flat*. The following definitions are needed in our model.

Definition 1. *An m-variable Boolean function f is k-normal (resp. k-weakly normal) if there exists a flat $V \subseteq F_2^m$ of dimension k such that f is constant (resp. affine) on V.*

For example, the 5-variable Boolean function $h(\cdot)$ in Grain-v1 is 2-normal and 3-weakly normal, and the 9-variable Boolean function $h(\cdot)$ in Sprout and Grain-128a is 5-normal.

Next, we study a natural generalization of the above definition for vectorial Boolean functions [5].

Definition 2. *An $(m,\ n)$-function F: $F_2^m \to F_2^n$ is called k-normal if there exists a flat $V \subseteq F_2^m$ of dimension k such that F is constant on V.*

In our analysis, we investigate the k-normality of the augmented function defined as follows.

Definition 3. *For a $(n_1 + n_2)$-variable Boolean function $h(pL^t, pN^t)$, the $(b+f+1)$-th augmented function of h, $H^{(b,f)} : F_2^{M_1+M_2} \to F_2^{b+f+1}$ is defined as*

$$H^{(b,f)}(PL^t, PN^t) = \big(h(pL^{t-b}, pN^{t-b}), ..., h(pL^t, pN^t), ..., h(pL^{t+f}, pN^{t+f})\big),$$

where b, f are two positive integers, and

$$PL^t \triangleq \bigcup_{i=-b}^{f} pL^{t+i},\ M_1 \triangleq |PL^t| \leq \sum_{i=-b}^{f} |pL^{t+i}| = n_1(b+f+1),$$
$$PN^t \triangleq \bigcup_{i=-b}^{f} pN^{t+i},\ M_2 \triangleq |PN^t| \leq \sum_{i=-b}^{f} |pN^{t+i}| = n_2(b+f+1).$$

3. Assumptions

– *3.1*: there exists two positive integers b, f such that $\bigcup_{i=-b}^{f} pqN^{t+i} \subseteq N^t$ for any $t \geq b$. In this case, the output segment $z_{t-b}, ..., z_t, ..., z_{t+f}$ can be computed from the complete state (L^t, N^t) at time t.

- *3.2*: $H^{(b,f)}$, the $(b+f+1)$-th augmented function of the filter function h, is a k-normal Boolean function such that $H^{(b,f)}(x_1, ..., x_n) = 0^{b+f+1}$ when x_j is fixed for all $j \in \Omega$, where Ω is a subset of $[1,\ n]$ and $|\Omega| = n - k$.
- *3.3*: there exists two positive integers d, e such that $\bigcup_{i=-d}^{e} rN^{t+i} \subseteq N^t$ for any $t \geq d$. In this case, $u(rL^{t+i}, rN^{t+i}), i = -d, ..., -1, 0, 1, ..., e$ can be computed from the complete state (L^t, N^t) at time t.
- *3.4*: assume $pqN^{t+f+1} \not\subset N^t$ and $pqN^{t+f+1} \subset N^{t+1}$ for any $t \geq b$, meaning that we cannot get pqN^{t+f+1} from the state (L^t, N^t). Note that the secret key is incorporated in the non-linear state updating selectively, if we assume a special state (L^t, N^t) such that $u(rL^t, rN^t) = 0$, N^{t+1} can be computed from (L^t, N^t), thus we further get the output bit z_{t+f+1}. Repeat this process for x steps, i.e., we assume a special state (L^t, N^t) such that $u(rL^{t+i}, rN^{t+i}) = 0$ for $i = 0, 1, ..., x-1$, then we get the output bits $z_{t+f+1}, ..., z_{t+f+x}$.
- *3.5*: assume $rN^{t+e+1} \not\subset N^t$ and $rN^{t+e+1} \subset N^{t+1}$ for any $t \geq d$. For the above special state (L^t, N^t) such that $u(rL^{t+i}, rN^{t+i}) = 0$ for $i = 0, 1, ..., x-1$, if $x - 1 \leq e$, we have only unknowns from (L^t, N^t); if $x - 1 > e$, then the unknowns from N^{t+1}, N^{t+2},... will appear with some nonlinear equations $N^{t+j+1} = G(N^{t+j} \oplus L_1(L^{t+j})) \oplus L_2(L^{t+j}) \oplus C_{t+j}$, $j = 0, 1, ..., x-e-2$.
- *3.6*: assume $pqN^{t-b-1} \not\subset N^t$ and $pqN^{t-b-1} \subset N^{t-1}$ for any $t \geq b$, which means we cannot get pqN^{t-b-1} from the state (L^t, N^t). If we assume a special state (L^t, N^t) such that $u(rL^{t-1}, rN^{t-1}) = 0$, N^{t-1} can be computed from (L^t, N^t), thus we further get the output bit z_{t-b-1}. Repeat this process for y steps, i.e., we assume a special state (L^t, N^t) such that $u(rN^{t-j}, rL^{t-j}) = 0$ for $j = 1, ..., y$, then we get the output bits $z_{t-b-1}, ..., z_{t-b-y}$.
- *3.7*: assume $rN^{t-d-1} \not\subset N^t$ and $rN^{t-d-1} \subset N^{t-1}$ for any $t \geq d$. For the above special state (L^t, N^t) such that $u(rL^{t-j}, rN^{t-j}) = 0$ for $j = 1, ..., y$, if $y \leq d$, we have only unknowns from (L^t, N^t); if $y > d$, then the unknowns from N^{t-1}, N^{t-2},... will appear with some nonlinear equations $N^{t-j-1} = G'(N^{t-j} \oplus L'_1(L^{t-j-1})) \oplus L'_2(L^{t-j-1}) \oplus C_{t-j-1}$, $j = 0, 1, ..., y-d-1$.

It is easy to check that the proposed model includes a number of primitives, e.g., Sprout and the Grain family. For Grain family, the term $u(rL^t, rN^t) = 0$ for any time t. For Sprout, $N^t = [n_t, n_{t+1}, ..., n_{t+39}]$, $L^t = [l_t, l_{t+1}, ..., l_{t+39}]$, and for any t, $u(rL^t, rN^t) = l_{t+4} \oplus l_{t+21} \oplus l_{t+37} \oplus n_{t+9} \oplus n_{t+20} \oplus n_{t+29}$. The positive integers b, f, d, e are $b = 1,\ f = 1,\ d = 9,\ e = 10$ respectively.

3 A TMD Tradeoff Attack Framework

In this section, we provide a systematic security analysis for Sprout-like stream ciphers. A dedicated TMD tradeoff attack framework is developed for such designs based on the k-normality of the augmented function.

The goal of cryptanalysis is to recover the internal state which has generated a sample segment, and if possible, given the internal state, to further restore the secret key. There are two phases in the framework: the pre-processing phase and the processing phase. The offline pre-processing phase is performed only once and is independent of the employed secret key and the keystream sample.

3.1 Pre-Processing Phase

In the offline pre-processing phase, some tables are prepared which will be used later in the processing phase. Given the parameters l_1, l_2 and b, f, d, e, x, y, define a two-dimensional counter array $\bar{\mathbf{C}} = [C_{t-y}, ..., C_{t-1}, C_t, C_{t+1}, ..., C_{t+(x-1)}]$, we construct the State-Keystream pair tables as follows.

1. Under the assumptions in the model, construct a system of equations which implies a "special" state (L^t, N^t) satisfying the following conditions.
   - (1.1) $H^{(b,f)}(PL^t, PN^t) = 0^{b+f+1}$ and $l(qL^{t+i}, qN^{t+i}) = 0$, for $i = -b, ..., -1, 0, 1, ..., f$.
   - (1.2) $u(rL^{t+i}, rN^{t+i}) = 0$ for $i = 0, 1, ..., x-1$, from which we can get the output bits z_{t+f+1},...,z_{t+f+x}.
   - (1.3) $u(rL^{t-j}, rN^{t-j}) = 0$ for $j = 1, ..., y$, from which we can get the output bits z_{t-b-1},...,z_{t-b-y}.
2. Suppose Assumptions 3.2, 3.5 and 3.7 hold,
   - if $x - 1 \leq e$ and $y \leq d$, the above system of equations has only unknowns from the state (L^t, N^t).
   - if $x - 1 > e$ and $y \leq d$, the unknowns from N^{t+1}, N^{t+2},... will appear with some non-linear equations:

     $$N^{t+j+1} = G(N^{t+j} \oplus L_1(L^{t+j})) \oplus L_2(L^{t+j}) \oplus C_{t+j},\ j = 0, 1, ..., x-e-2.$$

     Define another counter array $\bar{\mathbf{C}}' = [C_t, C_{t+1}, ..., C_{t+(x-e-2)}]$, note that the round constant vectors in $\bar{\mathbf{C}}'$ are involved in these equations.
   - if $x - 1 \leq e$ and $y > d$, the unknowns from N^{t-1}, N^{t-2},... will appear with some nonlinear equations:

     $$N^{t-j-1} = G'(N^{t-j} \oplus L_1'(L^{t-j-1})) \oplus L_2'(L^{t-j-1}) \oplus C_{t-j-1},\ j = 0, 1, ..., y-d-1.$$

     Define counter array $\bar{\mathbf{C}}' = [C_{t-(y-d)}, ..., C_{t-2}, C_{t-1}]$, the round constant vectors in $\bar{\mathbf{C}}'$ are involved in these equations.
   - if $x - 1 > e$ and $y > d$, the unknowns from N^{t+1}, N^{t+2},... and N^{t-1}, N^{t-2},... will appear with some nonlinear equations:

     $$\begin{array}{l} N^{t+j+1} = G(N^{t+j} \oplus L_1(L^{t+j})) \oplus L_2(L^{t+j}) \oplus C_{t+j}, j = 0, 1, ..., x-e-2, \\ N^{t-j-1} = G'(N^{t-j} \oplus L_{1'}(L^{t-j-1})) \oplus L_{2'}(L^{t-j-1}) \oplus C_{t-j-1}, j = 0, 1, ..., y-d-1. \end{array}$$

     Define counter array $\bar{\mathbf{C}}' = [C_{t-(y-d)}, ..., C_{t-1}, C_t, C_{t+1}, ..., C_{t+(x-e-2)}]$, the round constant vectors in $\bar{\mathbf{C}}'$ are involved in these equations.
3. For each possible counter array $\bar{\mathbf{C}}'$, solve the constructed system of equations and get the special states (L^t, N^t) satisfying 1 and 2. Memorize the special state (L^t, N^t) in the first column of a row in table $\mathrm{T}_{\bar{\mathbf{C}}'}$, further for this state and for each possible counter array $\bar{\mathbf{C}}^* = \bar{\mathbf{C}} \backslash \bar{\mathbf{C}}'$, get the corresponding $(x+y)$ output bits $\underbrace{z_{t-b-1}, ..., z_{t-b-y}}_{y}, \underbrace{z_{t+f+1}, ..., z_{t+f+x}}_{x}$ and store them in the second column as a sub-row in table $\mathrm{T}_{\bar{\mathbf{C}}'}$.

Remarks. Denote the number of rows (in the first column) of table $\mathrm{T}_{\bar{\mathbf{C}}'}$ as 2^r, if $r < x + y$, we only need to store $(x + y - r)$ output bits in the second column, indexed by r-bit of the output. Next, let $\mathrm{Z}_t^{(b+f+1)} = [z_{t-b}, ..., z_t, ..., z_{t+f}] \in F_2^{b+f+1}$, then an internal state satisfying the condition (1.1) implies $\mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1}$. Further, for each counter array $\bar{\mathbf{C}}'$, N^{t+1},...,N^{t+x} and N^{t-1},...,N^{t-y} can be computed directly from a "special" state (L^t, N^t) according to the non-linear state updating function without involving the secret key.

3.2 Processing Phase

Now we discuss how to recover the internal state which has generated a sample segment, and if possible, given the internal state, to further restore the secret key. The following two propositions have provided us a direct way of key recovery from an internal state candidate and some keystream bits.

Proposition 1. *For a special state (L^t, N^t) satisfying the conditions (1.1) and (1.2), N^{t+1},...,N^{t+x} can be computed directly from the complete state (L^t, N^t) and the non-linear state updating function without involving the secret key. Besides, if $u(rL^{t+x}, rN^{t+x}) = 1$, we may get some secret key information $R(t+x, K)$ when the keystream bit $z_{t+f+x+1}$ is known. Further, more key information $R(t+x+j, K)$, $j = 0, 1, ...$ will probably be obtained when more keystream bits $z_{t+f+x+j+1}$, $j = 0, 1, ...$ are known.*

Proof. The first half is clear from the condition (1.2).

For a special state (L^t, N^t), if $u(rL^{t+x}, rN^{t+x}) = 1$, the secret key information $R(t + x, K)$ is incorporated into the updating of the non-linear part from N^{t+x} to N^{t+x+1}. One can check that the keystream bit $z_{t+f+x+1}$ is dependent on N^{t+x+1}. In a word, $R(t + x, K)$ is likely to affect (if $u(rL^{t+x}, rN^{t+x}) = 1$) the keystream bit $z_{t+f+x+1}$. Accordingly, we may obtain some key information $R(t + x, K)$ from $z_{t+f+x+1}$. This procedure can be repeated many times. □

Similar to the proof of Proposition 1, we have the following proposition.

Proposition 2. *For a special state (L^t, N^t) satisfying the conditions (1.1) and (1.3), N^{t-1},...,N^{t-y} can be computed directly from the complete state (L^t, N^t) and the non-linear state updating function without involving the secret key. Besides, if $u(rL^{t-y-1}, rN^{t-y-1}) = 1$, we may get some key information $R(t-y-1, K)$ when the keystream bit $z_{t-b-y-1}$ is known. Further, more key information $R(t - y - j, K)$, $j = 1, 2, ...$ will probably be obtained when more keystream bits $z_{t-b-y-j}$, $j = 1, 2, ...$ are known.*

By utilizing the pre-computed tables and the given keystream sample, the processing phase is carried out as follows.

The Internal State Recovery Algorithm. Given the parameters b, f, x, y, the tables $\mathrm{T}_{\bar{\mathbf{C}}'}$, and the keystream sample $\{z_t\}_{t\geq 0}$, the processing steps are as follows.

1. Search the keystream sequence $\{z_t\}_t$ for the next non-considered block of $(b+f+1)$ zeros. If there are no more blocks, output a flag that the algorithm has failed.
2. For each detected block, compute the corresponding counter array $\bar{\mathbf{C}}$, $\bar{\mathbf{C}}'$ and $\bar{\mathbf{C}}^*$ from the time t, compare the x-bit segment of the keystream subsequent to the block and y-bit segment prior to the block with the memorized $(x+y)$-bit segments in the second column (sub-row is indexed by $\bar{\mathbf{C}}^*$) of the table $\mathrm{T}_{\bar{\mathbf{C}}'}$, and do the following:
   - If the matching does not exist, go the processing Step 1.
   - If the $(x+y)$-bit output segment matches with a segment in table $\mathrm{T}_{\bar{\mathbf{C}}'}$, go to Step 3.
3. Read the corresponding state, and if appropriate, recover (part of) the secret key according to Propositions 1 and 2 from this state and more keystream bits.

3.3 Complexity Analysis

In the Sprout-like model, the keystream bit is generated as $z_t = h(pL^t, pN^t) \oplus l(qL^t, qN^t)$. For the (b, f) derived from Assumption 3.3, we define a flat $V^{(b,f)}$ of dimension $dim(V^{(b,f)})$ such that $H^{(b,f)} = 0^{b+f+1}$ over it. i.e.,

$$V^{(b,f)} = \{(L^t, N^t) : H^{(b,f)}(PL^t, PN^t) = 0^{b+f+1}\},$$

We have the following lemma which is closely related to the time complexity of processing (table look-ups) of our proposed algorithm.

Lemma 1. *Suppose $Pr[u(\cdot) = 0] = p$, assume all the events in (1.1),(1.2) and (1.3) are independent, then the probability that an internal state (L^t, N^t) is a special state satisfying the conditions (1.1), (1.2) and (1.3) simultaneously when the keystream segment $\mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1}$ is*

$$\Pr\left[(L^t, N^t) \textit{ is a special state} \middle| \mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1}\right] = \frac{1}{2^{l_1+l_2-\dim(V^{(b,f)})}} \cdot p^{x+y},$$

where $V^{(b,f)}$ is a flat such that $H^{(b,f)} = 0^{b+f+1}$ over it.

Proof. For any internal state (L^t, N^t) and keystream segment $\mathrm{Z}_t^{(b+f+1)}$, the underlying assumptions directly imply the following:

$$\Pr\left[(L^t, N^t) \textit{ is a special state}\right] = \frac{1}{2^{l_1+l_2-\dim(V^{(b,f)})}} \cdot \frac{1}{2^{b+f+1}} \cdot p^{x+y},$$

and $\Pr\left[\mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1}\right] = 2^{-(b+f+1)}$, and

$$\Pr\left[\mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1} \middle| (L^t, N^t) \textit{ is a special state}\right] = 1.$$

On the other hand,

$$\begin{aligned}
&\Pr\left[(L^t, N^t) \textit{ is a special state} \middle| \, \mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1}\right] \\
&= \frac{\Pr\left[(L^t,N^t) \textit{ is a special state}\right] \cdot \Pr\left[\mathrm{Z}_t^{(b+f+1)}=0^{b+f+1} \middle| (L^t,N^t) \textit{ is a special state}\right]}{\Pr\left[\mathrm{Z}_t^{(b+f+1)}=0^{b+f+1}\right]} \\
&= \frac{1}{2^{l_1+l_2-\dim(V^{(b,f)})}} \cdot p^{x+y}
\end{aligned}$$

which yields the statement of the lemma. □

Theorem 1. *Suppose $V^{(b,f)}$ is a flat such that $H^{(b,f)} = 0^{b+f+1}$ over it, then the complexities of the proposed generic algorithm for cryptanalysis are as follows:*

(1) The processing data complexity is $D = 2^{l_1+l_2-\dim(V^{(b,f)})} \cdot 2^{b+f+1} \cdot p^{-(x+y)}$.

(2) The expected space complexity in the pre-processing phase is proportional to the sum of number of rows in each table $\mathrm{T}_{\bar{\mathbf{C}}'}$.

(3) The processing (table look-ups) time complexity is proportional to $2^{l_1+l_2-\dim(V^{(b,f)})} \cdot p^{-(x+y)}$.

(4) The pre-processing time complexity is equivalent to the workload for solving the system of equations constructed.

Proof. The data complexity is determined by the probability that an internal state is a special state satisfying conditions (1.1), (1.2) and (1.3) simultaneously in the pre-processing phase, which is given in the proof of Lemma 1 as $2^{-(l_1+l_2-dim(V^{(b,f)}))} \cdot 2^{-(b+f+1)} \cdot p^{x+y}$. Thus we have $D = 2^{l_1+l_2-\dim(V^{(b,f)})} \cdot 2^{b+f+1} \cdot p^{-(x+y)}$.

For each possible counter array $\bar{\mathbf{C}}'$, we have constructed the corresponding table $\mathrm{T}_{\bar{\mathbf{C}}'}$, thus the estimated space complexity is proportional to the sum of number of rows in each table $\mathrm{T}_{\bar{\mathbf{C}}'}$.

In the processing phase, the expected number of table look-ups depends on the probability that an internal state (L^t, N^t) is a special state satisfying the conditions (1.1), (1.2) and (1.3) simultaneously when the keystream segment $\mathrm{Z}_t^{(b+f+1)} = 0^{b+f+1}$, which is given in Lemma 1 as $2^{-(l_1+l_2-dim(V^{(b,f)}))} \cdot p^{x+y}$. Thus the number of table look-ups is $2^{l_1+l_2-dim(V^{(b,f)})} \cdot p^{-(x+y)}$.

The pre-processing time complexity is determined by the workload for solving the system of equations constructed. □

4 Cryptanalysis of Sprout

In this section, we apply the framework proposed in Sect. 3 to Sprout with the comparisons to the previous relevant attacks.

4.1 Fitting into the Model

Sprout fits into the model with the parameters $l_1 = l_2 = 40$, which are the length of LFSR and NFSR respectively. The keystream bit z_t at time t is generated as

$$\begin{aligned}
z_t &= h(n_{t+4}, l_{t+6}, l_{t+8}, l_{t+10}, l_{t+32}, l_{t+17}, l_{t+19}, l_{t+23}, n_{t+38}) \\
&\oplus l_{t+30} \oplus n_{t+1} \oplus n_{t+6} \oplus n_{t+15} \oplus n_{t+17} \oplus n_{t+23} \oplus n_{t+28} \oplus n_{t+34},
\end{aligned}$$

where $h(\cdot) = n_{t+4}l_{t+6} \oplus l_{t+8}l_{t+10} \oplus l_{t+32}l_{t+17} \oplus l_{t+19}l_{t+23} \oplus n_{t+4}l_{t+32}n_{t+38}$.

As described in Sect. 2, whether the secret key is involved in the NFSR state updating is determined by the value $u_t = l_{t+4} \oplus l_{t+21} \oplus l_{t+37} \oplus n_{t+9} \oplus n_{t+20} \oplus n_{t+29}$, to fit in the model, we have $rL^t = \{l_{t+4}, l_{t+21}, l_{t+37}\}$, $rN^t = \{n_{t+9}, n_{t+20}, n_{t+29}\}$ and the two parameters $d = 9$, $e = 10$ such that $\bigcup_{i=-9}^{10} rN^{t+i} \subseteq N^t$.

Let $pL^t = \{l_{t+6}, l_{t+8}, l_{t+10}, l_{t+17}, l_{t+19}, l_{t+23}, l_{t+32}\}$, $pN^t = \{n_{t+4}, n_{t+38}\}$, $qL^t = \{l_{t+30}\}$, and $qN^t = \{n_{t+1}, n_{t+6}, n_{t+15}, n_{t+17}, n_{t+23}, n_{t+28}, n_{t+34}\}$. From this we have $b = 1$, $f = 1$ to fit into the Sprout-like model. Given $(b, f) = (1, 1)$,

$$H^{(1,1)}(PL^t, PN^t) = \left(h(pL^{t-1}, pN^{t-1}), h(pL^t, pN^t), h(pL^{t+1}, pN^{t+1})\right),$$

where $PL^t = L^t_{[5,11]} \cup L^t_{[16,20]} \cup L^t_{[22,24]} \cup L^t_{[31,33]}$ and $PN^t = N^t_{[3,5]} \cup N^t_{[37,39]}$, thus $H^{(1,1)}(\cdot)$ is a $(24, 3)$-vectorial Boolean function.

Suppose $n_{t+3} = n_{t+4} = n_{t+5} = 0$, by a computer computation, there are $12096 (> 2^{13})$ possible values for the following 16-bit of LFSR such that $H^{(1,1)}(\cdot) = 0^3$:

$$\begin{aligned} P^t = [&l_{t+7}l_{t+8}l_{t+9}l_{t+10}||l_{t+11}l_{t+16}l_{t+17}l_{t+18} \\ &||l_{t+19}l_{t+20}l_{t+22}l_{t+23}||l_{t+24}l_{t+31}l_{t+32}l_{t+33}] \subseteq L^t. \end{aligned}$$

For example, $P^t = \texttt{0x0000}, \texttt{0x8000}, \texttt{0x4000}, \texttt{0xc000}, ...$ when denoted by hexadecimal digits. We denote all the $12096 (> 2^{13})$ values of P^t as a_1, a_2, a_3, a_4,...,a_{12096} such that $a_1 = \texttt{0x0000}$, $a_2 = \texttt{0x8000}$, $a_3 = \texttt{0x4000}$, $a_4 = \texttt{0xc000}$,... respectively.

For Sprout, we will use 2^{13} flats defined as follows:

$$V_i = \{(L^t, N^t) : P^t = a_i \text{ and } n_{t+j} = 0, j = 3, 4, 5\},\ i = 1, ..., 2^{13}$$

Note that in each V_i, 19 bits of (L^t, N^t) are fixed, then each V_i has a dimension of $dim(V_i) = 61$, and $H^{(1,1)}(\cdot) = 0^3$ over V_i. Further we define a flat V as $V = \bigcup_{i=1}^{2^{13}} V_i$. Thus the dimension of V is $dim(V) = 74$.

4.2 Cryptanalysis

We first discuss how to construct tables that will be used in the processing phase.

Pre-processing Phase. Given the parameters x, y and do the following:

1. Define a counter array as $C = [c^4_{t-y}, ..., c^4_{t-1}, c^4_t, c^4_{t+1}, ..., c^4_{t+(x-1)}]$ of size $|C| = x + y$. For an internal state (L^t, N^t) such that $n_{t+3} = n_{t+4} = n_{t+5} = 0$ (thus there are 77 unknowns), construct a system of equations which implies a state (L^t, N^t) satisfying the following conditions.
   - (a). $l(qL^{t+i}, qN^{t+i}) = 0$, for $i = -1, 0, 1$.
   - (b). $u_{t+i} = 0$ for $i = 0, 1, ..., x - 1$, from which we can get the output bits z_{t+2},...,z_{t+x+1} (suppose the round constants $c^4_t, c^4_{t+1}, ..., c^4_{t+(x-1)}$ are known).

- (c). $u_{t-j} = 0$ for $j = 1, ..., y$, from which we can get the output bits z_{t-2},...,z_{t-y-1} (suppose the round constants $c^4_{t-1}, c^4_{t-2}, ..., c^4_{t-y}$ are known).

2. We discuss it in the following situations:
   - **Case 1:** If $x \leq 11$ and $y \leq 9$, we have the corresponding system of $(3 + x + y)$ linear equations with only 77 unknowns from the state (L^t, N^t): 40 unknowns from L^t and 37 unknowns from N^t.

$$\begin{cases} l_{t+30+k} \oplus (\bigoplus_{i \in A} n_{t+i+k}) = 0, k = -1, 0, 1 \\ l_{t+4+i} \oplus l_{t+21+i} \oplus l_{t+37+i} \oplus n_{t+9+i} \oplus n_{t+20+i} \oplus n_{t+29+i} = 0, i = 0, 1, ..., x-1 \\ l_{t+4-j} \oplus l_{t+21-j} \oplus l_{t+37-j} \oplus n_{t+9-j} \oplus n_{t+20-j} \oplus n_{t+29-j} = 0, j = 1, ..., y \end{cases}$$

   - **Case 2:** If $x \geq 12$ and $y \leq 9$, in addition to the 77 unknowns from the state (L^t, N^t), the unknowns n_{t+40}, n_{t+41},...,$n_{t+40+(x-12)}$ will appear with some non-linear equations. Thus we obtain a system of equations with $(66 + x)$ unknowns, and $(2x + y - 8)$ equations ($(3 + x + y)$ linear equations and $(x - 11)$ non-linear equations). Define another counter array $C' = [c^4_t, c^4_{t+1}, ..., c^4_{t+(x-12)}]$ of size $|C'| = x - 11$, note that the round constants in C' are involved in this system.

$$\begin{cases} l_{t+30+k} \oplus (\bigoplus_{i \in A} n_{t+i+k}) = 0, k = -1, 0, 1 \\ l_{t+4+i} \oplus l_{t+21+i} \oplus l_{t+37+i} \oplus n_{t+9+i} \oplus n_{t+20+i} \oplus n_{t+29+i} = 0, i = 0, 1, ..., x-1 \\ l_{t+4-j} \oplus l_{t+21-j} \oplus l_{t+37-j} \oplus n_{t+9-j} \oplus n_{t+20-j} \oplus n_{t+29-j} = 0, j = 1, ..., y \\ n_{t+40+m} \oplus l_{t+m} \oplus c^4_{t+m} \oplus g(N^{t+m}) = 0, m = 0, 1, ..., x-12 \ (non-linear) \end{cases}$$

   - **Case 3:** If $x \leq 11$ and $y \geq 10$, in addition to the 77 unknowns from the state (L^t, N^t), the unknowns n_{t-1}, n_{t-2},...,$n_{t-(y-9)}$ will appear with some non-linear equations. Thus we obtain a system of equations with $(68 + y)$ unknowns, and $(x+2y-6)$ equations ($(3+x+y)$ linear equations and $(y-9)$ non-linear equations). Define $C' = [c^4_{t-(y-9)}, ..., c^4_{t-1}]$ of size $|C'| = y - 9$, the round constants in C' are involved in this system.

$$\begin{cases} l_{t+30+k} \oplus (\bigoplus_{i \in A} n_{t+i+k}) = 0, k = -1, 0, 1 \\ l_{t+4+i} \oplus l_{t+21+i} \oplus l_{t+37+i} \oplus n_{t+9+i} \oplus n_{t+20+i} \oplus n_{t+29+i} = 0, i = 0, 1, ..., x-1 \\ l_{t+4-j} \oplus l_{t+21-j} \oplus l_{t+37-j} \oplus n_{t+9-j} \oplus n_{t+20-j} \oplus n_{t+29-j} = 0, j = 1, ..., y \\ n_{t-n} \oplus l_{t-n} \oplus c^4_{t-n} \oplus g'(N^{t-n+1}) = 0, n = 1, ..., y-9 \ (non-linear) \end{cases}$$

   - **Case 4:** If $x \geq 12$ and $y \geq 10$, in addition to the 77 unknowns from the state (L^t, N^t), the unknowns n_{t+40}, n_{t+41},...,$n_{t+40+(x-12)}$ and n_{t-1}, n_{t-2},...,$n_{t-(y-9)}$ will appear with some non-linear equations. Thus we obtain a system of equations with $(57+x+y)$ unknowns, and $(2x+2y-17)$ equations ($(3+x+y)$ linear equations and $(x+y-20)$ non-linear equations). Define $C' = [c^4_{t-(y-9)}, ..., c^4_{t-1}, c^4_t, c^4_{t+1}, ..., c^4_{t+(x-12)}]$ of size $|C'| = x+y-20$, the round constants in C' are involved in the system.

$$\begin{cases} l_{t+30+k} \ (\bigoplus_{i \in A} n_{t+i+k}) = 0, k = -1, 0, 1 \\ l_{t+4+i} \oplus l_{t+21+i} \oplus l_{t+37+i} \oplus n_{t+9+i} \oplus n_{t+20+i} \oplus n_{t+29+i} = 0, i = 0, 1, ..., x-1 \\ l_{t+4-j} \oplus l_{t+21-j} \oplus l_{t+37-j} \oplus n_{t+9-j} \oplus n_{t+20-j} \oplus n_{t+29-j} = 0, j = 1, ..., y \\ n_{t+40+m} \oplus l_{t+m} \oplus c^4_{t+m} \oplus g(N^{t+m}) = 0, m = 0, 1, ..., x-12 \ (non-linear) \\ n_{t-n} \oplus l_{t-n} \oplus c^4_{t-n} \oplus g'(N^{t-n+1}) = 0, n = 1, ..., y-9 \ (non-linear) \end{cases}$$

3. For each possible counter array C', solve the constructed system of equations. Observe that all the round constants in C' are added to the system linearly, by guessing at most $2^{74-(x+y)}$ appropriate unknowns we can solve the system and get $2^{74-(x+y)}$ solutions (L^t, N^t) for each possible counter array C'.

4. For each possible counter array C', check each of the $2^{74-(x+y)}$ solutions (L^t, N^t). If $(L^t, N^t) \in V_i$, i.e., $P^t = a_i$ for any $i = 1, 2, ..., 2^{13}$, store the 61-bit (L^{*t}, N^{*t}) in the first column of a row in table $\mathrm{T}_{C',i}$, where $L^{*t} = L^t \backslash P^t$ and $N^{*t} = N^t \backslash \{n_{t+3}, n_{t+4}, n_{t+5}\}$. Further for this state and for each possible round constants of $C^* = C \backslash C'$, get the corresponding $(x + y)$ output bits $(z_{t-y-1}, ..., z_{t-2}, z_{t+2}, ..., z_{t+x+1})$ and put them in the second column as a sub-row in table $\mathrm{T}_{C',i}$. Thus there are expected 2^{58-x-y} rows in the first column and $2^{58-x-y} \times \frac{Count(|C|)}{Count(|C'|)}$ rows in the second column, where $Count(n)$ represents the number of all the possible counter arrays of size n.

We list in Table 1 the number of all the possible counter arrays $Count(n)$ of size n.

Table 1. The size of the counter array n and the number $Count(n)$ for all the possible counter arrays

n	6	7	8	9	10	11	12	13	14	15	16	17
$Count(n)$	12	14	16	18	20	22	24	26	28	30	32	33
n	18	19	20	21	22	23	24	25	26	27	28	29
$Count(n)$	35	37	39	41	43	45	47	49	51	53	55	57
n	30	31	32	33	34	35	36	37	38	39	40	41
$Count(n)$	59	61	63	64	65	66	67	68	69	70	71	72
n	42	43	44	45	46	47	48	49	50	51	52	53
$Count(n)$	73	74	75	76	77	78	79	80	80	80	80	80

Remarks. First, it can be seen that, the necessary and sufficient condition for a state (L^t, N^t) to be a "special" state is that $(L^t, N^t) \in V$ and the conditions (a), (b) and (c) hold. Second, for each possible counter array C, n_{t+40},...,$n_{t+40+(x-1)}$ and n_{t-1},...,n_{t-y} can be computed directly from a special state (L^t, N^t) according to the state updating of NFSR without involving the key information. Third, denote the number of rows (in the first column) of table $\mathrm{T}_{C',i}$ as 2^{r_i}, if $r_i < x+y$, we only need to store $(x + y - r_i)$ output bits in the second column, indexed by r_i-bit of the output. Finally, in the pre-processing phase, we have obtained $Count(|C'|) \times 2^{13}$ tables $\mathrm{T}_{C',i}$, each having 2^{58-x-y} rows in the first column to store "special" states and $2^{58-x-y} \times \frac{Count(|C|)}{Count(|C'|)}$ rows in the second column to store the corresponding output bits.

Lemma 2. *The probability that an internal state (L^t, N^t) is a special state (such that $(L^t, N^t) \in V$ and the conditions (*a*), (*b*) and (*c*) hold) when the keystream segment $\mathrm{Z}_t^{(3)} = 0^3$ is given by the following:*

$$\Pr\left[(L^t, N^t) \textit{ is a special state} \middle| \mathrm{Z}_t^{(3)} = 0^3\right] = 2^{-(6+x+y)}.$$

Proof. For any internal state (L^t, N^t) and keystream segment $\mathrm{Z}_t^{(3)}$, the underlying assumptions directly imply the following:

$$\Pr\left[(L^t, N^t) \textit{ is a special state}\right] = \frac{1}{2^{40+40-\dim(V)}} \times \frac{1}{2^{3+x+y}} = 2^{-(9+x+y)},$$

and $\Pr\left[\mathrm{Z}_t^{(3)} = 0^3\right] = 2^{-3}$, and

$$\Pr\left[\mathrm{Z}_t^{(3)} = 0^3 \middle| (L^t, N^t) \textit{ is a special state}\right] = 1.$$

On the other hand,

$$\begin{aligned}
&\Pr\left[(L^t, N^t) \textit{ is a special state} \middle| \mathrm{Z}_t^{(3)} = 0^3\right] \\
&= \frac{\Pr\left[(L^t, N^t) \textit{ is a special state}\right] \times \Pr\left[\mathrm{Z}_t^{(3)} = 0^3 \middle| (L^t, N^t) \textit{ is a special state}\right]}{\Pr\left[\mathrm{Z}_t^{(3)} = 0^3\right]} \\
&= 2^{-(6+x+y)}.
\end{aligned}$$

which yields the statement of the lemma. □

Next, we will present a *State Checking and Key Recovery Mechanism* specified for Sprout, by which we have the opportunity to check whether a state candidate is correct, and if so, further recover the key for a correct guess.

State Checking and Key Recovery Mechanism. For a state candidate at time t, $L^t = [l_t, l_{t+1}, ..., l_{t+39}]$, $N^t = [n_t, n_{t+1}, ..., n_{t+39}]$, create an 80-bit vector K for the possible values associated with it:

1. Compute the value of n_{t-1} given by the keystream bit z_{t-2} as $n_{t-1} = z_{t-2} \oplus h(n_{t+2}, l_{t+4}, l_{t+6}, l_{t+8}, l_{t+30}, l_{t+15}, l_{t+17}, l_{t+21}, n_{t+36}) \oplus l_{t+28} \oplus \left(\bigoplus_{i \in A'} n_{t+i-2}\right)$ where $A' = \{6, 15, 17, 23, 28, 34\}$. And compute l_{t-1} by the LFSR updating equation as $l_{t-1} = l_{t+39} \oplus l_{t+33} \oplus l_{t+24} \oplus l_{t+19} \oplus l_{t+14} \oplus l_{t+4}$, and deduce from n_{t-1}, l_{t-1} the value k^*_{t-1} by the NFSR updating equation as $k^*_{t-1} = n_{t+39} \oplus c^4_{t-1} \oplus l_{t-1} \oplus g(N^{t-1})$.
2. Compute the value of $u_{t-1} = l_{t+3} \oplus l_{t+20} \oplus l_{t+36} \oplus n_{t+8} \oplus n_{t+19} \oplus n_{t+28}$ and combine it with the value of k^*_{t-1} obtained in Step 1:
   - if $u_{t-1} = 0$ and $k^*_{t-1} = 0$, set $t \to t-1$ and go back to Step 1.
   - if $u_{t-1} = 0$ and $k^*_{t-1} = 1$, there is a contradiction, conclude that this guess for state is not correct and stop.
   - if $u_{t-1} = 1$ and $k^*_{t-1} = 0$, check if $k_{(t-1) \mod 80}$ has already been set in K. If no, set it to 0. Set $t \to t-1$ and go back to Step 1. Else, if there is a contradiction, conclude that this guess for state is not correct and stop.
   - if $u_{t-1} = 1$ and $k^*_{t-1} = 1$, check if $k_{(t-1) \mod 80}$ has already been set in K. If no, set it to 1. Set $t \to t-1$ and go back to Step 1. Else, if there is a contradiction, conclude that this guess for state is not correct and stop.

Similar to the statements in [8], the probability that a state candidate survives for $2r$ clocks is 2^{-r}. On average for each 2 clocks, half of the possible guesses will be eliminated. For 2^s candidate states, the average number of clocks for each elimination is

$$\sum_{i=0}^{s} \frac{2 \times 2^{s-i}}{2^s} = \sum_{i=0}^{s} \frac{1}{2^{i-1}} \approx 4$$

We can conclude that 4 clocks of output is enough for checking the validity of a candidate state and the recovery of the key bits for each candidate.

Next we illustrate the algorithm for the internal state and key recovery in the processing phase.

Processing Phase. Given the parameter x, y, the corresponding $Count(|C'|) \times 2^{13}$ tables $\mathrm{T}_{C',i}$ and the given keystream sample $\{z_t\}_{t\geq 0}$, the processing steps are as follows:

1. Search the keystream sequence $\{z_t\}_t$ for the next non-considered block of 3 zeros. If there are no more blocks, output a flag that the algorithm has failed to recover the key.

2. For each detected block, compute the corresponding counter array C, C' and C^* from the time t. For $i = 1, ..., 2^{13}$, compare the x-bit segment of the keystream subsequent to the block and y-bit segment prior to the block with the memorized $(x+y)$-bit segments in the second column (sub-row is indexed by C^*) of the table $\mathrm{T}_{C',i}$, and do the following:
   - If the matching does not exist, go to the processing Step 1.
   - If the $(x+y)$-bit sample segments match with a segment in table $\mathrm{T}_{C',i}$, go to Step 3.
3. Read the corresponding state, check whether it is a correct state or not and recover the secret key by the *State Checking and Key Recovery Mechanism* stated above. If this state survives, recover and output the key, else go to Step 1.

Theorem 2. *For two positive integers x, y, the dedicated TMD tradeoff on Sprout has complexities as follows: (1) The data complexity for the processing is $D = 2^{9+x+y}$; (2) The expected memory M (-bit) of pre-processing is computed as follows:*

$$M = \begin{cases} Count(|C'|) \times 2^{71-x-y} \times \left[61 + \frac{Count(x+y)}{Count(|C'|)} \cdot (x+y)\right], \ if\ x+y < 30, \\ Count(|C'|) \times 2^{71-x-y} \times \left[61 + \frac{Count(x+y)}{Count(|C'|)} \cdot (2x+2y-58)\right], \ if\ x+y \geq 30. \end{cases}$$

(3) The time complexity of processing is $2^{70.66-x-y}$ Sprout encryptions along with 2^{6+x+y} table look-ups. (4) The time complexity of pre-processing is proportional to 2^{74-x-y}.

Proof. The data complexity is determined by the probability that an internal state (L^t, N^t) is a special state (such that $(L^t, N^t) \in V$ and the conditions (a), (b) and (c) hold), which is given in the proof of Lemma 2 as $2^{-(9+x+y)}$. Thus we have $D = 2^{9+x+y}$.

As for the memory, we need $Count(|C'|) \times 2^{13}$ tables $\mathrm{T}_{C',i}$, each having 2^{58-x-y} rows in the first column to store 61-bit "special" states (3+16=19-bit are fixed for each table) and $2^{58-x-y} \times \frac{Count(|C|)}{Count(|C'|)}$ rows in the second column to store the corresponding output bits. If $x + y < 30$, each row in the second column contains $(x+y)$ output bits; if $x+y \geq 30$, each row in the second column

contains $2(x+y)-58$ output bits, indexed by $58-x-y$ bits of the output. Hence the memory M(-bit) is computed as follows:

$$M = \begin{cases} Count(|C'|) \times 2^{13} \times 2^{58-x-y} \times \left[61 + \frac{Count(x+y)}{Count(|C'|)} \cdot (x+y)\right], \ if\ x+y < 30, \\ Count(|C'|) \times 2^{13} \times 2^{58-x-y} \times \left[61 + \frac{Count(x+y)}{Count(|C'|)} \cdot (2x+2y-58)\right], \ otherwise. \end{cases}$$

In the processing phase, the expected number of table look-ups is determined by the probability that an internal state (L^t, N^t) is a special state when the keystream segment $Z_t^{(3)} = 0$, which is given in Lemma 2 as $2^{-(6+x+y)}$. Thus the number of table look-ups is 2^{6+x+y}. For each $(x+y)$-bit keystream bits, we have $2^{71-2(x+y)}$ state candidates producing the output. As stated before, 4 more clocks of output is enough for checking the validity of the state and the recovery of the key bits for each candidate. In total, the time complexity is $2^{6+x+y} \times 2^{71-2(x+y)} \times 4 = 2^{79-x-y}$, which is equivalent to $\frac{2^{79-x-y}}{324} = 2^{70.66-x-y}$ Sprout encryptions.

The pre-processing time complexity is equivalent to solving the constructed system of equations. We see that by guessing at most $2^{74-(x+y)}$ appropriate unknowns we can solve the system for each possible counter array C'. As the counter values are added to the systems linearly, we can do the Gauss elimination only once to store separate tables for each of the $Count(|C'|)$ counter arrays. □

4.3 Detailed Workload for $x = 16$, $y = 15$

We now focus on the workload to solve the system of equations for $x = 16$, $y = 15$. For a state (L^t, N^t), let $n_{t+j} = 0, j = 3, 4, 5$ and define $N^{*t} = N^t \backslash \{n_{t+3}, n_{t+4}, n_{t+5}\}$. We need to solve the following systems of equations, which amounts to 34 linear equations, 11 non-linear equations and 88 unknowns L^t,N^{*t},n_{t+40},n_{t+41},...,n_{t+44},n_{t-1},n_{t-2},...,n_{t-6}.

$$\begin{cases} 1 : l_{t+29} \oplus n_t \oplus n_{t+5} \oplus n_{t+14} \oplus n_{t+16} \oplus n_{t+22} \oplus n_{t+27} \oplus n_{t+33} = 0 \\ 2 : l_{t+30} \oplus n_{t+1} \oplus n_{t+6} \oplus n_{t+15} \oplus n_{t+17} \oplus n_{t+23} \oplus n_{t+28} \oplus n_{t+34} = 0 \\ 3 : l_{t+31} \oplus n_{t+2} \oplus n_{t+7} \oplus n_{t+16} \oplus n_{t+18} \oplus n_{t+24} \oplus n_{t+29} \oplus n_{t+35} = 0 \\ 4 : u_{t-15} = l_{t-11} \oplus l_{t+6} \oplus l_{t+22} \oplus n_{t-6} \oplus n_{t+5} \oplus n_{t+14} = 0 \\ 5 : u_{t-14} = l_{t-10} \oplus l_{t+7} \oplus l_{t+23} \oplus n_{t-5} \oplus n_{t+6} \oplus n_{t+15} = 0 \\ ..... \\ 18 : u_{t-1} = l_{t+3} \oplus l_{t+20} \oplus l_{t+36} \oplus n_{t+8} \oplus n_{t+19} \oplus n_{t+28} = 0 \\ 19 : u_t = l_{t+4} \oplus l_{t+21} \oplus l_{t+37} \oplus n_{t+9} \oplus n_{t+20} \oplus n_{t+29} = 0 \\ 20 : u_{t+1} = l_{t+5} \oplus l_{t+22} \oplus l_{t+38} \oplus n_{t+10} \oplus n_{t+21} \oplus n_{t+30} = 0 \\ ..... \\ 34 : u_{t+15} = l_{t+19} \oplus l_{t+36} \oplus l_{t+52} \oplus n_{t+24} \oplus n_{t+35} \oplus n_{t+44} = 0 \end{cases}$$

$$\begin{cases} 35: n_{t+40} \oplus l_t \oplus c_t^4 \oplus g(N^t) = 0 \\ 36: n_{t+41} \oplus l_{t+1} \oplus c_{t+1}^4 \oplus g(N^{t+1}) = 0 \\ ..... \\ 39: n_{t+44} \oplus l_{t+4} \oplus c_{t+4}^4 \oplus g(N^{t+4}) = 0 \\ 40: n_{t-1} \oplus l_{t-1} \oplus c_{t-1}^4 \oplus g'(N^t) = 0 \\ 41: n_{t-2} \oplus l_{t-2} \oplus c_{t-2}^4 \oplus g'(N^{t-1}) = 0 \\ ..... \\ 45: n_{t-6} \oplus l_{t-6} \oplus c_{t-6}^4 \oplus g'(N^{t-5}) = 0 \end{cases}$$

In the following part, L^t is treated as a column vector of size 40. First of all, we choose the 40 equations numbered by 1,2,...,34 and 40,41,...,45 from the above systems to represent L^t by the unknowns N^{*t},n_{t+40},n_{t+41},...,n_{t+44},n_{t-1},n_{t-2},..., n_{t-6} as $\mathrm{M} \cdot L^t = v$, where M is the 40×40 coefficient matrix of L^t, and v is a column vector of size 40, and

$$\mathrm{M} \cdot L^t = [l_{t+29}, l_{t+30}, l_{t+31}, l_{t-11} \oplus l_{t+6} \oplus l_{t+22}, ..., \\ l_{t+19} \oplus l_{t+36} \oplus l_{t+52}, l_{t-1}, ..., l_{t-6}]^T,$$

and

$$v = [\bigoplus_{i \in B} n_{t+i-1}, \bigoplus_{i \in B} n_{t+i}, \bigoplus_{i \in B} n_{t+i+1}, n_{t-6} \oplus n_{t+5} \oplus n_{t+14}, ..., \\ n_{t+24} \oplus n_{t+35} \oplus n_{t+44}, n_{t-1} \oplus c_{t-1}^4 \oplus g'(N^t), ..., n_{t-6} \oplus c_{t-6}^4 \oplus g'(N^{t-5})]^T.$$

We have checked that $\mathrm{rank}(\mathrm{M}) = 39$. Take l_t as a free variable, we obtain an invertible coefficient matrix of size 39×39. Let $L'^t = L^t \backslash \{l_t\}$, then each variable in L'^t can be uniquely represented as linear combinations of N^{*t},n_{t+40},n_{t+41},..., n_{t+44},n_{t-1},n_{t-2},...,n_{t-6} and l_t, together with 1 non-linear equation with these unknowns. Plugging in the values l_{t+1},l_{t+2},l_{t+3},l_{t+4} in equations numbered by 36,...,39, we get a system with 6 non-linear equations and 49 unknowns N^{*t},n_{t+40}, n_{t+41},...,n_{t+44},n_{t-1},n_{t-2},...,n_{t-5} and l_t. Define a set $\mathrm{GUESS} = \{n_{t+j} : j \in S\}$ of size 33, where

$$S = \{-1, 0, 1, 3, 4, 6, 7, 9, 11, 12, 13, 15, 16, 17, 19, 20, 21, \\ 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40, 41\}.$$

By guessing the 33 unknowns in the set GUESS,

- If $n_{t+9} = 0$, we come up with 2^{32} systems with 6 linear equations and 16 unknowns; For each of these systems, we do the Gauss elimination once by choosing an invertible coefficient matrix of 6×6. The systems can be solved with $2^{32} \times (6^3 + 2^{10}) = 2^{42.27}$ basic operations.
- If $n_{t+9} = 1$, we further guess n_{t+8}, thus we get 2^{33} systems with 6 linear equations and 15 unknowns. Similarly, the systems can be solved with $2^{33} \times (6^3 + 2^9) = 2^{42.51}$ basic operations.

In total, the pre-computation is approximately $2^{43.39}$ basic operations.

We list in Table 2 more instances that illustrate the complexities of the TMD tradeoff attacks on Sprout. The Comparison of our TMD tradeoff attacks with the previous ones in [8] and [12] are presented in Table 3. With carefully chosen attack parameters, our method is at least 2^{20} times faster than the attack in [12], 2^{10} times faster than the attack in [8] with much less memory.

Table 2. The complexity issues of the attack on Sprout

x, y	$Count(x+y)$	Data	Memory(-bit),(TB)	Time	Pre-computation
16,14	59	2^{39}	$2^{51.39}$-bit, 336 TB	$2^{40.66}$	$2^{44.03}$
16,15	61	2^{40}	$2^{50.63}$-bit, 198 TB	$2^{39.66}$	$2^{43.39}$
17,15	63	2^{41}	$2^{49.85}$-bit, 115 TB	$2^{38.66}$	$2^{43.81}$
17,16	64	2^{42}	$2^{49.03}$-bit, 65 TB	$2^{37.66}$	$2^{45.36}$
18,16	65	2^{43}	$2^{48.20}$-bit, 36 TB	$2^{36.66}$	$2^{47.09}$

Table 3. Comparison of our time/memory/data Tradeoff attacks with the previous ones

Attack	Data	Memory(-bit),(TB)	Time	Pre-computation
[12]	112	$\geq 2^{52.32}$-bit, $\geq$ 639 TB	$2^{66.80}$	$2^{68.87}$
[8]	2^{40}	$2^{52.58}$-bit, 770 TB	$2^{30.66}$	$2^{54.29}$
[8]	2^{41}	$2^{52.64}$-bit, 399 TB	$2^{29.66}$	$\approx 2^{56.70}$
[8]	2^{42}	$2^{50.69}$-bit, 207 TB	$2^{28.66}$	$\approx 2^{59.07}$
[8]	2^{43}	$2^{49.74}$-bit, 108 TB	$2^{27.66}$	$\approx 2^{61.42}$
ours	2^{39}	$2^{51.39}$-bit, 336 TB	$2^{40.66}$	$2^{44.03}$
ours	2^{40}	$2^{50.63}$-bit, 198 TB	$2^{39.66}$	$2^{43.39}$
ours	2^{41}	$2^{49.85}$-bit, 115 TB	$2^{38.66}$	$2^{43.81}$
ours	2^{42}	$2^{49.03}$-bit, 65 TB	$2^{37.66}$	$2^{45.36}$
ours	2^{43}	$2^{48.20}$-bit, 36 TB	$2^{36.66}$	$2^{47.09}$

5 Practical Implementation

To verify the validity of our attack, we experimentally test it on a reduced cipher with similar structure and properties as Sprout. In general, the simulation results match well with the theoretical estimates.

5.1 The Reduced Version of Sprout

Similarly, there is an 8-bit counter register, of which the lower 6 bits are a modulo 40 counter, denoted by $(c_t^5, c_t^4, c_t^3, c_t^2, c_t^1, c_t^0)$ at a given round t. The 3-th LSB c_t^3 of the counter is employed in the keystream generation. It should be noted that, c_t^3 has a cycle of length 40, i.e., in each cycle, this bit takes the values $\underbrace{0,0,...,0}_{8}\underbrace{1,1,...,1}_{8}\underbrace{0,0,...,0}_{8}\underbrace{1,1,...,1}_{8}\underbrace{0,0,...,0}_{8}$.

The reduced version of Sprout uses a 20-bit LFSR and a 20-bit NFSR. At time t, the LFSR state is $L^t = [l_t, l_{t+1}, ..., l_{t+19}]$, and it is updated recursively by f as $l_{t+20} = l_t \oplus l_{t+1} \oplus l_{t+14} \oplus l_{t+15} \oplus l_{t+16} \oplus l_{t+19}$. The NFSR state $N^t = [n_t, n_{t+1}, ..., n_{t+19}]$ is updated recursively by a nonlinear feedback function g as

$$\begin{aligned} n_{t+20} &= k_t^* \oplus c_t^3 \oplus l_t \oplus g(N^t) \\ &= k_t^* \oplus c_t^3 \oplus l_t \oplus n_t \oplus n_{t+13} \oplus n_{t+15} \oplus n_{t+17} \oplus n_{t+19} \\ &\oplus n_{t+2}n_{t+5} \oplus n_{t+3}n_{t+7} \oplus n_{t+8}n_{t+9} \oplus n_{t+1}n_{t+14} \oplus n_{t+16}n_{t+18} \oplus n_{t+6}n_{t+12} \\ &\oplus n_{t+13}n_{t+16}n_{t+17}n_{t+18} \oplus n_{t+10}n_{t+11}n_{t+12} \oplus n_{t+4}n_{t+7}n_{t+11}. \end{aligned}$$

Let u_t be $u_t = l_{t+1} \oplus l_{t+4} \oplus l_{t+17} \oplus n_{t+4} \oplus n_{t+10} \oplus n_{t+14}$, then

$$k_t^* = \begin{cases} k_t, \ 0 \le t \le 39 \\ k_{t(\text{mod } 40)} \cdot u_t, \ \text{otherwise} \end{cases}$$

Given the internal state at time t, the keystream bit z_t is generated as

$$z_t = h(n_{t+4}, l_{t+6}, l_{t+8}, l_{t+10}, l_{t+12}, l_{t+17}, l_{t+19}, l_{t+3}, n_{t+18}) \oplus l_{t+10} \oplus \left(\bigoplus\nolimits_{i \in A} n_{t+i} \right),$$

where $A = \{1, 3, 6, 15, 17\}$, and the filter function $h(\cdot)$ is defined as

$$h(\cdot) = n_{t+4}l_{t+6} \oplus l_{t+8}l_{t+10} \oplus l_{t+12}l_{t+17} \oplus l_{t+19}l_{t+3} \oplus n_{t+4}l_{t+12}n_{t+18}.$$

During the key/IV setup phase, since the key is fixed, first load the IV in the following way: $n_i = iv_i, 0 \le i \le 19$; $l_i = iv_{i+20}, 0 \le i \le 14$ and $l_i = 1, 15 \le i \le 18$, $l_{19} = 0$. Then run the cipher 160 rounds as follows.

- the LFSR update function is changed to $l_{t+20} = z_t \oplus f(L^t)$.
- the NFSR update function is changed to $n_{t+20} = z_t \oplus k_t^* \oplus l_t \oplus c_t^3 \oplus g(N_t)$.
- no keystream bit is generated.

After the initialization phase, the keystream generation phase starts and there is no feedback keystream anymore.

5.2 Attack Process

Suppose $l_{t+4} = 0$, $n_{t+3} = n_{t+4} = n_{t+5} = 0$, by a computer computation, there are 1728($> 2^{10}$) possible values for the following 13-bit of LFSR such that $H^{(1,1)}(\cdot)$ is 0 ($\in F_2^3$).

$$P^t = [l_{t+2}||l_{t+3}l_{t+7}l_{t+8}l_{t+9}||l_{t+10}l_{t+11}l_{t+12}l_{t+13}||l_{t+16}l_{t+17}l_{t+18}l_{t+19}] \subseteq L^t.$$

For example, P^t = `0x0000`, `0x0001`, ... We denote all the 1728 values of P^t as a_1,a_2,...,a_{1024},...,a_{1728}, where the first 1024 values are a_1 = `0x0000`, a_2 = `0x0001`,...,a_{1024} = `0x1ba1`. For convenience, several notations are defined as follows:

- $L^{*t} = L^t \backslash (\{l_{t+4}\} \cup P^t)$ of 6-bit.
- $N^{*t} = N^t \backslash \{n_{t+3}, n_{t+4}, n_{t+5}\}$ of 17-bit.

– Define $C = [c^3_{t-5}, ..., c^3_{t-1}, c^3_t, c^3_{t+1}, ..., c^3_{t+5}]$ of length 11, the employed counter array. There are 21 different counter arrays, denoted by hexadecimal numbers, they are

$$\texttt{0x007}, \texttt{0x00f}, \texttt{0x01f}, \texttt{0x03f}, \texttt{0x07f}, \texttt{0x0ff}, \texttt{0x1fe},$$
$$\texttt{0x3fc}, \texttt{0x7f8}, \texttt{0x7f0}, \texttt{0x7e0}, \texttt{0x7c0}, \texttt{0x780}, \texttt{0x700},$$
$$\texttt{0x601}, \texttt{0x403}, \texttt{0x600}, \texttt{0x400}, \texttt{0x000}, \texttt{0x001}, \texttt{0x003},$$

– Define $C' = [c^3_{t-1}]$ of length 1, and $C^* = [c^3_{t-5}, ..., c^3_{t-2}, c^3_t, c^3_{t+1}, ..., c^3_{t+5}]$ of length 10. There are 2 different values for C', denoted as $c'_1 = \texttt{0x0}$, $c'_2 = \texttt{0x1}$. If $c^3_{t-1} = \texttt{0x0}$, there are 13 different values for C^*, they are `0x007`, `0x00f`, `0x01f`,`0x03f`,`0x3c0`,`0x380`,`0x301`,`0x203`,`0x300`,`0x200`,`0x000`,`0x001`,`0x003`; If $c^3_{t-1} = \texttt{0x1}$, there are 8 different values for C^*, they are `0x03f`,`0x07f`,`0x0fe`, `0x1fc`,`0x3f8`,`0x3f0`,`0x3e0`,`0x3c0`.

Pre-processing Phase. For any state (L^t, N^t), suppose $l_{t+4} = 0$, $n_{t+3} = n_{t+4} = n_{t+5} = 0$. In the pre-processing phase, we construct 2×2^{10} tables $\mathrm{T}_{c'_j,a_i}$ indexed with c'_j and a_i, for $c'_1 = \texttt{0x0}$, $c'_2 = \texttt{0x1}$ and $a_1 = \texttt{0x0000}$, $a_2 = \texttt{0x0001}$,...,$a_{1024} = \texttt{0x1ba1}$. In Table $\mathrm{T}_{c'_j,a_i}$, 23-bit (L^{*t}, N^{*t}) are stored in the first column of a row such that

$$\begin{cases} P^t = [l_{t+2}||l_{t+3}l_{t+7}l_{t+8}l_{t+9}||l_{t+10}l_{t+11}l_{t+12}l_{t+13}||l_{t+16}l_{t+17}l_{t+18}l_{t+19}] = a_i \\ l_{t+10+i} \oplus n_{t+1+i} \oplus n_{t+3+i} \oplus n_{t+6+i} \oplus n_{t+15+i} \oplus n_{t+17+i} = 0, i = -1, 0, 1 \\ u_{t+j} = l_{t+1+j} \oplus l_{t+4+j} \oplus l_{t+17+j} \oplus n_{t+4+j} \oplus n_{t+10+j} \oplus n_{t+14+j} = 0, j = 0, 1, ..., 5 \\ u_{t-k} = l_{t+1-k} \oplus l_{t+4-k} \oplus l_{t+17-k} \oplus n_{t+4-k} \oplus n_{t+10-k} \oplus n_{t+14-k} = 0, k = 1, 2, ..., 5 \\ n_{t-1} \oplus l_{t-1} \oplus c^3_{t-1} \oplus g'(N^t) = 0 \ (non - linear) \end{cases}$$

Similarly, we can solve all the systems by choosing a set of unknowns as GUESS $= \{n_t, n_{t+1}, n_{t+6}, n_{t+7}, n_{t+10}, n_{t+11}, n_{t+15}, n_{t+16}, n_{t+17}\}$ with approximately 2^{23} basic operations. Besides, for each (c'_j, a_i) pair, there are expected 2^9 solutions, we store the 23-bit (L^{*t}, N^{*t}) of the internal state (4+13=17-bit are fixed for each table) in the first column of a row in table $\mathrm{T}_{c'_j,a_i}$. Further for this state and for each possible round constants C^*, get the corresponding 11-bit output $(z_{t-6}, ..., z_{t-2}, z_{t+2}, ..., z_{t+7})$ and put them in the second column as a sub-row indexed by C^*. The number of sub-row is 13 for $c'_1 = \texttt{0x0}$, while the number is 8 for $c'_2 = \texttt{0x1}$. In total, the memory needed is $M = 2^{10} \times 2^9 \times (23 + 11 \times 13) + 2^{10} \times 2^9 \times (23 + 11 \times 8) \approx 2^{27.11}$-bit, i.e., 17.3 MB[1].

Next, we present the *State Checking and Key Recovery Mechanism* specified for the reduced version of Sprout, which is similar to the one stated for Sprout.

State Checking and Key Recovery Mechanism. For a candidate state at time t, $L^t = [l_t, l_{t+1}, ..., l_{t+19}]$, $N^t = [n_t, n_{t+1}, ..., n_{t+19}]$, create a 40-bit vector K for the possible values associated with it:

[1] Since each table is expected to have 2^9 rows, we can only store 2 output bits in the second column of each row, indexed by 9 bits of the output. Thus, the memory can be reduced to $2^{10} \times 2^9 \times (23 + 2 \times 13) + 2^{10} \times 2^9 \times (23 + 2 \times 8) \approx 2^{25.46}$-bit, i.e., 5.5 MB.

1. Compute the value of n_{t-1} given by the keystream bit z_{t-2} as $n_{t-1} = z_{t-2} \oplus h(n_{t+2}, l_{t+4}, l_{t+6}, l_{t+8}, l_{t+10}, l_{t+15}, l_{t+17}, l_{t+1}, n_{t+16}) \oplus l_{t+8} \oplus \left(\bigoplus_{i \in A'} n_{t+i-2}\right)$ where $A' = \{3, 6, 15, 17\}$. And compute l_{t-1} by the LFSR updating equation as $l_{t-1} = l_{t+19} \oplus l_{t+18} \oplus l_{t+15} \oplus l_{t+14} \oplus l_{t+13} \oplus l_t$, and deduce from n_{t-1}, l_{t-1} the value k^*_{t-1} by the NFSR updating equation as $k^*_{t-1} = n_{t+19} \oplus c^3_{t-1} \oplus l_{t-1} \oplus g(N^{t-1})$.
2. Compute the value of $u_{t-1} = l_t \oplus l_{t+3} \oplus l_{t+16} \oplus n_{t+3} \oplus n_{t+9} \oplus n_{t+13}$ and combine it with the value of k^*_{t-1} obtained in Step 1:
   - if $u_{t-1} = 0$ and $k^*_{t-1} = 0$, set $t \to t-1$ and go back to Step 1.
   - if $u_{t-1} = 0$ and $k^*_{t-1} = 1$, there is a contradiction, conclude that this guess for state is not correct and stop.
   - if $u_{t-1} = 1$ and $k^*_{t-1} = 0$, check if $k_{(t-1) \mod 40}$ has already been set in K. If no, set it to 0. Set $t \to t-1$ and go back to Step 1. Else, if there is a contradiction, conclude that this guess for state is not correct and stop.
   - if $u_{t-1} = 1$ and $k^*_{t-1} = 1$, check if $k_{(t-1) \mod 40}$ has already been set in K. If no, set it to 1. Set $t \to t-1$ and go back to Step 1. Else, if there is a contradiction, conclude that this guess for state is not correct and stop.

By utilizing the pre-computed tables and the given keystream sample, the processing phase is carried out as follows.

The Internal State Recovery Algorithm. Given the 2×2^{10} tables $\mathrm{T}_{c'_j, a_i}$, and the keystream sample $\{z_t\}_{t \geq 0}$ having at least 2^{21} sample segments, the processing steps are as follows:

1. Search the keystream sequence $\{z_t\}_{t \geq 6}$ for the next non-considered block of 3 zeros, i.e., $z_{t-1} z_t z_{t+1} = 000$. If there are no more blocks, output a flag that the algorithm has failed.
2. For each detected block, compute the corresponding $C' = [c^3_{t-1}] \stackrel{\Delta}{=} c'$ and $C^* = [c^3_{t-5}, ..., c^3_{t-2}, c^3_t, c^3_{t+1}, ..., c^3_{t+5}] \stackrel{\Delta}{=} c^*$ from the time t. For $a_1 =$ `0x0000`, $a_2 =$ `0x0001`,...,$a_{1024} =$ `0x1ba1`, compare $(z_{t+2} z_{t+3} ... z_{t+7})$ after the zero-segment and $(z_{t-6} z_{t-5} ... z_{t-2})$ before the zero-segment with the memorized 11-bit segments in the second column of a sub-row indexed by c^* from the tables T_{c', a_i}, and do the following:
   - If the matching does not exist, go to the processing Step 1.
   - If the 11-bit sample segments match with a segment in table T_{c', a_i}, go to Step 3.
3. Read the corresponding state, check whether it is a correct state or not and recover the secret key by the *State Checking and Key Recovery Mechanism* stated before. If this state survives, recover and output the key, else go to Step 1.

5.3 Simulation Results

Our attacks have been fully implemented on one core of a single PC, running with Windows 7, Intel Core i3-2120 CPU @ 3.30 GHz and 4.00 GB RAM. In

general, the experimental results match the theoretical analysis quite well. We present the details as follows.

In our experiment, first of all, we constructed 2×2^{10} tables indexed by (c'_j, a_i) pairs for $c'_1 = \texttt{0x0}$, $c'_2 = \texttt{0x1}$ and $a_1 = \texttt{0x0000}$, $a_2 = \texttt{0x0001}$,...,$a_{1024} = \texttt{0x1ba1}$, storing the special internal states. We used 2×2^{10} text files to store the $(State, Keystream_1, keystream_2, ..., keystream_{count(|C^*|)})$ tuples named with the corresponding c'_j and a_i. Note that $count(|C^*|) = 13$ for $c'_1 = \texttt{0x0}$ and $count(|C^*|) = 8$ for $c'_2 = \texttt{0x1}$. Experimental results show that there are 496 or 504 or 520 or 528 rows in each table, and totally $524448 (\approx 2^{19})$ rows for $c'_1 = \texttt{0x0}$, $524128 (\approx 2^{19})$ rows for $c'_2 = \texttt{0x1}$. Thus the memory needed in the simulation is $524448 \times (23 + 11 \times 13) + 524128 \times (23 + 11 \times 8) \approx 2^{27.11}$-bit, i.e., 17.3 MB, which matches the theoretical estimate quite well.

For the key recovery algorithm illustrated above, the data complexity is estimated by the probability that an internal state (L^t, N^t) is a special state satisfying:

(1) $l_{t+4} = 0$, $n_{t+3} = n_{t+4} = n_{t+5} = 0$,
(2) $P^t = a_1$ or $P^t = a_2$ or ... $P^t = a_{1024}$,
(3) $l_{t+10+d} \oplus \left(\bigoplus_{i \in A} n_{t+i+d}\right) = 0$ for $d = -1, 0, 1$,
(4) $u_{t+j} = 0$, for $j = 0, 1, ..., 5$,
(5) $u_{t-k} = 0$, for $k = 1, 2, ..., 5$,

Thus the theoretical estimate is $D = 2^{21}$. In the experiment, we used the RC4 cipher to randomly generate 2^{15} (K, IV) pairs and for each randomly chosen (K, IV) pair, we ran the cipher and generated 2^{21} keystream bits. Results show that we can get a special state at time $t \leq 2^{21}$ for $20423 (\approx 2^{14.32})$ (K, IV) pairs. For example, suppose (K, IV) pair be

$$K = 10101001010110011010101100100110001101 10$$
$$IV = 11010101101001001110100110010111011$$

where the left-most bit represents the value for index 0. At time $t = 580697 (\approx 2^{19.14})$, a special state arises in Table $\mathrm{T}_{c',a_{140}}$, where $c' = \texttt{0x0}$ and $a_{140} = \texttt{0x0191}$, such that (1)(3)(4)(5) hold and $P^t = \texttt{0x0191}$. This internal state is

$$L^t = 11110000010001110000$$
$$N^t = 00100000011011010110$$

In the internal state and key recovery algorithm, we search the keystream sequence for the 3 zeros blocks, and for each block, we try to find matching pairs, and further recover the key. In the experiment, we first searched the given keystream sequence and collected the time instances t implying 3 zeros. The expected number of such instances is $2^{21} \times 2^{-3} = 2^{18}$. Besides, for each 11-bit output, the expected number of candidate states is $\frac{2^{19}}{2^{11}} = 2^8$ producing this output. Thus we go through all the time instances, and for each time instance, we go through all the candidate states. We have also verified by experiments that 4 more clocks of output is enough for checking the validity of the state and the recovery of the key bits for each candidate. In total, the estimate of the time

complexity is $2^{18} \times 2^8 \times 4 = 2^{28}$. In the simulation, for the (K, IV) pair above, we have recovered all the key bits within 1 hour.

6 Conclusion

In this paper, we have studied the security of Sprout-like stream ciphers in a unified framework from the viewpoint of k-normality of the augmented function. We made a systematic security analysis based on this property and developed a dedicated TMD tradeoff attack framework for such designs. In particular, it is shown that Sprout can be broken by various TMD tradeoffs. Our attack is highly flexible and compares favorably to all the previous attacks on Sprout, which demonstrates the superiority of the new method. We believe that stream ciphers with shorter internal state may suffer from the time/memory/data tradeoff attacks and the k-normality of the augmented function should be taken into account for new stream cipher designs.

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: A New Version of Grain-128 with Authentication, Symmetric Key Encryption Workshop 2011, DTU, Denmark
2. Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 451–470. Springer, Heidelberg (2015)
3. Banik Subhadeep., Some Results on Sprout. http://eprint.iacr.org/2015/327.pdf
4. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000)
5. Braeken, A., Wolf, C., Preneel, B.: Normality of vectorial functions. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 186–200. Springer, Heidelberg (2005)
6. Dobbertin, H.: Construction of bent functions and balanced Boolean functions with high nonlinearity. In: Preneel, B. (ed.) Fast Software Encryption. LNCS, vol. 1008, pp. 61–74. Springer, Heidelberg (1995)
7. http://www.ecrypt.eu.org/stream/
8. Esgin, M.F., Kara, O.: Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. http://eprint.iacr.org/2015/289.pdf
9. Hao, Y.: A Related-Key Chosen-IV Distinguishing Attack on Full Sprout Stream Cipher. http://eprint.iacr.org/2015/231.pdf
10. Hell, M., Johansson, T., Meier, W.: Grain - a Stream Cipher for Constrained Environments. Int. J. Wirel. Mobile Comput. **2**(1), 86–93 (2007)
11. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: grain-128. In: IEEE International Symposium on Information Theory - ISIT 2006 (2006)
12. Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of full sprout. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology – CRYPTO 2015. LNCS, vol. 9215, pp. 663–682. Springer, Heidelberg (2015)
13. Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key Recovery from State Information of Sprout: Application to Cryptanalysis and Fault Attack. http://eprint.iacr.org/2015/236.pdf

14. Mihaljevic, M.J., Gangopadhyay, S., Paul, G., Imai, H.: Internal state recovery of grain-v1 employing normality order of the filter function. Inf. Secur. IET **6**(2), 55–64 (2012)
15. Mihaljevic, M.J., Gangopadhyay, S., Paul, G., Imai, H.: Generic cryptographic weakness of k-normal boolean functions in certain stream ciphers and cryptanalysis of Grain-128. Periodica Math. Hung. **65**(2), 205–227 (2012)