CASC 3N vs. 4N: Effect of Increasing Cellular Automata Neighborhood Size on Cryptographic Strength

Fatima Ezzahra Ziani[1], Anas Sadak[2], Charifa Hanin[3], Bouchra Echandouri[4], Fouzia Omary[5]
Computer Science Department
University Mohammed V
Rabat, Morocco

***Abstract*—Stream ciphers are symmetric cryptosystems that rely on pseudorandom number generators (PRNGs) as a primary building block to generate a keystream. Stream ciphers have been extensively studied and many designs were proposed throughout the years. One of the popular designs used is the combination of linear feedback shift registers (LFSRs) and nonlinear feedback shift registers (NFSRs). Although this design is suitable for both software and hardware implementation and provides a good randomness behavior, it is still subject to attacks such as fault attacks and correlations attacks. Cellular automata (CAs) based stream ciphers are another design class that has been proposed. CAs display good cryptographic properties as well as a good randomness behavior, also high computational speed and a higher level of security. The use of CAs as cryptographic primitives is not recent and has been thoroughly investigated, especially the use of three-neighborhood one-dimensional cellular automata. In this article, the authors investigate the impact of increasing the neighborhood size of CAs on the security level and the cryptographic properties provided. Thereafter, four-neighborhood one-dimensional CAs are studied and a stream cipher algorithm is proposed. The security of the proposed algorithm is demonstrated by using the results of standard tests (i.e. NIST Test Suite and Dieharder Battery of Tests), particularly by computing the cryptographic properties of the used CAs and by showing the resistance of the suggested algorithm to mostly known attacks.**

Keywords*—Stream ciphers; cellular automata; neighborhood size; dieharder; NIST STS; cryptographic properties; attacks on stream ciphers

I. Introduction

In stream ciphers design, fast encryption and simplicity are particularly essential criteria. To get a ciphertext, a stream cipher processes by applying the XOR operation to the plaintext with the keystream. This latter is generated by a PRNG that should provide good randomness and a good security level. The strength of a stream cipher resides in the robustness of the strength of the PRNG [1]. The outstanding primitive recommended to use for the design of a PRNG is Cellular Automaton.

Thanks to the simplicity producing the complex behaviour of cellular automata (CA), especially the one-dimensional 3Neighborhood CAs which are widely used in the field of cryptography. They were studied [2-4] to ensure a good security level. However, some attacks are inevitable in 3Neighborhood configurations [4]. Accordingly, this article presents two versions of Cellular Automata-based Stream Cipher (3-CASC and 4-CASC). These versions were analysed and investigated to identify differences between their cryptographic properties and statistical analysis as well as their resistance against attacks targeting stream ciphers.

The study of the 4Neighborhood 1-dimensional CA rules is a challenging task. The authors chose the rules according to the recommendations in [5] for the 3-CASC version. Then, these rules are combined with a new variable to get the 4Neighborhood 1-dimensional CA rules. Section 2 details this step. The N-CASC design, which was inspired by grain-like CA-based ciphers, consists of three building blocks: a linear block, a nonlinear block, and a mixing block. For the linear block and nonlinear block, only linear rules and nonlinear rules are used respectively. For the mixing block, a hybrid ruleset with both linear and nonlinear rules is adopted.

The goal of the article is to look at the effect of transitioning from the 3N version to the 4N version on the cryptographic properties as well as the statistical features of the stream cipher proposed.

The rest of this article is organized as follows:

Section II presents cellular automata and cryptographic properties. Section III provides related works. Section IV details the design of the proposed scheme. Section V displays the results including the statistical test, the avalanche effect, and the cryptographic properties. Section VI shows the security analysis of the proposed scheme.

II. Background

A. Cellular Automata

Cellular automata are dynamic systems that were first introduced in the 1950s by John von Neumann and later popularized by Stephen Wolfram in the 1980s [6]. They were first studied for the modeling of biological self-reproduction by von Neumann upon Stanislas Ulam recommendations [7]. Since then, they were used in different fields such as physics, chemistry, mathematics, biology etc. … to model and solve physical, natural and real-life problems [6]. Researchers took interest in cellular automata because of the complex global behavior that stems from simple interactions and computations at the cellular level. Moreover, global properties such as

universality in computation and randomness explains the attraction of the scientific community [8].

A cellular automaton is a finite set of *n* cells arranged as a network that evolve in discrete space and time. Formally, a cellular automaton is a tuple (*L*, *S*, *N*, *f*, R) [6], where:

- L is the d-dimensional cellular space.
- S is the finite state set.
- N is the neighborhood vector linking each cell to its neighbors and represented by a radius r representing the number of consecutive cells a cell depends on.
- f is the local update rule or simply the rule that gives the next state of each cell.
- R is the rule vector consisting of the rule(s) applied to each cell.

The *L*, *S* and *N* parameters can be varied to define different types of CAs. For example, von Neumann studied 2-dimensional, 5-neighborhood, 29-state cellular automata. If the rule *f* is a linear Boolean function including only XOR logic, then the CA is called a linear CA. Otherwise, if *f* comprises also AND or OR logic, then the CA is called a non-linear CA. The rule vector *R* can consist of a single rule applied to all the cells (uniform CA) or a set of rules assigned to each cell (hybrid CA).

Despite the fact that multi-dimension cellular automata can display a more complex behavior, effectively characterizing them and mathematically analyzing them is much difficult than their 1-dimensional counterpart. This explains that much of the studies conducted on cellular automata and their application in cryptography has been done on 1-dimensional cellular automata, particularly a special kind of cellular automata introduced by Stephen Wolfram [2]. These cellular automata are called Elementary Cellular Automata (ECA) [8]. They are 1-dimensional, 3Neighborhood and 2-state cellular automata. For ECAs, there are $2^3 = 8$ neighborhood configurations and $2^{2^3} = 256$ total rules. Table I shows an example of a linear and a non-linear rule.

In Table I, x_{i-1}, $x_i$and $x_{i+1}$are the left neighbor, the cell and the right neighbor respectively. The rule name (e.g. Rule 120) is the decimal representation of the binary rule read from left to right. This naming convention was introduced by Wolfram [9].

For 4Neighborhood cellular automata, two possible neighborhood arrangement are possible [5]:

- Left skewed: each cell (x_i) depends on two left neighbors (x_{i-2} and x_{i-1}) and one right (x_{i+1}) neighbors.
- Right skewed: each cell (x_i) depends on one left neighbors (x_{i-1}) and two right (x_{i+1} and x_{i+2}) neighbors.

For 1-dimensional, 2-state, 4Neghborhood cellular automata, there are $2^4 = 16$ neighborhood configurations and $2^{2^4} = 65535$ total rules. Table II shows an example of a linear and a non-linear rule (left skewed).

One way to visualize the evolution of a cellular automaton is to use a space/time diagram. In a space/time diagram the cellular space lies on the *x*-axis, with different colors for each state, while time is represented by the *y*-axis. Space/time diagrams are a good tool to visualize the global behavior of a cellular automaton and the rule(s) associated with it. Fig. 1 represents the space time diagram of rule 90 for a configuration of 256 cells and 100 time steps (https://www.wolframalpha.com/input/?i=rule+90).

TABLE I. EXAMPLES OF 1-DIMENSIONAL, 2-STATE, 4N EXAMPLE OF ECA RULES

Neighborhood configuration	**Rule 120 (nonlinear)** $x_{i-1} \oplus x_i \cdot x_{i+1}$	**Rule 150 (linear)** $x_{i-1} \oplus x_i \oplus x_{i+1}$
111	0	1
110	1	0
101	1	0
100	1	1
011	1	0
010	0	1
001	0	1
000	0	0

TABLE II. EGHBORHOOD RULES

Neighborhood configuration	**Rule 32640 (nonlinear)** $x_{i-2} \oplus x_{i-1}.x_i.x_{i+1}$	**Rule 27030 (linear)** $x_{i-2} \oplus x_{i-1} \oplus x_i \oplus x_{i+1}$
1111	0	0
1110	1	1
1101	1	1
1100	1	0
1011	1	1
1010	1	0
1001	1	0
1000	1	1
0111	1	1
0110	0	0
0101	0	0
0100	0	1
0011	0	0
0010	0	1
0001	0	1
0000	0	0

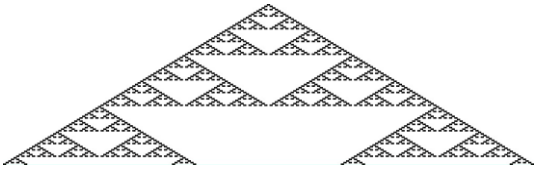

Fig. 1. Rule 90 Space Time Diagram.

For practical reasons, cellular automata are studied for a finite cellular space L. In this case, boundary conditions should be specified. Two broad categories of boundary conditions exist [6] for 1-dimensional cellular automata: open boundary conditions and periodic conditions. For open boundary conditions, the neighbors of the leftmost and rightmost cells are fixed to one of the possible state values. The most used open boundary condition is the null boundary configuration [6]. For periodic boundary conditions, the leftmost and rightmost cells are neighbors of each other.

B. Cryptographic Properties

In this section, some basic notions are defined and some properties of Boolean functions are provided. Satisfying these properties for cellular automata is a measure of their cryptographic strength and a good indication for their cryptographic suitability [10]. A more in-depth study of these cryptographic properties can be found in [11].

1) Basic Definitions

a) Hamming weight: The Hamming weight, denoted *wt(f)*, of a Boolean function is the number of 1s found in its truth table. A function has good hamming weight if $wt(f)=2^{n-1}$.

b) Hamming distance: The Hamming distance, denoted *d(f,g)*, between two Boolean functions is the Hamming weight of $f \oplus g$.

2) Cryptographic Properties

a) Nonlinearity: The nonlinearity of an *n*-variable Boolean function is defined as:

$$NL(f) = \min\{d(f,g) \mid g \in AF\}$$

where AF is the set of all *n*-variable affine Boolean functions. And $NL(f) < 2^{n-1}-2^{n/2-1}$.

b) Algebraic degree: The algebraic degree of an n-variable Boolean function is defined as the number of variables involved in the highest order term. It is bounded by n-1.

c) Balancedness: An n-variable Boolean function *f* is said to be balanced if $wt(f) = 2^{n-1}$.

d) Correlation Immunity: A Boolean function is m^{th} order correlation immune if its output is at most independent from any combination of *m* input variables.

e) Resiliency: An *m*-resilient Boolean function is a balanced m^{th} order correlation immune function.

III. Related Work

From the eSTREAM project, the LFSR/NFSR based Grain [12] and the simple LFSR based Trivium [13] designs were among the finalists designs that showed the most promising features in terms of speed, simplicity and security. However, since 2008 and the end of the eSTREAM project, many attacks [15] targeted at Grain and Trivium families of ciphers were mounted successfully. Those attacks, mainly fault attacks and correlation attacks, induced fault in the LFSR or the NFSR or exploit the dependence of the output and the initialization vector (IV). Cellular automata proved to be good cryptographic primitives due to their pseudo-randomness property and their cryptographic properties. Therefore, they were presented as good candidates to solve the problem of attacks related to LFSR/NFSR based stream cipher designs.

In [2] and [3], Stephen Wolfram was the first to propose the use of cellular automata as a keystream generator using rule 30. However, the proposed design was later attacked by Miere and Stafflebach in [4]. This attack, known as MS-attack, exploits the high correlation of the nonlinear rule 30. The majority of the subsequent proposals using cellular automata as keystream generator are based on 1-dimensional, 3Neighborhood, 2-state cellular automata. Examples include NOCAS in [16], CASTREAM in [17], CAvium in [18], CAR30 in [19] and CASca in [20]. All these designs are inspired by either Grain (NOCAS, CAR30 and CASca) or Trivium (CASTREAM and Cavium). The idea behind these designs is to replace LFSRs by hybrid linear cellular automata, NFSRs by uniform or hybrid nonlinear cellular automata and the filter function by the NMIX function [21] or a rotational symmetric bent function. Attempts at higher neighborhood radius can be found in [22] and [23] for 4Neigborhood and [24] for 5-neighborhood. In these works, it is suggested that as the neighborhood radius is increased, the randomness property and the cryptographic properties of CAs are strengthened and the resistance to fault, algebraic and correlation attacks is increased. Few examples of multidimensional CAs used as keystream generators are found in the literature. One such example can be found in [25]. This can be explained by the complexity of multidimensional CAs and the difficulty to properly study them mathematically.

IV. Proposed Stream Cipher Scheme

In this section, a detailed description of the system proposed in this article is presented. As the purpose of the article is to investigate the effect of increasing the neighborhood of cellular automata from 3Neighborhood (3N) to 4Neighborhood (4N) on the cryptographic properties and the quality of cellular automata, both the 3N and 4N versions of the system are presented here.

A. General Scheme

The general construction n-CASC (n-neighborhood one dimensional Cellular Automata based Stream Cipher) proposed in this article is a Grain-like stream cipher inspired by the cellular automata based stream cipher FResCA [22].

1) Encryption scheme: The encryption scheme consists of two phases: an initialization phase and an encryption phase. Both these phases comprise the same three building blocks, namely a nonlinear hybrid CA block, a linear hybrid CA block and a hybrid CA mixing function block. Those two phases along with the three building blocks are detailed below.

a) Initialization Phase: The initialization phase serves the purpose of putting the system in a good initial state, before starting the encryption phase, by running the three building blocks mentioned above multiple times. This allows the increase of the confusion properties and the cryptographic properties provided by the use of cellular automata within those building blocks.

The initialization phase starts with an initial 256-bit configuration C_0 at t_0. C_0 consists of a 128-bit key KEY and a

128-bit initial vector IV. Both KEY and IV are generated using the ThreadedSeedGenerator class of the Bouncy Castle Java Crypto Library. C_0 is fed to the system as follow:

- KEY is plugged as the starting configuration of the nonlinear hybrid CA block.
- IV is plugged as the starting configuration of the linear hybrid CA block.

The encryption phase, detailed in the next section, is then run n times until reaching the configuration C_n that serves as the initial configuration of the encryption phase. n, the number of times the encryption scheme is run during the initialization phase was the subject of a study. To determine a good n for the system, a 100 KEY/IV pairs were generated and the initialization phase was run for n=4, 8, 16, 32, 64 and 128. The average of the avalanche effect between C_0 and C_n was computed. Table III summarizes the results of this study. As shown in Table III, a good value for n is 64.

b) Encryption Phase: The initialization phase and encryption phases share the same building blocks, namely:

- A nonlinear hybrid CA block. This block is used to strengthen the system by the use of nonlinear-only ruleset carefully chosen for its cryptographic properties.
- A linear hybrid CA block. This block is used to extend the period of the system by the use of a carefully linear-only ruleset. The purpose of extending the period of the system is to decrease the frequency of going through a new initialization phase.
- A hybrid CA mixing function block. This block is used to further the confusion property provided by the use of CAs in the two other building blocks.

Fig. 2 and Fig. 3 show the initialization phase and the encryption phase, respectively.

TABLE III. NUMBER OF ROUNDS DURING INITIALIZATION PHASE

Number of Rounds	Average Avalanche Effect
4	48.80859
8	48.84375
16	48.89062
32	48.94141
64	49.01562
128	48.42187

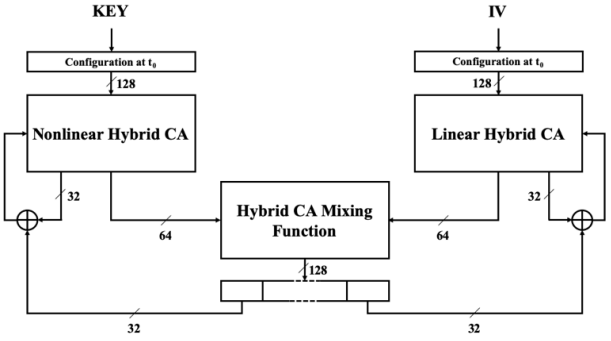

Fig. 2. Initialization Phase.

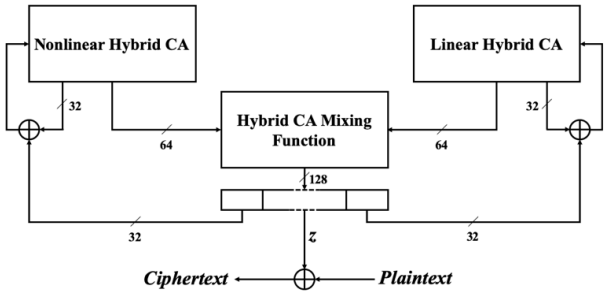

Fig. 3. Encryption Phase.

As shown in Fig. 3, the keystream *z* resulting from the combination of the three building blocks is then XORed with the plaintext to obtain a ciphertext. A portion (32 bits) of *z* is fed back to the system by XORing it to portions (32 bits) of both the nonlinear hybrid CA (32 rightmost bits) and the linear hybrid CA (32 leftmost bits) and injecting the results back in those blocks.

c) Building blocks: The description that follows details the building block of the encryption mechanism. The different rulesets used for each of these building blocks will be given for both the 3N and 4N versions, as it is the goal of the article to investigate the effect of increasing the neighborhood size on the strength and quality of the cellular automata properties.

The rulesets for 3N were carefully chosen following the recommendations found in [4] and [26] and the selection process outlined in [27]. For the 4N counterparts of these rulesets, the following process was used to determine the best candidates:

- First, for each rule of the 3N ruleset, all the similar 4N left-skewed rules were determined.
- Then, the cryptographic properties of each of the similar rules were computed and the best candidates were chosen according to the nonlinearity, algebraic degree, balancedness, correlation immunity and resiliency in this order.
- Finally, the space-time diagrams of the best candidates for each rule were compared to find the right fit.

1) Nonlinear Hybrid Ca

3N Version Case: The nonlinear hybrid CA block is a cellular automaton with n = 128 cells. The ruleset R for this cellular automaton is composed of only nonlinear rules. For the 3N version, R = {30, 120, 180, 45, 30, 120, 180, 45}. The cellular automaton is evolved $\frac{n}{2}$ = 64 time steps according to the recommendations of Wolfram [14]. At t0, the cellular automaton is filled with the 64 rightmost bits of Cn, the result of the initialization phase (Cn,0 to Cn,63).

4N Version Case: The ruleset of the 4N version is R = {43350, 38490, 25500, 22185, 43350, 38490, 25500, 22185}. For the 4N version, the cellular automaton is evolved for fewer time steps as the diffusion property of the cellular automaton spreads more rapidly for 4Neighborhood cellular automata compared to 3Neighborhood cellular automata. The cellular automaton must evolve for $\frac{n}{3}$ □ 43 time steps as

recommended in [23]. However, to compare 3N version with 4N, $\frac{n}{2}$ evolutions are performed.

2) Linear Hybrid CA

3N Version Case: The linear hybrid CA block is a cellular automaton with n =128 cells. The ruleset R for this cellular automaton is composed of only linear rules. For the 3N version, R = {90, 150}. The cellular automaton is evolved $\frac{n}{2}$ = 64 time steps. At t0, the cellular automaton is filled with the 64 leftmost bits of Cn, the result of the initialization phase (Cn,64 to Cn,127).

4N Version Case: The ruleset of the 4N version is R = {24330, 27030}. The cellular automaton is evolved for $\frac{n}{2}$ = 64 time steps.

3) Hybrid CA Mixing Function

3N Version Case: The hybrid CA mixing function block is a cellular automaton with n = 128 cells. The ruleset R for this cellular automaton is composed of only nonlinear rules. For the 3N version, R = {30, 60, 90, 120, 150, 180, 240, 15, 45}. The cellular automaton is evolved $\frac{n}{2}$ = 64 time steps according to the recommendations of Wolfram [14]. At t0, the cellular automaton is filled with the 64 rightmost bits from the result of the nonlinear hybrid CA block evolutions and the 64 leftmost bits from the result of the linear hybrid CA block evolutions.

4N Version Case: The ruleset of the 4N version is R = {43350, 49980, 42330, 38490, 27030, 25500, 65280, 255, 22185}. For the 4N version, the cellular automaton must evolve for fewer time steps as the diffusion property of the cellular automaton spreads more rapidly for 4Neighborhood cellular automata compared to 3Neighborhood cellular automata [23]. However the cellular automaton is evolved for $\frac{n}{2}$ = 64 time steps to be compared with the 3N version.

2) Decryption scheme: Since encryption and decryption are the same functions, the decryption scheme is realized by regenerating the same keystream *z* using the key *KEY* and the initial vector *IV*.

B. Design Motivation

In this section, the motivation behind some of the design criteria are presented.

1) Nonlinear Hybrid CA Block: The nonlinear hybrid CA block with only nonlinear rules is used to strengthen the system. The cryptographic robustness is provided by the nonlinear rules, carefully chosen for their high cryptographic properties such as the algebraic degree, nonlinearity and balancedness. These cryptographic properties increase the security of the system against attacks such as algebraic and fault attacks [28].

2) Linear Hybrid CA Block: Due to the use of rules 90 and 150, shown to produce maximum cycle length in [16], the linear hybrid CA block produces a maximum period.

3) Hybrid mixing function block: The mixing block is used to combine the output from the linear and nonlinear blocks. Its ruleset is made of both linear and nonlinear rules with good cryptographic properties and maximal length cycle. This block is used to further strengthen the system and increase its period.

V. Results

Several standard tests for evaluating the strength and quality of stream ciphers were conducted. The results of those tests are presented in this section.

A. Dieharder Battery of Tests

DIEHARDER battery of tests refers to a collection of standard tests compiled by Brown, Eddelbuettel and Bauer. The collection comprises tests written by Brown, Eddelbuettel and Bauer as well as other tests designed by Marsaglia and Tsang. It also includes some of the tests found in the NIST Statistical Test Suite (NIST STS). Since 2013, this test suite was updated multiple times, with each update comprising more and more tests. It is considered a strong test suite to assess the quality of random number generators and other cryptographic primitives such as stream ciphers, block ciphers and hash functions. For more information about this battery of tests, refer to [29].

The latest version, used in this article, includes 31 tests. The p-values, which are values ranging from 0 to 1, represent the results of each of the 31 tests. In order for a scheme/algorithm to pass a test, the p-value for that test should be in the range [α, 1- α], where α represents the significance level. α =0.005 is the significance level usually considered.

Table IV shows the results of the DIEHARDER battery of tests for both the 3N and 4N versions of CASC.

As can be seen from Table IV, both the versions pass all the tests. This demonstrates the good statistical properties as well as the randomness behavior and the indistinguishability property of the keystreams generated by both versions of CASC.

B. NIST Statistical Test Suite

The NIST Statistical Test Suite (NIST STS) is another collection of statistical tests, developed by the National Institute of Standards and Technology (NIST). It aims to test the randomness property of cryptographic primitives such as stream ciphers. It is used in this article to further show the good statistical properties and randomness behavior of CASC. For more details refer to the NIST special publication 800-22 [30].

As in the DIEHARDER battery of tests, a p-value is used to measure if a primitive passes a test or not. The significance level used is α=0.001.

The results of this test suite are shown in Table V for both versions of CASC.

TABLE IV. DIEHARDER BATTERY OF TESTS

Test Name	3N		4N	
	p-value	Pass?	p-value	Pass?
Diehard birthdays	0.84247	PASS	0.52067	PASS
Diehard OPERM5	0.51995	PASS	0.98337	PASS
Diehard 32x32 Binary Rank	0.90033	PASS	0.54218	PASS
Diehard 6x8 Binary Rank	0.04878	PASS	0.63344	PASS
Diehard_bitstream	0.42121	PASS	0.69582	PASS
Diehard OPSO	0.00513	PASS	0.48370	PASS
Diehard OQSO	0.81708	PASS	0.24016	PASS
Diehard DNA	0.16796	PASS	0.38120	PASS
Diehard Count the 1s (stream)	0.35543	PASS	0.12572	PASS
Diehard Count the 1s (byte)	0.59953	PASS	0.56446	PASS
Diehard Parking Lot	0.17297	PASS	0.57621	PASS
Diehard Minimum Distance (2d Circle)	0.99291	PASS	0.46500	PASS
Diehard 3d Sphere (Minimum Distance)	0.43597	PASS	0.88116	PASS
Diehard Squeeze	0.17122	PASS	0.51642	PASS
Diehard Sums	0.14394	PASS	0.51966	PASS
Diehard Runs	0.82783	PASS	0.47367	PASS
Diehard Craps	0.72554	PASS	0.82186	PASS
Marsaglia and Tsang GCD	0.78445	PASS	0.82275	PASS
STS Monobit	0.13080	PASS	0.19039	PASS
STS Runs	0.02633	PASS	0.59340	PASS
STS Serial Test (Generalized)	0.41695	PASS	0.55628	PASS
RGB Bit Distribution	0.58773	PASS	0.56630	PASS
RGB Generalized Minimum Distance	0.40435	PASS	0.63910	PASS
RGB Permutations	0.33968	PASS	0.48190	PASS
RGB Lagged Sum	0.55839	PASS	0.52245	PASS
RGB Kolmogorov-Smirnov	0.17476	PASS	0.57655	PASS
DAB Byte Distribution	0.49694	PASS	0.91177	PASS
DAB DCT (Frequency Analysis)	0.62319	PASS	0.58320	PASS
DAB Fill Tree	0.43359	PASS	0.76683	PASS
DAB Fill Tree 2	0.55191	PASS	0.44134	PASS
DAB Monobit 2	0.51141	PASS	0.94816	PASS

From Table V, it can be seen that both versions of CASC pass all the applicable tests. This further confirms the good statistical properties and the randomness behavior of both versions of CASC.

C. Avalanche Effect Test

Another common test for cryptographic primitives, such as stream ciphers, is the avalanche effect test.

This test was first introduced by Feistel in 1973[31] and states that a small difference in the input (1 bit in general) should translate into a substantial (around 50% in general) difference in the output. This concept is closely related to non-linearity. Formally, it can be formulated as follows:

f: $\{0,1\}^m \rightarrow \{0,1\}^n$ has the avalanche effect if:

$\forall$ M, M' $\in \{0,1\}^m$: Hamming(M,M')=1

$\Rightarrow$ average(Hamming(f(M), f(M'))=$\frac{n}{2}$

For CASC, the input is the initial 256-bit configuration C_0 including the 128-bit parameters KEY and IV. Therefore, to evaluate the avalanche effect for CASC, 100 different 256-bit initial configuration C_0 were generated ($C_{0,0}$ to $C_{0,99}$). For each of these configurations, the keystream of the original configuration $C_{0,i}$ and the keystreams of its one-bit change replicas (Hamming($C_{0,i}$, $C_{0,i,0\leq j\leq 255}$')=1, where j is the bit changed) are computed. Then the hamming distance between the keystreams are computed:

Hamming (f ($C_{0,i}$),f ($C_{0,i,\ 0\leq j\leq 255}$'))

The results of this test for both the 3N and 4N versions of CASC are presented in Fig. 4 and 5, respectively. The figures show the average value for each bit changed.

TABLE V. NIST STS

Test Name	3N		4N	
	p-value	Pass?	p-value	Pass?
The Frequency (Monobit) Test	0.58369	PASS	0.50865	PASS
Frequency Test within a Block	0.35048	PASS	0.45733	PASS
The Runs Test	0.49111	PASS	0.49209	PASS
Tests for the Longest-Run-of-Ones in a Block	0.57416	PASS	0.47883	PASS
The Binary Matrix Rank Test	0.53414	PASS	0.73991	PASS
The Discrete Fourier Transform (Spectral) Test	0.26486	PASS	0.48777	PASS
The Non-Overlapping Template Matching Test	0.54250	PASS	0.53420	PASS
The Overlapping Template Matching Test	0.28721	PASS	0.44916	PASS
Maurer's "Universal Statistical" Test	0.32383	PASS	0.32383	PASS
The Linear Complexity Test	0.53848	PASS	0.53414	PASS
The Serial p-value1 Test	0.34176	PASS	0.49896	PASS
The Serial p-value2 Test	0.91141	PASS	0.50188	PASS
The Approximate Entropy Test	0.66914	PASS	0.57476	PASS
The Cumulative Sums (Cusums) Forward Test	0.70265	PASS	0.65476	PASS
The Cumulative Sums (Cusums) Reverse Test	0.36499	PASS	0.50865	PASS
The Random Excursions Test	0.52242	PASS	0.42547	PASS
The Random Excursions Variant Test	0.58369	PASS	0.44899	PASS

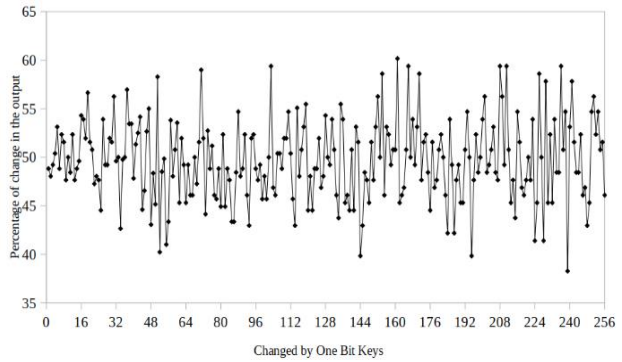

Fig. 4. Avalanche Effect Test Results for CASC 3N.

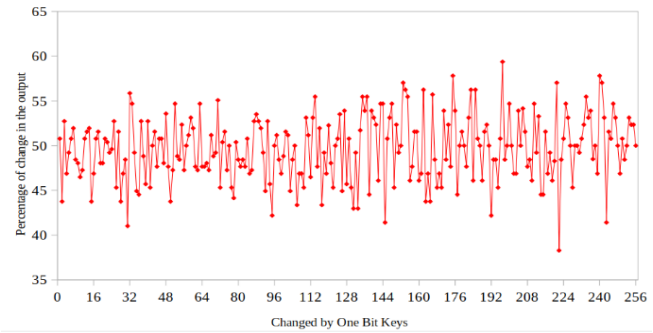

Fig. 5. Avalanche Effect Test Results for CASC 4N.

Fig. 4 and 5 show that for both versions the hamming distances concentrate around 50%, with the 4N version slightly showing better values. This shows that for both versions of CASC are statistically independent from inputs.

D. Cryptographic Properties of CASC 3N and CASC 4N

In conjunction with the previous tests, another good approach to measure the strength, statistical and randomness properties as well as the confusion property of a cryptographic primitive is to evaluate its cryptographic properties. In this article, the following main cryptographic properties are considered: algebraic degree, nonlinearity, balancedness, correlation immunity and resiliency. Tables VI to XV summarize these cryptographic properties for the nonlinear block and the mixing function block of both versions of CASC. The cryptographic properties are computed for eight cells, assumed to be unknown Boolean values x_i, and up to three clock cycles for the nonlinear block and for nine cells, assumed to be unknown Boolean values x_i, and up to three clock cycles for the mixing function block.

1) Nonlinear Block Cryptographic Properties

3N Ruleset: {30, 120, 180, 45, 30, 120, 180, 45}

4N Ruleset: {43350, 38490, 25500, 22185, 43350, 38490, 25500, 22185}

2) Mixing Function Block Cryptographic Properties

3N Ruleset: {30, 60, 90, 120, 150, 180, 240, 15, 45}

4N Ruleset: {43350, 49980, 42330, 38490, 27030, 25500, 65280, 255, 22185}

TABLE VI. NONLINEARITY

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
1	2	2	2	2	2	2	2	2	3N
	4	4	4	4	4	4	4	4	4N
2	8	8	8	8	8	8	8	8	3N
	32	56	48	48	48	56	48	48	4N
3	44	40	44	40	36	40	44	40	3N
	464	432	432	448	440	440	448	400	4N

TABLE VII. ALGEBRAIC DEGREE

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
1	2	2	2	2	2	2	2	2	3N
	2	2	2	2	2	2	2	2	4N
2	3 [1]	3	3	3	3	3	3	3	3N
	3	3	4	3	4	3	4	3	4N
3	5	5	4	4	5	5	4	4	3N
	5	5	6	5	5	5	6	5	4N

TABLE VIII. RESILIENCY

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
1	0	0	0	0	0	0	0	0	3N
	1	1	1	1	1	1	1	1	4N
2	0	1	0	0	1	1	0	0	3N
	1	1	0	0	1	1	0	0	4N
3	1	1	0	0	0	1	0	0	3N
	0	-1	0	0	0	-1	-1	1	4N

TABLE IX. CORRELATION IMMUNITY

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
1	0	0	0	0	0	0	0	0	3N
	1	1	1	1	1	1	1	1	4N
2	0	1	0	0	1	1	0	0	3N
	1	1	0	0	1	1	0	0	4N
3	1	1	0	0	0	1	0	0	3N
	0	0	0	0	0	0	0	1	4N

TABLE X. BALANCEDNESS

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
1	T	T	T	T	T	T	T	T	3N
	T	T	T	T	T	T	T	T	4N
2	T	T	T	T	T	T	T	T	3N
	T	T	T	T	T	T	T	T	4N
3	T	T	T	T	T	T	T	T	3N
	T	F	T	T	T	F	F	T	4N

TABLE XI. NONLINEARITY

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
1	2	0	0	2	0	2	0	0	2	3N
	4	0	0	4	0	4	0	0	4	4N
2	8	8	8	8	12	8	8	0	8	3N
	32	32	48	32	48	48	0	32	48	4N
3	32	32	48	48	48	32	32	32	32	3N
	384	256	384	448	448	384	384	384	384	4N

TABLE XII. ALGEBRAIC DEGREE

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
1	2	1	1	2	1	2	1	1	2	3N
	2	1	1	2	1	2	1	1	2	4N
2	2	2	2	3	2	2	2	1	3	3N
	3	2	2	3	2	2	1	2	3	4N
3	3	2	3	4	3	3	2	2	4	3N
	4	3	3	5	3	3	2	2	5	4N

TABLE XIII. RESILIENCY

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
1	0	1	1	0	2	0	0	0	0	3N
	1	2	2	1	3	1	0	0	1	4N
2	2	0	2	1	0	0	0	0	0	3N
	1	2	2	2	0	-1	3	1	3	4N
3	1	0	1	0	2	0	0	0	0	3N
	0	1	2	2	0	-1	0	-1	1	4N

TABLE XIV. CORRELATION IMMUNITY

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
1	0	1	1	0	2	0	0	0	0	3N
	1	2	2	1	3	1	0	0	1	4N
2	2	0	2	1	0	0	0	0	0	3N
	1	2	2	2	0	0	3	1	3	4N
3	1	0	1	0	2	0	0	0	0	3N
	0	1	2	2	0	0	0	0	1	4N

TABLE XV. BALANCEDNESS

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
1	T	T	T	T	T	T	T	T	T	3N
	T	T	T	T	T	T	T	T	T	4N
2	T	T	T	T	T	T	T	T	T	3N
	T	T	T	T	T	F	T	T	T	4N
3	T	T	T	T	T	T	T	T	T	3N
	T	T	T	T	T	F	T	F	T	4N

From these tables, it can be noted that, with the exception of correlation immunity and resiliency, the nonlinearity and the algebraic degree increase with each iteration or remain true in the case of balancedness for both 3N and 4N. The decrease of the values of correlation immunity and resiliency can be explained by the fact that these properties are in contradiction with the three others (nonlinearity, algebraic degree and balancedness) [4]. Therefore, a compromise should be found. For a stream cipher, the algebraic degree and the nonlinearity can be judged as more important properties to achieve than the correlation immunity and resiliency properties. Moreover, it can be noted that for these two properties (nonlinearity and algebraic degree), the 4N version of CASC displays better properties than the 3N version. This confirms the general tendency that going from 3N to 4N improves the statistical properties as well as the cryptographic properties of the stream cipher scheme presented here.

VI. SECURITY ANALYSIS

As the main criterion for a stream cipher is to resist all known attacks, a security analysis is conducted in this section. Resisting an attack means that the computational complexity of that attack is no less than that of an exhaustive key search attack which is $O(2^n)$.

This section covers some of the major known attacks against stream ciphers and the countermeasures used in the design of CASCA are pointed out.

A. Side Channel Attacks

Side channel attacks [32] are a class of attacks targeted at the hardware implementation of ciphers. By analyzing different physical characteristics, such as power consumption, noise or heat dissipation, during the execution time, these attacks try to recover the internal state of the keystream generator in the case of stream ciphers. In the case of the stream cipher presented in this article, the complexity of this kind of attacks is higher due to the use of a linear block, a nonlinear block and a mixing function block based on cellular automata that make the cipher quite difficult to reverse.

A. Time/Memory/Data Tradeoff Attack

In the time/memory/data tradeoff attack [32] the goal of the attacker is to lower the complexity of the exhaustive key search attack by establishing a lookup table of pairs of key/keystream during the offline phase and observing the keystreams generated by unknown keys during the online phase and trying to find matches.

The complexity of this attacks is $O(2^{n/2})$, where n is the inner state of the stream cipher. In the case of CASCA, n = 256 bits. Therefore, this attack is difficult to achieve.

B. Algebraic Attacks

In this type of attacks, the cryptographic system studied is modelled using algebraic equations. This is performed by first identifying an algebraic equation set relating the first configuration C_0 with the generated keystream. The maximum number of keystream bits is collected to construct the equations system. By solving this system, the initial configuration and consequently the secret key are recovered. To prevent this type of attacks, the algebraic degree and the

nonlinearity of the functions used in the keystream generation mechanism must be as high as possible [32].

From Tables VII and XII, it is clear that the algebraic degree and nonlinearity increase with each iteration. The number of cycles used is the recommended one in terms of the neighborhood size. These results make the system robust against this type of attacks.

C. Linear Approximation Attacks

The linear cryptanalysis technique is a known plaintext attack designed by Matsui to break DES encryption scheme [33]. This technique aims to find a linear approximation of a symmetric system's behavior from a number of plaintext bits and ciphertext bits in relation to the key bits.

To prevent this type of attacks, the nonlinear and the mixing blocks are useful to decrease the probability to find such an approximation. From Tables VII and XII, it is clear that the algebraic degree and nonlinearity increase with each iteration and thus make this kind of attacks difficult to realize.

D. Correlation Attacks

In 1985, Siegenthaler [34] proposed a known plaintext attack called the correlation attack which aims to recover the initial configuration of the internal state by using some known keystream bits.

High nonlinearity, balancedness, resiliency and correlation immunity are good countermeasures to avoid this kind of attack. As shown by the tables summarizing the cryptographic properties of CASCA 3N and 4N, the stream cipher presented in this article is quite robust against correlations attacks.

E. Fault Attacks

In fault attacks [23], faults are injected and the difference between ciphertexts containing faults and original ciphertexts without faults is exploited to recover the key. In the case of CASCA, the tracking of the faults is made difficult due to the high diffusion property of cellular automata used in the building blocks.

VII. Conclusion

In this article, a new stream cipher scheme (N-CASC) was presented. It is a grain-like stream cipher based on cellular automata comprising three building blocks: a linear block, a nonlinear block, and a mixing function block. The article details the internal functioning of the mechanism and provides a comparison of its 3N and 4N versions. The comparison, which is based on the statistical tests (Diharder and NIST STS) as well as the cryptographic properties and the security analysis, serves the purpose of outlining the advantages and disadvantages of each version. The 4N version presents better statistical results and displays a better nonlinearity and a higher algebraic degree than the 3N version. However, the 3N version shows better results regarding the correlation immunity and resiliency properties.

Based on the findings of the present paper, a new design can be proposed, in the future, combining building blocks of 3Neighborhood cellular automata and 4Neighborhood cellular automata taking advantage of the features of each of the configurations for better levels of security and higher levels of randomness. Another future prospect might be the investigation of higher neighborhood configurations (5-neighborhood cellular automata for example) and higher dimensions (2D and 3D).

References

[1] A. Klein, Stream Ciphers. Springer, 2013

[2] S. Wolfram, "Cryptography with Cellular Automata," Lecture Notes in Computer Science Advances in Cryptology — CRYPTO '85 Proceedings, pp. 429–432, 1985.

[3] S. Wolfram, "Random sequence generation by cellular automata," Advances in Applied Mathematics, vol. 7, no. 2, pp. 123–169, 1986.

[4] W. Meier and O. Staffelbach, "Analysis of Pseudo Random Sequences Generated by Cellular Automata," Advances in Cryptology – EUROCRYPT '91 Lecture Notes in Computer Science, pp. 186–199, 1991.

[5] K. Chakraborty and D. R. Chowdhury, "CSHR: Selection of Cryptographically Suitable Hybrid Cellular Automata Rule," Lecture Notes in Computer Science Cellular Automata, pp. 591–600, 2012.

[6] K. Bhattacharjee, N. Naskar, S. Roy, and S. Das, "A survey of cellular automata: types, dynamics, non-uniformity and applications," Natural Computing, 2018..

[7] J. T. Schwartz, J. V. Neumann, and A. W. Burks, "Theory of Self-Reproducing Automata," Mathematics of Computation, vol. 21, no. 100, p. 745, 1967

[8] D. Mukhopadhyay and A. Kundu, "Preliminaries on Cellular Automata," Web Searching and Mining Cognitive Intelligence and Robotics, pp. 29–35, 2018.

[9] S. Wolfram, "Universality and complexity in cellular automata," Physica D: Nonlinear Phenomena, vol. 10, no. 1-2, pp. 1–35, 1984

[10] T. W. Cusick, Cryptographic Boolean functions and applications. London: Academic Press, 2017.

[11] C.K. Wu and D. Feng , Boolean functions and their applications in cryptography. Place of publication not identified: SPRINGER, 2016.

[12] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A Stream Cipher Proposal: Grain-128," 2006 IEEE International Symposium on Information Theory, 2006.

[13] C. D. Cannière, "Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles," Lecture Notes in Computer Science Information Security, pp. 171–186, 2006

[14] S. Wolfram, A new kind of science. Champaign: Wolfram Media, 2002

[15] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and Y. Papaefstathiou, "A survey of lightweight stream ciphers for embedded systems," Security and Communication Networks, vol. 9, no. 10, pp. 1226–1246, 2015

[16] S. Karmakar and D. R. Chowdhury, "NOCAS : A Nonlinear Cellular Automata Based Stream Cipher," Discrete Mathematics and Theoretical Computer Science, pp. 135–146, 2012.

[17] S. Das and D. R. Chowdhury, "CASTREAM: A New Stream Cipher Suitable for Both Hardware and Software," Lecture Notes in Computer Science Cellular Automata, pp. 601–610, 2012.

[18] S. Karmakar, D. Mukhopadhyay, and D. R. Chowdhury, "CAvium - Strengthening Trivium stream cipher using Cellular Automata," Journal of cellular automata , vol. 7, no. 2, Jan. 2012.

[19] S. Das and D. Roychowdhury, "CAR30: A new scalable stream cipher with rule 30," Cryptography and Communications, vol. 5, no. 2, pp. 137–162, Jul. 2013.

[20] S. Ghosh and D. R. Chowdhury, "CASca:A CA Based Scalable Stream Cipher," Mathematics and Computing Springer Proceedings in Mathematics & Statistics, pp. 95–105, 2015.

[21] J. Bhaumik and D. R. Chowdhury, "Nmix: An Ideal Candidate For Key Mixing," Proceedings of the International Conference on Security and Cryptography, 2009.

[22] J. Jose and D. R. Chowdhury, "FResCA: A Fault-Resistant Cellular Automata Based Stream Cipher," Lecture Notes in Computer Science Cellular Automata, pp. 24–33, 2016.

[23] J. Jose and D. R. Chowdhury, "Investigating four neighbourhood cellular automata as better cryptographic primitives," Journal of Discrete

Mathematical Sciences and Cryptography, vol. 20, no. 8, pp. 1675–1695, 2017.

[24] R. Lakra, A. John, and J. Jose, “CARPenter: A Cellular Automata Based Resilient Pentavalent Stream Cipher,” Developments in Language Theory Lecture Notes in Computer Science, pp. 352–363, 2018.

[25] M. Perrenoud, M. Sipper, and M. Tomassini, “On the generation of high-quality random numbers by two-dimensional cellular automata,” IEEE Transactions on Computers, vol. 49, no. 10, pp. 1146–1151, 2000.

[26] S. Karmakar, D. Mukhopadhyay, and D. R. Chowdhury, “d-Monomial Tests of Nonlinear Cellular Automata for Cryptographic Design,” Lecture Notes in Computer Science Cellular Automata, pp. 261–270, 2010.

[27] A. Sadak, F. E. Ziani, B. Echandouri, C. Hanin, and F. Omary, “HCAHF: A New Family of CA-based Hash Functions,” International Journal of Advanced Computer Science and Applications, vol. 10, no. 12, 2019.

[28] S. Maiti, S. Ghosh, and D. R. Chowdhury, “On the Security of Designing a Cellular Automata Based Stream Cipher,” Information Security and Privacy Lecture Notes in Computer Science, pp. 406–413, 2017.

[29] Robert G. Brown's General Tools Page. [Online]. Available: https://phy.duke.edu/~rgb/General/dieharder.php. [Accessed: 16-Mar-2020].

[30] A. L. Rukhin, A statistical test suite for random and pseudorandom number generators for cryptographic applications. Gaithersburg, MD: U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2000.

[31] H. Feistel, “Cryptography and Computer Privacy,” Scientific American, vol. 228, no. 5, pp. 15–23, 1973.

[32] M. U.bokhari, S. Alam, and F. S. Masoodi, “Cryptanalysis Techniques for Stream Cipher: A Survey,” International Journal of Computer Applications, vol. 60, no. 9, pp. 29–33, 2012.

[33] M. Matsui, “Linear Cryptanalysis Method for DES Cipher,” Advances in Cryptology — EUROCRYPT ’93 Lecture Notes in Computer Science, pp. 386–397, 1994.

[34] T. Siegenthaler, “Decrypting a Class of Stream Ciphers Using Ciphertext Only,” IEEE Transactions on Computers, vol. C-34, no. 1, pp. 81–85, 1985.