Resynchronization Attacks on WG and LEX ⋆

Hongjun Wu and Bart Preneel

Katholieke Universiteit Leuven, ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{wu.hongjun,bart.preneel}@esat.kuleuven.be

Abstract. WG and LEX are two stream ciphers submitted to eStream – the ECRYPT stream cipher project. In this paper, we point out security flaws in the resynchronization of these two ciphers. The resynchronization of WG is vulnerable to a differential attack. For WG with 80-bit key and 80-bit IV, 48 bits of the secret key can be recovered with about $2^{31.3}$ chosen IVs . For each chosen IV, only the first four keystream bits are needed in the attack. The resynchronization of LEX is vulnerable to a slide attack. If a key is used with about $2^{60.8}$ random IVs, and 20,000 keystream bytes are generated from each IV, then the key of the strong version of LEX could be recovered easily with a slide attack. The resynchronization attack on WG and LEX shows that block cipher related attacks are powerful in analyzing non-linear resynchronization mechanisms.

Keywords: cryptanalysis, stream cipher, resynchronization attack, differential attack, slide attack, WG, LEX

1 Introduction

For the research on stream ciphers, resynchronization atacks have not been studied as thoroughly as the keystream generation algorithm itself. Ten years ago, Daemen, Govaerts and Vandewalle analyzed the weakness of linear resynchronization mechanism with known output Boolean function [5]. Later Golić and Morgari studied linear resynchronization mechanisms with unknown output function [7]. However almost all the stream ciphers proposed recently use non-linear resynchronization mechanisms, so the previous attacks on linear resynchronization mechanisms could no longer be applied. Recently Armknecht, Lano and Preneel applied algebraic attacks and linear cryptanalysis to the resynchronization mechanism and obtained lower bounds for the nonlinearity required from a secure resynchronization mechanism [1]. In this paper, we apply the differential attack and slide attack to stream ciphers with non-linear resynchronization. We show that the cryptanalysis techniques used to attack block ciphers are also useful in the analysis of non-linear resynchronization mechanisms.

⋆ This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

WG [11] and LEX [4] are two stream ciphers submitted to eStream, the ECRYPT stream cipher project [6]. The keystream generation algorithms of WG and LEX are quite strong. The keystream generation of WG is based on the WG transformation which has excellent cryptographic properties [8]. The keystream generation of LEX is based on the Advanced Encryption Standard [10]. However, the resynchronization mechanism of WG and LEX are insecure. The resynchronization mechanism of WG is vulnerable to the differential attack [2] and that of LEX is vulnerable to a slide attack [3]. Breaking WG requires $2^{31.3}$ chosen IVs, and breaking the strong version of LEX requires about $2^{60.8}$ random IVs.

This paper is organized as follows. WG and LEX are introduced in Sect. 2. The differential attack on WG is presented in Sect. 3, and the slide attack on LEX is described in Sect. 4. Section 5 concludes this paper.

2 Description of WG and LEX

WG and LEX are described in Sec. 2.1 and 2.2, respectively.

2.1 The stream cipher WG

WG is a hardware oriented stream cipher with key length up to 128 bits; it supports IV sizes from 32 bits to 128 bits. The main feature of the WG stream cipher is the use of the WG transformation to generate keystream from an LFSR.

Keystream Generation

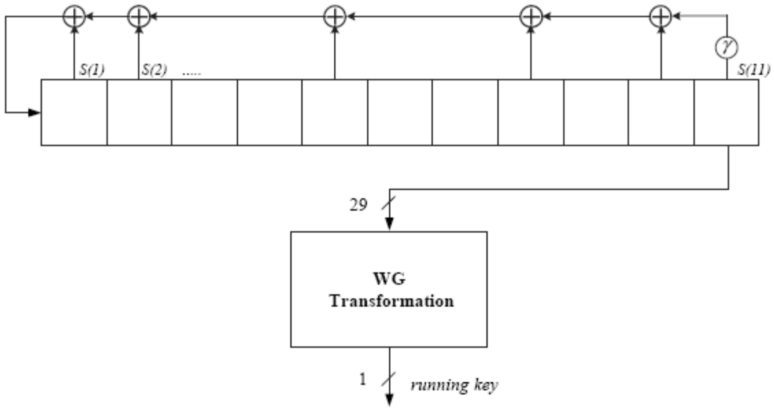

Fig. 1. Keystream generation diagram of WG [11]

The keystream generation diagram of WG is given in Fig. 1. WG has a regularly clocked LFSR which is defined by the feedback polynomial

$$p(x) = x^{11} + x^{10} + x^9 + x^6 + x^3 + x + \gamma \tag{1}$$

over $GF(2^{29})$, where $\gamma = \beta^{464730077}$ and β is the primitive root of $g(x)$

$$\begin{aligned} g(x) = x^{29} + x^{28} + x^{24} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + \\ x^{14} + x^{12} + x^{11} + x^{10} + x^7 + x^6 + x^4 + x + 1\,. \end{aligned} \tag{2}$$

Then the non-linear WG transformation, $GF(2^{29}) \rightarrow GF(2)$, is applied to generate the keystream from the LFSR.

Resynchronization (Key/IV setup)

The key/IV setup of WG is given in Fig. 2. After the key and IV have been loaded into the LFSR, it is clocked 22 steps. During each of these 22 steps, 29 bits from the middle of the WG transformation are XORed to the feedback of LFSR, as shown in Fig. 2.

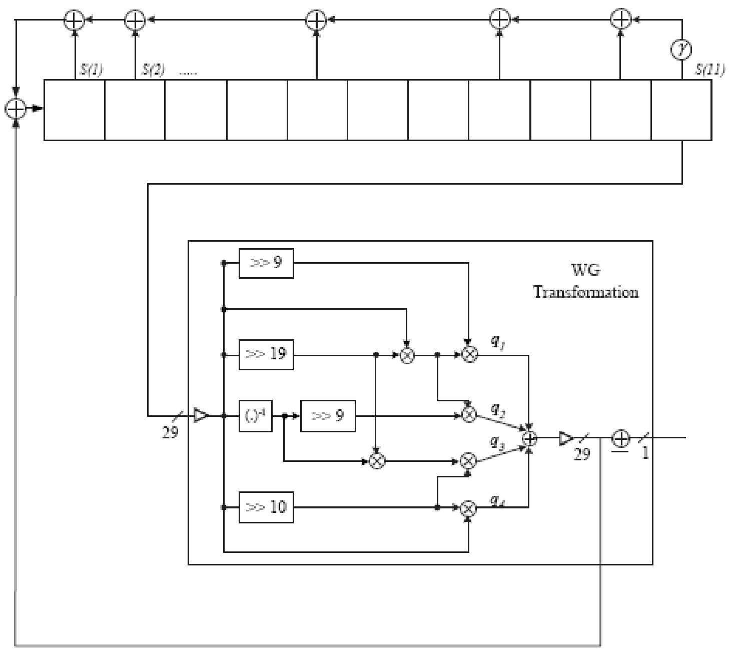

Fig. 2. Key/IV setup of WG [11]

One step of the key/IV setup can be expressed as follows:

$$T = S(1)\oplus S(2)\oplus S(5)\oplus S(8)\oplus S(10)\oplus(\gamma\times S(11))\oplus WG'(S(11))\,,$$
$$S(i) = S(i-1) \text{ for } i = 11\cdots 2;\ S(1) = T\ \ ,$$

where $WG'(S(11))$ denotes the 29 bits extracted from the WG transformation, as shown in Fig. 2.

The WG cipher supports several key and IV sizes: the key size can be 80 bits, 96 bits, 112 bits and 128 bits and the IV sizes can be 32 bits, 64 bits, 80 bits, 96 bits, 112 bits, and 128 bits. Slightly different resynchronization mechanisms are used for the different IV sizes. The details are given in Sect. 3.

2.2 The stream cipher LEX

LEX is based on the block cipher AES. The keystream bits are generated by extracting 32 bits from each round of AES in the 128-bit Output Feedback (OFB) mode [9]. LEX is about 2.5 times faster than AES. Fig. 3 shows how the AES is initialized and chained. First a standard AES key-schedule for a secret 128-bit key K is performed. Then a given 128-bit IV is encrypted by a single AES invocation: $S = AES_K(IV)$. The S and the subkeys are the output of the initialization process.

S is encrypted by K in the 128-bit OFB mode (for the more secure variant, K is changed every 500 AES encryptions). At each round, 32 bits of the middle value of AES are extracted to form the keystream. The bytes $b_{0,0}$, $b_{0,2}$, $b_{2,0}$, $b_{2,2}$ at every odd round and the bytes $b_{0,1}$, $b_{0,3}$, $b_{2,1}$, $b_{2,3}$ at every even round are selected, as shown in Fig. 4.

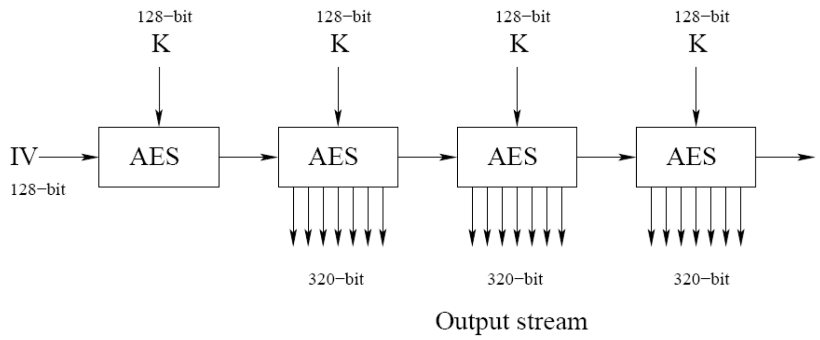

Fig. 3. Initialization and stream generation [4]

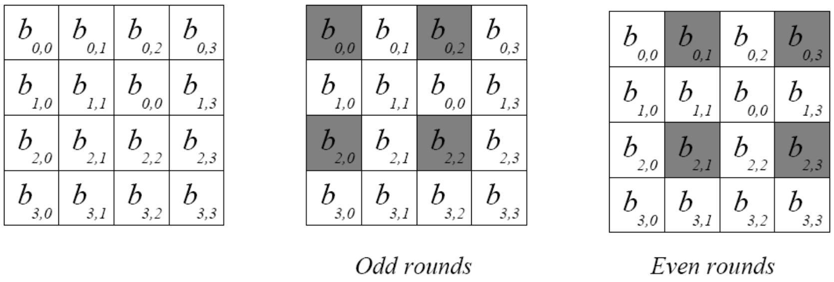

Fig. 4. The positions of the output extracted in the even and odd rounds [4]

3 The Differential Attacks on the Resynchronization of WG

The resynchronization of WG can be broken with a chosen IV attack based on differential cryptanalysis. (We remind the readers that the details of the differential attack given in this paper are slightly different from the standard differential attack on a block cipher, such as the generation of the differential pairs and the filtering of the wrong pairs.) WG with a 32-bit IV size is not vulnerable to the attack given in this section (since no special differential can be introduced into this short IV). In Sec. 3.1 the attack is applied to break an WG with an 80-bit key and an 80-bit IV. The attacks on WG with IV sizes larger than 80 bits are given in Sect. 3.2. The attack on a WG with a 64-bit IV size is given in Sec. 3.3.

3.1 An attack on WG with an 80-bit key and an 80-bit IV

We investigate the security of the key/IV setup of WG with an 80-bit key and an 80-bit IV. For this version of WG, denote the key as $K = k_1, k_2, k_3, \cdots, k_{80}$ and the IV as $IV = IV_1, IV_2, IV_3, \cdots, IV_{80}$. They are loaded into the LFSR as follows:

$$
\begin{array}{ll}
S_{1,\ldots,16}(1) = k_{1,\ldots,16} & S_{17,\ldots,24}(1) = IV_{1,\ldots,8} \\
S_{1,\ldots,8}(2) = k_{17,\ldots,24} & S_{9,\ldots,24}(2) = IV_{9,\ldots,24} \\
S_{1,\ldots,16}(3) = k_{25,\ldots,40} & S_{17,\ldots,24}(3) = IV_{25,\ldots,32} \\
S_{1,\ldots,8}(4) = k_{41,\ldots,48} & S_{9,\ldots,24}(4) = IV_{33,\ldots,48} \\
S_{1,\ldots,16}(5) = k_{49,\ldots,64} & S_{17,\ldots,24}(5) = IV_{49,\ldots,56} \\
S_{1,\ldots,8}(6) = k_{65,\ldots,72} & S_{9,\ldots,24}(6) = IV_{57,\ldots,72} \\
S_{1,\ldots,8}(7) = k_{73,\ldots,80} & S_{17,\ldots,24}(7) = IV_{73,\ldots,80}
\end{array}
$$

All the remaining bits of the LFSR are set to zero. Then the LFSR is clocked 22 steps with the middle value from the WG transformation being used in the feedback.

The chosen IV attack on WG goes as follows. For each secret key K, we choose two IVs, IV' and IV'', so that IV' and IV'' are identical in 8 bytes, but differ in two bytes: $IV'_{17,\ldots,24} \neq IV''_{17,\ldots,24}$ and $IV'_{49,\ldots,56} \neq IV''_{49,\ldots,56}$. The differences satisfy $IV'_{17,\ldots,24} \oplus IV''_{17,\ldots,24} = IV'_{49,\ldots,56} \oplus IV''_{49,\ldots,56}$.

Denote the value of $S(i)$ $(1 \le i \le 11)$ at the end of the j-th step by $S^j(i)$, and denote loading the key/IV as the 0th step. After loading the key and the chosen IV into LFSR, we know that the differences in $S(2)$ and $S(5)$ are the same, i.e., $S'^0(2) \oplus S''^0(2) = S'^0(5) \oplus S''^0(5)$. We denote this difference as $\triangle_1$, i.e., $\triangle_1 = S'^0(2) \oplus S''^0(2) = S'^0(5) \oplus S''^0(5)$.

Table 1. The differential propagation in the key/IV setup of WG

	S(1)	S(2)	S(3)	S(4)	S(5)	S(6)	S(7)	S(8)	S(9)	S(10)	S(11)
step 0	0	$\triangle_1$	0	0	$\triangle_1$	0	0	0	0	0	0
step 1	0	0	$\triangle_1$	0	0	$\triangle_1$	0	0	0	0	0
step 2	0	0	0	$\triangle_1$	0	0	$\triangle_1$	0	0	0	0
step 3	0	0	0	0	$\triangle_1$	0	0	$\triangle_1$	0	0	0
step 4	0	0	0	0	0	$\triangle_1$	0	0	$\triangle_1$	0	0
step 5	0	0	0	0	0	0	$\triangle_1$	0	0	$\triangle_1$	0
step 6	$\triangle_1$	0	0	0	0	0	0	$\triangle_1$	0	0	$\triangle_1$
step 7	$\triangle_2$	$\triangle_1$	0	0	0	0	0	0	$\triangle_1$	0	0
step 8	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0	0	0	0	0	$\triangle_1$	0
step 9	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0	0	0	0	0	$\triangle_1$
step 10	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0	0	0	0	0
step 11	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0	0	0	0
step 12	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0	0	0
step 13	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0	0
step 14	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0	0
step 15	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$	0
step 16	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$	$\triangle_1$
step 17	$\triangle_1\oplus\triangle_4$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$	$\triangle_2$
step 18	$\triangle_3\oplus\triangle_4\oplus\triangle_5$	$\triangle_1\oplus\triangle_4$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0	$\triangle_1\oplus\triangle_2$
step 19	$\triangle_1\oplus\triangle_2\oplus\triangle_3\oplus\triangle_5\oplus\triangle_6$	$\triangle_3\oplus\triangle_4\oplus\triangle_5$	$\triangle_1\oplus\triangle_4$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	0
step 20	$\triangle_4\oplus\triangle_6$	$\triangle_1\oplus\triangle_2\oplus\triangle_3\oplus\triangle_5\oplus\triangle_6$	$\triangle_3\oplus\triangle_4\oplus\triangle_5$	$\triangle_1\oplus\triangle_4$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$
step 21	$\triangle_4\oplus\triangle_5\oplus\triangle_7$	$\triangle_4\oplus\triangle_6$	$\triangle_1\oplus\triangle_2\oplus\triangle_3\oplus\triangle_5\oplus\triangle_6$	$\triangle_3\oplus\triangle_4\oplus\triangle_5$	$\triangle_1\oplus\triangle_4$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$	$\triangle_2\oplus\triangle_3$
step 22	$\triangle_2\oplus\triangle_3\oplus\triangle_4\oplus\triangle_5\oplus\triangle_6\oplus\triangle_7\oplus\triangle_8$	$\triangle_4\oplus\triangle_5\oplus\triangle_7$	$\triangle_4\oplus\triangle_6$	$\triangle_1\oplus\triangle_2\oplus\triangle_3\oplus\triangle_5\oplus\triangle_6$	$\triangle_3\oplus\triangle_4\oplus\triangle_5$	$\triangle_1\oplus\triangle_4$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2\oplus\triangle_3$	$\triangle_3$	$\triangle_2\oplus\triangle_3$	$\triangle_1\oplus\triangle_2$

We now examine the differential propagation during the 22 steps in the key/IV setup. The complete differential propagation is shown in Table 1, where the differences at the i-th step indicate the differences at the end of the i-th step. The difference $\triangle_2 = (\gamma \times S'^6(11) \oplus WG'(S'^6(11)) \oplus (\gamma \times S''^6(11) \oplus WG'(S''^6(11)) = (\gamma \times S'^0(5) \oplus WG'(S'^0(5)) \oplus (\gamma \times S''^0(5) \oplus WG'(S''^0(5))$. Similarly, we obtain that $\triangle_3 = (\gamma \times S'^0(2) \oplus WG'(S'^0(2)) \oplus (\gamma \times S''^0(2) \oplus WG'(S''^0(2))$.

From Table 1, we notice that at the end of the 22th step, the difference at $S^{22}(10)$ is $\triangle_2 \oplus \triangle_3$. From the above description of $\triangle_2$ and $\triangle_3$, we know that

$$\begin{aligned}\triangle_2 \oplus \triangle_3 = &((\gamma \times S'^0(5) \oplus WG'(S'^0(5)) \oplus (\gamma \times S''^0(5) \oplus WG'(S''^0(5))) \oplus \\ &((\gamma \times S'^0(2) \oplus WG'(S'^0(2)) \oplus (\gamma \times S''^0(2) \oplus WG'(S''^0(2)))\,. \quad (3)\end{aligned}$$

This shows that the value of $\triangle_2 \oplus \triangle_3$ is determined by $k_{17,\ldots,24}$, $k_{49,\ldots,64}$, $IV'_{9,\ldots,24}$, $IV'_{49,\ldots,56}$, $IV''_{9,\ldots,24}$, $IV''_{49,\ldots,56}$.

From the keystream generation of WG, we know that the first keystream bit is generated from $S^{22}(10)$ (after the key/IV setup, the LFSR is clocked, and $S^{23}(11)$ is used to generate the first keystream bit). If $\triangle_2 \oplus \triangle_3 = 0$, then the first keystream bits for IV' and IV'' should be the same. This property is applied in the attack to determine whether the value of $\triangle_2 \oplus \triangle_3$ is 0.

Assume that the value of $\triangle_2 \oplus \triangle_3$ is randomly distributed, then $\triangle_2 \oplus \triangle_3 = 0$ with probability 2^{-29}. We thus need to generate about 2^{29} pairs $(\triangle_2, \triangle_3)$ in order to obtain a pair satisfying $\triangle_2 \oplus \triangle_3 = 0$. Note that the key is fixed and that $S'^0(2) \oplus S''^0(2) = S'^0(5) \oplus S''^0(5)$ must be satisfied. Three bytes of IV ($IV'_{9,\ldots,24}$, $IV'_{49,\ldots,56}$) and one-byte difference ($\triangle_1$) can be freely chosen to generate different $(\triangle_2, \triangle_3)$, so there are about $2^{24} \times 255/2 \approx 2^{31}$ available pairs of $(\triangle_2, \triangle_3)$. Hence there is no problem to generate 2^{29} pairs of $(\triangle_2, \triangle_3)$.

Then we proceed to determine which pair $(\triangle_2, \triangle_3)$ satisfies $\triangle_2 \oplus \triangle_3 = 0$. For each pair $(\triangle_2, \triangle_3)$, we modify the values of $IV'_{1,\ldots,8}$ and $IV''_{1,\ldots,8}$, but we ensure that $IV'_{1,\ldots,8} = IV''_{1,\ldots,8}$. This modification does not affect the value of $\triangle_2 \oplus \triangle_3$, but it affects the value of $S^{22}(10)$. We generate keystream and examine the first keystream bits. If the values of the first keystream bits are the same, then the chance that $\triangle_2 \oplus \triangle_3 = 0$ is improved. In that case, we modify $IV'_{1,\ldots,8}$ and $IV''_{1,\ldots,8}$ again and observe the first keystream bits. This process ends when the first keystream bits are not the same or this process is repeated for 40 times. If one $(\triangle_2, \triangle_3)$ passes the test for 40 times, then we know that $\triangle_2 \oplus \triangle_3 = 0$ with probability extremely close to 1. (Each wrong pair could pass this filtering process with probability 2^{-40}. One pair of 2^{29} wrong pairs could pass this process with probability 2^{-11}.) Thus with about $2 \times 2^{29} \times \sum_{i=1}^{40} \frac{i}{2^i} = 2^{31}$ chosen IVs, we can find a pair $(\triangle_2, \triangle_3)$ satisfying $\triangle_2 \oplus \triangle_3 = 0$. Subsequently according to Eqn. (3) and $\triangle_2 \oplus \triangle_3 = 0$, we recover 24 bits of the secret key, $k_{17,\ldots,24}$ and $k_{49,\ldots,64}$.

The above attack can be improved if we consider the differences at $S^{22}(7)$ and $S^{22}(8)$. The differences there are both $\triangle_1 \oplus \triangle_2 \oplus \triangle_3$. If the value of $\triangle_1 \oplus \triangle_2 \oplus \triangle_3$ is 0, then the third and fourth bits of the two keystreams would be the same. If we only observe the third and fourth keystream bits, then $k_{17,\ldots,24}$ and $k_{49,\ldots,64}$ can be recovered with $2 \times 2^{29} \times \sum_{i=1}^{20} (\frac{1}{2^{i-1}} - \frac{1}{2^i}) \times i = 2^{30.4}$ chosen IVs.

In the attack, we observe the first, third and fourth keystream bits, then recovering $k_{17,\ldots,24}$ and $k_{49,\ldots,64}$ requires about $2 \times 2^{28} \times 2^{1.13} = 2^{30.1}$ chosen IVs (the value $2^{1.13}$ is obtained through numerical computation).

By setting the difference at $S^0(3)$ and $S^0(6)$ and observing the second and third bits of the keystream, we can recover another 24 bits of the secret key, $k_{25,\ldots,40}$ and $k_{65,\ldots,72}$. We need $2^{30.4}$ chosen IVs.

So with about $2^{30.1} + 2^{30.4} = 2^{31.3}$ chosen IVs, we can recover 48 bits of the 80-bit secret key. It shows that the key/IV setup of the WG stream cipher is insecure.

3.2 The attacks on WG with key and IV sizes larger than 80 bits

The WG ciphers with the key and IV sizes larger than 80 bits are all vulnerable to the chosen IV attacks. The attacks are very similar to the above attack. We omit the details of the attacks here. The results are given below:

1. For WG with 96-bit key and 96-bit IV, 48 bits of the key can be recovered with complexity about the same as the above attack.
2. For WG with IV sizes larger than 96 bits, 72 bits of the key can be recovered with complexity about 1.5 times that of the above attack.

3.3 The attacks on WG with 64-bit IV size

We use WG with an 80-bit key and a 64-bit IV as an example to illustrate the attack. For WG cipher with an 80-bit key and a 64-bit IV, the key and IV are loaded into the LFSR as follows:

$$
\begin{array}{ll}
S_{1,\ldots,16}(1) = k_{1,\ldots,16} & S_{1,\ldots,16}(2) = k_{17,\ldots,32} \\
S_{1,\ldots,16}(3) = k_{33,\ldots,48} & S_{1,\ldots,16}(4) = k_{49,\ldots,64} \\
S_{1,\ldots,16}(5) = k_{65,\ldots,80} & S_{1,\ldots,16}(9) = k_{1,\ldots,16} \\
S_{1,\ldots,16}(10) = k_{17,\ldots,32} \oplus 1 & S_{1,\ldots,16}(11) = k_{33,\ldots,48}
\end{array}
$$

$$
\begin{array}{ll}
S_{17,\ldots,24}(1) = IV_{1,\ldots,8} & S_{17,\ldots,24}(2) = IV_{9,\ldots,16} \\
S_{17,\ldots,24}(3) = IV_{17,\ldots,24} & S_{17,\ldots,24}(4) = IV_{25,\ldots,32} \\
S_{17,\ldots,24}(5) = IV_{33,\ldots,40} & S_{17,\ldots,24}(6) = IV_{41,\ldots,48} \\
S_{17,\ldots,24}(7) = IV_{49,\ldots,56} & S_{17,\ldots,24}(8) = IV_{57,\ldots,64}
\end{array}
$$

In the attack, we introduce differences at $S(2)$ and $S(5)$, but we can only generate about 2^{23} pairs of $(\triangle_2, \triangle_3)$ since we can only modify $IV_{9,\ldots,16}$ and $IV_{33,\ldots,40}$. Thus we can obtain a pair $(\triangle_2, \triangle_3)$ satisfying $\triangle_2 \oplus \triangle_3 = 0$ or $\triangle_1 \oplus \triangle_2 \oplus \triangle_3 = 0$ with probability 2^{-5}. Once we know $\triangle_2 \oplus \triangle_3 = 0$ or $\triangle_1 \oplus \triangle_2 \oplus \triangle_3 = 0$, we can recover 29 bits of information on $k_{17,\ldots,32}$ and $k_{65,\ldots,80}$. It shows that 29 bits of information of the secret key can be recovered with probability 2^{-5}. This attack requires about $2^{25.1}$ chosen IVs.

The attack on WG with 96-bit key and 64-bit IV is similar to the above attack. We introduce differences at $S(2)$ and $S(5)$ or at $S(3)$ and $S(6)$. In the attack 29 bits of information on $k_{17,\ldots,32}$ and $k_{65,\ldots,80}$ can be recovered with probability 2^{-5}, and another 29 bits of information on $k_{33,\ldots,48}$ and $k_{81,\ldots,96}$ can be recovered with probability 2^{-5}.

The attack on WG with 112-bit key and 64-bit IV is also similar. The result is that 29 bits of information on $k_{17,\ldots,32}$ and $k_{65,\ldots,80}$ can be recovered with probability 2^{-5}, 29 bits of information on $k_{33,\ldots,48}$ and $k_{81,\ldots,96}$ can be recovered with probability 2^{-5}, and 29 bits of information on $k_{49,\ldots,64}$ and $k_{97,\ldots,112}$ can be recovered with probability 2^{-5}.

The attack on WG with 128-bit key and 64-bit IV is also similar. The result is that 29 bits of information on $k_{17,\ldots,32}$ and $k_{65,\ldots,80}$ can be recovered with probability 2^{-5}, 29 bits of information on $k_{33,\ldots,48}$ and $k_{81,\ldots,96}$ can be recovered with probability 2^{-5}, 29 bits of information on $k_{49,\ldots,64}$ and $k_{97,\ldots,112}$ can be recovered with probability 2^{-5}, and 29 bits of information on $k_{64,\ldots,80}$ and $k_{113,\ldots,128}$ can be recovered with probability 2^{-5}.

4 A Slide Attack on the Resynchronization of LEX

The security of LEX depends heavily on the fact that only a small amount of information is released for each round (including the input and output) of AES. The slide attack intends to retrieve all the information of one AES round input (or output) in LEX.

Denote $S_i = E_K^i(IV)$, where $E^i(m)$ means that m is encrypted i times, $S_0 = IV$; denote the 320 bits extracted from the i-th encryption with k_i for $i \geq 2$. For two IVs, IV' and IV'', if $k'_2 = k''_j$ $(j > 2)$, then we know that $S'_1 = S''_{j-1}$. Immediately, we know that $S''_{j-2} = S'_0 = IV'$. Note that k''_{j-1} is extracted from $E_K(S''_{j-2})$, so k''_{j-1} is extracted from $E_K(IV')$; this means that we know the input to AES, and we know 32 bits from the output of the first round. In the following, we show that it is easy to recover the secret key from this 32 bits of information of the first round output.

Denote the 16-byte output of the r-th round of AES with $m^r_{i,j}$ $(0 \leq i, j \leq 3)$, and denote the 16-byte round key at the end of the r-th round with $w^r_{i,j}$ $(0 \leq i, j \leq 3)$. Now if $m^1_{0,0}$, $m^1_{0,2}$, $m^1_{2,0}$, $m^1_{2,2}$ are known, i.e, four bytes of the first round output are known, then we obtain the following four equations:

$$\begin{aligned}
m^1_{0,0} \oplus w^1_{0,0} &= \text{MixColumn}((m^0_{0,0} \oplus w^0_{0,0}) \parallel (m^0_{1,3} \oplus w^0_{1,3}) \\
&\quad \parallel (m^0_{2,2} \oplus w^0_{2,2}) \parallel (m^0_{3,1} \oplus w^0_{3,1}))\,\&\,0xFF && (4)\\
m^1_{2,0} \oplus w^1_{2,0} &= (\text{MixColumn}((m^0_{0,0} \oplus w^0_{0,0}) \parallel (m^0_{1,3} \oplus w^0_{1,3}) \\
&\quad \parallel (m^0_{2,2} \oplus w^0_{2,2}) \parallel (m^0_{3,1} \oplus w^0_{3,1}))\ >>\ 16)\,\&\,0xFF && (5)\\
m^1_{0,2} \oplus w^1_{0,2} &= \text{MixColumn}((m^0_{0,2} \oplus w^0_{0,2}) \parallel (m^0_{1,1} \oplus w^0_{1,1}) \\
&\quad \parallel (m^0_{2,0} \oplus w^0_{2,0}) \parallel (m^0_{3,3} \oplus w^0_{3,3}))\,\&\,0xFF && (6)\\
m^1_{2,2} \oplus w^1_{2,2} &= (\text{MixColumn}((m^0_{0,2} \oplus w^0_{0,2}) \parallel (m^0_{1,1} \oplus w^0_{1,1}) \\
&\quad \parallel (m^0_{2,0} \oplus w^0_{2,0}) \parallel (m^0_{3,3} \oplus w^0_{3,3})) >> 16)\,\&\,0xFF\,. && (7)
\end{aligned}$$

Each equation leaks one byte of information on the secret key. In the above four equations, 12 bytes of the subkey are involved. To recover all these 12 bytes, we need three inputs to AES and the related 32-bit first round outputs so that we can obtain 12 equations. These 12 equations can be solved with about $\alpha \times 2^{32}$ operations, where α is a small constant. With 96 bits of the key have been recovered, the rest of the 32 bits of AES can be recovered by exhaustive search.

We now compute the number of IVs required to generate three collisions. Suppose that a secret key is used with about $2^{65.3}$ random IVs, and each IV^i is used to generate a 640-bit keystream k_2^i, k_3^i. Since the block size of AES is 128 bits, we know that with high probability there are three collisions $k_2^i = k_3^j$ for different i and j since $\frac{2^{65.3} \times (2^{65.3}-1)}{2} \times 2^{-128} \approx 3$.

The number of IVs could be reduced if more keystream bits are generated from each IV. In [4], it is suggested to change the key every 500 AES encryptions for a strong variant of LEX. Suppose that each IV is applied to generate 500 320-bit outputs, then with $2^{60.8}$ IVs, we could find three collisions $k_2^i = k_x^j$ $(2 < x < 500)$ and recover the key of LEX. For the original version of LEX, the AES key is not changed during the keystream generation. Suppose that each IV is used to generate 2^{50} keystream bytes, then the key could be recovered with about 2^{43} random IVs (here we need to consider that the state update function of LEX is reversible; otherwise, the amount of IV required in the attack could be greatly reduced).

For a secure stream cipher with a 128-bit key and a 128-bit IV, each key would never be recovered faster than exhaustive key search no matter how many IVs are used together with that key. But for LEX each key could be recovered faster than exhaustive search if that key is used together with about 2^{61} random IVs. We thus conclude that LEX is theoretically insecure.

For a stream cipher with 128-bit key and 128-bit IV, if the attacker can choose the IV, then one of 2^{64} keys could be recovered with about 2^{64} pre-computations (based on the birthday paradox). The complexity of such an attack is close to our attack on LEX. However, there are two major differences between these two attacks. One difference is that the attack based on birthday paradox is a chosen IV attack while our attack is a random IV attack. Another difference is that the attack based on birthday paradox results in the recovery of one of n keys, while our attack recovers one particular key. Recovering one of n keys and recovering one particular key are two different types of attacks being used in different scenarios, so it is not meaningful to simply compare their complexities.

5 Conclusion

In this paper, we show that the resynchronization mechanisms of WG and LEX are vulnerable to a differential attack and a slide attack, respectively. It shows that the block cipher cryptanalysis techniques are powerful in analyzing the non-linear resynchronization mechanism of a stream cipher.

The designers of WG recommended to use 44 steps in the initialization to resist a differential attack [12]. It is a small modification to the design to achieve secure key/IV setup. However, it is inefficient. We recommend to change the primitive polynomial tap positions so that the tap distances are coprime, and to generate the first keystream bit from $S(1)$ instead of $S(10)$. Then we expect that WG with 22-step key/IV setup will be able to resist a differential attack.

Acknowledgements

The authors would like to thank the anonymous reviewers of SASC 2006 and FSE 2006 for their helpful comments. Special thanks go to Alex Biryukov for pointing out that the attack on LEX is slide attack and for helpful discussion.

References

1. F. Armknecht, J. Lano, and B. Preneel,"Extending the Resynchronization Attack," *Selected Areas in Cryptography – SAC 2004*, LNCS 3357, H. Handschuh, and A. Hasan (eds.), Springer-Verlag, pp. 19-38, 2004.
2. E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," in *Advances in Cryptology – Crypto'90*, LNCS 537, pp. 2-21, Springer-Verlag, 1991.
3. A. Biryukov, D. Wagner, "Slide Attacks," *Fast Software Encryption – FSE'99*, LNCS 1636, pp. 245-259, Springer-Verlag, 1999.
4. A. Biryukov, "A New 128-bit Key Stream Cipher LEX," *ECRYPT Stream Cipher Project Report 2005/013*. Available at http://www.ecrypt.eu.org/stream/
5. J. Daemen, R. Govaerts, J. Vandewalle, "Resynchronization weakness in synchronous stream ciphers," *Advances in Cryptology - EUROCRYPT'93*, Lecture Notes in Computer Science, vol. 765, pp. 159-167, 1994.
6. ECRYPT Stream Cipher Project, at http://www.ecrypt.eu.org/stream/
7. J. D. Golić, G. Morgari, "On the Resynchronization Attack," *Fast Software Encryption – FSE 2003*, LNCS 2887, pp. 100-110, Springer-Verlag, 2003.
8. G. Gong, A. Youssef. "Cryptographic Properties of the Welch-Gong Transformation Sequence Generators," *IEEE Transactions on Information Theory*, vol. 48, No. 11, pp. 2837-2846, Nov. 2002.
9. National Institute of Standards and Technology, "DES Modes of Operation," Federal Information Processing Standards Publication (FIPS) 81. Available at http://csrc.nist.gov/publications/fips/
10. National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication (FIPS) 197. Available at http://csrc.nist.gov/publications/fips/
11. Y. Nawaz, G. Gong. "The WG Stream Cipher," *ECRYPT Stream Cipher Project Report 2005/033*. Available at http://www.ecrypt.eu.org/stream/
12. Y. Nawaz, G. Gong. "Preventing Chosen IV Attack on WG Cipher by Increasing the Length of Key/IV Setup," *ECRYPT Stream Cipher Project Report 2005/047*. Available at http://www.ecrypt.eu.org/stream/