A New Hardware-Implemented Stream Encryption Algorithm

Rakhmatullayev Ilkhom Rakhmatullayevich[a]
Samarkand branch of the Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Samarkand, Uzbekistan
Ilhom9001@gmail.com

Keywords: AND, XOR, NHSA, algebraic cryptanalysis, differential cryptanalysis, algorithm cycle, NIST statistical tests.

Abstract: This article describes a streaming encryption algorithm called NHSA consisting of a total length of 3 shift registers, 6 AND elements and 14 XOR elements. The results of evaluating sequences developed using this algorithm for the level of randomness and conclusions about the portability of certain cryptanalysis methods, as well as information about the number of elements required to implement the algorithm on hardware devices are presented.

1 INTRODUCTION

Stream cipher algorithms are a category of encryption techniques that encrypt plaintext one bit or byte at a time, creating a stream of encrypted data. They are distinguished by their use of a keystream generator, which produces a pseudo-random sequence of bits known as the gamma key. Each bit of the plaintext is encrypted by performing an XOR (exclusive OR) operation with the corresponding bit of the gamma key (SHnajer, 2003).

$$c_i = p_i \oplus k_i \quad (1.1)$$

Exactly, the decryption process in stream cipher algorithms mirrors the encryption process, leveraging the symmetry of the XOR operation to recover the plaintext from the ciphertext. This symmetry hinges on both the sender and the recipient using the same pseudo-random keystream, which is generated by a keystream generator initialized with a shared secret symmetric key.

$$c_i \oplus k_i = p_i \oplus k_i \oplus k_i = p_i \quad (1.2)$$

Absolutely, the resilience of cryptosystems employing stream ciphers against different types of attacks hinges significantly on the properties of the keystream generator at the heart of the algorithm. The security and effectiveness of these ciphers are closely tied to two critical aspects of the generator: the period of the generated sequence and its degree of randomness. If the generator produces the same sequence in each session or has a short repetition period, it becomes possible to XOR two encrypted texts using the XOR operation, resulting in the XOR sum of two plaintexts $p_1 \oplus p_2$. Comparing the complexity of decrypting stream cipher encrypted text to that of decrypting a polyalphabetic cipher sheds light on how advances in cryptanalysis influence the security assessment of cryptographic algorithms. Polyalphabetic ciphers, such as the Vigenère cipher, are a step up from simple monoalphabetic substitution ciphers because they use multiple substitution alphabets, significantly increasing their resistance to frequency analysis. However, with the advent of more sophisticated cryptanalytic techniques, even the complexity of polyalphabetic ciphers has been effectively reduced, making them more vulnerable to decryption without the key (Knellwolf, Meier, Naya-Plasencia, 2011; De Canni`ere, Preneel, 2008; Quedenfeld, Wolf, 2014; Hitachi, 2001; Watanabe, Furuya, Yoshida, Takaragi, Preneel, 2002; Dawson, Carter, Gustafson, Henricksen, Millan, Simpson, 2002; Wong, Carter, Dawson, 2010; Orumiehchiha, Pieprzyk, Shakour, Steinfeld, 2013; Lee, Lee, Park, 2008; Zaynalov, 2020; Zaynalov, 2021; Zaynalov, 2020; Zaynalov, 2020;Rakhmatullaevich, Mardanokulovich, 2024; Kilichev, 2024; Huang, Susilo, Seberry, 2010).

The NHSA stream cipher algorithm, with its potential for hardware implementation, signifies a

[a] https://orcid.org/0000-0002-4872-4265

strategic development in the realm of cryptographic algorithms. Implementing cryptographic algorithms in hardware can offer several advantages over software implementations, including increased speed, lower power consumption, and enhanced security against certain types of attacks.

2 METHODS

The description of the New Hardware Stream Algorithm (NHSA) outlines a promising approach to stream ciphering that combines both hardware efficiency and cryptographic robustness. The initial state configuration, involving three shift registers with a total length of 269 bits, alongside the use of a 128-bit key and an initialization vector (IV), suggests a carefully designed mechanism to ensure both security and performance. The execution strategy of the New Hardware Stream Algorithm (NHSA), running the algorithm 4 times for each bit in the initial state, leading to a total of 1076 iterations, is an intriguing approach to ensuring a thoroughly mixed and secure initial setup. This method highlights a comprehensive mixing and diffusion strategy, aimed at maximizing the cryptographic strength of the cipher before actual data encryption begins.

The operational mechanism outlined for the New Hardware Stream Algorithm (NHSA) after its initialization phase showcases the fundamental principles of stream cipher encryption and decryption, focusing on the generation of keystream bits and their combination with the plaintext and ciphertext via XOR operations. This process encapsulates the core of stream ciphering, highlighting both its simplicity and elegance in securing data.

3 INITIALIZATION

To illustrate the initialization process of the New Hardware Stream Algorithm (NHSA) with a focus on loading the 128-bit key and the 128-bit initialization vector (IV) into the 269-bit shift registers, let's consider a structured approach to pseudocode. This process involves distributing these values across the registers, ensuring the algorithm begins with a richly mixed initial state. The total length of the shift registers is broken down into three distinct sizes: 89 bits, 83 bits, and 97 bits, respectively.

For the case where the indices of register cells are expressed sequentially:

$(K_0, K_1, \dots, K_{79}, 0, \dots, 0) \rightarrow (a_0, a_1, a_2, \dots, a_{88})$
$(I_0, I_1, \dots, I_{79}, 0,0,0) \rightarrow (a_{89}, a_{90}, a_{91}, \dots, a_{171})$
$(K_{80}, \dots, K_{127}, I_{80}, \dots, I_{127}, 0) \rightarrow (a_{172}, a_{173}, \dots, a_{268})$
$for\ i = 0\ to\ 1075\ do$
$a_{55} + a_{80} \cdot a_{81} + a_{82} + a_{72} \cdot a_{74} + a_{161} \rightarrow t_0$
$a_{151} + a_{164} \cdot a_{165} + a_{166} + a_{155} \cdot a_{157} + a_{259} \rightarrow t_1$
$a_{232} + a_{263} \cdot a_{264} + a_{265} + a_{242} \cdot a_{244} + a_{68} \rightarrow t_2$
$(t_2, a_0, \dots, a_{87}) \rightarrow (a_0, \dots, a_{88})$
$(t_1, a_{89}, \dots, a_{170}) \rightarrow (a_{89}, \dots, a_{171})$
$(t_0, a_{172}, \dots, a_{267}) \rightarrow (a_{172}, \dots, a_{268})$
$end\ for.$

For the case where the indices of register cells are expressed in a separate order for each register:

$(K_0, K_1, \dots, K_{79}, 0, \dots, 0) \rightarrow (a_0, a_1, a_2, \dots, a_{88})$
$(I_0, I_1, \dots, I_{79}, 0,0,0) \rightarrow (b_0, b_1, b_2, \dots, b_{82})$
$(K_{80}, \dots, K_{127}, I_{80}, \dots, I_{127}, 0) \rightarrow (c_0, c_1, \dots, a_{96})$
$for\ i = 0\ to\ 1075\ do$
$a_{55} + a_{80} \cdot a_{81} + a_{82} + a_{72} \cdot a_{74} + b_{72} \rightarrow t_0$
$b_{62} + b_{75} \cdot b_{76} + b_{77} + b_{66} \cdot b_{68} + c_{87} \rightarrow t_1$
$c_{60} + c_{91} \cdot c_{92} + c_{93} + c_{70} \cdot c_{72} + a_{59} \rightarrow t_2$
$(t_2, a_0, \dots, a_{87}) \rightarrow (a_0, \dots, a_{88})$
$(t_1, b_0, \dots, a_{81}) \rightarrow (b_0, \dots, b_{82})$
$(t_0, c_0, \dots, c_{95}) \rightarrow (c_0, \dots, c_{96})$
$end\ for.$

4 GENERATION

The stream generation phase of the New Hardware Stream Algorithm (NHSA) leverages a fascinating and intricate design, ensuring that the key and initialization vector (IV) significantly influence the generated keystream. This approach, using 21 specific bits from the 269-bit state to modify 3 bits of the state and compute 1 bit of the keystream, underscores a sophisticated method of ensuring security and randomness in the cipher. The process of generating an N-bit key N ($N \leq 2^{64}$) required for encryption can be expressed using the following pseudocode.

For the case where the indices of register cells are expressed sequentially:

$for\ i = 0\ to\ N\ do$
$a_{55} + a_{82} \rightarrow t_0$
$a_{151} + a_{166} \rightarrow t_1$
$a_{232} + a_{264} \rightarrow t_2$
$t_0 + t_1 + t_2 \rightarrow exit_i$
$a_{55} + a_{80} \cdot a_{81} + a_{82} + a_{161} \rightarrow t_0$

$a_{151} + a_{164} \cdot a_{165} + a_{166} + a_{259} \rightarrow t_1$
$a_{232} + a_{263} \cdot a_{264} + a_{265} + a_{68} \rightarrow t_2$
$(t_2, a_0, \dots, a_{87}) \rightarrow (a_0, \dots, a_{88})$
$(t_1, a_{89}, \dots, a_{170}) \rightarrow (a_{89}, \dots, a_{171})$
$(t_0, a_{172}, \dots, a_{267}) \rightarrow (a_{172}, \dots, a_{268})$
$end\ for.$

For the case where the indices of register cells are expressed in a separate order for each register:

$for\ i = 0\ to\ N\ do$
$a_{55} + a_{82} \rightarrow t_0$
$b_{62} + b_{77} \rightarrow t_1$
$c_{60} + c_{93} \rightarrow t_2$
$t_0 + t_1 + t_2 \rightarrow exit_i$
$a_{55} + a_{80} \cdot a_{81} + a_{82} + a_{72} \cdot a_{74} + b_{72} \rightarrow t_0$
$b_{62} + b_{75} \cdot b_{76} + b_{77} + b_{66} \cdot b_{68} + c_{87} \rightarrow t_1$
$c_{60} + c_{91} \cdot c_{92} + c_{93} + c_{70} \cdot c_{72} + a_{59} \rightarrow t_2$
$(t_2, a_0, \dots, a_{87}) \rightarrow (a_0, \dots, a_{88})$
$(t_1, b_0, \dots, a_{81}) \rightarrow (b_0, \dots, b_{82})$
$(t_0, c_0, \dots, c_{95}) \rightarrow (c_0, \dots, c_{96})$
$end\ for.$

In the context of stream ciphers and bitwise operations, using XOR and AND operations as analogs for addition and multiplication modulo 2 aligns perfectly with the principles of finite field arithmetic, specifically within the field GF(2). In this finite field, the elements are {0, 1}, and the addition and multiplication operations are defined as XOR and AND respectively. This alignment is due to the properties of XOR and AND operations mirroring those of addition and multiplication in a modulo-2 system. Using this algorithm, it is recommended to generate keys of length 2^{64} bits using a single key and initialization vector.

Below is an example of the correct organization of the generation process using the algorithm.

Input Parameters:

*Key:*00011100000001100011011000011001000010110001001001100000001000110011101100110101000100100101111100011110000111010000111000101111

*IV:*11110000111000001101000011000000101100001010000010010000100000000111000001100000000101010000000011000000100000000100000000000000

Values of the registers before the initialization process:

*a:*00011100000001100011011000011001000010110001001001100000001000110011101100110100000000000

*b:*11110000111000001101000011000000101100000101000000100100001000000011100000011000000000

*c:*00010010010111110001111000011101000011100010111101010100000000110000000100000000010000000000000000

Values of the registers after the initialization process:

*a:*10110010010011010110101001101110110101100011110000010010111111100011101011011001011010000

*b:*01001110011100100110011000110100011010000011101000111010000110110101010011011110100

*c:*00000010000010101001101000100100111100100101111101001010001110101001001000100111111011010000000000

*output:*00100101110100111110111110111011010110110110000110001100101000010110101000001110…

5 RESULTS

The New Hardware Stream Algorithm (NHSA) is conceptualized as a state-of-the-art solution catering to the modern demands of cryptographic security, particularly in environments where hardware resources are at a premium. Its design philosophy, focusing on compactness, energy efficiency, and speed, makes it especially suitable for a wide array of applications, from embedded systems to IoT devices and high-speed communication channels.

The requirement for a compact implementation of the New Hardware Stream Algorithm (NHSA), emphasizing a bit-oriented approach, non-linearity, energy efficiency, and rapid processing, sets a clear direction for its design and optimization strategies. These considerations are crucial for the algorithm's suitability in constrained environments, where the trade-offs between security, performance, and resource utilization are most acute.

Based on the figures provided in (ano, J., Mentens, Preneel, Verbauwhede, 2004) (i.e., 12 NAND gates for each flip-flop (shift register element), 2.5 NAND gates for XOR, and 1.5 NAND gates for AND), calculate the number of gates required for a possible hardware implementation. Comparative analysis results with the Trevium algorithm (De Canniere, Preneel, 2008), which is one of the top stream cipher algorithms participating in the EStream competition, are presented in Table 1.

Table 1: The number of gates required for the hardware implementation of the NHSA and Trevium algorithms is as follows.

Algorithm	Trevium	NHSA
Key length	80	128
IV length	80	128
Internal status register	288*12=3456	269*12=3228
AND gate	3*1.5=4.5	6*1.5=9
XOR gate	11*2.5=27.5	14*2.5=35
Total number of gates	**3488**	**3272**

Period of NHSA algorithms.

As the internal state of the algorithm evolves non-linearly, determining its exact period is challenging. However, the period of the algorithm's operation can be estimated through a series of observations. Firstly, it can be shown that any key/IV pair can produce a stream with a period of at least 2^{83-3}-1 if a complete linear circuit without NAND gates is obtained. This doesn't directly impact NHSA but can be considered an indication that the register lengths have been chosen correctly.

Secondly, the internal state of NHSA is updated in reverse order, and the initialization of the register $(c_0, .., c_{96})$ prevents state changes in fewer than 96 iterations. Assuming that NHSA behaves like a random permutation after a sufficient number of iterations (in this case, 269*4=1076 iterations), all cycle lengths up to 2^{269} are equally likely. Thus, for this key/IV pair, the probability that it generates a cycle shorter than 2^{128} is $2^{269-128}=2^{141}$.

However, it's recommended to generate a 2^{64} bit key using a single key/IV pair to ensure maximum reliability of the generated keys.

An algebraic attack on the NHSA algorithm.

At first glance, NHSA may appear to be a lightweight algorithm that could be efficiently targeted by algebraic attack methods. The complete scheme could be represented by very sparse low-order equations. However, due to the non-linear evolution of the algorithm's internal state, applying effective linearization methods used to solve equation systems created for schemes based on sufficiently long selected registers like LFSRs can be quite challenging.

Nevertheless, conducting research to draw conclusions about the algebraic properties of the algorithm would be prudent. It's known that the initialization process in the NHSA algorithm consists of 1076 steps. When applying algebraic cryptanalysis to this algorithm, the main goal is to construct and solve algebraic equations representing register cells after a certain iteration. From the following equations, it can be observed that in iterations 1 and 2, the bits of the key and initialization vector are involved in the equations representing the register cells.

Algebraic equation for iteration-1:

$$t_0 = k_{55} + 0 \cdot 0 + 0 + k_{72} \cdot k_{74} + IV_{72}$$
$$t_1 = IV_{62} + IV_{75} \cdot IV_{76} + IV_{77} + IV_{66} \cdot IV_{68} + IV_{119}$$
$$t_2 = IV_{92} + IV_{123} \cdot IV_{124} + IV_{125} + IV_{102} \cdot IV_{104} + k_{68}$$

Algebraic equation for iteration-2:

$$t_0 = k_{54} + 0 \cdot 0 + 0 + k_{71} \cdot k_{73} + IV_{71}$$
$$t_1 = IV_{61} + IV_{74} \cdot IV_{75} + IV_{76} + IV_{65} \cdot IV_{67} + IV_{118}$$
$$t_2 = IV_{91} + IV_{122} \cdot IV_{123} + IV_{124} + IV_{101} \cdot IV_{103} + k_{67}$$
$$(t_0, a_{172}, \dots, a_{267}) \rightarrow (a_{172}, \dots, a_{268})$$

Based on the formula provided, where the value of t_0 calculated from a quadratic nonlinear equation is written to the 172nd-order register cell before the 2nd iteration, the equations representing the cells of the third register begin to take on a quadratic form. Since the length of the third register is 97, 97 cycles are sufficient to transition this register to a quadratic form. In the second register, starting from the 63rd cycle, and in the first register, starting from the 73rd cycle, the algebraic level of equations representing the cells begins to increase. Thus, to achieve a complete quadratic representation, 73 + 89 = 162 iterations are required. The key bits are fully involved in the unknowns representing the equation system. The following table shows the change in the level of the equation system and the number of unknowns depending on the number of initialization process cycles:

Table 2: Parameters of the equations formed using the algebraic cryptanalysis method for the NHSA algorithm.

The minimum level of the equation system.	**Iteration step**	**The number of unknowns**	**Complexity $O(n^3)$**
Level-2	162	2^{14}	2^{42}
Level-3	162+97=259	2^{21}	2^{63}
Level-4	259+97=356	2^{28}	2^{84}
Level-5	356+97=453	2^{35}	2^{105}
Level-6	453+97=550	2^{42}	2^{126}
Level-7	550+97=647	2^{49}	2^{147}

It is evident that every 97 cycles after the 162nd iteration, the level of equations representing the register cells in algebraic form increases. From the Table 2 above, it is clear that the

NHSA encryption algorithm is resistant to algebraic cryptanalysis due to the sharp increase in the number of unknowns (2^{49}) and the complexity of solving (2^{147}) after the 647th iteration of the initialization process.

Correlation attack on the NHSA algorithm.

In the security analysis of a synchronous stream cipher, cryptanalysts typically consider two different types of correlations. The first type is the correlation between the key stream bits and linear combinations of bits in the internal state, which potentially could lead to a complete state recovery. The second type of correlation attack involves the correlation between the key stream bits themselves.

It is evident that finding a linear correlation between the current key bits and the bits of the internal state is straightforward, as the i output bit is assumed to be equal to ⟦output ⟧ _i=a_55+a_82+b_62+b_77+c_60+c_93. However, unlike other ciphers based on LFSR , the internal state of NHSA changes non-linearly, making the process of combining these equations for efficient state recovery a rather complex problem.

The simplest way to find correlations of the second type is to traverse the linear paths of the cipher and set the output values of all encountered AND gates close to 0. However, the organization of operations in NHSA is designed in such a way that any path leads to the complete prediction of the values of at least 144 AND gate outputs in this specific type of observation. Below is an example of a linear combination to determine the correlation of key stream bits:

If we assume that the correlation of this linear combination is fully expressed in the considered specific manner, this equation will have a correlation coefficient of 2–144. To determine such a correlation, at least 2288 key stream bits are required, which significantly exceeds the security requirements and the fu6`ll key selection method (2128).

6 DISCUSSION

Comparing the differential cryptanalysis method with the NHSA algorithm.

The idea of conditional differential cryptanalysis was expanded and applied to constructions based on NLFSR (Non-Linear Feedback Shift Registers).

In the main concept of the differential cryptanalysis method, when applied to the NHSA algorithm, it involves observing two key values that differ only by a few bits and determining the probability of the difference value after each iteration. The difference between the key values is referred to as the increment.

Let the considered increment be Δf(k,x) and the 1-bit difference x in the i-th bit state be e_i. By neutrality of x_i in Δf, we understand the probability that Δf(k,x)=Δf(k,x⊕e_i) for a random key k. Using a single neutral variable as a differentiator requires estimating Δf at least twice. If the neutrality of x_i is p, then the overall probability is |1/2 - p|.

To determine the conditions for data collection, the following algorithm proposed in (Knellwolf, Meier, Naya-Plasencia, 2010) was utilized:

Input: $a_1, a_2, \dots, a_d, r$
$J \leftarrow \emptyset$
foreach $a \in \{a1, \dots, ad\}$ **do**
for $i \leftarrow 0$ **to** r - 1 **do**
$f \leftarrow \triangle_a s_i(k, x) \bmod J$
if $f = 1$ **then**
add f to J
return J

In this case $a_1, a_2, \dots, a_d$ are binary vectors defined in space F_2^n.

During the analysis, we utilize a 24-bit adder. Conditions for 24 differences were determined using the algorithm mentioned above. We use the abbreviation $\triangle z_j = \triangle^{(24)}_{a_1, a_2, \dots, a_d} z_j$, here $z_j - j$ refers to the key stream generated in this iteration.

Using the algorithm provided above, conditions were analyzed for the case r = 200, where each increment is controlled for the first 200 iterations. After processing the first increment, the first increment (increment at point x_0) is generated as follows:

$$J = \langle x_0, x_{10}x_{11} + x_{13}, x_{13}x_{14} + x_{15}, x_{76} + k_{64}, x_{61} + x_{74}x_{75} + x_{74}k_{63} + x_{75}k_{62} + k_{49} + k_{62}k_{63} + k_{74}k_{75} + k_{76}, x_{63} + k_{51} + k_{76}k_{77} + k_{78}, k_{11}k_{12} + k_{13} + k_{55}, k_{13}k_{14} + k_{15} + k_{57}\rangle$$

At different algebraic parameters J, all pairs (k, x) have the same differential characteristic with respect to a_1 up to $r = 200$ periods. After processing all the increments, the value of J looks as follows:

$$J = \langle x_0, x_1, x_3, x_4, x_6, x_7, x_9, x_{10}, x_{12}, x_{13}, x_{15}, x_{16}, x_{18}, x_{19}, x_{21}, x_{22}, x_{24}, x_{25}, x_{27},$$
$$x_{28}, x_{30}, x_{31}, x_{33}, x_{34}, x_{36}, x_{37}, x_{39}, x_{40}, x_{42}, x_{43}, x_{45}, x_{46}, x_{48}, x_{49}, x_{51}, x_{52}, x_{54},$$
$$x_{55}, x_{57}, x_{58}, x_{60}, x_{61}, x_{63}, x_{64}, x_{66}, x_{67}, x_{69}, x_{70}, x_{72}, x_{73}, x_{75}, x_{76}, x_{78}, x_{79}, x_{81},$$
$$x_{82}, x_{84}, x_{85}, x_{87}, x_{88}, x_{90}, x_{91}, x_{93}, x_{94}, x_{96}, x_{97}, x_{99}, x_{100}, x_{102}, x_{103}, x_{105}, x_{106},$$
$$x_{108}, x_{109}, x_{111}, x_{112}, x_{114}, x_{115}, x_{117}, x_{118}, x_{120}, x_{121}, x_{123}, x_{124}, x_{126}, x_{127},$$
$$k_0, k_1, k_3, k_4, k_6, k_7, k_9, k_{10}, k_{12}, k_{13}, k_{15}, k_{16}, k_{18}, k_{19}, k_{21}, k_{22}, k_{24}, k_{25}, k_{27},$$
$$k_{28}, k_{30}, k_{31}, k_{33}, k_{34}, k_{36}, k_{37}, k_{39}, k_{40}, k_{42}, k_{43}, k_{45}, k_{46}, k_{48}, k_{49}, k_{51}, k_{52}, k_{54},$$
$$k_{55}, k_{57}, k_{58}, k_{60}, k_{61}, k_{63}, k_{64}, k_{66} + 1, k_{67}, k_{69}, k_{70}, k_{72}, k_{73}, k_{75}, k_{76}, k_{78}, k_{79}, k_{81},$$
$$k_{82}, k_{84}, k_{85}, k_{87}, k_{88}, k_{90}, k_{91}, k_{93}, k_{94}, k_{96}, k_{97}, k_{99}, k_{100}, k_{102}, k_{103}, k_{105}, k_{106},$$
$$k_{108}, k_{109}, k_{111}, k_{112}, k_{114}, k_{115}, k_{117}, k_{118}, k_{120}, k_{121}, k_{123}, k_{124}, k_{126}, k_{127}\rangle.$$

Only the bits $x_{61}, x_{64}, x_{67}, x_{70}$ in x are not specified and do not affect the increment. When all other variables x_i are set to zero, this makes them candidates for neutral bits for $\triangle z_j$. Empirical results confirm that they are likely to be neutral up to the 537th round. Table 3 shows the neutrality obtained in the experiment with 100 random keys. Note that neutrality equal to zero or one means that $\triangle z_j$ is linear in the corresponding variable (which can be used as an amplification coefficient with neutrality).

Table 3: Neutrality indicators defined for the bits $x_{61}, x_{64}, x_{67}, x_{70}$.

□	□$_{61}$	□$_{64}$	□$_{67}$	□$_{70}$
501	1.0	1.0	1.0	1.0
514	0.10	0.15	0.05	0.10
525	0.25	0.30	0.20	0.30
537	0.40	0.35	0.35	0.40
550	0.495	0.502	0.499	0.505

If experiments are conducted using weak keys, these neutrality values can be preserved until the iteration process (550+162=712), which is necessary before the complete update of the algorithm's register values. In subsequent cycles, the neutrality values approach 0.5. This parameter provides tolerance in statistical cryptanalysis methods. Therefore, it is sufficient to conclude the resilience of the NHSA stream cipher to differential cryptanalysis from the 712th cycle of the initialization loop.

Resynchronization Attack on the NHSA Algorithm.

Another type of attack is the resynchronization attack, where an attacker is allowed to manipulate the IV value and tries to gain information about the key by checking the corresponding key stream. The NHSA algorithm attempts to prevent this type of attack by performing a sufficient number of iterations in the initialization process before generating the key used for encryption. It can be shown that each bit of the state is nonlinearly dependent on each key and IV bit after two full cycles (i.e., $2 \cdot 269$ iterations). It can be noted that an additional two cycles are sufficient to protect the cipher from resynchronization attacks.

NHSA is a simple synchronous stream encryption algorithm that is well suited for applications requiring flexible hardware implementation. Despite the general rule to avoid sparse update functions at each algorithm iteration, only a few state bits are used. As a result, prediction and attack detection are undoubtedly complicated. A direct attack involves guessing the common 128-bit secret key (presumably the open key IV). Further research is needed to determine the extent to which more complex attacks can reduce this number. The conducted studies are crucial to draw conclusions about its security compliance.

Since the NHSA algorithm is analogous to the Trivium encryption algorithm in terms of its operation and structure, the evaluation results of cryptanalysis methods were compared with the Trivium algorithm (Table 4).

Table 4: The evaluation results of the Trivium and NHSA algorithms for cryptanalysis methods are as follows.

Algorithm	**Trivium**	**NHSA (this case)**
Algebraic cryptanalysis	$2^{42.2}$ is unknown in the 625th iteration of the initialization cycle 110	2^{42} is unknown in the 550th iteration of the initialization cycle

Differential cryptanalysis	Resilient after the 961st iteration of the initialization cycle.	Resilient after the 712th iteration of the initialization cycle.
Resynchronization attack	2^{80} need to choose	2^{128} need to choose
Correlational cryptanalysis	A key stream of 2^{144} bits is required to determine the correlation coefficient of 2^{-72}	A key stream of 2^{-144} bits is required to determine the correlation coefficient of 2^{288}

The results of evaluating sequences generated by the NHSA algorithm using NIST statistical tests for randomness.

The sequences generated using the mentioned key and IV in the NHSA algorithm were evaluated for randomness using NIST statistical tests. Below are the results of the approximate entropy test (Figure 1) and a total of 15 tests (Figure 2):

```
TEST: approximate_entropy_test
  n         =  2000000
  m         =  3
  Pattern 1 of 8, count = 249284
  Pattern 2 of 8, count = 250346
  Pattern 3 of 8, count = 249738
  Pattern 4 of 8, count = 250015
  Pattern 5 of 8, count = 250346
  Pattern 6 of 8, count = 249407
  Pattern 7 of 8, count = 250015
  Pattern 8 of 8, count = 250849
  phi(3)    = -4.382025
  Pattern 1 of 16, count = 124270
  Pattern 2 of 16, count = 125014
  Pattern 3 of 16, count = 124951
  Pattern 4 of 16, count = 125395
  Pattern 5 of 16, count = 125373
  Pattern 6 of 16, count = 124365
  Pattern 7 of 16, count = 124513
  Pattern 8 of 16, count = 125502
  Pattern 9 of 16, count = 125014
  Pattern 10 of 16, count = 125332
  Pattern 11 of 16, count = 124787
  Pattern 12 of 16, count = 124620
  Pattern 13 of 16, count = 124973
  Pattern 14 of 16, count = 125042
  Pattern 15 of 16, count = 125502
  Pattern 16 of 16, count = 125347
  phi(3)    = -5.075169
  AppEn(3)  = 0.693144
  ChiSquare =  11.61932763649176
  PASS
  P=0.16901368121036756
```

Figure 1: The calculated result of the entropy test.

```
SUMMARY
-------
monobit_test                          0.3828976248514478 PASS
frequency_within_block_test           0.6354841415737159 PASS
runs_test                             0.48512504500511644 PASS
longest_run_ones_in_a_block_test      0.4084594094693384 PASS
binary_matrix_rank_test               0.08050322562050827 PASS
dft_test                              0.6543472577764083 PASS
non_overlapping_template_matching_test 1.00000828105159   PASS
overlapping_template_matching_test    0.03940335533694327 PASS
maurers_universal_test                0.9991531960785148 PASS
linear_complexity_test                0.6468061465110148 PASS
serial_test                           0.16944057962301154 PASS
approximate_entropy_test              0.16901368121036756 PASS
cumulative_sums_test                  0.16791543598886172 PASS
random_excursion_test                 0.04385947941605159 PASS
random_excursion_variant_test         0.12339191184909339 PASS
```

Figure 2: Results of evaluation of NHSA algorithm by Nist statistical test.

From these results, it can be observed that the sequences generated using the NHSA algorithm meet the security requirements in terms of randomness.

7 SUMMARY

The description of the NHSA stream cipher algorithm, consisting of 3 shift registers with a total length of 269 bits, 6 AND gates, and 14 XOR gates, is provided. The evaluation results of sequences generated using the proposed algorithm in terms of randomness are presented, along with conclusions about the applicability of various cryptanalysis methods and information about the required hardware resources for implementing the algorithm.

It has been established that the NHSA encryption algorithm is resistant to algebraic cryptanalysis, differential cryptanalysis, correlation cryptanalysis, and resynchronization attacks. The sequences generated by the algorithm satisfy randomness criteria based on the evaluation of randomness levels using NIST statistical tests.

REFERENCES

SHnajer, B., 2003. Prikladnaya kriptografiya. Protokoly, algoritmy, iskhodnye teksty na yazyke Si. 816 b.

De Canniere, C., Preneel, B., 2008. Trivium – A Stream Cipher Construction Inspired by Block Cipher Design Principles.

Lano, J., Mentens, N., Preneel, B., Verbauwhede, I., 2004. Power Analysis of Synchronous Stream Ciphers with Resynchronization Mechanism. ECRYPT Workshop, SASC – The State of the Art of Stream Ciphers, pp. 327–333.

Knellwolf, S., Meier, W., Naya-Plasencia, M., 2010. Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg.

Knellwolf, S., Meier, W., Naya-Plasencia, M., 2011. Conditional Differential Cryptanalysis of Trivium and KATAN, 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12.

De Canni`ere, Ch., Preneel, B., 2008. Trivium. In New Stream Cipher Designs - The eSTREAM Finalists, pages 244–266.

Quedenfeld, F.-M., Wolf, Ch., 2014. Advanced Algebraic Attack on Trivium, Cryptology ePrint Archive.

Meiser, G., Eisenbarth, Th.,Lemke-Rust, K., Paar, Ch., 2008. Efficient implementation of eSTREAM ciphers on 8-bit AVR microcontrollers. 58 - 66.

Hitachi, Ltd., 2001. MUGI Pseudorandom Number Generator, Self-evaluation Report Ver. 1.1 12. 18, Copyright c 2001 Hitachi, Ltd.

Watanabe, D., Furuya, S., Yoshida, H., Takaragi, K., Preneel, B., 2002. A New Keystream Generator MUGI. In: Daemen, J., Rijmen, V. (eds) Fast Software Encryption. FSE 2002. Lecture Notes in Computer Science, vol 2365. Springer, Berlin, Heidelberg.

Dawson, E., Carter, G., Gustafson, H., Henricksen, M., Millan, B., Simpson, L., 2002. Evaluation of the MUGI Pseudo-Random Number Generator, Information Security Research Centre Queensland University of Technology, July 31.

Wong, K. K.-H., Carter, G., Dawson, E., 2010. An Analysis of the RC4 Family of Stream Ciphers against Algebraic Attacks, 8th Australasian Information Security Conference (AISC 2010), Brisbane, Australia.

Orumiehchiha, M.A., Pieprzyk, J., Shakour, E., Steinfeld, R., 2013. Cryptanalysis of RC4(n, m) Stream Cipher, Proceedings of the 6th International Conference on Security of Information and NetworksNovember. Pages 165–172.

Lee, J.-K., Lee, D.H., Park, S., 2008. Cryptanalysis of Sosemanuk and SNOW 2.0 Using Linear Masks, International Conference on the Theory and Application of Cryptology and Information Security. https://api.semanticscholar.org/CorpusID:18719819

Zaynalov, N. R., 2020. UNICODE For Hiding Information In A Text Document. 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT). C. 1-5.

Zaynalov, N. R., 2021. Analysis of the Frequency of Letters of the Uzbek Language Based on the Cyrillic Alphabet. 2021 International Conference on Information Science and Communications Technologies (ICISCT). C. 1-4.

Zaynalov, N. R., 2020. Combining Invisible Unicode Characters To Hide Information In A Text Document. JOIV: *International Journal on Informatics Visualization*. T. 4. №. 3. C. 148-153.

Zaynalov, N. R., 2020. Manipulating The Rotation Of Ms Excel Cells To Hide Information. *International Journal of Advanced Science and Technology*. T. 29. №. 8s. C. 2270-2273.

Rakhmatullaevich, R. I., Mardanokulovich, I. B., 2024. Analysis of cryptanalysis methods applied to stream encryption algorithms. Artificial Intelligence, Blockchain, Computing and Security Volume 1. CRC Press. C. 393-401.

Kilichev, D., 2024. Errors in SMS to hide short messages. Artificial Intelligence, Blockchain, Computing and Security Volume 2. CRC Press, C. 735-740.

Huang, J., Susilo, W., Seberry, J., 2010. Differential Fault Analysis of LEX. 6280. 55-72.