Cryptanalysis of ABC v2

Hongjun Wu and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC
{wu.hongjun,bart.preneel}@esat.kuleuven.be

Abstract. ABC v2 is a stream cipher with 128-bit key. In this paper, we show that there are about 2^{96} weak keys. The complexity to identify each weak key and to recover the internal state is low. To identify a weak key from about 2^{32} random keys, we need 2^{13} keystream bytes from each key, and $2^{13.5}$ operations are required for each keystream. Recovering the internal state of a weak key requires about $2^{29.5}$ keystream bytes and $2^{35.7}$ operations. The similar attack can be apply to break ABC v1 and its complexity is much lower than the previous attack on ABC v1.

1 Introduction

ABC v1 [2] is a submission to the ECRYPT eStream project. ABC v1 was broken by Berbain and Gilbert [1] (later by Khazaei [7]). That divide-and-conquer attack on ABC v1 exploits the short LFSR length in component A and the non-randomness in component C. To resist the attacks, the ABC designers modified component A. In ABC v2 [3–6], the length of the LFSR is 127 bits instead of the 63 bits in ABC v1.

In this paper, we analyze the weakness in component C of ABC v2 (component C in ABC v1 is the same as that in ABC v2). Component C is a key-dependent 32-bit-to-32-bit S-box. Vaudenay [10], Murphy and Robshaw [9] have stated that there could be weak key-dependent S-boxes due to the weak keys. Berbain and Gilbert's attack on ABC v1 deals with the weak keys that are related to the non-bijective S-box. That type of weak key exists with probability close to 1. To recover the internal state of a weak key requires about 2^{93} operations and 2^{34} keystream bytes.

In this paper, we found another type of weak key that exists with probability 2^{-32}. This new type of weak key is so weak that both the ABC v1 and ABC v2 fail completely. The identification of a weak key from 2^{32} random keys requires 2^{13} keystream bytes from each key, and $2^{13.5}$ operations for each keystream. Recovering the internal state of a weak key requires about $2^{27.5}$ keystream bytes and $2^{35.7}$ operations.

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

2 ABC v2

ABC v2 consists of three components – A, B and C. The keystream generation of the cipher is given in Fig. 1.

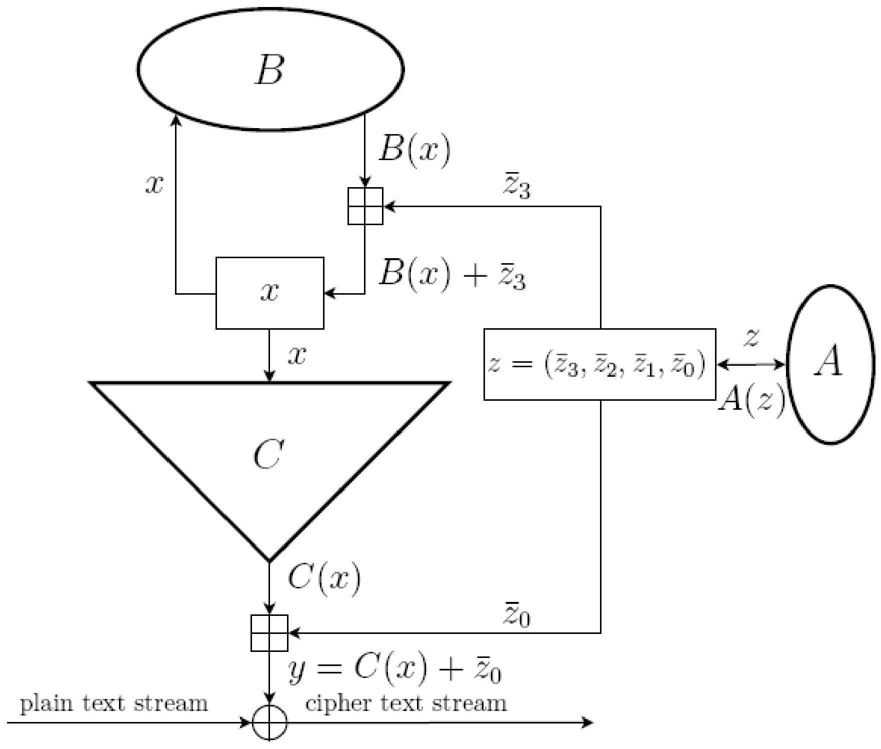

Fig. 1. Keystream generation of ABC v2 [6]

Compnent 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 linear feedback shift register. This register is key and IV dependent. During each step of ABC v2, 32 bits of this 128-bit regesiter are updated as follows:

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

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, $d_0 \equiv 0 \bmod 4$, $d_1 \equiv 1 \bmod 4$, $d_2 \equiv 0 \bmod 4$.

Component C is specified as $C(x) = S(x) >>> 16$, where 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.

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

3 Weak Key and Identification

In Subsection 3.1, we introduce some observation related to the bias in the keystream of ABC v2. Subsection 3.2 defines the ABC v2 weak key and gives the attack to identify the weak keys.

3.1 How the bias occurs

Lemma 1. Denote a, b as two random and independent n-bit integers. Let $c_n = (a+b) \bmod 2^n$, where c_n denotes the carry bit at the n-th least significant bit position. Denote the most significant bit of a as a_{n-1}. Then $\Pr(c_n \oplus a_{n-1} = 0) = \frac{3}{4}$.

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

Lemma 1 implies the bias given in the following theorem.

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 $c_i = (a_i + b_i) \bmod 2^n$. 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 simply apply Lemma 1, we obtain that $\Pr(c_1 \oplus c_2 \oplus c_3 = 0) = \frac{1}{2} + \frac{1}{16} = 0.5625$. The small diffrence in the biases (0.5714 and 0.5625) is due to the fact that a_3 is not an independent random number.

We show 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.

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

From Table 1, we see that the bias converges to 0.5714 as the value of n increases. For $n = 16$, we performed 2^{32} tests, and the bias is about 0.571424152. For $n = 32$, the bias is about 0.571434624 with 2^{32} tests. The experiment results show that Theorem 1 is valid.

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 is very small for $n = 16$, and this bias could not be exploited to break ABC v2.

3.2 Identify 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 $g(x) = x^{127} + x^{63} + 1$. Take the 2^5th power of $g(x)$, we 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 are updated, the distance between $\overline{z}_{0,j}^i$ and $\overline{z}_{0,j}^{i+k}$ is $|32(k-i)|$. 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 are related to the $S(x)$ in 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 that key as weak key**. Note that the least significant bit of e_{31} is always 0. Thus a weak key appears with probability 2^{-32}.

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

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

And 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 the 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), we know that $y_{i,16}$ is biased,

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

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 variance as σ. For random binary 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

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

then in 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 could 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. Thus with $3594+127 = 4081$ outputs from each key, we could successfully identify the weak keys of ABC v2.

The amount of outputs could be reduced if we consider 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 indentification of the weak keys is 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 keystream is with 1615 outputs, then the keystream could be distinguished from random. 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. In this experiment, we used the original ABC v2 source code provided by the ABC v2 designers. After testing 2^{34} random keys, we obtained five weak keys. One of the weak key is (fe 39 b5 c7 e6 69 5b 44 00 00 00 00 00 00 00 00). From this weak key we generated 2^{30} outputs, and the probability p_y (6) is 0.5714573. The experiment results confirm that a weak key exists with probability about 2^{-32}, and that the bias of the weak key keystream is large.

4 Recovering the Internal State of the Weak Keys

Once a weak key is identified, the internal state of the cipher could be recovered. The correlation between the keystream and the LFSR is disucssed in Subsection 4.1. In Subsection 4.2, we recover the internal state of the LFSR. The secret variables in component B and C are recovered in Subsection 4.3.

4.1 Correlation between the keystream and the LFSR

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:

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

Unfortunately (9) can not be used in fast correlation attack to increase the chance of the prediction of $\overline{z}^i_{0,16} \oplus \overline{z}^i_{0,15}$. The rason is given as follows. From the experiment, we found that

$$\Pr(\overline{z}^i_{0,16} \oplus \overline{z}^i_{0,15} = 0 \mid y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = 0) < \frac{1}{2} + 2^{-17}. \tag{10}$$

It shows that such relation cannot be used in the fast correlation attack. We need further study on this strange behavior.

According to (9), recovering the LFSR requires testing about $2^{99.4}$ states. To reduce the complexity, we need a correlation that can be exploited in the fast correlation attack.

Theorem 2. Denote a_i, b_i $(1 \leq i \leq 3)$ as n-bit $(n \geq 2)$integers. Denote c_i $(1 \leq i \leq 3)$ as binary values satisfying $c_i = (a_i + b_i) \bmod 2^n$. Denote h_i as $h_i = a_{i,n-1} \oplus a_{i,n-2}$, where $a_{i,j}$ denotes the jth least significant bit of a_i ($a_{i,n-1}$ is the most significant bit of a_i). Let a_1, a_2, b_1, b_2 and b_3 be random, but $a_3 = a_1 \oplus a_2$. If $c_1 \oplus c_2 \oplus c_3 = 0$, then h_i is biased. For $n = 16$, $\Pr(h_i = 0 \mid c_1 \oplus c_2 \oplus c_3 = 0) \approx 0.541$ $(1 \leq i \leq 3)$.

We show the validity of Theorem 2 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 Table 2. From Table 2, we see that as n increases, the values of p_0 converges to 0.541. For $n = 16$, we performed 2^{32} tests, and the values of p_0 is 0.54103037. The experiment results confirm that Theorem 2 is valid.

Table 2. The correlation $\Pr(h_i = 0 \mid c_1 \oplus c_2 \oplus c_3 = 0)$

n	correlation	n	correlation
2	0.54054054054	5	0.54101468625
3	0.54095563140	6	0.54101550765
4	0.54100811619	7	0.54101561033

Let $h_i = \overline{z}^i_{0,15} \oplus \overline{z}^i_{0,14}$. From (2), (5) and Theorem 2, we obtain that

$$\Pr(h_i = 0 \mid y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = 0) = 0.541 \tag{11}$$

(11) shows that there is strong correlation between the keystream and the LFSR.

Experiment 2. In this experiment, from the weak key (fe 39 b5 c7 e6 69 5b 44 00 00 00 00 00 00 00 00) we generated 2^{30} outputs. $\Pr(h_i = 0 \mid y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = 0) = 0.54685$. The experiment result confirms that for a weak key, there is strong correlation between the keystream and the LFSR.

4.2 Recovering the LFSR

The internal state of the LFSR could be recovered by exploiting the correlation between the keystream and the LFSR given in (11). The basic idea in this attack is similar to the fast correlation attack Algorithm A of Meier and Staffelbach [8]. But the details are completely different.

From (6), (11) and $\Pr(h_i = 0) = \frac{1}{2}$ (without knowing the keystream), we obtain

$$p_0 = \Pr(y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = 0 \mid h_i = 0) = 0.6183 \tag{12}$$

$$p_1 = \Pr(y_{i,16} \oplus y_{i+63,16} \oplus y_{i+127,16} = 0 \mid h_i = 1) = 0.5246. \tag{13}$$

Suppose N outputs are available. According to [8], the average amount of relations obtained for each h_i (by repeatly squaring $g^{2^5}(x)$) can be computed as

$$m = m(N, k, t) = \log_2(\frac{N}{2k})(t+1), \tag{14}$$

where $k = 127$ (the length of the LFSR), $t = 2$ (taps) for ABC v2. For the m relations related to h_i, if for v relations the XOR sum of those three $y_{j,16}$ bits are 0, then the probability that $h_i = 0$ is given as

$$p^*_{0,v} = \frac{p_0^v (1-p_0)^{m-v}}{p_0^v (1-p_0)^{m-v} + p_1^v (1-p_1)^{m-v}} \tag{15}$$

where p_0 and p_1 are defined in (12) and (13), respectively.

For h_i, the probability that for more than w relations the XOR sum of those three $y_{j,16}$ bits are 0 is given by

$$q_w = \sum_{i=w}^{m} \binom{m}{i} p_y^i (1-p_y)^{m-i} \tag{16}$$

where p_y is defined in (6).

With N outputs, the total number of h_i for which the XOR sum of those three $y_{j,16}$ bits are 0 for more than w relations is given as

$$s_w = N \cdot q_w \tag{17}$$

For each h_i satisfying that the XOR sum of those three $y_{j,16}$ bits are 0 for more than w relations, let $h_i = 0$. Among those s_w bits, the number of error bits is given by

$$e_w = \sum_{i=w}^{m} N \cdot \binom{m}{i} p_y^i (1-p_y)^{m-i} \cdot (1 - p^*_{0,i}) \tag{18}$$

For h_i, the probability that for less than w' relations the XOR sum of those three $y_{j,16}$ bits are 1 is given by

$$q'_{w'} = \sum_{i=0}^{w'} \binom{m}{i} p_y^i (1-p_y)^{m-i} \tag{19}$$

With N outputs, the total number of h_i for which the XOR sum of those three $y_{j,16}$ bits are 1 for less than w' relations is given as

$$s'_{w'} = N \cdot q'_{w'} \tag{20}$$

For each h_i satisfying that the XOR sum of those three $y_{j,16}$ bits are 0 for less than w' relations, let $h_i = 1$. Among those $s'_{w'}$ bits, the number of error bits is given by

$$e'_{w'} = \sum_{i=0}^{w'} N \cdot \binom{m}{i} p_y^i (1-p_y)^{m-i} \cdot p^*_{0,i} \tag{21}$$

The total number of guessed h_i bits is given by $s_w + s'_{w'}$. Among those $s_w + s'_{w'}$ guessed bits, the number of error bits is given by $e_w + e'_{w'}$. For $N = 2^{21}$, $m = 39$, $w = 34$, $w' = 9$, we get $s_w + s'_{w'} = 149.65$, and $e_w + e'_{w'} = 1.39$. Choose 127 bits from those 150 bits, there are about 1.2 bits are wrong. Each timing flipping one bit of those 127 bits, The LFSR could be recovered by solving 127 linear equations. The LFSR is recovered correctly with probability about $1.03 \times \frac{127}{212} = 0.617$. Thus the LFSR is successfully determined with 2^{20} outputs.

However, we found that there is big difference between our theoretical analysis and the experiment. In the experiment, we always get much more guessed h_i bits but also with more error bits (especially the error rate for $h_i = 1$ is too high). The reason is due to the noise introduced by each h_i bit to the other h_i bits. If one h_i is with value 0, then all the other bits that are in h_i's relations would tend to be guessed as 0. To compensate such noise, we need much more outputs to recover the LFSR, as illustrated in the following experiment.

Experiment 3. From the weak key (fe 39 b5 c7 e6 69 5b 44 00 00 00 00 00 00 00 00) with different IVs, we generated about 2^{31} outputs. We use 45 relations, and set $h_i = 0$ if more 43 relations are meet (the XOR sum of those three $y_{i,16}$ bits in a relation is zero). We determined 1548 h_i bits with 25 error bits.

Thus in order to find 127 bits of h_i, we need about $2 \cdot 127 \cdot 2^{\frac{45}{3}} + 2^{31} \cdot \frac{1548}{127} = 2^{27.5}$ outputs. Among those 127 h_i bits, there are about $127 \cdot \frac{25}{1548} = 2$ error bits in average. We obtain about 2^{15} internal states of the LFSR by solving 2^{15} sets of linear equations. The wrong internal states could be filtered out by generating a short binary sequence (less than 256 bits) and compare it with the keystream according to (9).

4.3 Recovering the components B and C

After recovering the LFSR, we proceed to recover component B. In the previous attack on ABC v1 [1], about 2^{77} operations are required to recover the components B and C. That complexity is too high. We give here a much simpler method to recover B and C with about $2^{33.3}$ operations.

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

$$x_i = (((x_{i-1} \oplus d_0) + d_1) \oplus d_2) + \overline{z}_2^i \bmod 2^{32}. \tag{22}$$

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

After knowing the LFSR, the value of each $\overline{z}_0^i$ can be determined, thus we know the value of each $S(x_i)$. For a weak key, the least significant bit of $S(x_i)$ is always 0. In average, the probability that $x_i = x_j$ is about 2^{-32}, and the probability that $S(x_i) = S(x_j)$ is about $2^{-32} + 2^{-31}$. Given 2^{18} outputs, there are about 8 cases that $x_i = x_j$ for $i \neq j$, and about 24 cases that $S(x_i) = S(x_j)$ for $i \neq j$. From those 24 cases that $S(x_i) = S(x_j)$ $(i \neq j)$, we choose 4 cases that $S(x_{i_u}) = S(x_{j_u})$ $(i_u \neq j_u$ and $0 \leq u < 4)$. The probability that $x_{i_u} = x_{j_u}$ for $0 \leq u < 4$ with probability $(\frac{8}{24})^4 = \frac{1}{81}$.

We know already the value of each $\overline{z}_2^i$. It is easy to 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 those four secret terms bit-by-bit from the least significant bit to the most significant bit. But those four most significant bits could not be determined exactly. (We mention here during this bit-by-bit approach, those four bits at each bit position may not be determined exactly, and further filtering is required in the consequent computations.) In average, we expect that solving those four equations gives about 8 possible values of x, d_0, d_1, and d_2. Also note that each set of those four equations holds true with proability $\frac{1}{81}$, we have about $81 \times 8 = 648$ possible values 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 needs to be repeated for 648 times since there are about 648 possible values of x, d_0, d_1, and d_2. The exact B and C could be determined by generating some outputs and comparing them to the original keystream.

4.4 Complexity of the attack

Recovering the LFSR requires about $2^{27.5}$ outputs. For each output we need to perform 135 XORs, 90 additions to guess the value of h_i. And we need about $\frac{127 \cdot 2 \cdot 2^{27.5}}{32}$ XOR operations (32-bit microprocessor) to generate the linear equations from the LFSR. We need to solve about 2^{15} sets of linear equations. About $127 \cdot 2 \cdot 5$ XOR operations are required to solve a set of 127 binary equations on the 32-bit microprocessor. Thus about $2^{35.4}$ operations are required to recover LFSR.

Recovering components B and C requires about 2^{18} outputs. To recover component B, solving 81 sets equations requires about $81 \cdot 32 \cdot 2^4 \cdot 2^{18} = 2^{33.3}$ operations. To recover component C, solving 648 set of equations (each set consists of 32 linear equations) requires about $648 \cdot 32 \cdot 2$ operations. Thus about $2^{33.3}$ operations are required to recover components B and C.

To recover the internal state of a weak key, $2^{27.5}$ outputs, and about $2^{35.4} + 2^{33.3} = 2^{35.7}$ operations are required.

There is tradeoff between the keystream length and the computation complexity. The keystream length could be reduced, then a number of LFSRs need to be tested with more computation.

5 Conclusion

Due to the large amount of weak keys and the severity of each weak key, ABC v2 is practically insecure.

Similar attack can be applied to break ABC v1. The complexity to break ABC v1 is almost the same as that to break ABC v2. ABC v2 is as insecure as ABC v1.

References

1. C. Berbain and H. Gilbert,"Cryptanalysis of ABC". Available at http://www.ecrypt.eu.org/stream/papersdir/048.pdf.
2. V. Anashin, A. Bogdanov and I. Kizhvatov, "ABC: A New Fast Flexible Stream Cipher". Available at http://www.ecrypt.eu.org/stream/ciphers/abc/abc.pdf.
3. V. Anashin, A. Bogdanov and I. Kizhvatov, "Increasing the ABC Stream Cipher Period". Available at http://www.ecrypt.eu.org/stream/papersdir/050.pdf.
4. V. Anashin, A. Bogdanov and I. Kizhvatov, "ABC Is Safe And Sound". Available at http://www.ecrypt.eu.org/stream/papersdir/079.pdf.
5. V. Anashin, A. Bogdanov and I. Kizhvatov, "Security and Implementation Properties of ABC v.2". *SASC 2006 - Stream Ciphers Revisited*, pp. 278-292, 2006.
6. V. Anashin, A. Bogdanov and I. Kizhvatov, "ABC: A New Fast Flexible Stream Cipher (Version 2)". Available at http://crypto.rsuh.ru/papers/abc-spec-v2.pdf
7. S. Khazaei, "Divide and Conquer Attack on ABC Stream Cipher". Available at http://www.ecrypt.eu.org/stream/papersdir/052.pdf.
8. W. Meier and O. Staffelbach, "Fast Correlation Attacks on Stream Ciphers". *Journal of Cryptology* 1(3), pp. 159-176, 1989.
9. S. Murphy and M.J.B. Robshaw. "Key-dependent S-boxes and differential cryptanalysis". *Designs, Codes, and Cryptography* 27(3), pp. 229-255, 2002.
10. S. Vaudenay, "On the Weak Keys of Blowfish", in *Fast Software Encryption (FSE'96)*, LNCS 1039, pp. 27-32, Springer-Verlag, 1996.