SCIENTIFIC AND TECHNICAL JOURNAL
Namangan Institute of Engineering and Technology

«A NEW KEY STREAM ENCRYPTION ALGORITHM AND ITS CRYPTANALYSIS»

Khudoykulov Zarif	Teacher
Rakhmatullaev Ilkhom	Teacher

Samarkand branch of Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi

https://doi.org/10.5281/zenodo.7951969

ISSN 2181-8622

Manufacturing technology problems

Scientific and Technical Journal Namangan Institute of Engineering and Technology

Volume 8
Issue 1
2023

Google Scholar

doi zenodo

SLIB.UZ
Scientific library of Uzbekistan

A NEW KEY STREAM ENCRYPTION ALGORITHM AND ITS CRYPTANALYSIS

KHUDOYKULOV ZARIF

Department of Cryptology, Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi
E-mail: zarif.xudoyqulov@mail.ru

RAKHMATULLAEV ILKHOM

Samarkand branch of Tashkent University of Information Technologies named after Muhammad al-Khwarizmi
E-mail: ilhom9001@mail.com

Abstract. The new stream encryption algorithm (NSA-New Stream Algorithm) is proposed in this work. The input parameters are considered a 128-bit secret key and 128-bit initialization vectors in the new algorithm. A 64-bit line is generated in each round as the output value. The architecture of the algorithm is particularly suitable for efficient hardware implementations, together with this, this algorithm is also suitable for software implementation. On the other hand, the security was evaluated for resynchronization attack, related key attack, and attack methods on the basis of linear correlation of the output sequence. Analysis confirms that this algorithm is a secure stream encryption algorithm.

Keywords. Linear Feedback Shift Registers (LFSR), Output Sequence Randomization, Resynchronization Attack, Related Key Attack, Linear Crypto analysis, Differential Crypto analysis, Integral Crypto analysis

Introduction. Several approaches to the design of pseudo-random number generators which can be used in stream encryption algorithms are known from the literatures. One of the popular approach is based on Linear Feedback Shift Registers (LFSR). They are suitable for very compact hardware applications and provide good randomness. However, due to their linearity and approximation, they cannot be used in pure forms. Several techniques have been developed in order to ensure their security, such as a combination generator, nonlinear filtering, and clock control. Numerous researches have been conducted to ensure the safety of these schemes. However, generators which are created on the basis of LFSR are not suitable for effective software implementation of algorithms which are developed on the basis of them.

On the other hand, software-oriented stream ciphers seem to be custom-designed, and we don't have the proper tools to evaluate them. The most important criterion is considered to check for deviations from randomness.

The algorithm made for security reasons is evaluated for its tolerance against resynchronization attacks and related key attacks.

In addition, the linear correlation of the output sequence was calculated.As a result, it was concluded that this algorithm is a reliable and effective cryptographic tool which can be used to provide encryption and message authentication.

From this point of view, the development of new stream encryption algorithms on the basis of a new approach is considered one of the actual researches.

The main part. The following new streamen cryption algorithm is proposed by analyzing existing stream encryption algorithms and generators,

Markings:

K –secret key;

I –initialization vector;

K_0–thefirst 64 bits of the key;

K_1–thesecond 64-bits part of the key;
I_0–thefirst 64-bits part of the initialization vector;
I_1–thesecond 64-bits part of the initialization vector;
x –length 64 bits array;
a_0, a_1, a_2–status arrays with a length of 64 bits;
b – 16 $(b_0, b_1, \dots, b_{15})$ buffer arrays with a length of 64 bits;
T –update function;
p –mixing (updating the state of arrays) function;
F –non-linear function in the mixing function;
λ –update function of $b$buffer value;
C_0, C_1, C_2, C_3, C_4 - constant numbers (constant values);
S - non-linear reflection (byte replacement - block);
M – linear reflection (matrix multiplication);
L -byte swapping function.

Input information:
Plain text: 64 bits;
Private key: 128 bits;
Initialization vector (I): 128 bits;
***T* –update function:**

$$(a^{t+1}, b^{t+1}) = T(a^t, b^t) = (p(a^t, b^t), \lambda(a^t, b^t))$$

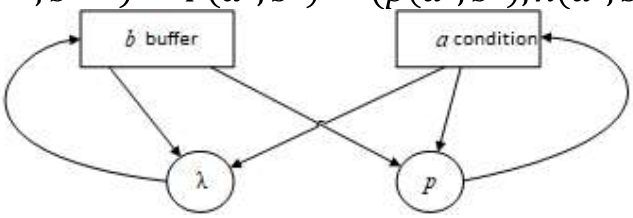

Figure 1. The schematic view of T - update function

***p* –mixing function:**

$$a_0^{(t+1)} = a_1^{(t)} \oplus C_0$$
$$a_1^{(t+1)} = a_2^{(t)} \oplus F(a_1^{(t)}, b_4^{(t)} \lll 15) \oplus C_1$$
$$a_2^{(t+1)} = a_0^{(t)} \oplus F(a_1^{(t)}, b_{10}^{(t)} \lll 15) \oplus C_2$$

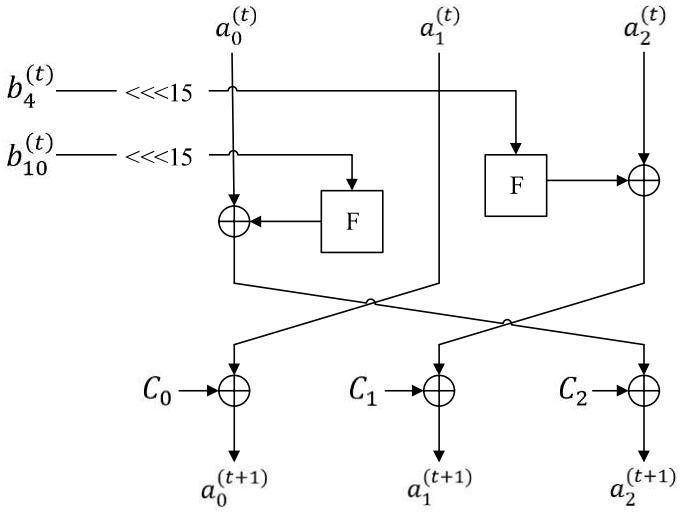

Figure 2. The schematic view of the p –mixing function

F function:

$$F(a_1^t, o) = L(M(S(a_1^t \oplus o)))$$

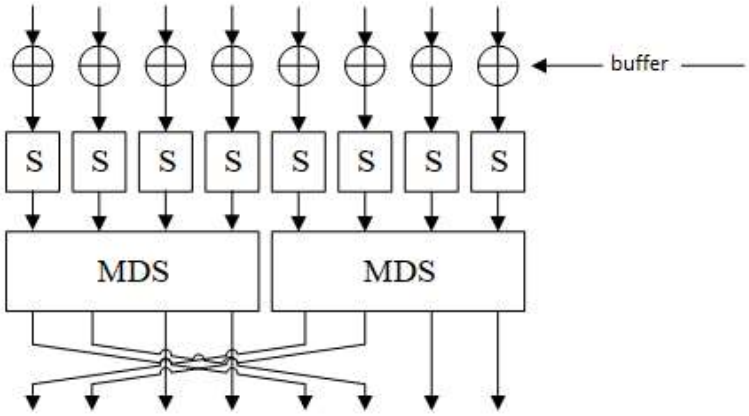

Figure 3. The schematic view of the F function

L function:

$L(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_4||x_5||x_2||x_3||x_0||x_1||x_6||x_7$,

$\boldsymbol{\lambda}$, function

$$b_{15}^{t+1} = a_1^t \oplus C_0$$
$$b_{14}^{t+1} = a_1^t \oplus F(b_{15}^t, 0) \oplus C_1$$
$$b_{13}^{t+1} = a_0^t \oplus a_2^t \oplus b_{14}^t \oplus F(b_{14}^t, 0) \oplus C_2$$
$$b_{\#}^{t+1} = a_0^t \oplus a_1^t \oplus a_2^t \oplus b_{i+1}^t \oplus b_{i+2}^t \oplus F(b_{i+1}^t, 0) \oplus C_3$$
$$i = 12, 11, \ldots, 0.$$

S non-linear reflection:

[fc, ee, dd, 11, cf, 6e, 31, 16, fb, c4, fa, da, 23, c5, 4, 4d, e9, 77, f0, db, 93, 2e, 99, ba, 17, 36, f1, bb, 14, cd, 5f, c1, f9, 18, 65, 5a, e2, 5c, ef, 21, 81, 1c, 3c, 42, 8b, 1, 8e, 4f, 5, 84, 2, ae, e3, 6a, 8f, a0, 6, b, ed, 98, 7f, d4, d3, 1f, eb, 34, 2c, 51, ea, c8, 48, ab, f2, 2a, 68, a2, fd, 3a, ce, cc, b5, 70, e, 56, 8, c, 76, 12, bf, 72, 13, 47, 9c, b7, 5d, 87, 15, a1, 96, 29, 10, 7b, 9a, c7, f3, 91, 78, 6f, 9d, 9e, b2, b1, 32, 75, 19, 3d, ff, 35, 8a, 7e, 6d, 54, c6, 80, c3, bd, d, 57, df, f5, 24, a9, 3e, a8, 43, c9, d7, 79, d6, f6, 7c, 22, b9, 3, e0, f, ec,

de, 7a, 94, b0, bc, dc, e8, 28, 50, 4e, 33, a, 4a, a7, 97, 60, 73, 1e, 0, 62, 44, 1a, b8, 38, 82, 64, 9f, 26, 41, ad, 45, 46, 92, 27, 5e, 55, 2f, 8c, a3, a5, 7d, 69, d5, 95, 3b, 7, 58, b3, 40, 86, ac, 1d, f7, 30, 37, 6b, e4, 88, d9, e7, 89, e1, 1b, 83, 49, 4c, 3f, f8, fe, 8d, 53, aa, 90, ca, d8, 85, 61, 20, 71, 67, a4, 2d, 2b, 9, 5b, cb, 9b, 25, d0, be, e5, 6c, 52, 59, a6, 74, d2, e6, f4, b4, c0, d1, 66, af, c2, 39, 4b, 63, b6].

Constants:

C_0 =0x6a09e667f3bcc908,

C_1 =0xbb67ae8584caa73b,

C_2 =0x3c6ef372fe94f82b,

C_3 =0xa54ff53a5f1d36f1,

C_4 =0x510e527fade682d1.

M line arreflection:

This reflection represents a 32-bit MDS matrix multiplication operation in the Rijndael algorithm. $(酸_0, x_1, x_2, x_3)$ and (y_0, y, y_2, y_3) represent the input and output matrices of the reflection:

$$y_0 = 0x02 \cdot x_0 \oplus 0x03 \cdot x_1 \oplus 0x01 \cdot x_2 \oplus 0x01 \cdot x_3,$$
$$y_1 = 0x01 \cdot x_0 \oplus 0x02 \cdot x_1 \oplus 0x03 \cdot x_2 \oplus 0x01 \cdot x_3$$
$$y_2 = 0x01 \cdot x_0 \oplus 0x01 \cdot x_1 \oplus 0x02 \cdot x_2 \oplus 0x03 \cdot x_3$$
$$y_3 = 0x03 \cdot x_0 \oplus 0x01 \cdot x_1 \oplus 0x01 \cdot x_2 \oplus 0x02 \cdot x_3$$

Initialization. The initialization process consists of 3 steps. In the first step, the arrays of b buffer is initialized using the K key, and in the second step, the buffer arrays of a_0, a_1, a_2 state is initialized using I initialization vector. In the third step, the internal state arrays are mixed.

Step 1: The 128-bit key is expanded to 192 bits and a_0, a_1, a_2written to state arrays:

$$档_0^{t_0} = K_0,\ a_1^{t_0} = K_1,\ a_2^{t_0} = (K_0 \lll 11) \oplus (K_1 \ggg 11) \oplus C_4$$

Here, t_0 indicates that the initialization processs has started.

After that, the values on the left side of $a_1^t$are shuffled using the p – shuffle function. a_0^t is placed into the array buffer as follows:

$$b_{15-i} = (p^{i+1}(a^{(t_0)}, 0))_0$$

here $p^i p$ denotes the i – iteration of the function. $p(a, 0)$ means that the values in the b buffer are not used at this stage.

Step 2: In this step, $a(K) = p^{16}(a(K, I), 0)$state arrays and I- initialization vector are required. I –the initialization vector is added to the a state arrays as follows:

$$a(K, I)_0 = a(K)_0 \oplus I_0 \oplus C_4\ ,$$
$$a(K, I)_1 = a(K)_1 \oplus I_1 \oplus C_4,$$
$$a(K, I)_2 = a(K)_2 \oplus (I_0 \lll 11) \oplus (I_1 \ggg 11) \oplus C_4$$

$a$state arrays are mixed 16 times by repeating the p function Then this process can be expressed as $p^{16}(a(K, I), 0)$.

Step 3. In the last step, the T update function is repeated 16 times. This process can be described as follows:

$$a^{(1)} = T^{16}(p^{16}(a(K, I), 0), b(K))$$

Here, $b(K)$ is a K key-initialized array buffer.

After initialization, the algorithm generates 64-bit random numbers and changes the internal state in each iteration. The output of cycle is taken as Output[t], the output vector is defined as follows:

$$exit[t] = a_2^{(t)}$$

In other words, the algorithm outputs the rightmost 64 bits of the $a$state arrays each time. After that, step 3 of the initialization process is performed again.

The steps from initialization to random number generation are shown in Table 1:

Table 1.

Representation of the algorithm process in a tabular form

	t round	process	input	output
Initialization	-49	*Input the key*	-	-
	-48, …, -33	*Mixing by using function P*	-	-
	-32	*Input IV*		
	-31, …, -16	*Mixing by using function P*	-	
	-15, …, 0	*Mixing by using function*	-	
Generation of a byte array	1,…	*Mixing and output*	-	*output[t]*

Evaluation results of the NSA algorithm to cryptanalysis methods. The security of an algorithm depends on the relationship between the input and output bits (or the relationship between the output bits). The internal state complete key selection attack, or attacks that facilitate the complete key selection attack applied to stream encryption algorithms, exploit some of these relationships and predict the internal state. It is assumed that the parser can observe some deviation between input and output bits (or only between output bits) and obtain information about the internal state, even if it cannot do so. This stems from the philosophy that it should be impossible to predict the output sequence of a secure pseu-dorandom number generator. Theafore mentioned relationships are divided into three cases:

Randomization of the output sequence. The parser observes changes in the output sequence by changing the secret key and the initialization vector.

Resynchronization attack. The parser observes the relationship between the initialization vector and the output sequences by changing the secret key.

Related key attack. The parser observes the relationship between the keys and the output sequence by changing the initialization vector. This attack involves tracking the relationship between the keys and the initialization vector [7].

On the other hand, the exhaustive key search requires an average of 2^{127} computations to find a valid key. The attack is considered successful if consumesless resources are used than the average of 2^{127} key selections.

Randomization of the output sequence.

Linearity should be one of the most important properties of certain estimation methods. Here, "linearity" does not mean linear complexity, but rather the maximum likelihood of linear combinations of output bits. It should be noted that the search for a linear combination is similar to the search for the best guess for a block cipher and uses the evaluation method used in linear cryptanalysis [5]. More precisely, this algorithm corresponds to the calculation of

active S-boxes in line arapproximations to estimate the linearity of the output sequence. However, applying this method to pseudo-random number generators is more difficult than applying it to block cipher algorithms, since the buffer is updated dynamically. Therefore, it is more efficient to calculate a lower bound on the number of active S-boxes needed for any linear approximation than to construct actual linear approximations.

The number of active S-boxes of line arapproximation is denoted by AS. The maximum linearity probability of the S-box of the proposed NSA algorithm is 2−6, so if there is not linear approximation with AS<22, it can be assumed that the linearity of the output sequence of the algorithm is small enough. Applying this method to the proposed NSA algorithm, the following theorem was established:

Theorem. The linear approximation of the NSA algorithm is AS ≥22.

The proof of this theorem is given in the following. The construction of a linear approximation consisting of output units is divided into two steps as follows:

1. Construction of linear approximations of p

2. Search for the path containing the buffer.

Construct linear approximations of p. Before starting the evaluation, the equivalent variants of $p$are selected for ease of analysis. Figure 4 shows the modified options of the transformation. The F function on the left is marked with a G function; These designations are used for convenience only. Firstly, F can be moved to the left in the next step. Then, the mask corresponding to the output unit can accept all values, so we divide this part into two masks, the output mask is the input mask. This transformation is not equivalent in the general sense, but it is equivalent in the sense that the mask templates do not change with the transformation. After that, we remove unnecessary links. The right side of Figure 4 shows a modified version of the $p$function. Then "p" means changed p. Note that the number of bindings is reduced by two, and the output masks of the F- and G-functions come directly from the "input" and "output" masks that the parser can choose.

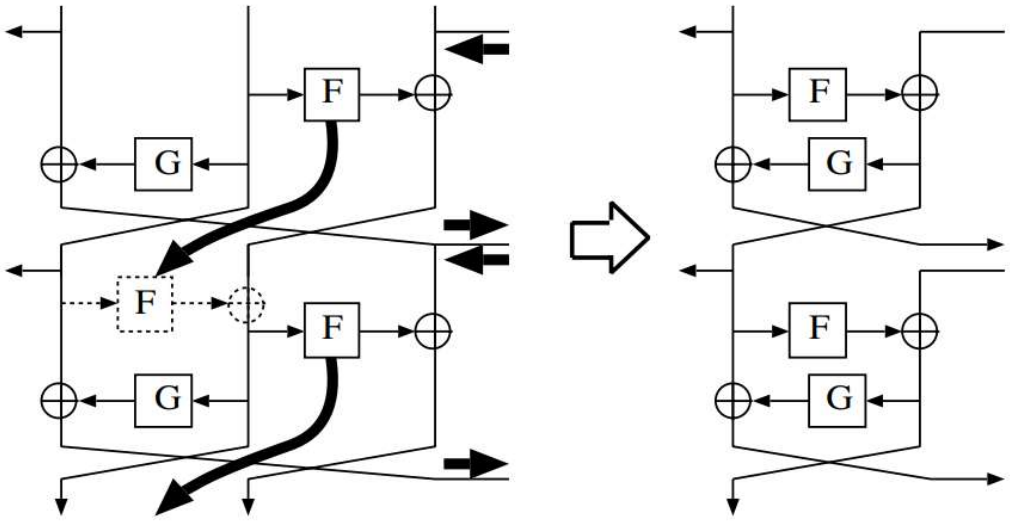

Figure 4. The modified version of the function p

Figure 4 shows some import ant paths of p . Only the five paths shown there ensure that the number of active S-boxes is greater than five. The grid number of the matrix M is defined by $min_{x\neq 0}(w_H(x) + w_H(M_x))$, here $w_H(x)$ is the Hamming weight per byte of x[1]. The grid number of the linear transformation is an important property for the diffusion properties of the block cipher.

But the number of grids of matrix M for PTSG does not guarantee a lower bound on the number of active S-boxes for linear approximation, even if it contains several active F-functions. This property is very different from that of block ciphers.

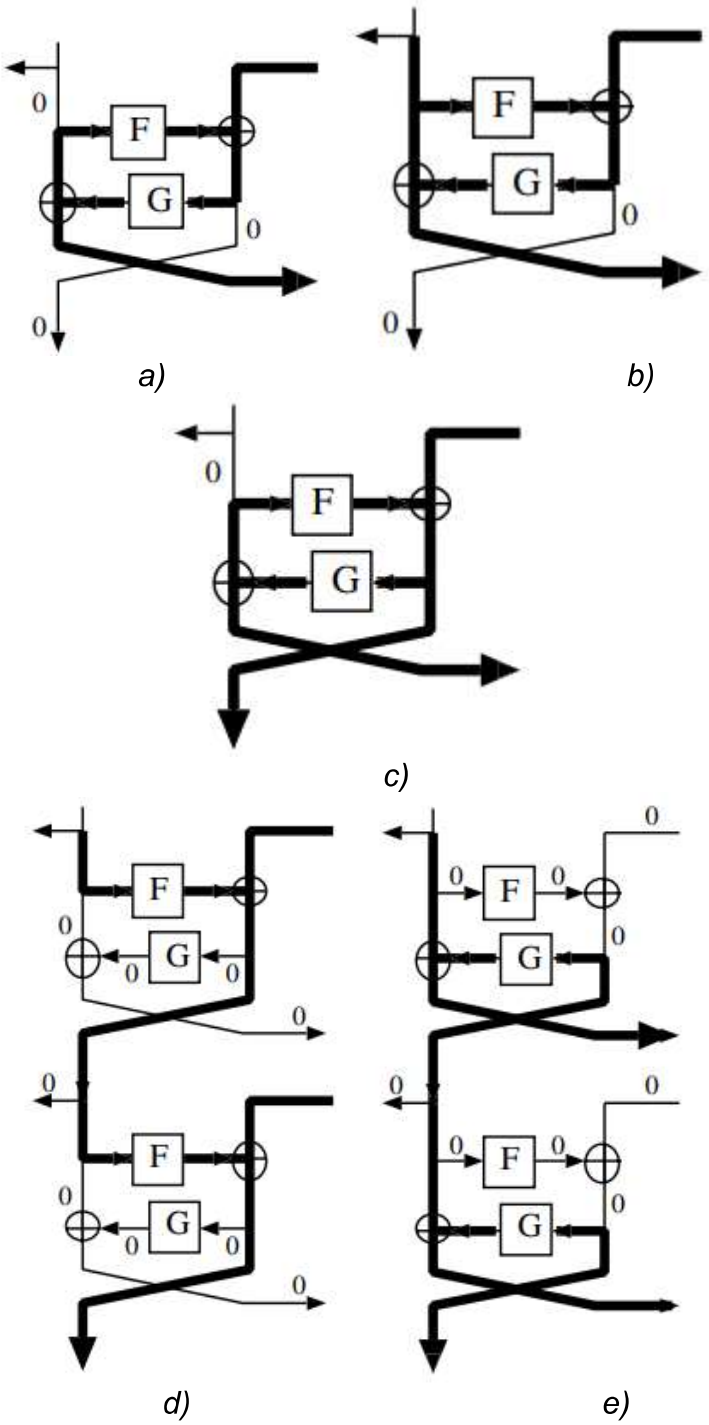

Figure 4. The linear approximations of a function

The *NSA* is linear path of the algorithm. Next, a path is searched that contains a buffer that gives a linear approximation of only the output bits. For PTSGs, the parser can track any number of rounds. Thus, a linear approximation can be constructed with the output of any round. In addition, some linear approximations may miss intermediate p function, which means that more rounds can be observed and the deviation increased. This feature makes sit difficult to search all paths.

The first and last rounds of the path can be marked $t_b$and t_o. The mask which is done XOR to the data from state a to buffer b is defined as $\Gamma(D)(t)$. In addition, an active F –function can be defined as 1, and F –function close to zero as 0. For example, if F –function is active, but the G –function is not active in the t –period, this is defined as a $\Gamma(a)^{(t)} = (1,0)$ state.

First of all, we need to pay special attention to the first and last stage of the road. The value of the input mask for all units of the buffer and their state is equal to zero in the first round, and only the mask of the output unit $Chiqish[t_b]$ is active. Only two paths, $a)$and $c)$cases in Figure 4, satisfy this condition. The last round is the same as the state $a)$, so the possible paths in the round $t_o$are shown only as $a)$and $b)$states.

In the next step, the effect of the buffer on p is analyzed. The value of $\Gamma(D)^{(t)}$is equal to 0 in round $t_b$to round $t_b + 4$because all input masks are equal to 0 for the first round. Also, the access mask from the buffer to G function must be active, so $\Gamma(D)^{(t_b+5)}$is active. Similarly, the value of $\Gamma(D)^{(t)}$ is equal to 0 in rounds $t_o - 5 \le t \le t_o$and is active in round $t_o - 5$

If $(\Gamma(a)^{(t_b+i)}, \Gamma(a)^{(t_b+i+1)}) = ((0,0),(1,1))$ is the rounds $i\ (1 \le i \le 4)$, it contains more active F –functions in the $a)$or $c)$ category, that's why it is AS ≥ 25. Therefore, it is only necessary to consider the case $\Gamma(a)^{(t_b+i)} = 0$ for all values of i from 1 to 4. Similarly, the mask of i in the last round must be $\Gamma(a)^{(t_o-i)} = 0$for all values from 1 to 6. According to this condition is $\Gamma(a)^{(t_b+5)} \neq 0$. Also, $\Gamma(a)^{(t_b+5)}$and $\Gamma(a)^{(t_o-6)}$is active and the value of $\Gamma(a)^{(t_o-6)}$is equal to 0. So, the number of rounds of $t_o - t_b$should be greater than 14. These results and the activity of $\Gamma(a)^{(t_o-6)}$indicate $\Gamma(a)^{(t_o-6)} \neq (0,0)$or $\Gamma(a)^{(t_o-7)} \neq (0,0)$. Therefore, it is AS ≥ 22 in this case.

Results. Resynchronization and related key attack. The resynchronization attack [2,6] is the most effective attack against PTSGs, so it is appropriate to evaluate the NSA algorithm with this method as well. The resynchronization attack can be used not only against the secret key, but also against basic stream cipher generators with a shared parameter. This is an effective attack if the algorithm is very simple to run. Under the assumption that the secret key is unchanged (fixed), the parser first looks for correlations between common parameters and related results. If the correlation probability is high, it can be used to guess the secret key information. For example, linear cryptanalysis in the counter mode of block ciphersis a type of resynchronization attack. The security assessment against the related key attack is similar to resynchronization by replacing the initial vectors with secret keys.

Differential and linear characteristics and integral cryptanalyst [3] options were chosen to evaluate the relationship between NSA inputs and outputs. Attacks against block ciphers using these properties are known as differential cryptanalysis [4] and linear cryptanalysis

[5]. The design of the NSA algorithm generator, especially its functionality, is very similar to the design of block ciphers. This shows that the above two statistical properties are very suitable for estimating the relationship between the initialization vector $I$and the corresponding internal state.

The maximum differential and linear characteristics of the repetition of pOnly the iteration of p is considered and its differential and linear properties are evaluated, ignoring the XOR to the buffer and output arrays. These evaluation methods can be applied similarly to block ciphers.

Table 2 shows the minimum number of active functions in all parts of state arrays for each attack.

Table 2.

The number of active F −functions for linear and differential paths of p

The number of round	…	11	12	13	14	15	16	17	18	19	20	21	22	23
Differensial	…	10	12	12	12	14	16	16	16	18	20	20	20	22
Linear	…	10	12	12	13	14	16	16	17	18	20	20	21	22

Resynchronization attack tolerance: Table 2 shows the relationship between the initialization vector I and the $a(t)$state arrays in $t-$ iterations. This means that with probability 2^{-128} greater than 23 iterations of p, the differential and non-linear properties.

When running the NSA algorithm, after the initialization vector I is set, only the p function is executed 16 times. Then, the p function is executed 16 times in the compound of the function T. However, the buffer affects the differential and linear characteristics of the ba state only after the 9th iteration, namely, 22 iterations after I is set. Therefore, due to the above characteristics, it can be concluded that it is difficult to observe the deviation after t > 0 iterations.

On the other hand, Table 2 shows that there is some relationship between the initialization vector I and some units of the corresponding buffer b in round 0. However, the differential characteristic is that the output is sequenced, and the buffer contains more than two buffer units. The correlation between these units is too small to observe and achieve the desired result. Therefore, it is impossible for the parser to use this correlation. These properties are also relevant for linear cryptanalysis.

Related Key Attack: Correlation between keys and corresponding outputs is more difficult to observe than correlation between initialization vector and corresponding outputs due to the first shuffling step. Thus, any security shortage was not found using differential and linear cryptanalysis.

Integral cryptanalysis. Due to the high byte-oriented structure, some variants of the Integral cryptanalysis attack [3] can be supported against the NSA algorithm. Integral cryptanalysisis currently the most successful attack against block ciphers with an SPN structure (such as Rijndael, AES, Kuznechik).

Integral cryptanalyst is considered against a block cipher is a chosen plain text attack in which the parser selects several related blocks of plain text, each of which usually differs by only one or two bytes. If a byte contains all values, this selected plain text is called the active set. A set is called a fully selected set if it contains all variants of a byte or two bytes that differ. Because of the complete selectivity in the input of a nonlinear function, the parser can expect to handle the intermediate values to some extent. The parser partially determines the intermediate value controlled by the cipher

text due to the fully selected plain text blocks. If the parser can partially determine the encryption key, he (or she) can distinguish between valid and invalid keys from among the possible key variants.

In stream encryption algorithms, the parser must try to choose a different value of the key orinitialization vector values to perform this attack. Therefore, an Integral cryptanalysis attack should be compared to a bound key attack or a chosen initialization vector attack.

Bind Key Attack: Initially, the attack model needs to be defined. It is assumed that the parser does not know the key value. In order to obtain full selectivity, the parser needs to initialize a set of keys, with the selected keys differing only by a portion of the key value. According to these opinions, the attention is paid to parts which are belong to keys, here the keys differ by one or two bytes. The attacker can't observe anything until a sequence of pseudo-random numbers is generated. It is necessary to check whether the parser can find any feature in the our put sequence between a number of iterations.

Fully selected key set. The set fully selected feature is introduced when initializing the buffer. We specify the property of the intermediate word in such a way that in each round the corresponding byte part of the set has a different value. For initialization, the passive elements of the set is marked with O which the value is constant. We also introduce the weakest "balanced" property which we mark with Φ, that is, the sum of all values of the corresponding element of the entire set of XOR is equal to zero. If the element of the set is neither active, nor passive, nor balanced, namely, uncontrollable, it is marked with the $*$. If the word triple (A, B, C) has the properties Λ, O and Φ for words A, B, C, then it is defined as, $(A, B, C) \xrightarrow{p} \Lambda, O, \Phi$ or as $A \xrightarrow{p} \Lambda, B \xrightarrow{p} O, C \xrightarrow{p} \Phi$.

Obviously, the most effective element for introducing saturation is the last word to affect other elements. Therefore, it is necessary to analyze the situation. $a_0, a_1, a_2 \xrightarrow{p} (\Lambda, O, O)$ the outcome of the t −round is determined with (a_0^t, a_1^t, a_2^t). The results of the changes in the characteristics of the set are given in Table 3.

Table 3.

Changes in the properties of a set of elements at intermediate values

Intermediate values	*Properties of elements*
(a_0^0, a_1^0, a_2^0)	(Λ, O, O)
(a_0^1, a_1^1, a_2^1)	(O, O, Λ)
(a_0^2, a_1^2, a_2^2)	(O, Λ, O)
(a_0^3, a_1^3, a_2^3)	$(\Lambda, \Lambda, \Lambda)$
(a_0^4, a_1^4, a_2^4)	(Λ, Φ, Φ)
(a_0^5, a_1^5, a_2^5)	$(\Phi, *, *)$
$(a_0^{6+}, a_1^{6+}, a_2^{6+})$	$(*, *, *)$

So, the initial values of the buffer $b_i$will have the following properties depending on the index i:

$$b_i \xrightarrow{p} \begin{cases} O : i = 15,14 \\ \Lambda : i = 13,12 \\ \Phi : i = 11 \\ * : i = 10,9,8, \dots ,0 \end{cases} \quad (1)$$

It should be noted that the properties in (1) do not mean that the parser can control intermediate values upto b_{11}. In fact, b_{11}can be represented by other buffer

values and the evaluation of F –function (non-linear buffer relations mentioned above). However, due to the randomness in the iterations after the initialization vector is introduced, this feature is eliminated before the output sequence is produced. Therefore, it can be considered that an appropriate key attack based on the Integral cryptanalysis method does not pose any threat to the NSA algorithm.

Resynchronization Attack: This attack may be more practical than cryptanalysis based on key selection above. However, the Initialization Vector does not insert any values into the buffer until 16 iterations of shuffling are complete. Given the number of controlled rounds specified above, 16 iterations are sufficient to eliminate the complete selectivity feature in the selected sets of the shuffling initialization vector.

Conclusion. A new NSA key stream encryption algorithm was proposed in this work. NSA is effective for both hardware and software implementations. Evaluation results of cryptanalytic methods show that NSA is resistant to appropriate key-based attacks and resynchronization attacks. But it is appropriate to evaluate the security of NSA using other analysis methods. Further studies aim to evaluate the security of this algorithm compared to other cryptanalysis methods.

References:

1. J. Daemen, “Cipher and hash function design strategies based on linear and differential cryptanalysis,” Doctoral Dissertation, March 1995, K. U. Leuven
2. J. Daemen, R. Govaerts, J. Vandewalle, “Resynchronization weaknesses in synchronous stream ciphers,” Advances in Cryptology, Proceedings Eurocrypt’93, Springer-Verlag, LNCS 765, pp. 159-169, 1994.
3. J. Daemen, L. Knudsen, V. Rijmen, “The BlockCipher Square,” Fast Software Encryption, Springer-Verlag, LNCS 1267, pp. 149–165, 1997.
4. E. Biham, A. Shamir, “Differential Cryptanalysis of the Data Encryption Standard,” Springer-Verlag, 1993
5. M. Matsui, “Linear cryptanalysis method for DES cipher,” Advances in Cryptology, Eurocrypt’93, Springer-Verlag, LNCS 765, pp. 159–169, 1994.
6. J. Daemen, R. Govaerts, J. Vandewalle, “Resynchronization weaknesses in synchronous stream ciphers,” Advances in Cryptology, Proceedings Eurocrypt’93, Springer-Verlag, LNCS 765, pp. 159-169, 1994.
7. S. Fluhrer, M. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4,” Selected in Areas in Cryptography, SAC 2001, Springer-Verlag, LNCS 2259, pp. 1– 24, 2001.
8. A. Clark, J. Golic, W. Millan, L. Penna, L. Simpson, “The LILI-128 Keystream Generator,” NESSIE project submission, 2000, available at http://www.cryptonessie.org
9. S. Fluhrer, “Cryptanalysis of the SEAL 3.0 Pseudorandom Function Family,” Fast Software Encryption, FSE 2001, Proceedings, pp. 142–151, 2001.
10. T. Jacobsen and L. R. Knudsen, “The Interpolation AttackonBlockCiphers,” Fast Software Encryption, FSE’97, Springer-Verlag, LNCS 1267, pp. 28–40, 1997.
11. P. Rogaway, D. Coppersmith, “A Software-Optimized Encryption Algorithm,” Journal fo Cryptography, Vol. 11, No. 4, pp. 273–287, 1998.
12. P. Rogaway, D. Coppersmith, “A Software-Optimized Encryption Algorithm,” Fast Software Encryption, FSE’94, Springer-Verlag, LNCS 809, pp. 56–63, 1994
13. Suwais K., Samsudin A. New Classification of Existing Stream Ciphers. Universiti Sains Malaysia(USM), Malaysia 2010.
14. M. Hassan Shirali-Shahreza, Mohammad Shirali-Shahreza. (2006) A new approach to persian/arabic text steganography. In: 5th IEEE/ACIS international conference on computer and information science and 1st IEEE/ACIS international

workshop on component-based software engineering, software architecture and reuse, pp 310-315.

15. Bala Krishnan, Prasanth Kumar Thandra, M. Sai Baba (2017). An overview of text steganography. 4th International Conference on Signal Processing, Communications and Networking (ICSCN -2017), March 16 - 18, 2017, Chennai, INDIA.

16. N.R.Zaynalov, U.Kh.Narzullaev, A.N. Muhamadiev, O.N.Mavlonov, J.Kiyamov, D.Qilichev. Hiding Short Message Text in the Uzbek Language. 2020 International Conference on Information Science and Communications Technologies (ICISCT). https://ieeexplore.ieee.org/document/9351521.

17. Por LY, Delina B (2008) Information hiding—a new approach in text steganography. In:7th WSEAS international conference on applied computer and applied computational science. Hangzhou China,689

18. N.R. Zaynalov, O.N. Mavlonov, A.N. Muhamadiev, D. Kilichev, I.R. Rakhmatullayev. UNICODE For Hiding Information In A Text Document. 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT) | 978-1-7281-7386-3/20 IEEE | DOI: 10.1109/AICT50176.2020.9368819

19. Akbarov D.Ye. Cryptographic methods of information security and their application. - Tashkent, "Mark of Uzbekistan" publishing house, 2009. - 432 p

20. Kharin Yu.S., Bernik V.I., Matveev G.V., Agievich S.V. Mathematical and computer bases of cryptology: Textbook. - Minsk, LLC "New Knowledge", 2003. - 382 p.

COORDINATION OF THE MOVEMENT OF TRANSPORT TYPES IN AREAS WITH HIGH PASSENGER FLOW

MOMINOV TOLKIN

Assistant of Tashkent State Transport University
E-mail: tulqinmuminov643@gmail.com, phone.: (+99890) 168-68-35

YULDOSHEV DAVRON

Doctoral student of Tashkent State Transport University
E-mail: davron.yoldoshev@bk.ru, phone.: (+99897) 411-41-69

Abstract:

Objective. In providing reliable transport service to passengers, it is important to coordinate the movement of transport types taking into account the flow of passengers. In the article, the literature on this issue is analyzed and tasks for mutual coordination of the movement of transport types are determined based on experience.

Methods. Traffic schedules were analyzed to coordinate the traffic of surface public transport and metropolitan routes, types of transport. The main issue is to determine the results of the distribution of the total time of the passenger during the journey and the components of this time.

Results. A general mathematical expression of the arrival time of the passenger at the destination is derived. Research was conducted using the expression and the results were analyzed. The indicators of the passenger's movement as a pedestrian in reaching his destination and his movements in the transport as a passenger were made on the basis of the "Geo Tracker" program. The results of the distribution of the total travel time to the passenger's destination and the components of this time were obtained.

Keywords: Railway, bus, metro, transport infrastructure, transport links, simulation model, transport system, station, bus station.

Introduction. Today, many measures are being implemented to improve public transport infrastructure development. In particular, based on the decision of the President of the Republic of Uzbekistan №. PQ-111 dated February 2,

CONTENTS