ABC Is Safe And Sound

Vladimir Anashin, Andrey Bogdanov, Ilya Kizhvatov

Faculty of Information Security
Institute for Information Sciences and Security Technologies
Russian State University for the Humanities
Kirovogradskaya Str. 25/2, 117534, Moscow, Russia

{anashin,bogdanov,kizhvatov}@rsuh.ru

Abstract

ABC is a synchronous stream cipher submitted to eSTREAM [4]. In the previous paper [3] we suggested minor tweaks that increase the period and the secret state of ABC. Here we describe how these tweaks make ABC v.2 [5] resistant to certain attacks, including the ones presented in [6] and [7]. We also note that the paper [8] contains multiple errors and the distinguisher for ABC v.2 has complexity greater than the brute force attack.

1 Introduction

ABC is a synchronous stream cipher optimized for software applications. Techniques used in the ABC design enable one to build platform-independent ciphers offering fast and natural implementations in C or most other algorithmic languages. The version of ABC with a 128-bit key and 32-bit internal variables, offering 128-bit security, was submitted to eSTREAM [4].

Actually ABC is a family of stream ciphers. The flexibility of the ABC design enabled us in [3] to suggest the tweaks raising its keystream period from $2^{32} \cdot (2^{63} - 1)$ 32-bit words to $2^{32} \cdot (2^{127} - 1)$ 32-bit words while keeping all the other properties of ABC stated in [4], including guaranteed uniform distribution and high linear complexity of the keystream. For further details refer to the ABC v.2 specification [5].

This paper outlines how the tweaks make ABC v.2 resistant to certain distinguishing attacks. Another possible update is also discussed. We show that 'Theorem 1' from the paper [7] by S. Khazaei describing an attack on ABC is wrong.

We also note that the paper [8] by S. Khazaei and M. Kiaei contains multiple errors and does not present any distinguishing attack both on ABC v.1 and v.2. Moreover, our experiments indicate that the distinguisher for ABC v.2 has the complexity greater than that of a brute force attack.

2 ABC modifications

Here the tweaks in the ABC keystream generator making ABC v.2 out of ABC v.1 are briefly outlined. The adjusted setup procedures[1] described in [3, 5] are not discussed here. The notation is the same as in [4, 5, 3].

The keystream generator of ABC v.2 is illustrated in Figure 1.

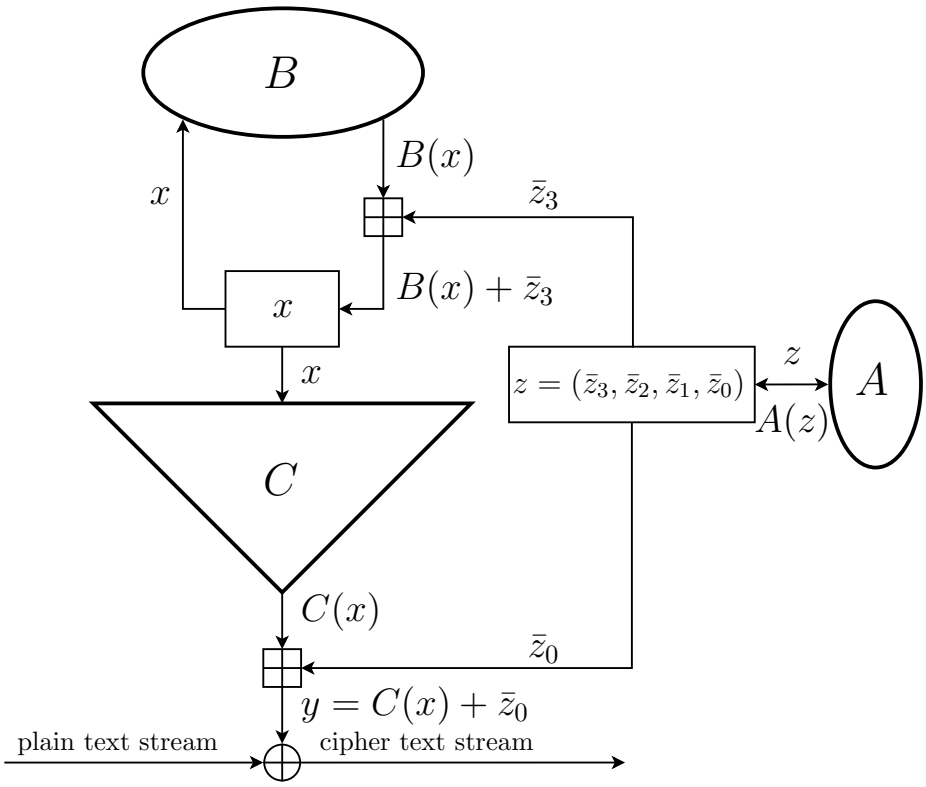

Figure 1: ABC v.2 keystream generator

The formal description of the routine is as follows:

ABC v.2 keystream generation routine

Input: $z \in \mathbb{Z}/2^{128}\mathbb{Z}$, $x \in \mathbb{Z}/2^{32}\mathbb{Z}$

$z \leftarrow A(z)$

$x \leftarrow \bar{z}_3 + B(x) \bmod 2^{32}$

$y \leftarrow \bar{z}_0 + C(x) \bmod 2^{32}$

Output: $z \in \mathbb{Z}/2^{128}\mathbb{Z}$, $x \in \mathbb{Z}/2^{32}\mathbb{Z}$, $y \in \mathbb{Z}/2^{32}\mathbb{Z}$

[1] Some inaccuracy mentioned in [6] was corrected in [3, 5].

In both versions of ABC A is a linear transformation of the vector space $\mathbb{V}_n = \mathrm{GF}(2)^n$ with a cycle of length $2^{n-1} - 1$ (where $n = 128$ for ABC v.2, and $n = 64$ for ABC v.1), B is a single cycle T-function on 32-bit words, and $C\colon \mathbb{Z}/2^{32}\mathbb{Z} \to \mathbb{Z}/2^{32}\mathbb{Z}$ is a filter function: C takes x as argument and produces y in the following way:

$$\begin{aligned} \zeta &= S(x), \\ y &= \zeta \ggg 16, \end{aligned} \tag{1}$$

where $S : \mathbb{Z}/2^{32}\mathbb{Z} \to \mathbb{Z}/2^{32}\mathbb{Z}$ is a mapping defined by

$$S(x) = e + \sum_{i=0}^{31} e_i \delta_i(x) \bmod 2^{32}, \tag{2}$$

$e_{31} \equiv 2^{16} \pmod{2^{17}}$. Coefficients $e, e_0, \ldots, e_{31} \in \mathbb{Z}/2^{32}\mathbb{Z}$ are obtained from the key during the initialization procedure.

The single cycle function B used in the ABC v.2 cipher can be specified through the following equation:

$$B(x) = ((x \mathbin{\mathsf{XOR}} d_0) + d_1) \mathbin{\mathsf{XOR}} d_2 \bmod 2^{32}, \tag{3}$$

where $d_0,\ d_1,\ d_2 \in \mathbb{Z}/2^{32}\mathbb{Z}$, $d_0 \equiv 0 \pmod 4$, $d_1 \equiv 1 \pmod 4$, $d_2 \equiv 0 \pmod 4$. In the non-modified ABC v.1 the function B was of the form

$$B(x) = d_0 + 5(x \mathbin{\mathsf{XOR}} d_1) \bmod 2^{32}, \tag{4}$$

with $d_0,\ d_1 \in \mathbb{Z}/2^{32}\mathbb{Z}$, $d_0 \equiv 1 \pmod 2$, $d_1 \equiv 0 \pmod 4$.

Under the restrictions mentioned above the following properties of the keystream produced by the ABC v.2 keystream generator are proven:

- The length P of the shortest period of the keystream sequence of 32-bit words is $P = 2^{32} \cdot (2^{127} - 1)$.
- The distribution of the keystream sequence of 32-bit words is uniform in the following sense: For each 32-bit word a the number $\mu(a)$ of occurrences of a at the period of the keystream satisfies the following inequality:
$$\left| \frac{\mu(a)}{P} - \frac{1}{2^{32}} \right| < \frac{1}{\sqrt{P}}.$$
- The linear complexity λ of the keystream bit sequence satisfies the inequality $2^{31} \cdot (2^{127} - 1) + 1 \geq \lambda \geq 2^{31} + 1$.

Proofs are based on the results presented in [2] and can be found in the updated ABC specification [5].

3 Attacks and remedies

In this section we describe some attacks that lead to recovering the internal state of the (non-modified) ABC v.1, and which are more efficient than a brute force attack. We describe the corresponding remedies as well.

3.1 Attacks

Suppose that one has a statistical test $\mathcal{T}$ (which is further called a *distinguisher*) that could tell the keystream sequence $Y = \{y_j \in \mathbb{Z}/2^{32}\mathbb{Z}\}_{j=0}^{\infty}$ from the intermediate sequence $C(X) = \{C(x_j) \in \mathbb{Z}/2^{32}\mathbb{Z}\}_{j=0}^{\infty}$, which is the output of the function C. Then trying different initial states $\hat{z}$ of the LFSR A and testing the sequences $\overline{C(X)}(z) = \{y_j - \bar{z}_{0,j}(\hat{z}) \bmod 2^{32}\}$ with $\mathcal{T}$, where $\bar{z}_{0,j}(\hat{z})$ is the the 32 low order bits of the output of the LFSR A at the j-th step, one finds $\bar{z}$. Under the assumption that $\mathcal{T}$ makes no errors in distinguishing, the computational cost of finding the true initial state of the LFSR is $(2^n - 1)T$ computations of AB, where T is the computational cost of testing one sequence with the test $\mathcal{T}$, and n is the length of the LFSR registry (i.e., $n = 63$ in non-modified ABC, and $n = 127$ in the modified one). After finding the true initial state $\hat{z}$ of the LFSR, one tests coefficients of the function B and then, solving the corresponding congruences modulo 2^{32} with respect to the unknown values of $e, e_0, \ldots, e_{31}$, totally recovers the internal state of the ABC.

This idea was quite clear to the designers from the very beginning of the development of ABC [2]. We did not find such a distinguisher with low computational cost T before the submission of our specification; we were not even convinced that such a distinguisher exists. However, later[3] we found some distinguishers with relatively low computational costs, which thus make ABC vulnerable to the described attack. Immediately after that we found a remedy and prepared a modified version [3] of ABC (that is, ABC v.2)[4].

We must point out here that attacks of this kind were recently described by Berbain and Hilbert in [6], and by Shahram Khazaei in [7]. By the time[5] the paper [6] was posted at the ECRYPT site our paper [3] had been submitted to ECRYPT and the specification [5] of ABC v.2 was already available at `http://crypto.rsuh.ru`.[6] So attacks of the kind described in

[2] This was clearly marked in the first draft of the ABC specification, which was submitted to ECRYPT. However, in the final version the corresponding subsection 'Some Special Attacks' containing the description of the idea of the attack was totally omitted in order to make our quite lengthy submission shorter.

[3] It was in June 2005.

[4] This modified version was submitted to ECRYPT in the very beginning of July 2005.

[5] That was on July 18-th, 2005.

[6] We are puzzled that the paper [6] was published at the ECRYPT site one day earlier than our paper [3], which was published at the ECRYPT site a week after it had been

[6] and [7] had been thwarted before they were published.

3.2 Remedies

We need only those remedies that do not worsen the important properties of ABC (long period, uniform distribution and high linear complexity of the keystream) and/or significantly reduce its performance. There are several such remedies; two of them are described below.

3.2.1 Remedy 1: Special coefficients

Since the coefficients $e, e_0, \ldots, e_{31}$ of the function S of (2) are produced in a pseudorandom way during the initialization stage, the probability the mapping C of (1) is bijective is too small; see Corollary 1 below for the exact value of that probability[7]. Hence, with high probability the distribution of the sequence $C(X) = \{C(x_j) \in \mathbb{Z}/2^{32}\mathbb{Z}\}_{j=0}^{\infty}$, is not uniform since the distribution of the sequence $X = \{x_j \in \mathbb{Z}/2^{32}\mathbb{Z}\}_{j=0}^{\infty}$, which is the output if the function B, is uniform[8]. At the same time, the distribution of the keystream sequence $Y = \{y_j \in \mathbb{Z}/2^{32}\mathbb{Z}\}_{j=0}^{\infty}$ is uniform (see Section 2). Hence, the distribution of the sequence $\overline{C(X)}(\hat{z})$ is not uniform in case of the right guess of the initial state $\hat{z}$ of the LFSR A, since the distribution of the output sequence of the LFSR A is uniform.

Thus, a distinguisher $\mathcal{T}$ just tests the uniformity of distribution of the sequence $\overline{C(X)}(z)$ for various z; in case the distribution is not uniform, the corresponding $z = \hat{z}$ is accepted as a true one. Note that distinguishers of [6] and of [7] are exactly of this sort.

To make sequences $\overline{C(X)}(\hat{z})$ indistinguishable one from another with respect to the test $\mathcal{T}$ for all the choices of $\hat{z}$ it suffices to choose coefficients of S in some special way to ensure that S is bijective.

Thus one needs criteria the coefficients should satisfy to make S bijective. In [7, Theorem 1] the following 'criterion' is stated: The function

$$\begin{array}{l} S(x) = e + \sum_{i=0}^{k-1} e_i \delta_i(x) \pmod{2^k}, \\ x, e, e_i \in \mathbb{Z}/2^k\mathbb{Z}, i = 0, \ldots, k-1, \end{array} \tag{5}$$

induces a permutation of the reside ring $\mathbb{Z}/2^k\mathbb{Z}$ iff for each non-empty subset $M \subset \{0, 1, \ldots, k-1\}$

$$\sum_{i \in M} e_i \not\equiv 0 \pmod{2^k}.$$

submitted.

[7] Note that in both [6] and [7] the authors use in their arguments only very rough estimates of this probability: The estimate of [6] is just an empirical conjecture, the one of [7] is based on the erroneous 'Theorem 1' of [7].

[8] The latter statement follows from the results stated [2] and can be found in [5].

However, it could be immediately shown that the above 'criterion' (as well as the whole 'Theorem' 1 of [7]) *are merely wrong*: Take $k = 3$, put $e_0 = 1$, $e_1 = 2$, $e_2 = 3$ and verify that the mapping $x \mapsto \delta_0(x) + 2 \cdot \delta_1(x) + 3 \cdot \delta_2(x)$ is *not* a permutation of the residue ring modulo 8.

The right criterion reads the following.

Theorem 1 [9] *The function* (5) *induces a permutation on the ring* $\mathbb{Z}/2^k\mathbb{Z}$ *if and only if*

$$e_{j_0} \equiv 1 \pmod 2, \quad e_{j_1} \equiv 2 \pmod 4, \ldots, e_{j_{k-1}} \equiv 2^{k-1} \pmod{2^k},$$

for some permutation $(j_0, j_1, \ldots, j_{k-1})$ *of* $(0, 1, \ldots, k-1)$.

Corollary 1 *There are exactly* $k! \cdot 2^{\frac{k(k+1)}{2}}$ *permutations among all* $2^{k(k+1)}$ *pairwise distinct transformations of form* (5) *of the residue ring* $\mathbb{Z}/2^k\mathbb{Z}$. *Hence, the probability that* S *is a permutation is* $k! \cdot 2^{-\frac{k(k+1)}{2}}$.

In other words, S of (2) is a permutation iff $e_0, \ldots, e_{31}$ could be reordered so that $e_i = 2^i \cdot e'_i$, where e'_i are odd, $i = 0, 1, \ldots, 31$. Note that our condition $e_{31} \equiv 2^{16} \pmod{2^{17}}$ is in a certain sense a 'remnant' of our Theorem 1.

Thus, just to avoid the kind of attack described in [6] and [7] it is sufficient *only to make minor modifications to the initialization procedure* so that one of $e_0, \ldots, e_{31}$ always has 1 in the least significant bit position, another has 01 in its two rightmost bit positions, a further one has 001 in the three rightmost bit positions, etc. *The modification does not change the ABC keystream generation routine at all*, leaving both the performance and other properties (period length, uniform distribution, linear complexity) unchanged.

So the assumption of [7] by S. Khazaei that 'The designers of ABC have not neither evaluated C function theoretically nor using statistical simulations and just have designed C function to provide a provably minimum period for its output sequences' is just not true. We certainly could make S (whence, C) balanced (that is, bijective) at the very first stage of the ABC design procedure: We had mathematical tools to construct balanced mappings. Note that these tools have been developed long before[10] (and are more effective) than the ones of paper [9], which the author of [7] mentions. However, the *arbitrary* choice of coefficients in accordance with our Theorem 1 *could lead to some very effective algebraic attacks* unless some special countermeasures are undertaken. Yet so far the only countermeasures that either decrease performance or make the initialisation procedure too complicated are known to us.

[9] Theorem 1 follows immediately from a (more than 10 year old) result of one of us, see [1, Proposition 4.8]. Also, it could be easily deduced from the older result of DeBruijn, see [10, Section 4.1, Exercise 30]. Of course, it is not difficult to prove this theorem directly.

[10] See e.g. the bibliography in [4] and [5]

Moreover, *making S bijective per se does not make ABC resistant to attacks based on other distinguishers that exploit statistical properties other than the distribution of 32-tuples.* So we consider the following modification as a much better one.

3.2.2 Remedy 2: Long LFSR

This solution is based on the usage of LFSR with period $2^{127}-1$ instead of the LFSR with period $2^{63}-1$ in the keystream generator, see Figure 1. In spite of the fact that it implies modification of the keystream routine (we had also to modify the B function to compensate some speed reduction), the solution makes the ABC resistant to *all possible* attacks of the described kind *independently* of concrete distinguishers $\mathcal{T}$ they are based on: The computational cost is then $(2^{127}-1)\cdot T \approx 2^{127}\cdot T \geq 2^{128}$, since we could hardly imagine a distinguisher with computational cost $T=1$, under every reasonable definition of what is computational cost. Thus, every attack of the described type becomes less effective than a brute force attack. As a bonus we obtain certain increase of security of the function B, since some extra bits of security are added (cf. (3) and (4)).

4 A note on distinguishers

In their paper [8] Shahram Khazaei and Mohammad Kiaei claimed that there is a distingusher on both versions of ABC with the complexity of about 2^{32}. The claim was supported by the empirical results of computer experiments with a set of reduced versions of ABC.

The analysis of the C code[11] used for performing the computer experiments showed that the authors of [8] have made multiple conceptual errors and implementation bugs:

1. Incorrect calculation of the function A
   (a) using a linear congruential generator `rand()` provided by standard C++ library instead of the LFSR;
   (b) using non-primitive polynomials in attempt to emulate LFSR A.
2. Incorrect calculation of the function B
   (a) replacing single-cycle function B by a simple counter $f(x)=x+1$;
   (b) lack of counter-dependency, i. e. no addition of part of LFSR state while calculating the next value of x;

[11] provided by Shahram Khazaei and available at http://crypto.rsuh.ru

(c) therefore, the sequence at the input of the function C has a period of 2^m m-bit words for any λ, i. e. the authors use λ consequential identical subsequences $\{1, 2, ..., 2^m - 1\}$ to "emulate" the behavior of B while analyzing the keystream of $N = 2^m\lambda$ words.

3. Incorrect calculation of the output of the ABC cipher

   (a) 4 consequential outputs of C are identical, as the single input value of C is consequentially used 4 times to calculate C output;

   (b) lack of rotation of C output.

Hence the paper [8] presents a distinguisher for some construction with catastrophical cryptographic properties which has nothing to do with ABC v.1 or v.2. Shahram Khazaei and Mohamad Kiaei disproved their claims against ABC at the eSTREAM forum[12].

Our computer experiments with the accurately implemented reduced versions of ABC v.2[13] modelled the same distinguishing algorithm and employed good truly random sequences. The latter were obtained from a physical source of randomness. The experiments showed that distinguishing is completely impossible with time and data compexities of about 2^m for ABC with m-bit words.

Moreover, the results of extensive simulations on a high-performance computing cluster[14] indicate that distinguishing of m-bit ABC for $m \geqslant 12$ with a negligible error probability cannot be performed with time and data complexities less or equal to 2^{4m}, which is the size of the key space. That is, distinguishing of full-scaled ABC v.2 from the random sequence requires significantly more resources than the 2^{128} brute force key search, which makes the distinguishers nonsensical and totally impractical.

5 Conclusion

In this paper we have shown that the simple way to increase the period of the ABC stream cipher, which was described in [3], has already totally eliminated the attacks described in [6] and [7] prior to their publication. Also we have studied another way of thwarting these attacks and explained why we have preferred the way of [3]. Finally, it was noted that results stated in [8] are erroneous and distinguishing ABC v.2 from the random sequence is actually harder than the exhaustive key search.

[12] see http://www.ecrypt.eu.org/stream/phorum

[13] also available at http://crypto.rsuh.ru

[14] the cluster of Research Computing Center of Moscow State University, see http://srcc.msu.su/nivc/index_engl.htm

References

[1] Vladimir Anashin. Uniformly distributed sequences over p-adic integers. In I. Shparlinsky A. J. van der Poorten and H. G. Zimmer, editors, *Number theoretic and algebraic methods in computer science. Proceedings of the Int'l Conference (Moscow, June–July, 1993)*, pages 1–18. World Scientific, 1995. 6

[2] Vladimir Anashin. Pseudorandom number generation by p-adic ergodic transformations, 2004. Available from http://arXiv.org/abs/cs.CR/0401030. 3, 5

[3] Vladimir Anashin, Andrey Bogdanov, and Ilya Kizhvatov. Increasing the ABC stream cipher period. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/050, 2005. http://www.ecrypt.eu.org/stream. 1, 2, 4, 8

[4] Vladimir Anashin, Andrey Bogdanov, Ilya Kizhvatov, and Sandeep Kumar. ABC: A new fast flexible stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. http://www.ecrypt.eu.org/stream. 1, 2, 6

[5] Vladimir Anashin, Andrey Bogdanov, Ilya Kizhvatov, and Sandeep Kumar. ABC: A new fast flexible stream cipher. Version 2, 2005. http://crypto.rsuh.ru/papers/abc-spec-v2.pdf. 1, 2, 3, 4, 5, 6

[6] Côme Berbain and Henry Gilbert. Cryptanalysis of ABC. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/048, 2005. http://www.ecrypt.eu.org/stream. 1, 2, 4, 5, 6, 8

[7] Shahram Khazaei. Divide and conquer attack on ABC stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/052, 2005. http://www.ecrypt.eu.org/stream. 1, 4, 5, 6, 8

[8] Shahram Khazaei and Mohammad Kiaei. Distinguishing attack on the ABC v.1 and v.2. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/061, 2005. http://www.ecrypt.eu.org/stream. 1, 7, 8

[9] Alexander Klimov and Adi Shamir. A new class of invertible mappings. In B.S.Kaliski Jr.et al., editor, *Cryptographic Hardware and Embedded Systems 2002*, volume 2523 of *Lect. Notes in Comp. Sci*, pages 470–483. Springer-Verlag, 2003. 6

[10] Donald Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, third edition, 1998. 6