Increasing the ABC Stream Cipher Period

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 ECRYPT Stream Cipher Project [2]. The changes proposed in this paper increase ABC keystream period to $2^{32} \cdot (2^{127} - 1)$ words and the size of ABC internal state to 1287 bits while keeping all the guaranteed properties of the keystream without a considerable overhead.

1 Introduction

ABC is a synchronous stream cipher optimized for software applications. The version of ABC with 128-bit key and 32-bit internal variables, offering 128-bit security, was submitted to ECRYPT Stream Cipher Project [2].

The flexibility of the ABC design provides a way to raise the period of ABC keystream without a considerable overhead. This is achieved by increasing the length of LFSR A. The internal state of single cycle function B is increased. The key and IV setup procedures are adjusted appropriately.

The changes we suggest raise the period of the ABC keystream from $2^{32} \cdot (2^{63} - 1)$ words to $2^{32} \cdot (2^{127} - 1)$ words while keeping all other properties of the ABC stated in [2], including guaranteed uniform distribution of the keystream and its high linear complexity, exceeding 2^{31}. Moreover, the ABC internal state is enlarged to 1287 bits. The performance of the updated ABC on an Intel Pentium 4 processor is 3.7 clocks per byte.

This paper is a brief summary of changes. For further details refer to the updated ABC specification [3].

2 ABC modifications

Here the changes to LFSR A and single cycle function B are outlined in detail. Then the inclusion of these changes into the ABC keystream generator

is depicted and adjusted ABC setup procedures are described. Throughout this section the notation from [2] is used with the following changes:

z is a 128-bit integer value that allows several equivalent representations:

$$z = (z_{127}, \ldots, z_0) = \sum_{i=0}^{127} z_i 2^i \in \mathbb{Z}/2^{128}\mathbb{Z}, \;\; z_i \in \{0,1\}, \;\; i = 0, \ldots, 127;$$
$$z \in \mathbb{V}_{128} = \mathrm{GF}(2)^{128};$$
$$z = (\bar{z}_3, \bar{z}_2, \bar{z}_1, \bar{z}_0) \in (\mathbb{Z}/2^{32}\mathbb{Z})^4, \quad \bar{z}_3, \bar{z}_2, \bar{z}_1, \bar{z}_0 \in \mathbb{Z}/2^{32}\mathbb{Z}.$$

$d_j = (d_{j,31}, \ldots, d_{j,0}) \in \mathbb{Z}/2^{32}\mathbb{Z}$, $j = 0, 1, 2$, is a 32-bit integer value.

$\delta_i : \mathbb{Z}/2^{128}\mathbb{Z} \to \{0,1\}$, $\delta_i(z) = z_i, \;\; i = 0, \ldots, 127$, is an i-th bit selection.

2.1 A primitive

A is modified to be a linear transformation of the vector space $\mathbb{V}_{128} = \mathrm{GF}(2)^{128}$, $z = A(z)$, defined as in [4] by a word-oriented LFSR of length 128 with characteristic polynomial $\phi(\theta) = \psi(\theta)\theta$, where $\psi(\theta) = \theta^{127} + \theta^{63} + 1$ is primitive. The next 32 bits are produced at once as follows:

$$\begin{aligned} \zeta &\leftarrow \bar{z}_2 \,\mathsf{XOR}\, (\bar{z}_1 \ll 31) \,\mathsf{XOR}\, (\bar{z}_0 \gg 1) \bmod 2^{32}, \\ \bar{z}_0 &\leftarrow \bar{z}_1, \\ \bar{z}_1 &\leftarrow \bar{z}_2, \\ \bar{z}_2 &\leftarrow \bar{z}_3, \\ \bar{z}_3 &\leftarrow \zeta. \end{aligned} \tag{1}$$

The cycle length of this LFSR is $2^{127} - 1$. As the cycle length becomes 1 in case the initial state $z = (\bar{z}_3, \bar{z}_2, \bar{z}_1, \bar{z}_0)$ is either $(0,0,0,0)$ or $(0,0,0,1)$, $\delta_1(z)$ is forced to 1 in the ABC key setup and IV setup procedures, thus reducing the secret state of A primitive to 126 bits.

Two outputs from A are obtained, $\bar{z}_3$ for the state transition procedure and $\bar{z}_0$ for updating the output function (see Subsection 2.3).

2.2 B primitive

The single cycle function B is modified as follows:

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

where $d_0 \equiv 0 \pmod 4$, $d_1 \equiv 1 \pmod 4$, $d_2 \equiv 0 \pmod 4$. These restrictions guarantee that B is a single cycle map modulo 2^{32} (the proof can be found in [1, Proposition 3.15] and in [5]).

The form of $B(x)$ is determined at the initialization stage through setting the 90 bits of d_0, d_1 and d_2 in a key- and IV-dependent manner (see Subsection 2.4). Along with 32 bits of x this leads to a larger B secret state, encompassing 122 bits.

2.3 ABC keystream generator

The keystream generation routine of ABC is adapted to the changes in A and B primitives as follows.

ABC KEYSTREAM GENERATOR

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

The following properties of the keystream produced by modified ABC keystream generator are proved:

- 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, that is, 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 [1] and can be found in the updated ABC specification [3].

2.4 Key expansion and nonce setup

The change in the length of the LFSR A and in the number of secret bits of B is adopted in ABC setup procedures by making 248 bits of z, x, d_0, d_1 and d_2 key- and IV-dependent. The number of warm-up iterations with feedback is increased to 8 in order to touch each of x, d_0, d_1, d_2, $\bar{z}_0$, $\bar{z}_1$, $\bar{z}_2$ and $\bar{z}_3$. The modified setup routine is outlined below.

ABC SETUP ROUTINE

KEY EXPANSION

INPUT: $k = (\bar{k}_3, \bar{k}_2, \bar{k}_1, \bar{k}_0)$, $z = (\bar{z}_3, \bar{z}_2, \bar{z}_1, \bar{z}_0), x, d_0, d_1, d_2, e, \{e_i\}_{i=0}^{31}$, $\{\mathfrak{T}_i\}_{i=0}^{t-1}$

TEMPORARY VARIABLES: $z' = (\bar{z}'_3, \bar{z}'_2, \bar{z}'_1, \bar{z}'_0), x', d'_0, d'_1, d'_2, i, \zeta$

Initialization:

$d'_0 \leftarrow \bar{k}_3 \;\mathsf{AND}\; \underbrace{1\ldots1}_{30}00_2;$

$d'_1 \leftarrow (\bar{k}_2 \;\mathsf{AND}\; \underbrace{1\ldots1}_{30}00_2) \;\mathsf{OR}\; 1;$

$d'_2 \leftarrow \bar{k}_1 \;\mathsf{AND}\; \underbrace{1\ldots1}_{30}00_2;$

$x' \leftarrow \bar{k}_0;$

$\bar{z}'_0 \leftarrow (\bar{k}_0 \ggg 16) \;\mathsf{OR}\; 2;$

$\bar{z}'_1 \leftarrow \bar{k}_1 \ggg 16;$

$\bar{z}'_2 \leftarrow \bar{k}_2 \ggg 16;$

$\bar{z}'_3 \leftarrow \bar{k}_3 \ggg 16;$

Initial state warm-up with feedback:

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$x' \leftarrow x' \;\mathsf{XOR}\; \zeta;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$d'_0 \leftarrow (d'_0 \;\mathsf{XOR}\; \zeta) \;\mathsf{AND}\; \underbrace{1\ldots1}_{30}00_2;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$d'_1 \leftarrow ((d'_1 \;\mathsf{XOR}\; \zeta) \;\mathsf{AND}\; \underbrace{1\ldots1}_{30}00_2) \;\mathsf{OR}\; 1;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$d'_2 \leftarrow (d'_2 \;\mathsf{XOR}\; \zeta) \;\mathsf{AND}\; \underbrace{1\ldots1}_{30}00_2;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$z'_2 \leftarrow z'_2 \;\mathsf{XOR}\; \zeta;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$z'_2 \leftarrow z'_2 \;\mathsf{XOR}\; \zeta;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$z'_2 \leftarrow z'_2 \;\mathsf{XOR}\; \zeta;$

$\zeta \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1});$

$z'_2 \leftarrow z'_2 \;\mathsf{XOR}\; \zeta;$

$z'_0 \leftarrow z'_0 \;\mathsf{OR}\; 2;$

Main state filling:

$e \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$;

for i from 0 to 30 do

$e_i \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$;

end for

$e_{31} \leftarrow (g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$ AND $\underbrace{1\ldots1}_{16}\underbrace{0\ldots0}_{16}{}_2)$ OR $\underbrace{0\ldots0}_{15}1\underbrace{0\ldots0}_{16}{}_2$;

$d_0 \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$ AND $\underbrace{1\ldots1}_{30}00_2$;

$d_1 \leftarrow (g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$ AND $\underbrace{1\ldots1}_{30}00_2)$ OR 1;

$d_2 \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$ AND $\underbrace{1\ldots1}_{30}00_2$;

$x \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$;

$\bar{z}_0 \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$ OR 2;

$\bar{z}_1 \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$;

$\bar{z}_2 \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$;

$\bar{z}_3 \leftarrow g(z', x', d'_0, d'_1, d'_2, \{\mathfrak{T}_i\}_{i=0}^{t-1})$;

OUTPUT: $z, x, d_0, d_1, d_2, e, \{e_i\}_{i=0}^{31}$

OPTIMIZATION TABLES PRECOMPUTATION

INPUT: $w, t, e, \{e_i\}_{i=0}^{31}$

TEMPORARY VARIABLES: i, j, l

for i from 1 to $t-1$ do

for j from 0 to $2^w - 1$ do

$T_i[j] = 0$

for l from 0 to $w-1$ do

$T_i[j] \leftarrow T_i[j] + \delta_l(j) \cdot e_{w \cdot i + l}$;

for j from 0 to $2^w - 1$ do

$T_0[j] = e$;

for l from 0 to $w-1$ do

$T_0[j] \leftarrow T_0[j] + \delta_l(j) \cdot e_l$;

OUTPUT: $T_0, \ldots, T_{t-1}$

IV SETUP

INPUT: $iv = (\bar{iv}_3, \bar{iv}_2, \bar{iv}_1, \bar{iv}_0)$,$\{T_i\}_{i=0}^{t-1}$

TEMPORARY VARIABLES: ζ

IV application:

$d_0 \leftarrow (d_0 \,\mathsf{XOR}\, \bar{iv}_3) \,\mathsf{AND}\, \underbrace{1\ldots1}_{30}00_2;$

$d_1 \leftarrow ((d_1 \,\mathsf{XOR}\, \bar{iv}_2) \,\mathsf{AND}\, \underbrace{1\ldots1}_{30}00_2) \,\mathsf{OR}\, 1;$

$d_2 \leftarrow (d_2 \,\mathsf{XOR}\, \bar{iv}_1) \,\mathsf{AND}\, \underbrace{1\ldots1}_{30}00_2;$

$x \leftarrow x \,\mathsf{XOR}\, \bar{iv}_0;$

$\bar{z}_0 \leftarrow (\bar{z}_0 \,\mathsf{XOR} (\bar{iv}_0 \ggg 16)) \,\mathsf{OR}\, 2;$

$\bar{z}_1 \leftarrow \bar{z}_1 \,\mathsf{XOR} (\bar{iv}_1 \ggg 16);$

$\bar{z}_2 \leftarrow \bar{z}_2 \,\mathsf{XOR} (\bar{iv}_2 \ggg 16);$

$\bar{z}_3 \leftarrow \bar{z}_3 \,\mathsf{XOR} (\bar{iv}_3 \ggg 16);$

Warm-up with feedback:

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$x \leftarrow x \,\mathsf{XOR}\, \zeta;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$d_0 \leftarrow (d_0 \,\mathsf{XOR}\, \zeta) \,\mathsf{AND}\, \underbrace{1\ldots1}_{30}00_2;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$d_1 \leftarrow ((d_1 \,\mathsf{XOR}\, \zeta) \,\mathsf{AND}\, \underbrace{1\ldots1}_{30}00_2) \,\mathsf{OR}\, 1;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$d_2 \leftarrow (d_2 \,\mathsf{XOR}\, \zeta) \,\mathsf{AND}\, \underbrace{1\ldots1}_{30}00_2;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$z_2 \leftarrow z_2 \,\mathsf{XOR}\, \zeta;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$z_2 \leftarrow z_2 \,\mathsf{XOR}\, \zeta;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$z_2 \leftarrow z_2 \,\mathsf{XOR}\, \zeta;$

$\zeta \leftarrow g(z, x, d_0, d_1, d_2, \{T_i\}_{i=0}^{t-1});$

$z_2 \leftarrow z_2 \,\mathsf{XOR}\, \zeta;$

$z_0 \leftarrow z_0 \,\mathsf{OR}\, 2;$

OUTPUT: $z = (\bar{z}_3, \bar{z}_2, \bar{z}_1, \bar{z}_0), x, d_0, d_1, d_2$

3 Performance

Throughput values for modified ABC reference implementation can be found in Table 1. Costs of modified setup routines are given in Table 2. The tables contain results for different optimization window sizes obtained on a 3.2 GHz Intel Pentium 4 Northwood processor under the same measurement conditions as described in [2].

Table 1: ABC throughput for Intel Pentium 4

w	Speed, Gbps	Cycles per byte	Memory for lookup tables, bytes
2	2.19	11.68	256
4	3.36	7.65	512
8	6.91	3.70	4096

Table 2: Cost of ABC setup routines in processor cycles for Intel Pentium 4

w	Key setup with precomputation	IV setup	Table precomputation	Key setup without precomputation
2	2056	372	301	1755
4	4792	259	3632	1160
8	90519	207	89758	761

The results indicate that the proposed changes lead to a speedup in some cases and cause some overhead in the others.

4 Conclusion

In this paper we have presented a simple way to increase the period of the ABC stream cipher. The suggested changes illustrate the natural flexibility of the ABC design. The techniques developed in [1] that are used for proving the properties of the ABC keystream like period length, uniform distribution and high linear complexity provide easy scaling of some of these properties without worsening the others. We recommend using the updated version of ABC [3] instead of the original one [2].

References

[1] Vladimir Anashin. Pseudorandom number generation by p-adic ergodic transformations, 2004. Available from http://arXiv.org/abs/cs.CR/0401030. 2, 3, 7

[2] 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. 1, 2, 6, 7

[3] 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, 3, 7

[4] C. Carroll, A. Chan, and M. Zhang. The software-oriented stream cipher SSC2. In *Proceedings of Fast Software Encryption – FSE 2000, LNCS*, volume 1978, 2001. 2

[5] L. Kotomina. Fast nonlinear congruential generators. Diploma Thesis, Russian State University for the Humanities, Moscow, 1999. 2