Cryptanalysis of ABC

Côme Berbain, Henri Gilbert

France Telecom R&D,
38-40 rue du Général Leclerc,
F-92794 Issy-les-Moulineaux, France.
{come.berbain, henri.gilbert}@francetelecom.com

Abstract. ABC ([1]) is a synchronous stream cipher submitted by Anashin, Bogdanov, Kizhvatov and Kumar to the ECRYPT call for Stream Cipher Primitives. In this paper, we present an attack against ABC which retrieves the complete internal state of the keystream generator after the key setup and before the IV setup (and thus provides an equivalent key). The attack requires 2^{95} computations and 2^{32} 32-bit keystream words.

1 Brief description of ABC

ABC is a synchronous stream cipher which uses a 128-bit key and a 128-bit IV and claims a security level of 2^{128}. ABC consists in three components:

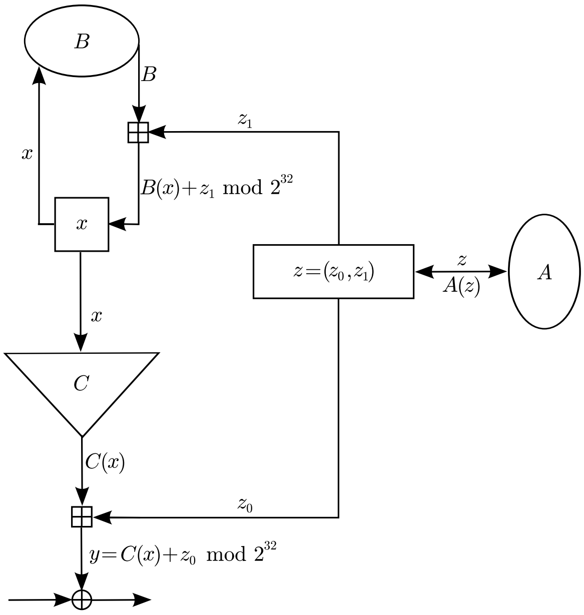

Fig. 1. ABC

Component A is a linear transformation on $GF(2)^{64}$. It uses a 64-bit Linear Feedback Shift Register which current state is represented by two 32-bit words z_0 and z_1.

At each clock the LFSR is updated as follows:

$$\zeta = z_1 \oplus (z_0 >> 1) \oplus (z_1 << 31), z_0 = z_1, z_1 = \zeta$$

Component B is the following single cycle T-function where d_0 and d_1 represent 32-bit key and IV dependent words.

$$B(x) = d_0 + 5(x \oplus d_1) \mod 2^{32}.$$

Component C is a mapping from $GF(2)^{32}$ to $GF(2)^{32}$ which definition involves key dependent constant 32-bit vectors e and e_i, $0 \leq i \leq 31$. It works as a selector depending on x: if bit i of x (denoted by $\delta_i(x)$) is equal to 1 then the vector e_i is added. The obtained result is rotated:

$$C(x) = \left(e + \sum_{i=0}^{31} e_i \delta_i(x) \mod 2^{32} \right) \ggg 16 \tag{1}$$

To generate a 32 bit word of keystream, component A is first applied on $z = (z_0, z_1)$, then component B is applied on x and the result is added modulo 2^{32} to z_1 to update x. Then component C is applied on x and the result is added modulo 2^{32} to z_0 to produce the keystream.

2 Description of the attack

We denote by z_0^0, z_1^0 and x^0 the values of z_0, z_1 and x after the IV setup and by z_0^t, z_1^t and x^t the values of z_0, z_1 and x after the t-th keystream word has been produced.

We first present a distinguisher on ABC, we can convert efficiently into an attack which recovers the whole internal state.

2.1 A distinguishing attack which recovers z

Expression (1) suggests that the output of component C can be expected to be strongly biased for nearly all values of the e_i constants, due to the fact that it is extremely unlikely to represent a one to one mapping. This was confirmed by computer experiments. Consequently, the deviation from the behavior one would observe if the component C output was uniformly distributed is easy to detect. Let us consider $N = 2^{32}$ consecutive C output words. If the distribution of these words were uniform, the average number of words reached k times would be:

$$N \binom{N}{k} \frac{1}{N^k} (1 - \frac{1}{N})^{N-k}$$

We can approximate the underlying binomial law by a Poisson law of parameter 1. Now for the actual output distribution of component C, the detected bias depends on the e_i. In exceptional cases, for instance if we had $e_i = 2^i$, $0 \leq i \leq 31$, one nearly uniform distribution would be obtained. However because the e_i are generated as 32 consecutive output words of ABC, this is extremely unlikely to happen and a rough heuristic argument suggests an order of magnitude of $2^{32}\left(1-\frac{1}{e}\right)$ for the average number of possible C output values[1].

To illustrate the bias, we give the frequency table of $N = 2^{32}$ output words of component C and compare it with the average values one would obtain from a Poisson distribution:

k	Poisson law	experimental result
0	1580030168	2283981230
1	1580030168	838584300
2	790015084	573760447
3	263338361	313263951
4	65834590	155322135
5	13166918	73033783
6	2194486	32879000
7	313498	14230621
8	39187	5953672
9	4354	2417630
10	435	953271
11	39	367907
12	3	139207
13		51327
14		18610
15		6682
16		2324
17		768
18		280
19		109
20		29
21		10
22		1
23		2

Let us now outline how to exploit the former bias to derive the state words z_0^0 and z_1^0 based on N consecutive keystream words s^t, $0 \leq t \leq N-1$.

One can notice that because of the shift operation by 1 in the update of the component A, bit 0 of z_0^0 does not have any influence on the keystream. So we only have to guess 63 bits to recover z_0^1 and z_1^1.

For each of the 2^{63} considered values of z_0^0 and z_1^0, we derive the associated sequence (z_0^t) and with the keystream sequence (s^t), we compute the sequence $(\tilde{c^t})$ given by:

$$\tilde{c^t} = (s^t - z_0^t) >>> 16$$

If we have made the right guess, then $\tilde{c^t}$ is the actual output of component C: $\tilde{c^t} = C(x^t)$. Otherwise the sequence can be expected to be more uniformly distributed.

[1] One can argue that we are in an intermediate situation between the one where 2^{32} C output values would result from choosing 2^{32} 32-bit words at random and the one which would occur if modulo 2^{32} addition were replaced by xor's in the expression of C. The number of distinct C output values would then be 2^r, where r is the dimension of the vector space spanned by the e_i.

We use the bias found in the output of component C to detect whether we have made the correct guess or not.

For each guess we can build the table T which entry $T[i]$ $(0 \leq i \leq 2^{32}-1)$ is defined as the number of occurrences of value i in the sequence $\tilde{c^t}$. Then we can compute the indicator:

$$\chi^2 = \sum_{i=0}^{2^{32}-1} (T[i]-1)^2$$

One can expect χ^2 to be much larger in the case of the unbalanced $(\tilde{c^t})$ distribution corresponding to the right guess than for the more uniform distribution corresponding to incorrect guesses. Therefore the maximum value of χ^2 can be expected to correspond to (z_0^0, z_1^0) or to $(z_0^0 \oplus \texttt{0x00000001}, z_1^0)$. In both cases we can compute (z_0^1, z_1^1). This was confirmed experimentally by applying the above test to the right value of (z_0^0, z_1^0) and to numerous incorrect assumptions. Moreover it was also confirmed experimentally on a reduced version of ABC, where 32-bit words of the actual ABC were replaced by 8-bit words, by doing the described exhaustive search for a very large number of keys.

This first part of the attack costs 2^{95} operations and requires 2^{32} 32-bit words of keystream. It is certainly possible to detect the bias with less than 2^{32} 32-bit words of keystream (and consequently to reduce the complexity of the attack) but then false alarm candidates should be taken into account.

2.2 Retrieving the complete internal state

Once we know the complete sequences (z_0^t) and (z_1^t), we need to retrieve the values of x^0, d_0, d_1, e and e_i, $0 \leq i \leq 31$ to get the complete internal state after key and IV loading.

First we retrieve the 16 least significant bits of x^0, d_0 and d_1. Because the designers want the T-function of component B to be single cycle, we know that $d_0 = 1 \mod 2$ and $d_1 = 0 \mod 4$. So we only have to guess 45 bits. For each guess we can derive the 16 least significant bits for all the values of x^t and when those bits are all zero (which happens with probability 2^{-16}) we have access to the value $e + \sum_{i=16}^{31} \delta_i(x)e_i \mod 2^{32}$ if we have made the right guess or to a less strongly biased value otherwise. Because we already have 2^{32} words of keystream, we obtain for each guess about 2^{16} words where the 16 least significant bits of x^t are all zero. In the case of an incorrect guess, we can expect these about 2^{16} words to be nearly all distinct, since they are drawn from the set of all possible $C(x)$ values, which size is much larger than 2^{16}, though strictly lower than 2^{32}. In the case of a correct guess, many multiple occurrences can be expected, since the about 2^{16} selected values are drawn from the set of possible $e + \sum_{i=16}^{31} \delta_i(x)e_i \mod 2^{32}$ values, which size is less or equal to 2^{16}.

We give the frequency table of 2^{32} output words of component C, which were obtained for an incorrect guess and a correct guess respectively:

k	incorrect guess	correct guess
0	4294901558	4294901844
1	65737	24062
2	1	12017
3		4004
4		1042
5		182
6		37
7		4
8		2

We can use the same estimator as in the first part of the attack (except that we do not consider all the not reached values) to distinguish the correct guess from the incorrect ones and we get the 16 least significant bits of x^0, d_0 and d_1. An even more simple but also effective test would consist in selecting the guess which maximizes the number of multiple occurrences. Recovering the 16 least significant bits of x^0, d_0 and d_1 costs 2^{77} operations and requires the same 2^{32} words of keystream as previously.

To recover the end of the internal state, we guess the end of x^0, d_0, d_1 and for each hypothesis, we derive the entire values of x^t. So we can write linear equations over e and e_i, $0 \leq i \leq 31$ and as soon as we have 33 independent equations we can solve the system and instantly check whether the guess is correct. So we have recovered the complete internal state in $2^{93} + 2^{77} + 2^{63}$ operations.

2.3 Study of the IV setup

Firstly we have noticed some flaws in the IV setup: for example one can change bit number 33 of the IV without changing the keystream, which represents an undesirable property. Moreover because bit 0 of z_0^t does not affect the later $(z_0^{t'}, z_1^{t'})$ and because of some bits of the internal state fixed to special values during the IV setup, several IV can lead to the same internal state after the IV setup. Consequently we can go backward from the internal state just after the IV setup to find one of the possible internal states after the key setup. This only costs 2^{32} operations.

3 Conclusion

We have presented a distinguishing attack on ABC which needs 2^{32} words of keystream, requires 2^{93} operations and retrieves the complete internal state after key and IV setup. This attack can be transformed into an attack that retrieves the internal state before the IV setup (and thus provides an equivalent key) for a negligible increase of the cost.

References

1. Vladimir Anashin, Andrey Bogdanov, Ilya Kizhvatov, and Sandeep Kumar. ABC: A new fast flexible stream cipher. ECRYPT Stream Cipher Project Report 2005/001, 2005. http://www.ecrypt.eu.org/stream.