Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 457275, 7 pages
http://dx.doi.org/10.1155/2014/457275

Research Article

Cryptanalysis of Loiss Stream Cipher-Revisited

Lin Ding, Chenhui Jin, Jie Guan, and Qiuyan Wang

Information Science and Technology Institute, Zhengzhou 450000, China

Correspondence should be addressed to Lin Ding; dinglin_cipher@163.com

Received 15 November 2013; Revised 5 May 2014; Accepted 5 May 2014; Published 27 May 2014

Academic Editor: Renat Zhdanov

Loiss is a novel byte-oriented stream cipher proposed in 2011. In this paper, based on solving systems of linear equations, we propose an improved Guess and Determine attack on Loiss with a time complexity of 2^{231} and a data complexity of 2^{68}, which reduces the time complexity of the Guess and Determine attack proposed by the designers by a factor of 2^{16}. Furthermore, a related key chosen *IV* attack on a scaled-down version of Loiss is presented. The attack recovers the 128-bit secret key of the scaled-down Loiss with a time complexity of 2^{80}, requiring 2^{64} chosen *IV*s. The related key attack is minimal in the sense that it only requires one related key. The result shows that our key recovery attack on the scaled-down Loiss is much better than an exhaustive key search in the related key setting.

1. Introduction

Many stream ciphers have been proposed over the past 20 years. Most of them are constructed using a linear feedback shift register (LFSR), which is easily implemented in hardware, but the software implementations are mostly slow. In recent years, several word-oriented stream ciphers have been proposed and standardized, such as ZUC [1], proposed for use in the 4G mobile networks, SNOW3G [2] deployed in the 3GPP networks, and also four software-oriented finalists of eSTREAM project (i.e., SOSEMANUK [3], HC-128 [4], Rabbit [5], and Salsa 20/12 [6]).

In 2011, the Loiss stream cipher [7] was proposed by a team from the State Key Laboratory of Information Security in China. Loiss is a novel byte-oriented stream cipher, which takes a 128-bit secret key and a 128-bit initial vector as inputs and outputs a keystream of bytes. Loiss is based on a linear feedback shift register and utilizes a structure called byte-oriented mixer with memory (BOMM) in the filter generator, which aims to improve the resistance against algebraic attacks, linear distinguishing attacks, and fast correlation attacks. The designers hope Loiss can enrich applications of orthomorphic permutations in cryptography and motivate the research on cryptographic properties of orthomorphic permutations. By exploiting some differential properties of the BOMM structure during the cipher initialization phase, two related key attacks on Loiss were independently proposed in [8, 9]. These results show that the additional design complication, that is, the addition of the BOMM mechanism, weakens the cipher instead of strengthening it. Naturally, an open problem was left for future research, that is, whether the scaled-down Loiss, obtained by getting rid of the BOMM from Loiss and keeping other parts same as Loiss, is resistant against related key attack.

No attack on Loiss has been published except for the two related key attacks showed in [8, 9]. In the specification of Loiss stream cipher, the designers present a Guess and Determine attack on Loiss, which has a time complexity of 2^{247} with a data complexity of 2^{52}. In fact, the time complexity can be reduced at the cost of increased data complexity. In this paper, based on solving systems of linear equations, we propose an improved Guess and Determine attack on Loiss, which has a time complexity of 2^{231} with a data complexity of 2^{68}. Furthermore, by exploiting the weakness of a scaled-down version of Loiss during its initialization phase, a related key chosen *IV* attack on the scaled-down Loiss is given. The attack recovers the 128-bit secret key of the scaled-down Loiss with time complexity of 2^{80}, requiring one related key and 2^{64} chosen *IV*s. The related key attack is minimal in the sense that it only requires one related key. The result shows that our key recovery attack on the scaled-down Loiss is much better than an exhaustive key search in the related key setting.

The rest of the paper is organized as follows. A brief description of Loiss stream cipher is given in Section 2. In Section 3, an improved Guess and Determine attack on full Loiss is presented. Section 4 gives a related key chosen *IV* attack on scaled-down Loiss. Concluding remarks are given in Section 5.

2. Brief Description of the Loiss Stream Cipher

In this section, we recall the Loiss stream cipher briefly; for more details, refer to [7]. Loiss consists of three parts: linear feedback shift register (LFSR), the nonlinear function F, and a structure called byte-oriented mixer with memories (BOMM) as a part of the nonlinear filter generator; see Figure 1 [7].

2.1. Keystream Generator of Loiss. The LFSR contains 32-byte registers. Denote by $(s_0^{(t)}, s_1^{(t)}, \ldots, s_{31}^{(t)})$ the state of LFSR at time t $(t \geqslant 0)$. Then the state at time $t+1$ satisfies

$$\begin{aligned} s_{31}^{(t+1)} &= s_{29}^{(t)} \oplus \alpha s_{24}^{(t)} \oplus \alpha^{-1} s_{17}^{(t)} \oplus s_{15}^{(t)} \\ &\quad \oplus s_{11}^{(t)} \oplus \alpha s_5^{(t)} \oplus s_2^{(t)} \oplus \alpha^{-1} s_0^{(t)}, \qquad (1) \\ s_i^{(t+1)} &= s_{i+1}^{(t)}, \quad i = 0, 1, 2, \ldots, 30, \end{aligned}$$

where α is a root of the primitive polynomial $\pi(x) = x^8 + x^7 + x^5 + x^3 + 1$ in F_{2^8}.

The nonlinear function F (the dotted rectangle in Figure 1) is a compressing function from 32 bits to 8 bits, which contains a 32-bit memory unit R. Denote by $R^{(t)}$ and $R^{(t+1)}$ the values of the memory unit R at times t and $t+1$, respectively. Let w be the output of F. The output of the nonlinear function is obtained as $w^{(t)} = T(R^{(t)})$, where $T(\cdot)$ is a truncation function which truncates the leftmost 8 bits from $R^{(t)}$ as output. Then, the state of the memory unit R is updated by

$$R^{(t+1)} = \theta\left(\gamma\left(X \oplus R^{(t)}\right)\right), \qquad (2)$$

where $X = s_{31}^{(t)} \parallel s_{26}^{(t)} \parallel s_{20}^{(t)} \parallel s_7^{(t)}$. γ is obtained by paralleling 4 S-box S_1 of size 8×8; that is,

$$\gamma\left(x_1 \parallel x_2 \parallel x_3 \parallel x_4\right) = S_1\left(x_1\right) \parallel S_1\left(x_2\right) \parallel S_1\left(x_3\right) \parallel S_1\left(x_4\right), \qquad (3)$$

where x_i $(0 \leqslant i \leqslant 3)$ is a byte. θ is a linear transformation on 32-bit strings defined as

$$\begin{aligned} \theta(x) &= x \oplus (x \lll 2) \oplus (x \lll 10) \\ &\quad \oplus (x \lll 18) \oplus (x \lll 24), \end{aligned} \qquad (4)$$

where $\lll$ denotes the left cyclic shift on 32-bit strings. As for the BOMM structure, it utilizes 16-byte memory units, denoted by $y_i, 0 \leqslant i \leqslant 15$. Let $w^{(t)}$ and $v^{(t)}$ be the input and the output of BOMM at time t, respectively. BOMM works as follows:

$$h^{(t)} = w^{(t)} \gg 4, \qquad l^{(t)} = w^{(t)} \bmod 16, \qquad (5)$$

$$v^{(t)} = y_h^{(t)} \oplus w^{(t)}, \qquad (6)$$

$$y_l^{(t+1)} = y_l^{(t)} \oplus S_2\left(w^{(t)}\right), \qquad (7)$$

$$y_h^{(t+1)} = \begin{cases} y_h^{(t)} \oplus S_2\left(y_l^{(t+1)}\right), & \text{if } h \neq l, \\ y_l^{(t+1)} \oplus S_2\left(y_l^{(t+1)}\right), & \text{if } h = l, \end{cases} \qquad (8)$$

$$y_i^{(t+1)} = y_i^{(t)}, \quad \text{for } i = 0, 1, \ldots, 15, \ i \neq h, l, \qquad (9)$$

where the symbol $\gg$ denotes the right shift operator and S_2 is an S-box of size 8×8.

2.2. Initialization and Keystream Generation

2.2.1. Initialization. The initialization process of Loiss consists of two stages.

In the first stage, it initializes LFSR using a 128-bit secret key and a 128-bit initial vector and then sets $R^{(0)} = 0$.

Set

$$IK = IK_0 \parallel IK_1 \parallel \cdots \parallel IK_{15}, \qquad IV = IV_0 \parallel IV_1 \parallel \cdots \parallel IV_{15}, \qquad (10)$$

where both IK_i and IV_i are bytes, $0 \leqslant i \leqslant 15$.

Denote the initial states of LFSR by $(s_0^{(0)}, s_1^{(0)}, \ldots, s_{31}^{(0)})$. Then, for $0 \leqslant i \leqslant 15$,

$$s_i^{(0)} = IK_i, \qquad s_{16+i}^{(0)} = IK_i \oplus IV_i. \qquad (11)$$

After that, Loiss runs 64 times without the keystream generated and the output of BOMM takes part in the feedback calculation of LFSR.

2.2.2. Keystream Generation. After the initialization process, Loiss starts to generate keystream. Loiss generates one byte of keystream when it runs one time. Let $z^{(t)}$ be the output of Loiss at time t $(t \geqslant 0)$. Then,

$$z^{(t)} = s_0^{(t)} \oplus v^{(t)}, \qquad (12)$$

where $s_0^{(t)}$ and $v^{(t)}$ are the value of the register s_0 of LFSR and the output of BOMM, respectively, at time t.

2.3. Scaled-Down Loiss. The scaled-down Loiss is obtained by getting rid of the BOMM from Loiss and keeping other parts same as Loiss. For convenience, the scaled-down Loiss is denoted by SD-Loiss in the paper. SD-Loiss consists of two parts: LFSR and the nonlinear function F; see Figure 2.

3. Improved Guess and Determine Attack on Full Loiss

Guess and Determine attack is a common attack on stream ciphers. Guess and Determine attacks exploit the relationship

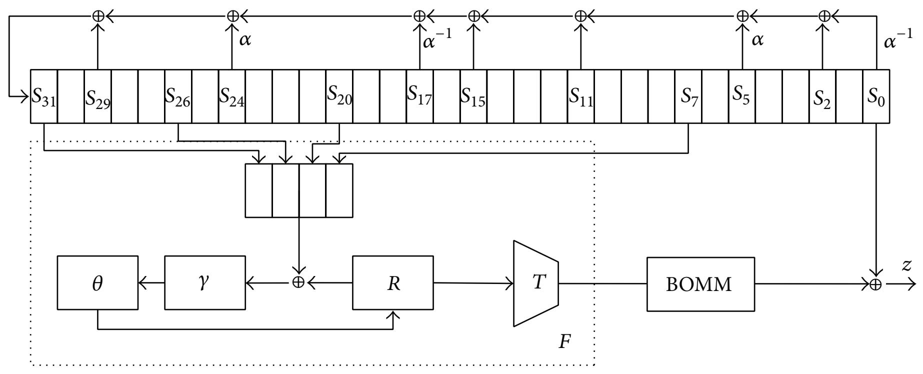

FIGURE 1: The structure of Loiss stream cipher.

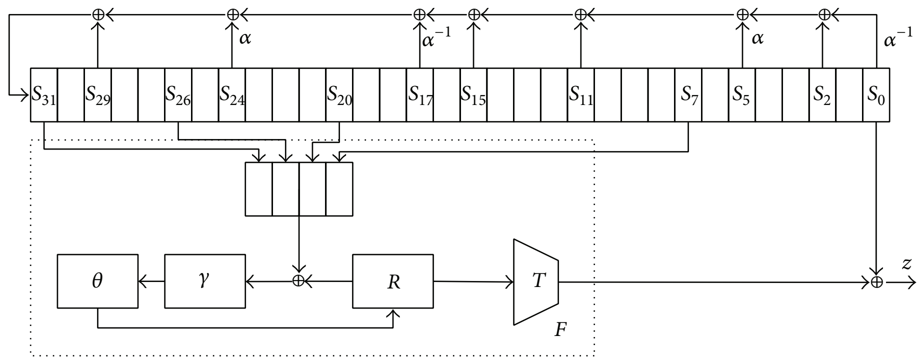

FIGURE 2: The structure of SD-Loiss stream cipher.

between internal and the keystream values. In Guess and Determine attacks, some internal values are guessed, and then other internal values are determined using keystream values. Guess and Determine attacks generally consist of three phases, that is, guessing, determining, and the test phase. The efficiency of Guess and Determine attacks can be discussed in terms of two complexities, namely, a time and a data complexity. Guess and Determine attack is one of the general attacks which have been effective on some stream ciphers, for example, A5/1 [10], SNOW 1.0 [11], Sober-t32 [12], SOSEMANUK [13], Rabbit [14], ZUC [15], and so forth.

In the specification of Loiss stream cipher [7], the designers present a Guess and Determine attack on Loiss which has a time complexity of 2^{247} with a data complexity of 2^{52}. In fact, the time complexity can be reduced at the cost of increased data complexity.

Here, we assume that the attacker has observed a portion of keystream words $\{z_t\}$, $t = 1, 2, \ldots, N$, where N is large enough for the attack to succeed. For convenience, we denote by $A \overset{(*)}{\Rightarrow} B$ the deduction of B from A by equation $(*)$.

For convenience, we denote by $(s_{(t)}, s_{(t+1)}, \ldots, s_{(t+31)})$ the state of LFSR in Loiss at time t ($t \geqslant 0$). Then the state at time $t+1$ is $(s_{(t+1)}, s_{(t+2)}, \ldots, s_{(t+32)})$. The recurrence function of LFSR is redefined as follows:

$$s_{(t+32)} = s_{(t+29)} \oplus \alpha s_{(t+24)} \oplus \alpha^{-1} s_{(t+17)} \oplus s_{(t+15)} \oplus s_{(t+11)} \oplus \alpha s_{(t+5)} \oplus s_{(t+2)} \oplus \alpha^{-1} s_{(t)}. \quad (13)$$

Before the description of our attack, we make an assumption as follows.

Assumption 1. The following conditions occur at nine successive times starting from time t:

(1) $h^{(t)} = l^{(t)}$, where $h^{(t)} = w^{(t)} \gg 4$, $l^{(t)} = w^{(t)} \bmod 16$;
(2) $w^{(t)} = w^{(t+1)} = \cdots = w^{(t+8)}$.

In the attack, the attacker guesses the values of $h^{(t)}$, $s_{(t)}$, $s_{(t+12)}$, $s_{(t+13)}$, $s_{(t+21)}$, $s_{(t+25)}$, $s_{(t+26)}$, $s_{(t+27)}$ and the values of the rightmost three bytes of each $R^{(t+i)}$ $(2 \leqslant i \leqslant 5)$. The whole description of our Guess and Determine attack on Loiss can be divided into three phases as follows.

Phase One. For a given guess, all the bits of the following components can be immediately determined by exploiting the relationships in the cipher:

(i) $z^{(t)}, s_{(t)} \overset{(12)}{\Rightarrow} v^{(t)}$;

(ii) $v^{(t)}, w^{(t)} \overset{(6)}{\Rightarrow} y^{(t)}_{h^{(t)}}$;

(iii) $y^{(t)}_{h^{(t)}} \overset{(7,8)}{\Rightarrow} y^{(t+1)}_{h^{(t)}}$;

(iv) $y^{(t+1)}_{h^{(t+1)}}, w^{(t+1)} \overset{(6)}{\Rightarrow} v^{(t+1)}$;

(v) $z^{(t+1)}, v^{(t+1)} \overset{(12)}{\Rightarrow} s_{(t+1)}$.

The last three steps above can be repeated for $i = 1, \ldots, 7$ to determine more components:

(i) $y_{h^{(t)}}^{(t+i)} \overset{(7,8)}{\Rightarrow} y_{h^{(t)}}^{(t+i+1)}$;

(ii) $y_{h^{(t+i+1)}}^{(t+i+1)}, w^{(t+i+1)} \overset{(6)}{\Rightarrow} v^{(t+i+1)}$;

(iii) $z^{(t+i+1)}, v^{(t+i+1)} \overset{(12)}{\Rightarrow} s_{(t+i+1)}$.

After that, we deduce more components as follows:

(i) $R^{(t+2)}, R^{(t+3)} \overset{(1)}{\Rightarrow} s_{(t+33)}, s_{(t+28)}, s_{(t+22)}, s_{(t+9)}$;

(ii) $R^{(t+3)}, R^{(t+4)} \overset{(1)}{\Rightarrow} s_{(t+34)}, s_{(t+29)}, s_{(t+23)}, s_{(t+10)}$;

(iii) $R^{(t+4)}, R^{(t+5)} \overset{(1)}{\Rightarrow} s_{(t+35)}, s_{(t+30)}, s_{(t+24)}, s_{(t+11)}$.

Phase Two. Then, we can determine more components as follows.

We know that

$$R^{(t+2)} = \theta\left(\gamma\left(s_{(t+32)} \oplus R_3^{(t+1)} \parallel s_{(t+27)} \oplus R_2^{(t+1)} \parallel s_{(t+21)} \oplus R_1^{(t+1)} \parallel s_{(t+8)} \oplus R_0^{(t+1)}\right)\right), \quad (14)$$

where $R^{(t+i)}$ are divided into four bytes as $R^{(t+i)} = R_3^{(t+i)} \parallel R_2^{(t+i)} \parallel R_1^{(t+i)} \parallel R_0^{(t+i)}$.

Thus,

(i) $R^{(t+2)} \overset{(1)}{\Rightarrow} s_{(t+32)} \oplus R_3^{(t+1)}, s_{(t+27)} \oplus R_2^{(t+1)}, s_{(t+21)} \oplus R_1^{(t+1)}, s_{(t+8)} \oplus R_0^{(t+1)}$;

(ii) $R_3^{(t+1)}, s_{(t+27)}, s_{(t+21)}, s_{(t+8)} \overset{(1)}{\Rightarrow} s_{(t+32)}, R_2^{(t+1)}, R_1^{(t+1)}, R_0^{(t+1)}$.

Since $R^{(t+1)}$ has been recovered, then

(i) $R^{(t+1)} \overset{(1)}{\Rightarrow} s_{(t+31)} \oplus R_3^{(t)}, s_{(t+26)} \oplus R_2^{(t)}, s_{(t+20)} \oplus R_1^{(t)}, s_{(t+7)} \oplus R_0^{(t)}$,

(ii) $R_3^{(t)}, s_{(t+31)} \oplus R_3^{(t)} \overset{(1)}{\Rightarrow} s_{(t+31)}$,

(iii) $s_{(t+26)}, s_{(t+26)} \oplus R_2^{(t)} \overset{(1)}{\Rightarrow} R_2^{(t)}$,

(iv) $s_{(t+7)}, s_{(t+7)} \oplus R_0^{(t)} \overset{(1)}{\Rightarrow} R_0^{(t)}$.

We know that

$$\begin{aligned} R^{(t+6)} &= \theta\left(\gamma\left(s_{(t+36)} \oplus R_3^{(t+5)} \parallel s_{(t+31)} \oplus R_2^{(t+5)} \parallel s_{(t+25)} \oplus R_1^{(t+5)} \parallel s_{(t+12)} \oplus R_0^{(t+5)}\right)\right) \\ &= \theta\left(S_1\left(s_{(t+36)} \oplus R_3^{(t+5)}\right) \parallel S_1\left(s_{(t+31)} \oplus R_2^{(t+5)}\right) \parallel S_1\left(s_{(t+25)} \oplus R_1^{(t+5)}\right) \parallel S_1\left(s_{(t+12)} \oplus R_0^{(t+5)}\right)\right). \end{aligned} \quad (15)$$

In this equation, the values of $s_{(t+31)}$, $s_{(t+25)}$, and $s_{(t+12)}$ have been obtained and the values of $R_2^{(t+5)}, R_1^{(t+5)}$, and $R_0^{(t+5)}$ have been guessed, and $R_3^{(t+6)}$ is also known. Thus, we can easily recover the value of $S_1(s_{(t+36)} \oplus R_3^{(t+5)})$ by solving a system of eight bitwise linear equations. After that, we can recover the value of $s_{(t+36)}$, since $R_3^{(t+5)}$ is known. At last, we can recover the value of $R^{(t+6)}$.

Similarly, we can recover the values of $s_{(t+37)}$ and $R^{(t+7)}$.

Phase Three. Then, we can determine the remaining components as follows. In this phase, we have to solve two systems of three byte-wise linear equations.

The first system is described as follows:

$$\begin{aligned} \alpha^{-1}s_{(t+17)} \oplus s_{(t+15)} &= s_{(t+32)} \oplus s_{(t+29)} \oplus \alpha s_{(t+24)} \\ &\quad \oplus s_{(t+11)} \oplus \alpha s_{(t+5)} \oplus s_{(t+2)} \oplus \alpha^{-1}s_{(t)}, \\ \alpha^{-1}s_{(t+19)} \oplus s_{(t+17)} &= s_{(t+34)} \oplus s_{(t+31)} \oplus \alpha s_{(t+26)} \oplus s_{(t+13)} \\ &\quad \oplus \alpha s_{(t+7)} \oplus s_{(t+4)} \oplus \alpha^{-1}s_{(t+2)}, \\ s_{(t+19)} \oplus s_{(t+15)} &= s_{(t+36)} \oplus s_{(t+33)} \oplus \alpha s_{(t+28)} \oplus \alpha^{-1}s_{(t+21)} \\ &\quad \oplus \alpha s_{(t+9)} \oplus s_{(t+6)} \oplus \alpha^{-1}s_{(t+4)}. \end{aligned} \quad (16)$$

In this system, only three variables are unknown, that is, $s_{(t+19)}$, $s_{(t+17)}$, and $s_{(t+15)}$. Obviously, this system can be easily solved. Thus, we can recover the values of $s_{(t+19)}$, $s_{(t+17)}$, and $s_{(t+15)}$ by solving this system.

Then, we deduce $s_{(t+38)}$ as follows:

(i) $s_{(t+35)}, s_{(t+30)}, s_{(t+23)}, s_{(t+21)}, s_{(t+17)}, s_{(t+11)}, s_{(t+8)}, s_{(t+6)} \overset{(13)}{\Rightarrow} s_{(t+38)}$.

We know that

$$\begin{aligned} R^{(t+8)} &= \theta\left(\gamma\left(s_{(t+38)} \oplus R_3^{(t+7)} \parallel s_{(t+33)} \oplus R_2^{(t+7)} \parallel s_{(t+27)} \oplus R_1^{(t+7)} \parallel s_{(t+14)} \oplus R_0^{(t+7)}\right)\right) \\ &= \theta\left(S_1\left(s_{(t+38)} \oplus R_3^{(t+7)}\right) \parallel S_1\left(s_{(t+33)} \oplus R_2^{(t+7)}\right) \parallel S_1\left(s_{(t+27)} \oplus R_1^{(t+7)}\right) \parallel S_1\left(s_{(t+14)} \oplus R_0^{(t+7)}\right)\right). \end{aligned} \quad (17)$$

In this equation, the values of $s_{(t+38)}, s_{(t+33)}$, and $s_{(t+27)}$ have been obtained and the value of $R^{(t+7)}$ has been determined, and $R_3^{(t+8)}$ is also known. Thus, we can easily recover the value of $S_1(s_{(t+14)} \oplus R_0^{(t+7)})$ by solving a system of eight bitwise linear equations. After that, we can recover the value of $s_{(t+14)}$, since $R_0^{(t+7)}$ is known. At last, we can recover the value of $R^{(t+8)}$.

Then, we should solve another system of three linear equations, which is described as follows:

$$\begin{aligned}\alpha^{-1}s_{(t+18)}\oplus s_{(t+16)} &= s_{(t+33)}\oplus s_{(t+30)}\oplus\alpha s_{(t+25)}\oplus s_{(t+12)}\\ &\quad\oplus\alpha s_{(t+6)}\oplus s_{(t+3)}\oplus\alpha^{-1}s_{(t+1)},\\ \alpha^{-1}s_{(t+20)}\oplus s_{(t+18)} &= s_{(t+35)}\oplus s_{(t+32)}\oplus\alpha s_{(t+27)}\oplus s_{(t+14)}\\ &\quad\oplus\alpha s_{(t+8)}\oplus s_{(t+5)}\oplus\alpha^{-1}s_{(t+3)},\\ s_{(t+20)}\oplus s_{(t+16)} &= s_{(t+37)}\oplus s_{(t+34)}\oplus\alpha s_{(t+29)}\oplus\alpha^{-1}s_{(t+23)}\\ &\quad\oplus\alpha s_{(t+10)}\oplus s_{(t+7)}\oplus\alpha^{-1}s_{(t+5)}.\end{aligned}\tag{18}$$

In this system, only three variables are unknown, that is, $s_{(t+20)}, s_{(t+18)}$, and $s_{(t+16)}$. Obviously, this system can be easily solved. Thus, we can recover the values of $s_{(t+20)}, s_{(t+18)}$, and $s_{(t+16)}$ by solving this system.

Finally, we deduce $R^{(t)}$ as follows, since the value of $s_{(t+20)}$ has been recovered:

(i) $R^{(t+1)}, s_{(t+31)}, s_{(t+26)}, s_{(t+20)}, s_{(t+7)} \overset{(1)}{\Rightarrow} R^{(t)}$.

Thus, all internal states of LFSR and F, that is, $s_{(t)}, s_{(t+1)}, \ldots, s_{(t+31)}, R^{(t)}$, have been recovered. After all the internal states of LFSR and F are recovered, the attacker runs Loiss for about another 128 times and then can recover the values of all memory units of BOMM.

Up to now, all internal states of LFSR, F, and BOMM have been recovered. And then the attacker has to check the correctness of those values by producing a keystream using the above recovered values and comparing it with the observed keystream. If the keystreams agree, it shows that the recovered states are correct. If the keystreams do not agree, then we will repeat the above process until the correct internal state is found. Since the probability that the assumption satisfies is 2^{-68} and the attacker has to guess 156-bit internal state in the guessing stage, so the time complexity of our Guess and Determine attack on Loiss is $2^{68}\cdot 2^{156}\cdot 2^{7} = 2^{231}$ with a data complexity of 2^{68}. Compared with the Guess and Determine attack proposed by the designers, the time complexity of our attack on Loiss has been reduced by a factor of 2^{16}.

4. Related Key Chosen *IV* Attack on Scaled-Down Loiss

By exploiting some differential properties of the BOMM structure during the cipher initialization phase, two related key attacks on Loiss were independently proposed in [8, 9]. These results show that the additional design complication, that is, the addition of the BOMM mechanism, weakens the cipher instead of strengthening it. Naturally, an open problem was left for future research, that is, whether the scaled-down Loiss, obtained by getting rid of the BOMM from Loiss and keeping other parts same as Loiss, is resistant against related key attack. In this section, based on the idea of slide (key, IV) pairs, a related key chosen IV attack on scaled-down Loiss is presented.

4.1. Some Properties of SD-Loiss

4.1. Some Properties of SD-Loiss. In this subsection, we will present some properties of SD-Loiss. Let $I^{(t)}$ be the internal state of SD-Loiss at time t. Denote by $I^{(0)}$ and $I^{(64)}$ the internal state of SD-Loiss just after the first stage of initialization and the full initialization, respectively. Let IK and IV be the 128-bit secret key and a 128-bit initial vector into the memory units of LFSR. Let (IK', IV') pair be the related pair of (IK, IV). The relation between them is defined as follows:

$$\begin{aligned}IK' &= IK \lll 16 = IK_2 \parallel IK_3 \parallel \cdots \parallel IK_{15} \parallel IK_0 \parallel IK_1\\ IV' &= IV \lll 16 = IV_2 \parallel IV_3 \parallel \cdots \parallel IV_{15} \parallel IV_0 \parallel IV_1\end{aligned}\tag{19}$$

Let $I'^{(t)}$ be the internal state of SD-Loiss at time t using (IK', IV') pair. Then, we can get the following proposition which discusses the probability that $I^{(2)} = I'^{(0)}$ holds.

Proposition 2. *For the fixed key IK and 2^{48} chosen IVs where $IV_0 = IV_1 = 0$, $(IV_4, IV_5, IV_{11}, IV_{13}, IV_{14}, IV_{15})$ are all 48-bit values and the remaining bytes are fixed to $c \in \{0, 1, \ldots, 255\}$; there exactly exists one IV satisfying $I^{(2)} = I'^{(0)}$.*

Proof. According to the structure of SD-Loiss, we know $I^{(2)} = I'^{(0)}$ can be written as the following system of 33 equations:

$$\begin{gathered}s_i^{(2)} = s_i'^{(0)}, \quad i = 0, 1, 2, \ldots, 31,\\ R^{(2)} = R'^{(0)}.\end{gathered}\tag{20}$$

Since there are 29 equations which always hold in system (20), we can simplify the system (20) as follows:

$$\begin{aligned}&IV_0 = IV_1 = 0,\\ s_{30}^{(2)} = s_{31}^{(1)} &= \left(IK_{13}\oplus IV_{13}\right)\oplus\alpha\left(IK_8\oplus IV_8\right)\\ &\quad\oplus\alpha^{-1}\left(IK_1\oplus IV_1\right)\oplus IK_{15}\oplus IK_{11}\oplus\alpha\left(IK_5\right)\\ &\quad\oplus IK_2\oplus\alpha^{-1}\left(IK_0\right)\oplus w^{(0)} = IK_0\oplus IV_0,\\ s_{31}^{(2)} &= \left(IK_{14}\oplus IV_{14}\right)\oplus\alpha\left(IK_9\oplus IV_9\right)\oplus\alpha^{-1}\left(IK_2\oplus IV_2\right)\\ &\quad\oplus IK_1\oplus IV_1\oplus IK_{12}\oplus\alpha\left(IK_6\right)\oplus IK_3\oplus\alpha^{-1}\left(IK_1\right)\\ &\quad\oplus w^{(1)} = IK_1\oplus IV_1 R^{(2)} = \theta\left(\gamma\left(X^{(1)}\oplus R^{(1)}\right)\right) = 0,\end{aligned}\tag{21}$$

where $w^{(0)} = 0$ and $w^{(1)} = T(R^{(1)})$.

Since θ is a linear transformation on 32-bit strings, we can simplify the equation $R^{(2)} = 0$ as

$$\gamma\left(X^{(1)}\oplus R^{(1)}\right) = 0.\tag{22}$$

That is,

$$\begin{aligned}&\gamma\left[\left(s_{31}^{(1)} \parallel IK_{11}\oplus IV_{11} \parallel IK_5\oplus IV_5 \parallel IK_8\right)\right.\\ &\qquad\left.\oplus\theta\left(\gamma\left(\parallel IK_{15}\oplus IV_{15} \parallel IK_{10}\oplus IV_{10} \parallel IK_4\oplus IV_4 \parallel IK_7\right)\right)\right]\\ &\quad= 0.\end{aligned}\tag{23}$$

Let $Y = Y_0 \parallel Y_1 \parallel Y_2 \parallel Y_3 = \theta(\gamma(IK_{15} \oplus IV_{15} \parallel IK_{10} \oplus IV_{10} \parallel IK_4 \oplus IV_4 \parallel IK_7))$ be a 32-bit string, where Y_i $(0 \leqslant i \leqslant 3)$ is a byte. According to the S-box S_1 in Loiss, we know that $S_1^{-1}(0) = 0x17$.

Then, we know

$$\begin{aligned} s_{31}^{(1)} \oplus Y_0 &= 0x17, \\ IK_{11} \oplus IV_{11} \oplus Y_1 &= 0x17, \\ IK_5 \oplus IV_5 \oplus Y_2 &= 0x17, \\ IK_8 \oplus Y_3 &= 0x17. \end{aligned} \tag{24}$$

Thus, when we try all 32-bit values of $(IV_4, IV_5, IV_{11}, IV_{15})$, there exists exactly one IV satisfying the system (24). Considering the second and third equations of system (21), when we try 2^{48} chosen IVs where $IV_0 = IV_1 = 0$, $(IV_4, IV_5, IV_{11}, IV_{13}, IV_{14}, IV_{15})$ are all 48-bit values and the remaining bytes are fixed to $c \in \{0, 1, \ldots, 255\}$; there exactly exists one IV satisfying $I^{(2)} = I'^{(0)}$. □

If $I^{(2)} = I'^{(0)}$ holds, we easily know $I^{(i)} = I'^{(i-2)}$ always holds for $2 \leqslant i \leqslant 64$. Similarly, if IV satisfies $I^{(i)} = I'^{(i-2)}$, we say that the IV is *valid*. Otherwise, we say that the IV is *invalid*.

Let Z and Z' be keystream sequences generated from (IK, IV) and (IK', IV'), respectively. Let $Z[t, L]$ be the keystream sequences generated from time t to $t + L$; that is, $Z[t, L] = (z^{(t)}, z^{(t+1)}, \ldots, z^{(t+L-1)})$. Similarly, let $Z'[t, L]$ be the keystream sequences generated from time t to $t + L$; that is, $Z'[t, L] = (z'^{(t)}, z'^{(t+1)}, \ldots, z'^{(t+L-1)})$. Then, we know that $I^{(65)}$ and $I^{(66)}$ are updated by the keystream generation mode, while $I'^{(63)}$ and $I'^{(64)}$ are updated by the initialization mode. Thus, both $I^{(65)} = I'^{(63)}$ and $I^{(66)} = I'^{(64)}$ hold, if and only if both $w'^{(63)} = 0$ and $w'^{(64)} = 0$ hold. If $w'^{(63)} = w'^{(64)} = 0$, $I^{(i)} = I'^{(i-2)}$ always holds for $i \geqslant 65$, and then $Z[2, L] = Z'[0, L]$ always holds, for $L \geqslant 1$ (we call it IV-Test).

Theoretically, the probability that a valid IV passes the IV-Test is equal to $\Pr(w'^{(63)} = w'^{(64)} = 0) = 2^{-16}$, and then there exists one IV passing IV-Test among 2^{16} valid IVs. Thus, we can find at least one IV passing IV-Test with high probability among 2^{64} chosen IVs where $IV_0 = IV_1 = 0$; $(IV_4, IV_5, IV_{10}, IV_{11}, IV_{12}, IV_{13}, IV_{14}, IV_{15})$ are all 64-bit values and the remaining bytes are fixed to $c \in \{0, 1, \ldots, 255\}$, while for an invalid IV, after the initialization of SD-Loiss, $z^{(i+2)} = z'^{(i)}$ is uniformly distributed for $0 \leqslant i \leqslant L-1$. Thus, an invalid IV passes IV-Test with probability 2^{-8L}. Obviously, when L is large enough, the probability 2^{-8L} is much smaller than 2^{-16}, which means that we can distinguish valid IV and invalid IVs with very high probability.

4.2. Related Key Chosen IV Attack on Scaled-Down Loiss. Our attack on SD-Loiss can be divided into two phases. In the first phase, we should find a valid IV. In the second phase, we will recover the 128-bit secret key IK.

We choose 2^{64} IVs where $IV_0 = IV_1 = 0$; $(IV_4, IV_5, IV_{10}, IV_{11}, IV_{12}, IV_{13}, IV_{14}, IV_{15})$ are all 64-bit values and the remaining bytes are fixed to $c \in \{0, 1, \ldots, 255\}$. Then, we can find a valid IV among these 2^{64} chosen IVs by the Finding Valid IV algorithm for SD-Loiss which is described as follows.

Finding Valid IV Algorithm for SD-Loiss

```
(1) For each IV in these $2^{64}$ chosen IVs, repeat the following:
    (a) generate $L + 2$ bytes of keystream $Z[2, L]$ by using $(IK, IV)$;
    (b) generate $L$ bytes of keystream $Z'[0, L]$ by using $(IK', IV')$;
    (c) check $IV$-Test for $Z[2, L]$ and $Z'[0, L]$; if they pass $IV$-Test, go to Step (2); otherwise, return to Step (1) and choose another $IV$.
(2) Return Valid $IV$ and store this $IV$.
```

This algorithm requires 2^{64} chosen IVs and runs the encryption process of SD-Loiss $2^{64} + 2^{64} = 2^{65}$ times. Thus, the time complexity of this algorithm is 2^{65}. Then, for each single key and IV, this algorithm only requires $L + 2$ bytes of keystream at most. To make the probability that an invalid IV passes IV-Test negligible, we choose L to be 15, and then 2^{-8L} is equal to 2^{-120} which is small enough to distinguish valid IVs and invalid IVs with very high probability. Hence, our attack requires 17 bytes of keystream at most for each single key and IV.

In the second phase, using the Finding Valid IV algorithm for SD-Loiss above, we can recover the 128-bit secret key IK using a simple Guess and Determine attack. We guess the values of IK_1, IK_2, IK_3, IK_4, IK_6, IK_7, IK_9, IK_{10}, IK_{12}, and IK_{15} (a total of 80 bits). Then, we can determine the remaining 48 bits of the secret key IK by system (21) easily.

Recall the two phases of our attack on SD-Loiss. The time complexity of our attack on SD-Loiss is $2^{65} + 2^{80} \approx 2^{80}$, requiring 2^{64} chosen IVs. The result shows that slide (key, IV) pairs can be also used for related key attack.

5. Conclusions

In this paper, an improved Guess and Determine attack on Loiss is proposed, which reduces the time complexity of the attack proposed by the designers by a factor of 2^{16}. Furthermore, by exploiting the weakness of a scaled-down version of Loiss during its initialization phase, a related key chosen IV attack on the scaled-down Loiss is given. The attack recovers the 128-bit secret key of the scaled-down Loiss with time complexity of 2^{80}, requiring one related key and 2^{64} chosen IVs. We hope our results can be helpful in evaluating the security of the Loiss stream cipher against Guess and Determine attack and related key attack, and we look forward to further works evaluating it against other kinds of cryptanalytic attacks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Natural Science Foundation of China (Grant nos. 61202491, 61272041, and 61272488), the Foundation of Science and Technology on Information Assurance Laboratory (Grant no. KJ-13-007), and the Science and Technology on Communication Security Laboratory Foundation of China under Grant no. 9140C110303140C11003.

References

[1] ETSI/SAGE Specification, "Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3," Document 2: ZUC Specification, Version: 1.6, 2011.

[2] ETSI/SAGE, "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2," Document 2: SNOW 3G Specification, version 1.1, 2006.

[3] C. Berbain, O. Billet, A. Canteaut et al., "Sosemanuk, a fast software-oriented stream cipher," in *New Stream Cipher Designs*, vol. 4986 of *Lecture Notes in Computer Science*, pp. 98–118, Springer, Heidelberg, Germany, 2008.

[4] H. J. Wu, "The stream cipher HC-128," in *New Stream Cipher Designs*, vol. 4986 of *Lecture Notes in Computer Science*, pp. 39–47, Springer, Heidelberg, Germany, 2008.

[5] M. Boesgaard, M. Vesterager, and E. Zenner, "The rabbit stream cipher," in *New Stream Cipher Designs*, vol. 4986 of *Lecture Notes in Computer Science*, pp. 69–83, Springer, Heidelberg, Germany, 2008.

[6] D. J. Bernstein, "The salsa 20 family of stream ciphers," in *New Stream Cipher Designs*, vol. 4986 of *Lecture Notes in Computer Science*, pp. 84–97, Springer, Heidelberg, Germany, 2008.

[7] D. Feng, X. Feng, W. Zhang, X. Fan, and C. Wu, "Loiss: a byte-oriented stream cipher," in *Coding and Cryptology*, vol. 6639 of *Lecture Notes in Computer Science*, pp. 109–125, Springer, Heidelberg, Germany, 2011.

[8] L. Ding and J. Guan, "Cryptanalysis of Loiss stream cipher," *The Computer Journal*, vol. 55, no. 10, pp. 1192–1201, 2012.

[9] A. Biryukov, A. Kircanski, and A. M. Youssef, "Cryptanalysis of the Loiss stream cipher," in *Selected Areas in Cryptography 2012*, vol. 7707 of *Lecture Notes in Computer Science*, pp. 119–134, Springer, Heidelberg, Germany, 2012.

[10] J. Golić, "Cryptanalysis of alleged A5 stream cipher," in *Advances in Cryptology—EUROCRYPT '97*, vol. 1233 of *Lecture Notes in Computer Science*, pp. 239–255, Springer, Heidelberg, Germany, 1997.

[11] P. Hawkes and G. G. Rose, "Guess-and-determine attacks on SNOW," in *Selected Areas in Cryptography*, vol. 2595 of *Lecture Notes in Computer Science*, pp. 37–46, Springer, Heidelberg, Germany, 2002.

[12] S. Babbage, C. de Cannière, J. Lano, B. Preneel, and J. Vandewalle, "Cryptanalysis of SOBER-t32," in *Fast Software Encryption*, vol. 2887 of *Lecture Notes in Computer Science*, pp. 111–128, Springer, Heidelberg, Germany.

[13] X. Feng, J. Liu, Z. Zhou, C. Wu, and D. Feng, "A byte-based guess and determine attack on SOSEMANUK," in *Advances in Cryptology—ASIACRYPT 2010*, vol. 6477 of *Lecture Notes in Computer Science*, pp. 146–157, Springer, Heidelberg, Germany, 2010.

[14] X. Feng, Z. Shi, C. Wu, and D. Feng, "On guess and determine analysis of Rabbit," *International Journal of Foundations of Computer Science*, vol. 22, no. 6, pp. 1283–1296, 2011.

[15] L. Ding, S. K. Liu, Z. Y. Zhang, and J. Guan, "Guess and determine attack on ZUC based on solving nonlinear equations," in *Proceedings of the 1st International Workshop on ZUC Algorithm*, 2010, report 2010/007.