Cryptanalysis of the Stream Cipher ABC v2*

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. ABC v2 is a software-efficient stream cipher with 128-bit key. In this paper, we apply a fast correlation attack to break ABC v2 with weak keys. There are about 2^{96} weak keys in ABC v2. The complexity to identify a weak key and to recover the internal state of a weak key is low: identifying one weak key from about 2^{32} random keys requires 6460 keystream bytes and $2^{13.5}$ operations for each random key. Recovering the internal state of a weak key requires about $2^{19.5}$ keystream bytes and $2^{32.8}$ operations. A similar attack can be applied to break ABC v1 with much lower complexity than the previous attack on ABC v1.

Keywords: Fast correlation attack, key-dependent S-box, stream cipher, ABC v2.

1 Introduction

ABC [1] is a stream cipher submitted to the ECRYPT eStream project. It is one of the fastest submissions with encryption speed about 3.5 cycles/byte on the Intel Pentium 4 microprocessor.

ABC v1 was broken by Berbain and Gilbert [3] (later by Khazaei [8]). Their divide-and-conquer attack on ABC exploits the short length (63 bits) of the LFSR in the component A and the non-randomness in the component C: all the possible initial values of the LFSR get tested, and the correct value results in the biased binary stream that matches the non-random output from the component C. The component C is a key-dependent 32-bit-to-32-bit S-box. Vaudenay [12], Murphy and Robshaw [11] have stated that the key-dependent S-boxes may be weak. Berbain and Gilbert's attack on ABC v1 deals with the weak keys that are related to the non-bijective S-box. This type of weak key exists with probability close to 1. Recovering the internal state of a weak key requires about 2^{95} operations and 2^{34} keystream bytes.

In order to resist these attacks, the ABC designers introduced ABC v2 with the improved components A and B. In ABC v2 [2], the length of the LFSR is 127 bits instead of the 63 bits in ABC v1. The increased LFSR length makes it impossible to test all the states of the LFSR, thus the attack on ABC v1

* 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.

E. Biham and A.M. Youssef (Eds.): SAC 2006, LNCS 4356, pp. 56–66, 2007.
© Springer-Verlag Berlin Heidelberg 2007

can no longer be applied to ABC v2. However ABC v2 is still insecure due to the low weight of the LFSR and the non-randomness in the component C (the component C in ABC v1 is the same as in ABC v2).

In this paper, we find a new type of weak key that exists with probability 2^{-32}. This new type of weak key results in a heavily biased output of the component C. Due to the low weight of the LFSR and the strong correlation resulting from the component C, a fast correlation attack can be applied to recover the LFSR. After recovering the LFSR, the internal state of the cipher can be recovered easily. The identification of a weak key from 2^{32} random keys requires 6460 keystream bytes from each key, and $2^{13.5}$ instructions for each keystream. Recovering the internal state of a weak key requires about $2^{27.5}$ keystream bytes and $2^{35.7}$ instructions. Both the ABC v1 and ABC v2 are vulnerable to this attack.

This paper is organized as follows. In Sect. 2, we illustrate the ABC v2. In Sect. 3, we define the weak keys and show how to identify them. Section 4 recovers the internal state of a weak key. Section 5 concludes this paper.

2 The Stream Cipher ABC v2

The stream cipher ABC v2 consists of three components – A, B and C, as shown in Fig. 1. The component A is a regularly clocked LFSR, the component B is a finite state machine (FSM), and the component C is a key-dependent S-box. ABC v1 has the same structure as ABC v2 except that the LFSR in ABC v1 is 63-bit, and the FSM in ABC v1 has less elements than that in ABC v2. The component C in ABC v1 is the same as that in ABC v2.

The component A is based on a linear feedback shift register with primitive polynomial $g(x) = x^{127} + x^{63} + 1$. Denote the register in component A as $(\overline{z}_3, \overline{z}_2, \overline{z}_1, \overline{z}_0)$, where each $\overline{z}_i$ is a 32-bit number. Note that this 128-bit register itself is not a linear feedback shift register. Its initial value depends on the key and IV. At each step of ABC v2, 32 bits of this 128-bit register get updated:

$$\zeta = (\overline{z}_2 \oplus (\overline{z}_1 \ll 31) \oplus (\overline{z}_0 \gg 1)) \bmod 2^{32}$$
$$\overline{z}_0 = \overline{z}_1 \ , \ \overline{z}_1 = \overline{z}_2 \ , \ \overline{z}_2 = \overline{z}_3 \ , \ \overline{z}_3 = \zeta \ ,$$

where $\ll$ and $\gg$ indicates left shift and right shift, respectively.

The component B is specified as $B(x) = ((x \oplus d_0) + d_1) \oplus d_2 \bmod 2^{32}$, where x is the 32-bit input, d_0, d_1 and d_2 are key and IV dependent 32-bit numbers, $d_0 \equiv 0 \bmod 4$, $d_1 \equiv 1 \bmod 4$, $d_2 \equiv 0 \bmod 4$. The x is updated as $x = B(x) + \overline{z}_3$.

The component C is specified as $C(x) = S(x) \ggg 16$, where $\ggg$ indicates rotation, x is the 32-bit input, $S(x) = e + \sum_{i=0}^{31}(e_i \times x[i])$, where $x[i]$ denotes the ith least significant bit of x, and e and e_i are key dependent 32-bit random numbers, except that $e_{31} \equiv 2^{16} \bmod 2^{17}$. Note that e and e_i are not related to the initialization vector.

Each 32-bit keystream word is given as $y = C(x) + \overline{z}_0$.

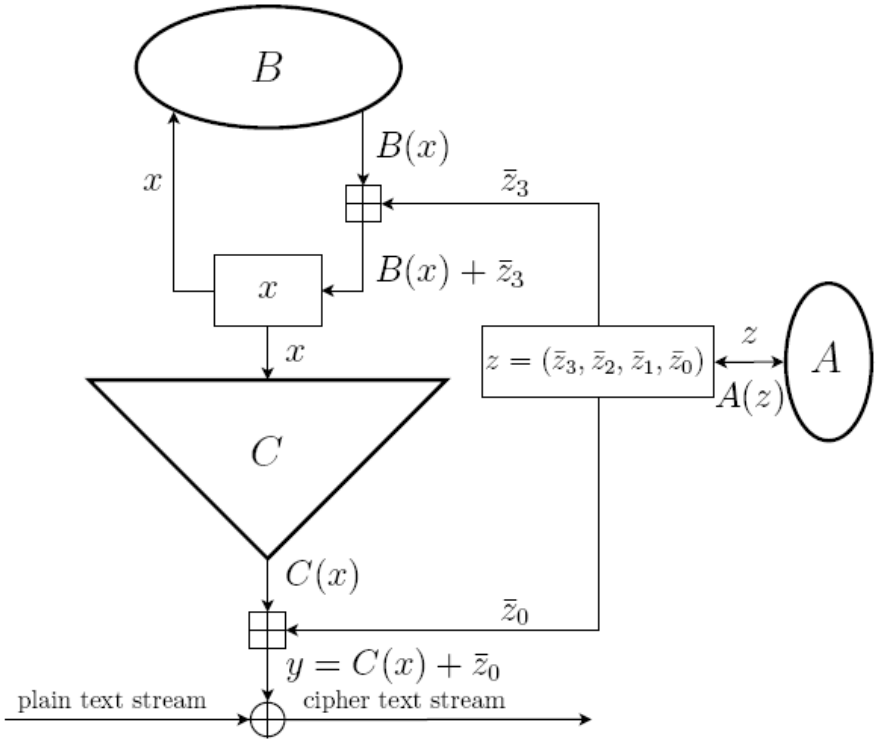

Fig. 1. Keystream generation of ABC v2 [2]

The details of the initialization of ABC v2 are not described here. We are only interested in the generation of the key-dependent S-box in the component C. The above specifications of the component C are sufficient for the illustration of the attacks presented in this paper.

3 The Weak Keys of ABC v2

In Sect. 3.1, we introduce some observation related to the bias of carry bits. Section 3.2 defines the ABC v2 weak keys and gives an attack to identify them.

3.1 The Large Bias of Carry Bits

Carry bits are always biased even if the two addends are random. The bias of the carry bit at the n-th least significant bit position is $\frac{1}{2} + 2^{-n-1}$ $(n \geq 1)$. However, this bias is very small for large n. In the following, we look for the large bias of carry bits when the addends are not random.

Table 1. The probability of $c_1 \oplus c_2 \oplus c_3 = 0$ (denote the probability as $\frac{1}{2} + \epsilon$)

n	ϵ	n	ϵ
1	0.125	5	0.071441650390625
2	0.078125	6	0.071430206298828125
3	0.072265625	7	0.071428775787353515625
4	0.071533203125	8	0.071428596973419189453125

Lemma 1. Denote u and v as two random and independent n-bit integers. Let $c_n = (u + v) \gg n$, where c_n denotes the carry bit at the nth least significant bit position. Denote the most significant bit of u as u_{n-1}. Then $\Pr(c_n \oplus u_{n-1} = 0) = \frac{3}{4}$.

Proof. $c_n = (u_{n-1} \cdot v_{n-1}) \oplus ((u_{n-1} \oplus v_{n-1}) \cdot c_{n-1})$. If $c_{n-1} = 0$, then $c_n \oplus u_{n-1} = u_{n-1} \cdot \overline{v}_{n-1}$, where $\overline{v}_{n-1}$ denotes the inverse of v_{n-1}. If $c_{n-1} = 1$, then $c_n \oplus u_{n-1} = \overline{u}_{n-1} \cdot v_{n-1}$. Thus $\Pr(c_n \oplus u_{n-1} = 0) = \frac{3}{4}$.

Lemma 1 implies the following bias.

Theorem 1. Denote a_i, b_i $(1 \le i \le 3)$ as n-bit integers. Denote c_i $(1 \le i \le 3)$ as binary values satisfying $c_i = (a_i + b_i) \gg n$. Let a_1, a_2, b_1, b_2 and b_3 be random and independent, but $a_3 = a_1 \oplus a_2$. Then $c_1 \oplus c_2 \oplus c_3$ is biased. For $n = 16$, $\Pr(c_1 \oplus c_2 \oplus c_3 = 0) \approx 0.5714$.

If we apply Lemma 1 directly, we obtain that $\Pr(c_1 \oplus c_2 \oplus c_3 = 0) = \frac{1}{2} + \frac{1}{16} = 0.5625$. (The u_{n-1}'s in Lemma 1 are eliminated since they are linearly related in Theorem 1.) The small difference between these two biases (0.5714 and 0.5625) is due to the fact that a_3 is not an independent random number.

We illustrate the validity of Theorem 1 with numerical analysis. For small n, we try all the values of a_1, a_2, b_1, b_2 and b_3 and obtain the following table. From Table 1, we see that the bias ϵ converges to 0.0714 as the value of n increases. For $n = 16$, we performed 2^{32} tests, and the bias is about 0.071424. For $n = 32$, the bias is about 0.071434 with 2^{32} tests. The experimental results show that Theorem 1 is valid. Recently, the complete proof of Theorem 1 is given in [16]. It was shown that $\Pr(c_1 \oplus c_2 \oplus c_3 = 0) = \frac{4}{7} + \frac{3}{7} \times \frac{1}{8^n}$, which confirms the correctness of Theorem 1.

Remarks. In Theorem 1, if a_1, a_2, a_3, b_1, b_2 and b_3 are all random and independent, then $\Pr(c_1 \oplus c_2 \oplus c_3 = 0) = \frac{1}{2} + 2^{-3n-1}$, which is very small for $n = 16$. This small bias cannot be exploited to break ABC v2.

3.2 Identifying the Weak Keys

We start the attack with analyzing the linear feedback shift register used in ABC v2. The register $(\overline{z}_3, \overline{z}_2, \overline{z}_1, \overline{z}_0)$ is updated according to the primitive polynomial $g(x) = x^{127} + x^{63} + 1$. Note that each time the 127-bit LFSR advances

32 steps. To find a linear relation of the 32-bit words, we take the 2^5th power of $g(x)$, and obtain

$$g^{2^5}(x) = x^{127\times 32} + x^{63\times 32} + 1\,. \tag{1}$$

Denote the z_0 at the i-th step as z_0^i, and denote the jth significant bit of z_0^i as $z_{0,j}^i$. Since each time 32 bits get updated, the distance between $\overline{z}_{0,j}^i$ and $\overline{z}_{0,j}^{i+k}$ is $|32\cdot(k-i)|$. According to (1), we obtain the following linear recurrence

$$\overline{z}_0^i \oplus \overline{z}_0^{i+63} \oplus \overline{z}_0^{i+127} = 0\,. \tag{2}$$

The weak keys of ABC v2 are related to the $S(x)$ in the component C. $S(x)$ is defined as $S(x) = e + \sum_{i=0}^{31}(e_i \times x[i])$, where e and e_i are key dependent 32-bit random numbers, except that $e_{31} \equiv 2^{16} \bmod 2^{17}$. **If the least significant bits of e and e_i $(0 \le i < 32)$ are all 0, then the least significant bit of $S(x)$ is always 0, and we consider the key as weak key**. Note that the least significant bit of e_{31} is always 0. Thus a randomly chosen key is weak with probability 2^{-32}.

In the following, we describe how to identify the weak keys. Denote the 32-bit keystream word at the ith step as y_i, the jth significant bit of y_i as $y_{i,j}$. And denote x_i as the input to function S at the i-th step. Then $y_i = (S(x_i) \ggg 16) + \overline{z}_0^i$. Let $c_{i,j}$ denote the carry bit at the j-th least significant bit position of $(S(x_i) \ggg 16) + \overline{z}_0^i$, i.e., $c_{i,j} = (((S(x_i) \ggg 16) \bmod 2^j) + (\overline{z}_0^i \bmod 2^j)) \gg j$. Assume that $((S(x_i) \ggg 16) \bmod 2^{16}$ is random. According to Theorem 1 and (2), we obtain

$$\Pr(c_{i,16} \oplus c_{i+63,16} \oplus c_{i+127,16} = 0) = \frac{1}{2} + 0.0714\,. \tag{3}$$

Due to the rotation of $S(x_i)$, we know that

$$y_{i,16} = S(x_i)_0 \oplus z_{0,16}^i \oplus c_{i,16}\,, \tag{4}$$

where $S(x_i)_0$ denotes the least significant bit of $S(x_i)$. Note that $S(x_i)_0$ is always 0 for a weak key. From (2) and (4), we obtain

$$y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = c_{i,16} \oplus c_{i+63,16} \oplus c_{i+127,16}\,. \tag{5}$$

From (3) and (5), $y_{i,16}$ is biased as

$$\Pr(y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = 0) = \frac{1}{2} + 0.0714\,. \tag{6}$$

We use (6) to identify the weak keys. Approximate the binomial distribution with the normal distribution. Denote the total number of samples as N, the mean as μ, and the standard deviation as σ. For the binomial distribution, $p = \frac{1}{2}$, $\mu = Np$ and $\sigma = \sqrt{Np(1-p)}$. For (6), $p' = \frac{1}{2} + 0.0714$, $\mu' = Np'$ and $\sigma' = \sqrt{Np'(1-p')}$. For the normal distribution, the cumulative distribution function gives value $1 - 2^{-39.5}$ at 7σ, and value 0.023 at -2σ. If the following relation holds

$$u' - u \ge 7\sigma + 2\sigma'\,, \tag{7}$$

then on average, each strong key is wrongly identified as weak key (false positive) with probability $2^{-39.5}$, and each weak key is not identified as weak key (false negative) with probability 0.023. It means that the weak keys can be successfully identified since one weak key exists among 2^{32} keys. Solving (7), the amount of samples required is $N = 3954$. For each sample, we only need to perform two XORs and one addition. With 3594+127 = 4081 outputs from each key, we can successfully identify the weak keys of ABC v2.

The number of outputs can be reduced if we consider the 2^ith power of $g(x)$ for $i = 5, 6, 7, 8$. With 1615 outputs, we can obtain 3956 samples. Thus the keystream required in the identification of the weak keys can be reduced to 1615 outputs.

The identification of a weak key implies directly a distinguishing attack on ABC v2. If there are 2^{32} keystreams generated from 2^{32} random keys, and each key produces 1615 outputs, then the keystream can be distinguished from random with high probability. In order to find one weak key, the total amount of keystream required are $2^{32} \times 1615 \times 4 = 2^{44.7}$ bytes, and the amount of computations required are $2^{32} \times 3956 \times 2 \approx 2^{45}$ XORs and 2^{44} additions.

Experiment 1. We use the original ABC v2 source code provided by the ABC v2 designers in the experiment. After testing 2^{34} random keys, we obtain five weak keys, and one of them is (fe 39 b5 c7 e6 69 5b 44 00 00 00 00 00 00 00 00). From this weak key we generate 2^{30} outputs, and the bias defined in (6) is 0.5714573. The experimental results confirm that the probability that a randomly chosen key is weak is about 2^{-32}, and the bias of a weak key keystream is large.

4 Recovering the Internal State

Once a weak key is identified, we proceed to recover the internal state resulting from the weak key. In Sect. 4.1, we apply the fast correlation attack to recover the LFSR. The components B and C are recovered in Sect. 4.2. The complexity of the attack is given in Sect. 4.3. Section 4.4 applies the attack to ABC v1.

4.1 Recovering the Initial Value of the LFSR

The initial value of the LFSR is recovered by exploiting the strong correlation between the LFSR and the keystream. From Lemma 1, we get

$$\Pr(\overline{z}^i_{0,15} \oplus c_{i,16} = 0) = \frac{3}{4}. \tag{8}$$

From (8) and (4), we obtain the following correlation:

$$\Pr(\overline{z}^i_{0,16} \oplus \overline{z}^i_{0,15} \oplus y_{i,16} = 0) = \frac{3}{4}. \tag{9}$$

The strong correlation in (9) indicates that the cipher is very weak.

The fast correlation attack Algorithm A of Meier and Staffelbach [9] can be applied to recover the LFSR. There are some advanced fast correlation attacks [10,6,7], but the original attack given by Meier and Staffelbach is sufficient here since we are dealing with a strong correlation and a two-tap LFSR.

We now apply the fast correlation attack Algorithm A [9] to recover the LFSR. Let $p = \frac{3}{4}$, $u_i = \overline{z}^i_{0,16} \oplus \overline{z}^i_{0,15}$, and $w_i = y_{i,16}$. By squaring the polynomial (1) iteratively, we obtain a number of linear relations for every u_i:

$$u_i \oplus u_{i+63\cdot 2^j} \oplus u_{i+127\cdot 2^j} = 0 \quad (j \geq 0). \tag{10}$$

From (9) and (10), we obtain

$$s = \Pr(w_i \oplus w_{i+63\cdot 2^j} \oplus w_{i+127\cdot 2^j} = 0 \mid u_i = w_i) = p^2 + (1-p)^2\,, \tag{11}$$

where each value of j indicates one relation for w_i (also for $w_{i+63\cdot 2^j}$ and $w_{i+127\cdot 2^j}$). On average there are m relations for w_i as

$$m = m(N,k,t) = (t+1)\cdot \log_2(\frac{N}{2k})\,, \tag{12}$$

where N is the number of outputs, $k = 127$ (the length of the LFSR), $t = 2$ (taps) for ABC v2. The probability that w_i satisfies at least h of the m relations equals

$$Q(h) = \sum_{i=h}^{m} \binom{m}{i} \cdot (p\cdot s^i \cdot (1-s)^{m-i} + (1-p)\cdot (1-s)^i \cdot s^{m-i})\,. \tag{13}$$

If $u_i = w_i$, then the probability that w_i satisfies h of these m relations is equal to

$$R(h) = \sum_{i=h}^{m} \binom{m}{i} \cdot p\cdot s^h \cdot (1-s)^{m-h}\,. \tag{14}$$

According to [9], $N \cdot Q(h)$ is the number of u_i's being predicted in the attack, and $N \cdot R(h)$ is the number of u_i's being correctly predicted.

For $N = 4500$, there are on average about 12 relations for each w_i. For $h = 11$, 98.50 bits can be predicted with 98.32 bits being predicted correctly. For $h = 10$, 384.99 bits can be predicted with 383.21 bits being predicted correctly. To predict 127 bits, we can predict 98.50 bits for $h = 11$, then predict 127-98.50 = 28.50 bits using the w_i's satisfying only 10 relations. Then in average there are $98.32 + 28.50 \times \frac{383.21-98.31}{384.99-98.50} = 126.66$ bits being predicted correctly. It shows that 127 u_i's can be determined with about 0.34 wrong bits. Then the LFSR can be recovered by solving 127 linear equations.

We carry out an experiment to verify the above analysis. In order to reduce the programming complexity, we consider only the w_i's with 12 relations, thus we use 8000 outputs in the experiment. Using more outputs to recover the LFSR has no effect on the overall attack since recovering the component B requires about $2^{17.5}$ outputs, as shown in Sect. 4.2.

Experiment 2. From the weak key (fe 39 b5 c7 e6 69 5b 44 00 00 00 00 00 00 00 00), we generate 8000 outputs, but consider only those $8000 - 2 \cdot 2^{\frac{12}{3}} \cdot 127 = 3936$ w_i's with 12 relations. We repeat the experiments 256 times with different IVs. For $h = 11$, 104.66 bits can be predicted with 104.35 bits being predicted correctly. For the w_i's satisfying only 10 relations, 278.37 bits can be predicted with 276.08 bits being predicted correctly. To predict 127 bits, $127 - 104.66 = 22.34$ w_i's satisfying only 10 relations should be used. Among the 127 predicted bits, $104.35 + 22.34 \times \frac{276.08}{278.37} = 126.51$ bits are correct.

4.2 Recovering the Components B and C

After recovering the LFSR, we proceed to recover the component B. In the previous attack on ABC v1 [3], about 2^{77} operations are required to recover the components B and C. That complexity is too high. We give here a very simple method to recover the components B and C with about $2^{33.3}$ operations.

In ABC v2, there are four secret terms in the component B: x, d_0, d_1, and d_2, where d_0, d_1 and d_2 are static, x is updated as

$$x_i = (((x_{i-1} \oplus d_0) + d_1) \oplus d_2) + \overline{z}_3^i \bmod 2^{32}. \tag{15}$$

Note that the higher significant bits never affect the less significant bits. It allows us to recover x, d_0, d_1, and d_2 bit-by-bit.

Since the initial value of the LFSR is known, the value of each $\overline{z}_0^i$ can be computed, thus we know the value of each $S(x_i)$. In average, the probability that $x_i = x_j$ is about 2^{-32}. For a weak key, the least significant bit of $S(x_i)$ is always 0, and the probability that $S(x_i) = S(x_j)$ is about $2^{-32} + (1 - 2^{-32}) \cdot 2^{-31} \approx 2^{-32} + 2^{-31}$. Given $2^{17.5}$ outputs, there are about $\binom{2^{17.5}}{2} \times (2^{-32} + 2^{-31}) \approx 12$ cases that $S(x_i) = S(x_j)$ $(i \neq j)$. And there are about $\binom{2^{17.5}}{2} \times 2^{-32} \approx 4$ cases that $x_i = x_j$ among those 12 cases. Choose four cases from those 12 cases randomly, the probability that $x_{i_u} = x_{j_u}$ for $0 \leq u < 4$ is $(\frac{4}{12})^4 = \frac{1}{81}$ (here (i_u, j_u) indicates one of those 12 pairs (i, j) satisfying $S(x_i) = S(x_j)$ $(i \neq j)$).

The value of each $\overline{z}_3^i$ in (15) is already known. When we solve the four equations $x_{i_u} = x_{j_u}$ $(0 \leq u < 4)$ to recover x, d_0, d_1, and d_2, we obtain the four unknown terms bit-by-bit from the least significant bit to the most significant bit. The four most significant bits cannot be determined exactly, but the four least significant bits can be determined exactly since only the least significant bit of x is unknown. (We mention here during this bit-by-bit approach, the four bits at each bit position may not be determined exactly, and further filtering is required in the consequent computations.) On average, we expect that solving each set of four equations gives about 8 possible values of x, d_0, d_1, and d_2. Also note that each set of four equations holds true with probability $\frac{1}{81}$, we have about $81 \times 8 = 648$ possible solutions for x, d_0, d_1, and d_2.

After recovering the component B, we know the input and output of each $S(x_i)$, so the component C can be recovered by solving 32 linear equations. This process is repeated 648 times since there are about 648 possible solutions of x, d_0, d_1, and d_2. The exact B and C can be determined by generating some outputs and comparing them to the keystream.

4.3 The Complexity of the Attack

According to the experiment, recovering the LFSR requires about 8000 outputs. For each w_i, testing 12 relations requires about $\frac{12 \cdot 2}{3} = 8$ XORs and 12 additions. After predicting 127 u_i's, each u_i should be expressed in terms of the initial state of the LFSR. It is almost equivalent to running the LFSR $8000 \cdot 32$ steps, with the LFSR being initialized with only one non-zero bit $\overline{z}^0_{1,31}$. Advancing the LFSR 32 steps requires 2 XORs and 2 shifts. Solving a set of 127 binary linear equations requires about $\frac{2 \cdot 127^3}{3} \cdot \frac{1}{32} \approx 42675$ operations on the 32-bit microprocessor. So about $2^{17.8}$ operations are required to recover the LFSR.

Recovering the component B requires about $2^{17.5}$ outputs and solving 81 sets of equations. Each set of equations can be solved bit-by-bit, and it requires about $32 \cdot 2^4 \cdot 2^{17.5} = 2^{26.5}$ operations. Recovering the component C requires solving 648 sets of equations. Each set of equations consists of 32 linear equations with binary coefficients, and solving it is almost equivalent to inverting a 32×32 binary matrix which requires about $\frac{2 \cdot 32^3}{3} \cdot \frac{1}{32} \approx 683$ operations. So $81 \cdot 2^{26.5} + 648 \cdot 683 = 2^{32.8}$ operations are required to recover the components B and C.

Recovering the internal state of a weak key requires $2^{17.5}$ outputs and $2^{17.8} + 2^{32.8} = 2^{32.8}$ operations in total.

4.4 The Attack on ABC v1

The previous attack on ABC v1 deals with a general type of weak keys [3], but the complexity is too high (2^{95} operations). The above attack can be slighty modified and applied to break ABC v1 (with the weak keys defined in Sect. 3) with much lower complexity. We outline the attack on ABC v1 below.

The LFSR in ABC v1 is 63 bits. The shorter LFSR results in more relations for the same amount of keystream. Identifying a weak key requires 1465 outputs from each key instead of the 1615 outputs required in the attack on ABC v2. In theory, recovering the LFSR with the fast correlation attack requires 2500 outputs instead of the 4500 outputs required in the attack on ABC v2. The component B in ABC v1 has only three secret variables. Recovering the component B requires $2^{17.3}$ outputs, with the complexity reduced to $2^{30.1}$ operations, smaller than the $2^{32.8}$ operations required to recover the component B of ABC v2. In total the attack to recover the internal state of ABC v1 with a weak key requires $2^{17.3}$ outputs and $2^{30.1}$ operations.

5 Conclusion

Due to the large amount of weak keys and the serious impact of each weak key, ABC v1 and ABC v2 are practically insecure.

In order to resist the attack presented in this paper, a straightforward solution is to ensure that at least one of the least significant bits of the 33 elements in the component B should be nonzero. However, ABC v2 with such improvement is still insecure. A new type of weak keys with all the two less (but not least)

significant bits being 0 still exists. After eliminating all the similar weak keys, the linear relation in (2) can still be applied to exploit the non-randomness in the outputs of the component C to launch a distinguishing attack. ABC v3 is the latest version of ABC, and it eliminates the weak keys described in this paper. However, a recent attack exploiting the non-randomness in the outputs of the component C is still able to identify a new weak key with about 2^{60} outputs [15]. It seems difficult to improve the ABC cipher due to the risky design that the 32-bit-to-32-bit S-box is generated from only 33 key-dependent elements.

We recommend updating the secret S-box of ABC v2 frequently during the keystream generation process. In ABC v2, the key-dependent S-box is fixed. For block cipher design, the S-box has to remain unchanged, but such restriction is not applicable to stream cipher. Suppose that the size of the key-dependent S-box of a stream cipher is large (it is risky to use the small randomly generated key-dependent S-box). We can update the S-box frequently, such as updating at least one element of the S-Box at each step (in a cyclic way to ensure that all the elements of the S-box get updated) with enough random information in an unpredictable way. When a weak S-box appears, only a small number of outputs are generated from it before the weak S-box disappears, and it becomes extremely difficult for an attacker to collect enough outputs to analyze a weak S-box. Thus an attacker has to deal with the average property of the S-box, instead of dealing with the weakest S-box. For example, the eSTREAM submissions HC-256 [13], HC-128 [14], Py [4] and Pypy [5] use the frequently updated large S-boxes to reduce the effect resulting from the weak S-boxes. The security of ABC stream cipher can be improved in this way, but its performance will be affected.

Acknowledgements. The authors would like to thank the anonymous reviewers of SAC 2006 for their helpful comments. Thanks also go to Adi Shamir and others for their comments on the security of updating a key-dependent S-box at the SAC 2006 conference.

References

1. Anashin, V., Bogdanov, A., Kizhvatov, I.: ABC: A New Fast Flexible Stream Cipher. Available at `http://www.ecrypt.eu.org/stream/ciphers/abc/abc.pdf`
2. Anashin, V., Bogdanov, A., Kizhvatov, I.: Security and Implementation Properties of ABC v.2. SASC 2006 - Stream Ciphers Revisited, pp. 278–292, (2006), Available at `http://www.ecrypt.eu.org/stream/papersdir/2006/026.pdf`
3. Berbain, C., Gilbert, H.: Cryptanalysis of ABC. Available at `http://www.ecrypt.eu.org/stream/papersdir/048.pdf`
4. Biham, E., Seberry, J.: Py: A Fast and Secure Stream Cipher Using Rolling Arrays. Available at `http://www.ecrypt.eu.org/stream/p2ciphers/py/py_p2.ps`
5. Biham, E., Seberry, J.: Pypy: Another Version of Py. Available at `http://www.ecrypt.eu.org/stream/p2ciphers/py/pypy_p2.ps`
6. Chepyzhov, V.V., Johansson, T., Smeets, B.: A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 181–195. Springer, Heidelberg (2001)

7. Johansson, T., Jönsson, F.: Fast Correlation Attacks through Reconstruction of Linear Polynomials. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 300–315. Springer, Heidelberg (2000)
8. Khazaei, S.: Divide and Conquer Attack on ABC Stream Cipher. Available at http://www.ecrypt.eu.org/stream/papersdir/052.pdf
9. Meier, W., Staffelbach, O.: Fast Correlation Attacks on Stream Ciphers. Journal of Cryptology 1(3), 159–176 (1989)
10. Mihaljević, M., Fossorier, M.P.C., Imai, H.: A Low-Complexity and High-Performance Algorithm for Fast Correlation Attack. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 196–212. Springer, Heidelberg (2001)
11. Murphy, S., Robshaw, M.J.B.: Key-dependent S-boxes and differential cryptanalysis. Designs, Codes, and Cryptography 27(3), 229–255 (2002)
12. Vaudenay, S.: On the Weak Keys of Blowfish. In: Gollmann, D. (ed.) Fast Software Encryption. LNCS, vol. 1039, pp. 27–32. Springer, Heidelberg (1996)
13. Wu, H.: A New Stream Cipher HC-256. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 226–244. Springer, Heidelberg (2004), Full version available at http://eprint.iacr.org/2004/092.pdf
14. Wu, H.: The Stream Cipher HC-128. Available at http://www.ecrypt.eu.org/stream/p2ciphers/hc256/hc128_p2.pdf
15. Zhang, H., Li, L., Wang, X.: Fast Correlation Attack on Stream Cipher ABC v3 (2006), Available at http://www.ecrypt.eu.org/stream/papersdir/2006/049.pdf
16. Zhang, H., Wang, S., Wang, X.: The Probability Advantages of Two Linear Expressions in Symmetric Ciphers (2006), Available at http://www.ecrypt.eu.org/stream/papersdir/2006/046.pdf