A Related-Key Chosen-IV Distinguishing Attack on Full Sprout Stream Cipher

Yonglin Hao

Department of Computer Science and Technology, Tsinghua Universtiy, Beijing 100084, China
haoyl14@mails.tsinghua.edu.cn

Abstract. Sprout is a new lightweight stream cipher proposed at FSE 2015. According to its designers, Sprout can resist time-memory-data trade-off (TMDTO) attacks with small internal state size. However, we find a weakness in the updating functions of Sprout and propose a related-key chosen-IV distinguishing attacks on full Sprout. Under the related-key setting, our attacks enable the adversary to detect non-randomness on full 320-round Sprout with a practical complexity of $\tilde{O}(2^4)$ and find collisions in 256 output bits of full Sprout with a complexity of $\tilde{O}(2^7)$.
Furthermore, when considering possible remedies, we find that only by modifying the updating functions and output function seems unlikely to equip Sprout with better resistance against this kind of distinguisher. Therefore, it is necessary for designers to give structural modifications.

1 Introduction

In the modern society, the wide use of RFID tags and sensor networks has stimulated the need of lightweight cryptographic primitives that require very limited resources (the area size on the chip, memory, power consumption etc.) while still providing good security. During the past few years, several lightweight block ciphers are proposed (eg. PRESENT [1], CLEFIA [2], LED [3], LBlock [4], mCrypton [5] etc.) followed by enormous cryptanalysis results evaluating their secure margins (such as [6,7,8,9,10] etc).

Besides block ciphers, many stream ciphers (such as Trivium [11], Grain-v1 [12], Grain-128a [13] etc.) are also suitable for lightweight applications. Comparing with lightweight block ciphers, the stream ciphers have advantages in providing higher throughput. However, there is a requirement that the internal size of stream ciphers must be at least twice the security parameter in order to resist the time-memory-data trade-off (TMDTO) attacks [14,15,16] aiming at the recovery of the internal states. This limitation burdens the stream ciphers with a larger area size than lightweight block ciphers and barricades the widespread of stream ciphers in lightweight applications.

However, recently at FSE 2015, Frederik and Vasily [17] proposed a new paradigm for lightweight stream cipher design to resist TMDTO using shorter internal state which in turn breaks the previous limitation in internal state size. To achieve this goal, the authors involves the secret key not only in the initialization process but also in the key-stream generation phase. They also claim that even the a internal state were to be recovered, it would not be likely for the adversary

to further retrieve the secret key bits with a complexity lower than exhaustive search. To support their idea, they also specified an instance of this new stream cipher family called "Sprout".

The Sprout stream cipher features a 80-bit secret key and a 80-bit internal state composed of a 40-bit NFSR and a 40-bit LFSR. As soon as its proposal, Sprout receives severe challenges. Firstly, Maitra et al. cryptanalyzed Sprout and refuted many claims made in [17]. In their paper [18], they proved that it is possible to recover all the secret key bits from around 850 key-stream bits, the complete knowledge of NFSR and a partial knowledge of LFSR with a complexity of 2^{54}. Based on their findings, they also launched a fault attack on Sprout requiring around 120 faults in random locations. Then, Lallemand and Naya-Plasencia give a key recovery attack on full Sprout [19]. They exploit the non-linear influence of the key bits on the updating functions and, evolving some divide-and-conqure technique, manage to recover the whole 80-bit secret key wit a time complexity of $2^{69.36}$ and very low data complexity [19]. They implement the attack on a toy version of Sprout that conserves the main properties exploited in the attack.

The two previous cryptanalysis results on Sprout are both considering security under the single key model aiming at recovering secret key bits. The complexities of their whole attacks are high and beyond the reach of practical implementation. In this paper, we evaluate the secure margin of Sprout under the related-key model. Instead of recovering secret key bits, we simply try to detect non-randomness of key-stream bits generated by full Sprout. Comparing with those under the single-key model, our attacks are far more practical and can verified easily using a PC within seconds.

Our contribution. We find a weakness in the updating functions of Sprout. Based on our finding, we manage to launch a practical related-key chosen-IV distinguishing attacks on full Sprout. Our method can distinguish Sprout from random stream with $\tilde{O}(2^4)$ key-IV pairs. We can also find two key-IV pairs that collides at the first 256 output bits with a complexity $\tilde{O}(2^7)$. Furthermore, we prove that this kind of distinguishing attacks can still work even if the updating functions and output function are changed. This finding suggests that the designers should revisit the structure of Sprout rather than simply modifying small details.

Organization of the Paper. Section 2 provides the description of Sprout and some notations used in this paper. Section 3 give our findings about Sprout. In Section 4, we describe the procedure of our attack. We further prove the necessity of structural modification in Section 5. Finally, we summarize our paper in Section 6.

2 Description of Sprout

The Sprout stream cipher generates keystream bits from 70 public IV bits, denoted by $v_0, \cdots, v_{79}$, and an 80 secret key bits, denoted by $x_0, \cdots, x_{79}$. It consists of a 40-bit NFSR and a 40-bit LFSR, denoted by $\mathbf{N}$ and $\mathbf{L}$ respectively. $\mathbf{N}$ and $\mathbf{L}$ are first initialized as follows:

$$\mathbf{N} = (n_0, \cdots, n_{39}) = (v_0, \cdots, v_{39}),$$
$$\mathbf{L} = (l_0, \cdots, l_{39}) = (v_{40}, \cdots, v_{69}, 1, \cdots, 1, 0).$$

For $t \geq 0$, the feedback function of LFSR is defined as:

$$l_{t+40} = l_t + l_{t+5} + l_{t+15} + l_{t+20} + l_{t+25} + l_{t+34} \tag{1}$$

and that of NFSR is

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

where the key k_t^* is assigned according to a round key function as follow:

$$k_t^* = \begin{cases} x_t & 0 \le t \le 79 \\ (x_{t \mod 80}) \cdot (l_{t+9} + l_{t+21} + l_{37} + n_{t+9} + n_{t+20} + n_{t+29}), t \ge 80 \end{cases} \tag{3}$$

and the counter bit c_t^4 is the 4-th significant bit of the integer $(t \mod 80)$. The output function is defined as

$$z_t = \sum_{j \in B} n_{t+j} + l_{t+30} + h(t) \tag{4}$$

where $B = \{1, 6, 15, 17, 23, 28, 34\}$ and

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

After $l_0, \cdots, l_{39}$ and $n_0, \cdots, n_{39}$ are settled, the Sprout state then runs 320 initialization rounds not producing an output but feeding the output back into both LFSR and NFSR. So the actual output bits are $z_{320}, z_{321}, \cdots$ We demonstrate the overall structure of Sprout in Figure 1. As can be seen, the structure of Sprout resembles those of the Grain-x stream cipher family [20,13,12].

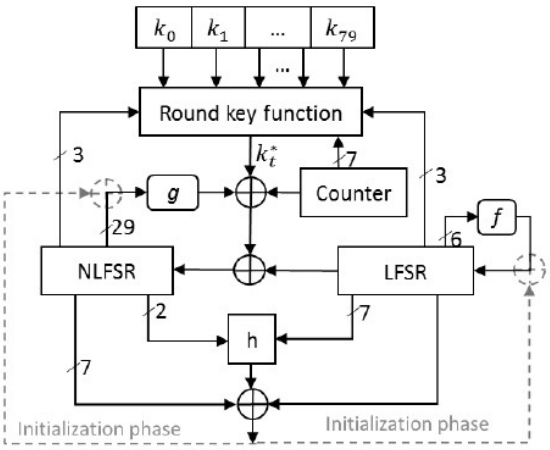

Figure 1. The overall structure of Sprout.

Before our descriptions, we give some notations used throughout this paper.

b We use small letters to represent 0-1 bits. More specifically, the 80 secret key bits are represented by letter x_i $(i = 0, \cdots, 79)$ and the 70 public IV bits are represented by letter v_j $(j = 0, \cdots, 69)$.

b We denote the 0-1 vectors by bold letters such as $\mathbf{k}$ for the 80-bit key and $\mathbf{v}$ for 70-bit IV. We also denote the internal states of NFSR and LFSR after round t as

$$\mathbf{N}_t = (n_t, \cdots, n_{t+39})$$
$$\mathbf{L}_t = (l_t, \cdots, l_{t+39}).$$

b[i] We refer to the i-th (i starts from 0) bit of vector $\mathbf{b}$ as $\mathbf{b}[i]$. For example, the i-th bit of secret key $\mathbf{k}$ defined above, we have $\mathbf{k}[i] = x_i$ $(i = 0, \cdots, 79)$.

Besides, for $t \geq 80$, we specifically denote the value λ_t as

$$\lambda_t = l_{t+9} + l_{t+21} + l_{37} + n_{t+9} + n_{t+20} + n_{t+29}.$$

The definition of λ_t is for the convenience of our interpretations in the following section.

3 The Main Observations on Sprout

Before interpreting our observations about Sprout, we further give a notation commonly used in the remainder of this paper. the following definition. For any secret key $\mathbf{k} = (x_0, \cdots, x_{79})$ and public IV $\mathbf{v} = (v_0, \cdots, v_{69})$, we denote corresponding $\hat{\mathbf{k}}$ and $\hat{\mathbf{v}}$ as follows:

$$\hat{\mathbf{k}} = (x_0 + 1, x_1, \cdots, x_{79}) \tag{5}$$
$$\hat{\mathbf{v}} = (v_0 + 1, v_1, \cdots, v_{69}). \tag{6}$$

We denote the intermediate state bits n_i, l_i, z_i as the bits deduced from $(\mathbf{k}, \mathbf{v})$ and denote $\hat{n_i}$, $\hat{l_i}$, $\hat{z_i}$ as their counterparts deduced from $(\hat{\mathbf{k}}, \hat{\mathbf{v}})$. $\mathbf{N}_t$ $(\mathbf{L}_t)$ and $\hat{\mathbf{N}}_t$ $(\hat{\mathbf{L}}_t)$ are defined in the same way.

Our main finding about Sprout can be summarized as Proposition 1.

Proposition 1. *For any randomly chosen key-IV pair* $(\mathbf{k}, \mathbf{v})$ *and its corresponding* $(\hat{\mathbf{k}}, \hat{\mathbf{v}})$*, after* $80t$ *Sprout rounds* $(t \geq 1)$*, we have*

$$Pr\{z_{80t} = \hat{z}_{80t}\} = 2^{-1} + 2^{-t} \tag{7}$$

Proof. In the first 80 rounds, x_0 only takes part in the updating of n_{40}. We found that

$$n_{40} = (v_0 + x_0) + C = (v_0 + x_0 + 1 + 1) + C = \hat{n}_{40}$$

where C is irrelevant with v_0 and x_0. So we have

$$Pr\{n_{40} = \hat{n}_{40}\} = Pr\{\mathbf{L}_1 = \hat{\mathbf{L}}_1\} = Pr\{\mathbf{N}_1 = \hat{\mathbf{N}}_1\} = 1.$$

In the following 79 rounds, besides $\mathbf{L}_1$ and $\mathbf{N}_1$ only the rest 79 identical key bits are involved in the updating process, so we have

$$Pr\{\mathbf{L}_j = \hat{\mathbf{L}}_j\} = Pr\{\mathbf{N}_j = \hat{\mathbf{N}}_j\} = 1.$$

for $j = 1, \cdots, 80$ and (7) holds when $t = 1$.

Then, when updating n_{120} ($n_{\hat{120}}$), the bit $\mathbf{k}[0]$ ($\hat{\mathbf{k}}[0]$) will not be involved with probability

$$Pr\{\lambda_{80} = \hat{\lambda}_{80} = 0\} = Pr\{n_{120} = \hat{n}_{120} = 0\} = 2^{-1}.$$

If we $n_{120} = \hat{n}_{120}$, we can also deduce that

$$Pr\{\mathbf{L}_{80+j} = \hat{\mathbf{L}}_{80+j} | n_{120} = \hat{n}_{120}\} = Pr\{\mathbf{N}_{80+j} = \hat{\mathbf{N}}_{80+j} | n_{120} = \hat{n}_{120}\} = 1.$$

for $j = 1, \cdots, 80$ and (7) holds when $t = 2$.

More generally, for $t > 1$ and $j = 1, \cdots, 80$, we have

$$\begin{aligned} &Pr\{\mathbf{L}_{80(t-1)+j} = \hat{\mathbf{L}}_{80(t-1)+j} | \bigwedge_{i=1}^{t-1} (n_{80i+40} = \hat{n}_{80i+40})\} \\ &= Pr\{\mathbf{N}_{80(t-1)+j} = \hat{\mathbf{N}}_{80(t-1)+j} | \bigwedge_{i=1}^{t-1} (n_{80i+40} = \hat{n}_{80i+40})\} = 1. \end{aligned} \tag{8}$$

and

$$Pr\{\bigwedge_{j=1}^{t-1} (n_{80j+40} = \hat{n}_{80j+40})\} = Pr\{\bigwedge_{j=1}^{t-1} (\lambda_{80j} = \hat{\lambda}_{80j} = 0)\} = 2^{-(t-1)}. \tag{9}$$

So we can deduce that

$$Pr\{z_{80t} = \hat{z}_{80t}\} = 2^{-1} Pr\{\overline{\bigwedge_{j=1}^{t-1} (\lambda_{80j} = \hat{\lambda}_{80j} = 0)}\} + Pr\{\bigwedge_{j=1}^{t-1} (\lambda_{80j} = \hat{\lambda}_{80j} = 0)\} = 2^{-1} + 2^{-t}$$

which is exactly (7). □

With the proof of Proposition 1, we can deirectly deduce the following Corollary 1 which is the basis of our known-key collision attack.

Corollary 1. *For $t \geq 1$, we have*

$$Pr\{\bigwedge_{j=0}^{79} (z_{80t-j} = \hat{z}_{80t-j})\} \geq 2^{-(t-1)}. \tag{10}$$

Proof. It is obvious that (10) holds for $t = 1$ since $\mathbf{L}_j = \hat{\mathbf{L}}_j$ and $\mathbf{N}_j = \hat{\mathbf{N}}_j$ for $j = 1, \cdots, 80$.

For $t > 1$, with (8), we can easily deduce that

$$Pr\{\bigwedge_{j=0}^{79} (z_{80t-j} = \hat{z}_{80t-j}) | \bigwedge_{i=1}^{t-1} (n_{80i+40} = \hat{n}_{80i+40})\} = 1.$$

Adding the knowledge of (9), we have

$$\begin{aligned} Pr\{\bigwedge_{j=0}^{79} (z_{80t-j} = \hat{z}_{80t-j})\} &= 2^{-80} Pr\{\overline{\bigwedge_{j=1}^{t-1} (\lambda_{80j} = \hat{\lambda}_{80j} = 0)}\} + Pr\{\bigwedge_{j=1}^{t-1} (\lambda_{80j} = \hat{\lambda}_{80j} = 0)\} \\ &> Pr\{\bigwedge_{j=1}^{t-1} (\lambda_{80j} = \hat{\lambda}_{80j} = 0)\} = 2^{-(t-1)}. \end{aligned}$$

and therefore we have proved (10). □

4 Related-Key Chosen-IV Distinguishing Attack on Full Sprout

With Proposition 1, we can naturally conduct a related-key chosen-IV distinguishing attack on full 320-round Sprout stream cipher. With practical complexity, we can distinguish full Sprout from random 0-1 stream. The procedure is as follows:

1. We randomly select m key-IV pairs $(\mathbf{k}_i, \mathbf{v}_i)$ and compute their corresponding $(\hat{\mathbf{k}}_i, \hat{\mathbf{v}}_i)$ $(i = 1, \cdots, m)$ as (5)(6).
2. For all $i = 1, \cdots, m$, compute their first output bit of Sprout stream, denoted as z^i_{320} and $\hat{z}^i_{320}$. Count the number of i's satisfying $z^i_{320} = \hat{z}^i_{320}$ and denoted by ξ, which means

$$\xi := \#\{i | z^i_{320} = \hat{z}^i_{320}, i = 1, \cdots, m\}.$$

   According to Proposition 1, the Sprout output streams satisfy

$$\lim_{m\to\infty} \frac{\xi}{m} = \frac{1}{2} + (\frac{1}{2})^4 = 0.5625.$$

   For random bit streams, the ratio $\frac{\xi}{m}$ will approach 0.5 instead.

Since the bias for full 320 round Sprout is 2^{-4}, the success probability is significant enough for $m \sim \tilde{O}(2^4)$. For precise consideration, we set $m = 2^{20}$ and experimentally verified the correctness of our attack. Some of the statistics are shown in Table 1.

Table 1. Experiment Verifications ($m = 2^{20}$)

Output	$\frac{\xi}{m}$
z_{80}	1
z_{160}	0.749512
z_{240}	0.635986
z_{320}	0.563721

By utilizing Corollary 1, we can also detect non-randomness by launching a related-key chosen-IV collision attack with practical complexity. For example, by testing $\tilde{O}(2^7)$ key-IV pairs $(\mathbf{k}, \mathbf{v})$, we can find one pair that collides with its corresponding $(\hat{\mathbf{k}}, \hat{\mathbf{v}})$ in the 256 output bits $z_{320}, \cdots, z_{575}$. We give a collision of such in Table 2.[1]

[1] For $\mathbf{k}$, the two words are represented as $x_{63}\| \cdots \|x_0, x_{79}\| \cdots \|x_{64}$. For $\mathbf{v}$, the two words are represented as $v_{63}\| \cdots \|v_0, v_{69}\| \cdots \|v_{64}$. Output bits, such as $z_{320\sim447}$, are represented as $z_{383}\| \cdots \|z_{320}$, $z_{447}\| \cdots \|z_{384}$ etc.

Table 2. Collision at $z_{320}, \cdots, z_{575}$

$\mathbf{k}$	0xea6daa88c5cf3c5e, 0xf7d0
$\hat{\mathbf{k}}$	0xea6daa88c5cf3c5f, 0xf7d0
$\mathbf{v}$	0xaff0e857756369ff, 0x17
$\hat{\mathbf{v}}$	0xaff0e857756369fe, 0x17
$z_{320\sim447}$	0x49e6c49522ba16e7, 0x4eb385fceb377844
$z_{448\sim575}$	0xa51d1fce4327928e, 0x1bf995f95e2f2a81

5 The Necessity of Structural Modifications

With the above findings, it is natural for us to consider the countermeasure to resist such a kind of distinguisher. The simplest and most direct thought is modifying the updating functions of NFSR and LFSR. The output function may be modified as well. However, we will illustrate in this section that it is not likely for Sprout to resist this kind of distinguishing attacks only by modifying the updating functions and the output function. It is necessary to consider structural modifications.

For $t \geq 0$ and $0 \leq i \leq 39$, we define vectors $\mathbf{N}_t \backslash n_{t+i}$ and $\mathbf{L}_t \backslash l_{t+i}$ as

$$\begin{aligned} \mathbf{N}_t \backslash n_{t+i} &= (n_t, \cdots, n_{t+i-1}, n_{t+i+1}, \cdots, n_{t+39}) \\ \mathbf{L}_t \backslash l_{t+i} &= (l_t, \cdots, l_{t+i-1}, l_{t+i+1}, \cdots, l_{t+39}). \end{aligned}$$

According to the structure shown in Figure 1, with $t < 320$, we can present the output function in a more general manner as

$$z_t = n_t \cdot h_1(\mathbf{N}_t \backslash n_t, \mathbf{L}_t \backslash l_t) + l_t \cdot h_2(\mathbf{N}_t \backslash n_t, \mathbf{L}_t \backslash l_t) + n_t l_t \cdot h_3(\mathbf{N}_t \backslash n_t, \mathbf{L}_t \backslash l_t) + h_4(\mathbf{N}_t \backslash n_t, \mathbf{L}_t \backslash l_t).$$

The updating functions of NFSR and LFSR can be represented respectively as follows:

$$n_{t+40} = (k_t^* + n_t \cdot g_1(\mathbf{N}_t \backslash n_t) + l_t + z_t) + g_2(\mathbf{N}_t \backslash n_t), \text{ where } g_1 \not\equiv 0 \tag{11}$$

$$l_{t+40} = (l_t + z_t) + f_1(\mathbf{L}_t \backslash l_t). \tag{12}$$

As can be seen from the previous sections, the effectiveness of the distinguisher relies on the fact that the differences $\Delta k_0^* = \mathbf{k}[0] \oplus \hat{\mathbf{k}}[0]$ and $\Delta n_0 = \mathbf{N}[0] \oplus \hat{\mathbf{N}}[0]$ fail in affecting the newly generated bits n_{40}, l_{40} ($\Delta n_{40} = \Delta l_{40} = 0$) which can be represented according to (11) (12) as follows

$$\begin{aligned} n_{40} &= (k_0^* + n_0 \cdot g_1 + l_0 + z_0) + g_2 \\ l_{40} &= (l_0 + z_0) + f_1 \\ z_0 &= n_0 \cdot h_1 + l_0 \cdot h_2 + n_0 l_0 \cdot h_3 + h_4 \end{aligned}$$

where g_1, g_2 are functions of $v_1, \cdots, v_{39}$, f_1 is a function of $v_{40}, \cdots, v_{69}$ and h_1, h_2, h_3, h_4 are functions of $v_1, \cdots, v_{69}$. Since $g_1 \neq 0$, we consider the updating functions and the output function in the following conditions:

1. If $h_1 \equiv 0$, this is the situation of Sprout. Even if $h_3 \not\equiv 0$, we can set $l_0 = 0$ by assigning $\mathbf{v}[40] = \hat{\mathbf{v}}[40] = 0$ and eliminate the term involving h_3.

2. If $h_1 \not\equiv 0$ and $h_1 \not\equiv g_1$, we can select the $v_1, \cdots, v_{69}$ to make $h_1 = 0$ and $g_1 = 1$ (about 2^{67} values) and the attack can still work;
3. If $h_1 \equiv g_1$, then h_1 only relies on $v_1, \cdots, v_{39}$. We modify the $\mathbf{v}$ and $\hat{\mathbf{v}}$ as $\mathbf{v}[40] = \hat{\mathbf{v}}[40] = 0$ to enable $l_0 = \hat{l}_0 \equiv 0$. Then we have an even more efficient distinguisher that for all secret key $\mathbf{k} \in \{0,1\}^{80}$, we have

$$z_{320}(\mathbf{k}, \mathbf{v}) \equiv z_{320}(\mathbf{k}, \hat{\mathbf{v}})$$

since the bit n_0 has totally been eliminated from the updating functions.

Therefore, it is unlikely for Sprout to resist this kind of distinguishers only by modifying its updating functions and output function. Structural modifications are necessary.

6 Conclusion

In this paper, we find a weakness in the design of newly proposed stream cipher Sprout. Based on the weakness, we manage to launch a related-key chosen-IV attack on the full cipher with practical complexity and 100% success probability. We also prove that the effectiveness of this distinguishing attack relies more on the structural weakness of Sprout. It seems unlikely for Sprout to resist this kind of distinguishers only by modifying its updating functions and output function. We appeal that the designers should modify the overall structure of Sprout for its thorough resistance against related-key chosen-IV distinguishing attacks.

References

1. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. Springer (2007)
2. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher clefia. In: Fast software encryption, Springer (2007) 181–195
3. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The led block cipher. In: Cryptographic Hardware and Embedded Systems–CHES 2011. Springer (2011) 326–341
4. Wu, W., Zhang, L.: Lblock: a lightweight block cipher. In: Applied Cryptography and Network Security, Springer (2011) 327–344
5. Lim, C.H., Korkishko, T.: mcrypton–a lightweight block cipher for security of low-cost rfid tags and sensors. In: Information Security Applications. Springer (2006) 243–258
6. Nakahara Jr, J., Sepehrdad, P., Zhang, B., Wang, M.: Linear (hull) and algebraic cryptanalysis of the block cipher present. In: Cryptology and Network Security. Springer (2009) 58–75
7. Li, W., Gu, D.W., Zhao, C., Liu, Z.Q., Liu, Y.: Security analysis of the led lightweight cipher in the internet of things. Jisuanji Xuebao(Chinese Journal of Computers) **35**(3) (2012) 434–445
8. Shibayama, N., Kaneko, T.: A new higher order differential of clefia. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **97**(1) (2014) 118–126
9. Karakoç, F., Demirci, H., Harmancı, A.E.: Impossible differential cryptanalysis of reduced-round lblock. In: Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems. Springer (2012) 179–188
10. Park, J.H.: Security analysis of mcrypton proper to low-cost ubiquitous computing devices and applications. International Journal of Communication Systems **22**(8) (2009) 959–969

11. De Canniere, C., Preneel, B.: Trivium. In: New Stream Cipher Designs. Springer (2008) 244–266
12. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. International Journal of Wireless and Mobile Computing **2**(1) (2007) 86–93
13. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. International Journal of Wireless and Mobile Computing **5**(1) (2011) 48–59
14. Babbage, S.: Improved exhaustive search attacks on stream ciphers. In: Security and Detection, 1995., European Convention on, IET (1995) 161–166
15. Golić, J.D.: Cryptanalysis of alleged a5 stream cipher. In: Advances in CryptologyEUROCRYPT97, Springer (1997) 239–255
16. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Advances in CryptologyASIACRYPT 2000. Springer (2000) 1–13
17. Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states, FSE (2015)
18. Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key recovery from state information of sprout: Application to cryptanalysis and fault attack. Cryptology ePrint Archive, Report 2015/236 (2015) `http://eprint.iacr.org/`.
19. Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of full sprout. Cryptology ePrint Archive, Report 2015/232 (2015) `http://eprint.iacr.org/`.
20. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: IEEE International Symposium on Information Theory, ISIT, Citeseer (2006)