UDC 004.056.55

"STRUMOK" STREAM CIPHER

Ivan Gorbenko[1,2], Alexandr Kuznetsov[1,2], Yuriy Gorbenko[2], Anton Alekseychuk[2,3], Vlad Tymchenko[1]

[1] V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine
[2] JSC "Institute of Information Technologies", 12 Bakulin St., Kharkiv, 61166, Ukraine
[3] National Technical University of Ukraine "Kyiv Polytechnic Institute", 37 Prospect Peremogy, Kiev 03056, Ukraine

gorbenkoi@iit.kharkov.ua, kuznetsov@karazin.ua, gorbenkou@iit.kharkov.ua, alex-dtn@ukr.net, tvlad.tyma@gmail.com

Reviewer: Nikołay Karpinskiy, Doctor of Sciences (Eng.), Full Prof., University of Bielsko-Biala, Willowa 2, Bielsko-Biala, 43-309, Poland.
mkarpinski@ath.bielsko.pl

Received on February 2018

Анотація: *This work presents the main developing results of a new keystream generator, which named "Strumok", and offered as a candidate for the national symmetric encryption standard of Ukraine. "Strumok" is built on SNOW 2.0-likes schema of the summation generator. Increased secret key length and the initialization vector allow using reliably the stream cipher even taking into account quantum cryptographic analysis methods. Unlike SNOW 2.0, Strumok is designed for use in more powerful 64-bit computing systems. The conducted comparative tests have shown that the "Strumok" on 32-bit computing systems also shows good performance results. There are basic transformation and individual results from the cipher performance research, here is it is shown the generator, which is capable of forming a keystream at speeds exceed of 10 Gbit per sec.*

Keywords: *encryption; stream cipher; synchronous keystream generator; pseudorandom sequence.*

1 Introduction

An important cryptographic information protection mechanism is a stream symmetric encryption [1,2]. It uses for providing services of information confidentiality and integrity (*as an additional service*) during data processing in information, telecommunication and information-telecommunication systems [1].

Recent years the requirements for modern streaming encryption algorithms have significantly increased [3-5]: on the one side, it is necessary to provide a high-speed cryptographic transformation (*more than 10 Gbit per sec*), on the other side, it have to withstand effectively for the latest methods of cryptographic analysis including methods using of quantum computation. Therefore, the development, research and a gradual introduction of new stream symmetric encryption methods is an actual and extremely important the national level scientific and applied problem. Usually a streaming encryption operation is a bitwise XOR operation between a keystream and a message. ISO/IEC 18033-4:2011 describes the output functions for different stream ciphers and certain pseudorandom numbers generators, which used for restricted information protecting, in particular to ensure the information confidentially when processing is going on [6]. The work objective is to present the main results of new pseudorandom number generator (a keystream) developing, which named "Strumok", and is proposed as a candidate for the national symmetric encryption standard of Ukraine [7-18]. The "Strumok" generator provides a high forming keystream rates (*over 10 Gbit per sec*) that exceeds most known algorithms and is suitable for using in a post-quantum environment.

2 General Parameters

In basics of Strumok algorithm lies the classical summing generator schema [1,2,6,7], which similar to the SNOW2.0 generator as defined in ISO/IEC 18033-4:2011 [6]. The Strumok algorithm uses the 256-bit initialization vector *IV* and the 256 or 512-bit secret key *K* and provides high and ultra-high resistance, taking into the possible using account of quantum cryptographic analysis. The crypto algorithm is oriented on 64-bit computing systems, so the word size is set to 64 bits.

The main generator components are the linear feedback shift register (*LFSR*) and the finite-state machine (*FSM*), in which performed a non-linear transformation. The input data are used for initializing the variable state $S_i = (s^{(i)}, r^{(i)})$, $i \geq 0$, which consists of eighteen 64-bit blocks (words) that has two components: 16 variables $s^{(i)}$ – words of the *LFSR*: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, ..., s_0^{(i)})$; two words of the FSM $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$. On the output, it gets a keystream, which formed with 64-bit words Z_i.

A schematic operation representation of keystream Strumok generator is in an arbitrary moment of the time *i*, it is represented in Fig. 1.

The feedback taps in the *LFSR* are constructed over the finite field $GF(2^{64})$ by a primitive polynomial: $f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha$, where α is the root over the finite field $GF(2^8)$ of the primary polynomial:

$$g(z) = z^8 + \beta^{170}z^7 + \beta^{166}z^6 + \beta^2 z^5 + \beta^{224}z^4 + \beta^{70}z^3 + \beta^2.$$

In turn, the finite field $GF(2^8)$ is constructed over a $GF(2)$ field by a primitive polynomial:

$$p(y) = y^8 + y^4 + y^3 + y^2 + 1,$$

and the polynomial coefficients $g(z)$ are submitted through degree of a primitive element β of the finite field $GF(2^8)$ that is β – the root of a polynomial $p(y)$ Thus, we have an extension:

$$GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024}),$$

where: the finite field $GF(2^{1024})$ is given by output of feedback the *LFSR* as a quotient ring $GF(2^{64})[x]/(f(x))$; the finite field $GF(2^{64})$ is given as a quotient ring $GF(2^8)[z]/(g(z))$; the finite field $GF(2^8)$ is given as a quotient ring $GF(2)[y]/(p(y))$. Therefore, output sequence period of the *LFSR* is maximal and equals of $2^{1024} - 1$.

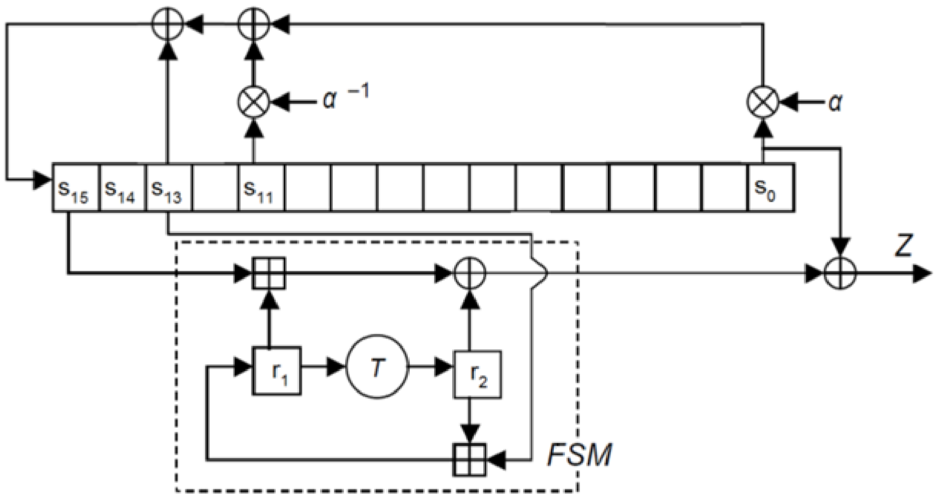

Fig.1 - Conceptual scheme of the *"Strumok"* keystream generator in generating gamma cipher mode (a keystream)

Structurally in the Strumok algorithm, it can distinguish three main functions:

- The initializing function *Init* that takes the key *K* (256 or 512 bits) and initialization vector *IV* (256 bits) as the input data, and produces the initial value of the variable state $S_0 = (s^{(0)}, r^{(0)})$;
- The next-state function *Next*, which takes the variable state S_i into the input and produces the next value of the variable state $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$. The *Next* function can be in two modes,

which depending on how the iteration performed – as part of the implementation or as normal mode of generating output data.

- The keystream function *Strm* that takes the variable state S_i into the input and produces at the output 64-bit a keystream Z_i.

3 Initialization function Init

The initializing function of the internal state is described as follows.

Input: 256 or 512-bit a key *K*, 256-bit an initialization vector *IV*.

Output: the initial value of the variable state $S_0 = (s^{(0)}, r^{(0)})$.

The key for the stream cipher version Strumok-256 can represented as four 64-bit words $K = (K_3, K_2, K_1, K_0)$ and for the 512-bit a key – in the form $K = (K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0)$, where K_3 and K_7s respectively 256 and 512 bits, the most significant words, and K_0 – the least significant.

The initialization vector can represented in the form of four 64-bit words $IV = (IV_3, IV_2, IV_1, IV_0)$, where IV_3 and IV_0 is respectively the most and the least significant words.

1. It is entered the key value in 16 words of the *LFSR*.

For a 256-bit version, the following operations are performed:

$$s_{15}^{(-33)} = \neg K_0,\ s_{14}^{(-33)} = K_1,\ s_{13}^{(-33)} = \neg K_2,\ s_{12}^{(-33)} = K_3,\ s_{11}^{(-33)} = K_0,\ s_{10}^{(-33)} = \neg K_1,\ s_{9}^{(-33)} = K_2,\ s_{8}^{(-33)} = K_3,$$
$$s_{7}^{(-33)} = \neg K_0,\ s_{6}^{(-33)} = \neg K_1,\ s_{5}^{(-33)} = K_2 \oplus IV_3,\ s_{4}^{(-33)} = K_3,\ s_{3}^{(-33)} = K_0 \oplus IV_2,\ s_{2}^{(-33)} = K_1 \oplus IV_1,$$
$$s_{1}^{(-33)} = K_2,\ s_{0}^{(-33)} = K_3 \oplus IV_0.$$

For a 512-bit version:

$$s_{15}^{(-33)} = K_0,\ s_{14}^{(-33)} = \neg K_1,\ s_{13}^{(-33)} = K_2,\ s_{12}^{(-33)} = K_3,\ s_{11}^{(-33)} = \neg K_7,\ s_{10}^{(-33)} = K_5,\ s_{9}^{(-33)} = \neg K_6,$$
$$s_{8}^{(-33)} = K_4 \oplus IV_3,\ s_{7}^{(-33)} = \neg K_0,\ s_{6}^{(-33)} = K_1,\ s_{5}^{(-33)} = K_2 \oplus IV_2,\ s_{4}^{(-33)} = K_3,\ s_{3}^{(-33)} = K_4 \oplus IV_1,$$
$$s_{2}^{(-33)} = K_5,\ s_{1}^{(-33)} = K_6,\ s_{0}^{(-33)} = K_7 \oplus IV_0$$

2. It performs 32 tacts of triggering without generating a key stream, i.e. two full cycles. Formally, it is presented as follows: $S_{-1} = Next^{32}(S_{-33}, INIT)$ which means 32 iterations to perform the *Next* function in the initialization mode *INIT*, $S_{-33} = (s^{(-33)}, r^{(-33)})$ the values of the variable state are calculated in previous step.

3. The initial value of the variable state S_0 calculated, according to the rule: $S_0 = Next(S_{-1})$ i.e. by executing the *Next* function in normal mode.

4. Get the output value of the variable state S_0.

4 Next-state function Next

The function of next-state, is described as follows.

Input: the variable state $S_i = (s^{(i)}, r^{(i)})$ selected mode (normal or initialization mode).

Output: the next value of the variable state $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$.

1. A nonlinear substitution performed to update the value of the word $r_2^{(i+1)}$ the FSM. For this value, the *T* function is calculated: $r_2^{(i+1)} = T(r_1^{(i)})$.

2. The value of the word $r_1^{(i+1)}$ FSM is updated. For this value calculated, as following: $r_1^{(i+1)} = r_2^{(i+1)} +_{64} s_{13}^{(i)}$, where $+_{64}$ denotes the operation of adding integers by modulus 2^{64} (in the scheme of the cipher, see Fig. 1 this operation is marked as $\boxplus$).

3. The 15 words value *LFSR* is updated $s_j^{(i+1)} = s_{j+1}^{(i)}$ where $j = 0, 1, \ldots, 14$.

4. The value of the 16-th word the *LFSR* is updating. If the normal mode the Next function, the value of this word computing by the rule:

$$s_{15}^{(i+1)} = (s_0^{(i)} \otimes \alpha) \oplus (s_{11}^{(i)} \otimes \alpha^{-1}) \oplus s_{13}^{(i)}.$$

If initialization mode *INIT* of the *Next* function is set, the value is computing by the rule:

$$s_{15}^{(i+1)} = FSM(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}) \oplus (s_0^{(i)} \otimes \alpha) \oplus (s_{11}^{(i)} \otimes \alpha^{-1}) \oplus s_{13}^{(i)}.$$

The multiplication operations $\otimes$ on α and on α^{-1}, as well as the essence of the *FSM* function explained further.

5. The variable state $S_i = (s^{(i)}, r^{(i)})$ value is calculated and outputs.

The conceptual scheme of keystreams "Strumok" generator, when performing the *Next* function in the initialization mode is shown in Fig. 2.

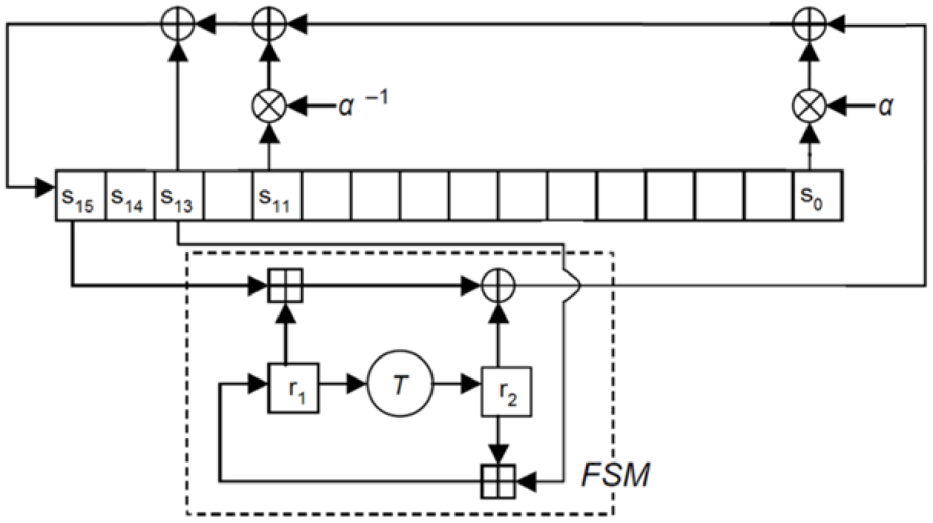

Fig.2 - The conceptual scheme of keystreams *"Strumok"* generator in initialization mode of the *Next* function

5 Keystream function Strm

Input: the variable state $S_i = (s^{(i)}, r^{(i)})$.

Output: 64-bit a key stream Z_i

1. The value is calculated

$$Z_i = FSM(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}) \oplus s_0^{(i)}.$$

2. Get the output value Z_i.

6 Function finite-state machine FSM

The finite-state machine function is marked as *FSM (x,y,z)* and described as following.
Input*:* three 64-bit words *x*, *y* and *z*.
Output*:* 64-bit word *q*.
1. The value is calculated $q = (x +_{64} y) \oplus z$.
2. Get the output value *q*.

7 Function nonlinear substitution T

The nonlienar substitution of the *T* function implements finite field $GF(2^{64})$ rearrangment elements using the components of the block symmetric cryptographic transformation national standard DSTU 7624:2014 [19].

Input: 64-bit word w.

Output: 64-bit word $T = T(w)$.

1. The input word w is divided into 8-bit sub-blocks w_j : $w = (w_7, w_6, ..., w_0)$.

2. For each sub-blocks w_j is performed the DSTU 7624:2014 algorithm substitution using four table transformation $\pi_0, \pi_1, \pi_2, \pi_3$. As a result an output vector formed $r = (r_7, r_6, ..., r_0)$, $r_j = \pi_{j \bmod 4}\left[w_j\right]$, where j = 0, 1, …,7.

3. The vector calculated $q = (q_7, q_6, ..., q_0)$ using the rule (as in DSTU 7624:2014): $q_i = (\upsilon >>> i) r^T$, where υ given in hexadecimal form and $\upsilon = (01, 01, 05, 01, 08, 06, 07, 04)$, $>>> i$ - a cyclic shift operation on i bits to the right, $i = 0, 1, ..., 7$, $r^T = (r_0, r_1, ..., r_7)^T$ and the elements of the vectors r and q interpreted as elements of the final field $GF(2^8)$, which is given as a quotient ring $GF(2)[y]/(p(y))$.

4. Get the output value q, which interpreted as a 64-bit word.

The vector quick calculation $(q_0, q_1, ..., q_7) = Q$ is implemented by the rule:

$$Q^T = T_0[w_0] \oplus T_1[w_1] \oplus T_2[w_2] \oplus T_3[w_3] \oplus T_4[w_4] \oplus T_5[w_5] \oplus T_6[w_6] \oplus T_7[w_7],$$

where:

$$T_0[a] = \begin{pmatrix} 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \end{pmatrix} \cdot \pi_0[a], \quad T_1[a] = \begin{pmatrix} 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \end{pmatrix} \cdot \pi_1[a],$$

$$T_2[a] = \begin{pmatrix} 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \end{pmatrix} \cdot \pi_2[a], \quad T_3[a] = \begin{pmatrix} 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \end{pmatrix} \cdot \pi_3[a],$$

$$T_4[a] = \begin{pmatrix} 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \end{pmatrix} \cdot \pi_0[a], \quad T_5[a] = \begin{pmatrix} 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \end{pmatrix} \cdot \pi_1[a],$$

$$T_6[a]=\begin{pmatrix}07\\06\\08\\01\\05\\01\\01\\04\end{pmatrix}\cdot\pi_2[a],\ T_7[a]=\begin{pmatrix}04\\07\\06\\08\\01\\05\\01\\01\end{pmatrix}\cdot\pi_3[a].$$

The constant tables $T_i[a]$, $i=0,1,...,7$ using enables significantly reducing the operations number, in particular the nonlinear substitution function, is calculated seven XOR operations over 64-bit words.

8 Multiplications of α in $GF(2^{64})$

Multiplication on α in the finite field $GF(2^{64})$ arithmetic is implemented by a table of pre-calculus Mul_α, which contains 256 rows of 64 bits in each.

1. The value is calculated

$$w'=(w<<8)\oplus Mul_\alpha[w>>56] \tag{1}$$

where:

- $w<<8$ is the result of a shift to the left (towards the higher significant bits) 64-bit word w on 8 bits with the filling of the less significant bits with zero values.
- $w>>56$ is the result of a shifting to the right (*towards the less significant bits*) 64-bit word w on 56 bits with the filling of the higher significant bits with zero values. Eight less bits of the vector $w>>56$ is interpreted as the finite field $GF(2^8)$ element for indexing the table of pre-calculus Mul_α;
- $Mul_\alpha[c]$ - 64-bit value the table of pre-calculus in the row with the index $c\in GF(2^8)$, where $Mul_\alpha[c]\in GF(2^{64})$.

2. Get the output value w'.

9 Multiplications of α^{-1} in $GF(2^{64})$

Multiplication on α^{-1} in arithmetic of the finite field $GF(2^{64})$ is implemented by a table of pre-calculus $Mul_{\alpha^{-1}}$, which contains 256 rows of 64 bits in each.

1. The value is calculated

$$w'=(w>>8)\oplus Mul_{\alpha^{-1}}[w\&\gamma] \tag{2}$$

where:

- $w>>8$ is the result of a shifting to the right (towards the less significant bits) 64-bit word w on 8 bits with the filling of the higher significant bits with zero values.
- $w\&\gamma$ is a bitwise conjunction result of words w and γ, which is in the hexadecimal form $\gamma=00000000000000\text{FF}$. The eight less significant bits of the vector $w\&\gamma$ is interpreted as an element of the finite field $GF(2^8)$ for indexing the table of pre-calculus $Mul_{\alpha^{-1}}$.

3. Get the output value w'.

10 Value tables-constant Mul_α and $Mul_{\alpha^{-1}}$

For the fast encryption there are used tables of pre-calculus Mul_α and $Mul_{\alpha^{-1}}$. This allows significantly reduce the number of operation to handle the input data.

The polynomial that defines feedback of the LFSR has the expression $g(z)$. Thus, each word of the LFSR stores a 64-bit sequence w, which is represented as eight sub-blocks w_j of 8 bits in each: $w=\left(w_7, w_6, \ldots, w_0\right)$, which is interpreted as coefficients of a polynomial $w(z) \in GF(2^8)[z]/(g(z))$.

If $\alpha = z$ is a root with a primitive polynomial of the $GF(2^8)$:

$$g(z) = z^8 + g_7 z^7 + \ldots + g_0$$

then:

$$w(z)\cdot\alpha = w_{<<8}(z) + w_{>>56}(z)\cdot g'(z),$$

where $w_{<<8}(z)$, $w_{>>56}(z)$ and $g'(z)$ are polynomials, and have the form:

$$w_{<<8}(z) = w_6 z^7 + w_5 z^6 + \ldots + w_0 z\,,\ w_{>>56}(z) = w_7\,,$$

$$g'(z) = g_7 z^7 + g_6 z^6 + \ldots + g_0\,.$$

The binary representation of the polynomials $w_{<<8}(z)$ and $w_{>>56}(z)$ forms coefficients is considered in (1) binary sequences $w<<8$ and $w>>56$. So computing $w(z)\cdot\alpha$ in the finite field $GF(2^{64})$ arithmetic corresponds to the formula (1), where 256 values of the table $Mul_\alpha[w_7]$ is calculated as 64-bit sequences with the binary representation of coefficients $(w_7 g_7, w_7 g_6, \ldots, w_7 g_0)$ a polynomial

$$w_{>>56}(z)\cdot g'(z) = w_7(g_7 z^7 + g_6 z^6 + \ldots + g_0)$$

for each with 256 possible values $w_7 \in GF(2^8)$.

If $\alpha = z$ then:

$$\alpha^8 = g_7\alpha^7 + \ldots + g_1\alpha + g_0\alpha^0\,,$$

so

$$g_0^{-1}\alpha^7 + g_0^{-1}g_7\alpha^6 + \ldots + g_0^{-1}g_1\alpha^0 = \alpha^{-1} = z^{-1}\,,$$

then

$$w(z)\alpha^{-1} = w_{>>8}(z) + w_0(z)\cdot g''(z),$$

where $w_{>>8}(z)$, $w_0(z)$ and $g''(z)$ are polynomials, and have the form:

$$w_{>>8}(z) = w_7 z^6 + w_6 z^5 + \ldots + w_1,$$

$$w_0(z) = w_0\,,$$

$$g''(z) = g_0^{-1}z^7 + g_0^{-1}g_7 z^6 + \ldots + g_0^{-1}g_1\,.$$

The binary representation of the polynomials coefficients $w_{>>8}(z)$ and $w_0(z)$ forms is considered in (2) binary sequences $w>>8$ and γ. So computing $w(z)\cdot\alpha^{-1}$ in arithmetic of the finite field $GF(2^{64})$ corresponds to formula (2), where 256 values of the table $Mul_{\alpha^{-1}}[w_0]$ is calculated as 64-bit sequences with the binary representation of coefficients $(w_0 g_0{}^{-1}, w_0 g_0{}^{-1} g_7, \ldots, w_0 g_0{}^{-1} g_1)$ a polynomial

$$w_0(z)\cdot g''(z) = w_0(g_0^{-1}z^7 + g_0^{-1}g_7 z^6 + \ldots + g_0^{-1}g_1)$$

for each with 256 possible values $w_0 \in GF(2^8)$.

11 Software performance of "*Strumok*"

For the generation key stream rate research, we realized experiment as the well-known symmetric cryptographic transformation on equal terms. List of algorithm, source of specification and brief information are given in the table 1. The testing results based on the criterion for long streams encryption [20] are shown in the table 2. As we can see from the data in the table, the keystream "Strumok" generator enables pseudo-random sequences forming with speeds exceeding of 10 Gbit per sec. By this measure, it is ahead of almost all the most common ciphers including the algorithm SNOW 2.0.

Table 1 – List of algorithm

Names cipher	**Source of specification**	**State size, bit**	**Key size, bit**	**Size IV, bit**
AES	FIPS-197, CRYPTREC, ISO/IEC 18033-4	128	128, 256	256
Kalyna	DSTU 7624:2014	128, 256, 512	128, 256, 512	128, 256, 512
HC	eSTREAM	128, 256	128, 256	128, 256
MICKEY	eSTREAM	160	128	128
RABBIT	ISO/IEC 18033-4, eSTREAM	513	128	64
SALSA-20	eSTREAM	512	128	64
SNOW2.0	ISO/IEC 18033-4	512	128, 256	128, 56
SOSEMANUK	eSTREAM	512	128	128
TRIVIUM	eSTREAM, ISO/IEC 29192-3	288	80	80
Enocoro	ISO/IEC 29192-3	272	80, 128	64
CRYPTMT3	eSTREAM	128	128	64
DECIMv2	ISO/IEC 18033-4, eSTREAM	288	128	128
RC4	Список рассылки Cypherpunks	256	256	–
KCIPHER-2	ISO/IEC 18033-4, CRYPTREC	640	128	128
GRAIN	eSTREAM	128	128	96
MUGI	ISO/IEC 18033-4	128	128	128
Strumok-256	This article	1024	256	256
Strumok-512			512	

Table 2 – Performance evaluation of ciphers

CIPHERS	**Intel Core i7-6820HQ 2.7 Gh**	**Intel Core i7-5500u 2.4 Gh**	**Intel Pentium P6200 2.13 Gh**
AES-128	2,48	1,75	1,12
AES-256	1,66	1,18	0,80
Kalyna-128	2,56	1,79	0,83
Kalyna-256	1,71	1,21	0,57
Kalyna-512	1,42	0,99	0,46
HC-128	11,46	7,69	4,25
HC-256	5,13	3,88	2,03

a continuation Table 2

CIPHERS	Intel Core i7-6820HQ 2.7 Gh	Intel Core i7-5500u 2.4 Gh	Intel Pentium P6200 2.13 Gh
MICKEY-128	0,07	0,05	0,03
RABBIT	3,65	2,77	1,64
SALSA-20	3,02	2,06	1,41
SNOW2.0-128	8,76	5,43	3,67
SNOW2.0-256	8,72	5,54	3,59
SOSEMANUK	4,07	2,56	1,82
TRIVIUM	3,89	2,78	1,89
CryptMT3	5,92	4,63	4,04
DECIM-128	0,01	0,01	0,01
RC4	3,58	3,21	1,67
KCIPHER-2	0,40	0,40	0,31
GRAIN	0,01	0,01	0,00
MUGI	3,62	2,98	2,58
Strumok-256	13,31	10,04	5,10
Strumok-512	13,70	9,74	5,08

Conclusions

Strumok in its conceptual scheme similar to the *SNOW* 2.0. But *SNOW* 2.0 focused on the use of 32-bit computing systems, while "*Strumok*" is intended for use in more powerful 64-bit computing systems. With this in "*Strumok*" increased rate of formation of the pseudo-random sequence, as used by 64-bit words to store encryption keystream. The conducted comparative tests have shown that the "*Strumok*" on 32-bit computing systems also shows good performance results. Using a pre-computation increases the speed of the algorithm, since there is no need to make complicated calculations during the generation of the keystream.

In the "*Strumok*" algorithm, compared with SNOW 2.0, it is increased the length of the secret key and the initialization vector. This allows us reliably apply a stream cipher even with taking into account quantum methods of cryptographic analysis. Thus, in aggregate of properties, the "*Strumok*" can be considered as a candidate for the symmetric encryption national standard of Ukraine.

References

[1] N. Ferguson and B. Schneier. Practical Cryptography. John Wiley & Sons, 2003, 432 p.

[2] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997, 794 p.

[3] N. Koblitz and A.J. Menezes. "A Riddle Wrapped in an Enigma". Internet: https://eprint.iacr.org/2015/1018.pdf, Oct. 20, 2015 [Aug. 21, 2016]

[4] D. Bernstein, J. Buchmann and E.Dahmen. Post-Quantum Cryptography. Springer-Verlag, Berlin-Heidleberg, 2009, 245 p.

[5] D. Moody. "Post-Quntum Cryptography: NIST's Plan for the Future". The Seventh International Conference on Post-Quntum Cryptography, Japan, 2016. [On-line]. Internet: https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf

[6] ISO/IEC 18033-4:2011. "Information technology – Security techniques – Encryption algorithms – Part 4: Stream ciphers". On-line]. Internet: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54532 [Dec., 2012]

[7] O. Kuznetsov, M. Lutsenko and D. Ivanenko, "Strumok stream cipher: Specification and basic properties". 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 59-62.

[8] O. Kuznetsov, Y. Gorbenko and I. Kolovanova, "Combinatorial properties of block symmetric ciphers key schedule". 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 55-58.

[9] I. Gorbenko, A. Kuznetsov, M. Lutsenko and D. Ivanenko, "The research of modern stream ciphers". 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, 2017, pp. 207-210.
[10] A. Kuznetsov, I. Svatovskij, N. Kiyan and A. Pushkar'ov, "Code-based public-key cryptosystems for the post-quantum period". 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, 2017, pp. 125-130.
[11] A. Kuznetsov, I. Kolovanova and T. Kuznetsova, "Periodic characteristics of output feedback encryption mode". 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, 2017, pp. 193-198.
[12] A. Kuznetsov, Y. Gorbenko, A. Andrushkevych and I. Belozersev, "Analysis of block symmetric algorithms from international standard of lightweight cryptography ISO/IEC 29192-2". 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, 2017, pp. 203-206.
[13] Y. Izbenko, V. Kovtun and A. Kuznetsov, "The Design of Boolean Functions by Modified Hill Climbing Method". 2009 6th International Conference on Information Technology: New Generations, Las Vegas, NV, 2009, pp. 356-361.
[14] A. Kuznetsov, R. Serhiienko and D. Prokopovych-Tkachenko, "Construction of cascade codes in the frequency domain". 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, 2017, pp. 131-136.
[15] A. Andrushkevych, T. Kuznetsova, I. Bilozertsev and S. Bohucharskyi, "The block symmetric ciphers in the post-quantum period". 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 43-46.
[16] I.D. Gorbenko, V.I. Dolgov, V.I. Rublinetskii, K.V. Korovkin. “Methods of Information Protection in Communications Systems and Methods of Their Cryptoanalysis”. Telecommunications and Radio Engineering, Vol. 52, Issue 4, (1998), pp. 89-96.
[17] I. Gorbenko, V. Ponomar. “Examining a possibility to use and the benefits of post-quantum algorithms dependent on the conditions of their application”. EasternEuropean Journal of Enterprise Technologies. Vol 2, No. 9 (86) (2017), pp. 21-32.
[18] Yu.V. Stasev, A.A. Kuznetsov. “Asymmetric code-theoretical schemes constructed with the use of algebraic geometric codes”. Kibernetika i Sistemnyi Analiz, No. 3, pp. 47-57, May-June 2005.
[19] DSTU 7624:2014. “Informacijni tehnologii'. Kryptografichnyj zahyst informacii'. Algorytm symetrychnogo blokovogo peretvorennja”. (in Ukrainian). [On-line]. Internet: http://shop.uas.org.ua/ua/informacijni-tehnologii-kriptografichnij-zahist-informacii-algoritm-simetrichnogo-blokovogo-peretvorennja.html
[20] “eSTREAM Optimized Code HOWTO”. [On-line]. Internet: http://www.ecrypt.eu.org/stream/perf/ [Nov. 1, 2005].

Рецензент: Микола Карпінський, д.т.н., проф., університет Бельсько-Бяла, Польща.
E-mail: mkarpinski@ath.bielsko.pl

Надійшло: Лютий 2018.

Автори:
Іван Горбенко, д.т.н., проф., Харківський національний університет імені В.Н. Каразіна, м. Харків, Україна.
E-mail: gorbenkoi@iit.kharkov.ua
Олександр Кузнецов, д.т.н., проф., академік Академії наук прикладної радіоелектроніки, ХНУ імені В.Н. Каразіна, м. Харків, Україна. E-mail: kuznetsov@karazin.ua
Юрій Горбенко, к.т.н., акціонерне товариство “Інститут інформаційних технологій” (АТ "ІІТ"), м. Харків, Україна.
E-mail: gorbenkou@iit.kharkov.ua
Антон Олексійчук, д.т.н., національний технічний університет України "КПІ", (НТУУ "КПІ"), м. Київ , Україна.
E-mail: alex-dtn@ukr.net
Владислав Тімченко, студент факультету комп’ютерних наук, Харківський національний університет імені В.Н. Каразіна, м. Харків, Україна.
E-mail: tvlad.tyma@gmail.com

Потоковий шифр "Струмок".

Анотація. У роботі представлені основні результати розробки нового генератора ключів «Струмок», який пропонується в якості кандидата на національний стандарт симетричного шифрування України. «Струмок» побудований на схемі SNOW 2.0-like генератора підсумовування. Збільшена довжина секретного ключа і вектор ініціалізації дозволяють надійно використовувати шифр потоку навіть з урахуванням методів квантового криптографічного аналізу. На відміну від SNOW 2.0 «Strumok» призначений для використання в більш потужних 64-розрядних обчислювальних системах. Проведені порівняльні тести показали, що «Strumok» на 32-розрядних обчислювальних системах також покази хороші результати. Представлені основні перетворення і окремі результати дослідження продуктивності шифрування, розглянуто генератор, якій забезпечує формування потоку ключів зі швидкістю, що перевищує 10 Гбіт / сек.

Ключові слова: шифрування; потоковий шифр; синхронний генератор ключів; псевдовипадкова послідовність.

Рецензент: Николай Карпинский д.т.н., проф., университет Бельсько-Бяла, Польша.
E-mail: mkarpinski@ath.bielsko.pl

Поступила: Февраль 2018.

Авторы:

Иван Горбенко, д.т.н., проф., Харьковский национальный университет имени В.Н. Каразина, г. Харьков, Украина.
E-mail: gorbenkoi@iit.kharkov.ua

Александр Кузнецов, д.т.н., проф., академик Академии наук прикладной радиоэлектроники, ХНУ имени В.Н. Каразина, г. Харьков, Украина. E-mail: kuznetsov@karazin.ua

Юрий Горбенко, к.т.н., АТ "Институт информационных технологий" (АТ "ИИТ"), г. Харьков, Украина.
E-mail: gorbenkou@iit.kharkov.ua

Антон Алексейчук, д.т.н., национальный технический университет Украины "КПИ", (НТУУ "КПИ"), г. Киев, Украина.
E-mail: alex-dtn@ukr.net

Владислав Тимченко, студент факультета компьютерных наук, Харьковский национальный университет имени В.Н. Каразина, г. Харьков, Украина.
E-mail: tvlad.tyma@gmail.com

Потоковый шифр "Струмок".

Аннотация. В работе представлены основные результаты разработки нового генератора ключей «Струмок», который предлагается в качестве кандидата на национальный стандарт симметричного шифрования Украины. «Струмок» построен на схеме SNOW 2.0-like генератора суммирования. Увеличенная длина секретного ключа и вектор инициализации позволяют надежно использовать шифр потока даже с учетом методов квантового криптографического анализа. В отличие от SNOW 2.0, «Струмок» предназначен для использования в более мощных 64-разрядных вычислительных системах. Проведенные сравнительные тесты показали, что «Струмок» на 32-разрядных вычислительных системах также показывает хорошие результаты. Представлены основные преобразования и отдельные результаты исследования производительности шифрования, рассмотрен генератор, обеспечивающий формирование потока ключей со скоростью, превышающей 10 Гбит / сек.

Ключевые слова: шифрование; потоковый шифр; синхронный генератор ключей; псевдослучайная последовательность.