CASca:A CA Based Scalable Stream Cipher

Shamit Ghosh and Dipanwita Roy Chowdhury

Abstract This paper presents a scalable stream cipher based on Cellular Automata. The cipher uses linear and nonlinear cellular automata as crypto primitives. The properties of maximum length nonlinear cellular automata have been exploited to design the cipher. Rotational symmetric bent function is used in the final combiner of the cipher which is proven to be secured against certain kind of fault attacks. The scalability provides different security level for different applications. Finally the cipher is shown to be very hardware efficient.

Keywords Cellular automata · Stream cipher · Pseudo random sequence generator · Scalable stream cipher

1 Introduction

Stream Cipher is an important branch in symmetric key cryptography. The goal of a stream cipher design is that it must provide high-speed encryption and less design overhead in comparison with block ciphers. The conventional stream ciphers used linear feedback shift registers (LFSR) for randomness and sufficiently large period. However, attempts are made to replace LFSR with linear CA to get excellent random sequences with a high speed of execution. Nonlinearity is another essential property for security, which is typically introduced by nonlinear feedback shift registers (NFSR) in stream cipher designs. The challenge of designing a crypto-system is, in addition to providing the required security, the crypto-system should be easy to implement in both hardware and software together with high performance and minimal resource usage. Stream ciphers have gained a lot of attention in the past few years.

S. Ghosh (✉) · D.R. Chowdhury
Department of Computer Science and Engineering, Indian Institute of Technology Karagpur, Kharagpur 721302, India
e-mail: shamit.ghosh@cse.iitkgp.ernet.in

D.R. Chowdhury
e-mail: drc@cse.iitkgp.ernet.in

R.N. Mohapatra et al. (eds.), *Mathematics and Computing*,
Springer Proceedings in Mathematics & Statistics 139,
DOI 10.1007/978-81-322-2452-5_7

The eSTREAM [4] project was launched in 2004 in search of a good stream cipher. The eSTREAM project has been instrumental for this attention. The eSTREAM portfolio ciphers fall into two profiles. Profile 1 contains stream ciphers more suitable for software applications with high throughput requirements. The winners in this profile are HC-128 [11], Rabbit [6], Salsa20/12 [5], SOSEMANUK [3]. Profile 2 stream ciphers are particularly suitable for hardware applications with restricted resources such as limited storage, gate count, or power consumption. Grain v1 [10], MICKEY 2.0 [1] and Trivium [7] are the winner in this category.

The basic philosophy of a stream cipher is to generate pseudo-random sequences from a secret key or a seed. There is an optional provision for an initial value (IV) which provides security for multiple encryptions using the same secret key. This fact is the motivation of designing fast pseudo-random sequence generators. The cellular automata (CA) provide very good pseudo-random sequences which exhibit excellent statistical properties. A necessary requirement for such a sequence generator is large period. In our design, both linear and nonlinear part are maximum length sequence generator. Moreover, scalability is an important aspect any cryptographic design. Due to advancement of computing speed, the current security standard may be obsolete after a few years. Only for a scalable design, the new security standard can be achieved by increasing the key size without discarding the whole algorithm.

Our Contribution: In this paper we have designed a new CA-based scalable stream cipher CASca. The design specification and design rationale of the cipher is portrayed. Crypto properties of CASca is shown in detail which proves its security against all existing attacks. The design is shown to be suitable for constrained hardware environment.

The remainder of the paper is organized as follows. Section 2 draws the idea about some notions, definitions, and basic studies on CA. The design of CASca is depicted in Sect. 3. Section 4 shows the scalability and initialization of the cipher. Section 5 gives a security analysis of CASca against some popular cryptanalysis techniques. Finally Sect. 6 concludes the work.

2 Preliminaries

In this section we discuss some basic notions required for security analysis. Some definitions and properties related to CA are also highlighted. Based on these theoretical studies, we further proceed to our proposed scheme.

2.1 Notions

Throughout the paper, we use '+' to represent Boolean XOR operation in **GF**(2). In this subsection, some basic security properties for evaluating a cryptographic primitive are given. The entire theoretical studies and analysis of our scheme is done based on these properties.

Definition 1 Hamming Weight: Number of Boolean 1's in a Boolean function's truth table is called the Hamming weight of the function.

Definition 2 Affine Function in $GF(2)$: A Boolean function which can be expressed as XOR of some or all of its input variables and a Boolean constant is an affine function.

In this paper the term *Affine Function* simply refers to Affine Function in $GF(2)$.

Definition 3 Nonlinearity: Let, f be a Boolean function of variables, $x_1, x_2, \cdots, x_n$ and A be the set of all affine functions in $x_1, x_2, \cdots, x_n$. The minimum of all the Hamming distances between f and the Boolean functions in A is the nonlinearity of f.

Definition 4 Algebraic Normal Form: Any Boolean function can be expressed as XOR of conjunctions and a Boolean constant, True or False. This form of the Boolean function is called its Algebraic Normal Form (ANF).

Definition 5 Correlation Immunity : A function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is kth order correlation immune if for any independent n binary random variables $X_0 \ldots X_{n-1}$, the random variable $Z = f(X_0, \ldots, X_{n-1})$ is independent of any random vector $(X_{i_1} \ldots X_{i_k})$ with $0 \leq i_1 < \ldots < i_k < n$.

Definition 6 Resiliency : A function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is kth order resistant if it is balanced and correlation immune of order k.

2.2 CA Basics

Cellular automata are studied as mathematical model for self-organizing statistical systems. CA can be one-dimensional or multi-dimensional. In this paper, we discuss only one-dimensional two state CA. They can be considered as an array of cells where each cell is a one-bit memory element.

The neighbor set $\mathbf{N}(i)$ is defined as the set of cells on which the ith cell is dependent on each iteration. The simplest class of CA are *elementary CA* or *three-neighborhood CA* where each cell evolves in every time step based on some combinatorial logic on the cell itself and its two nearest neighbors. More formally, for a three-neighborhood CA, $\mathbf{N}(i) = \{i-1, i, i+1\}$. So, if the value of ith cell at tth time step is $q_i(t)$, then

$$q_i(t+1) = f(q_{i-1}(t), q_i(t), q_{i+1}(t))$$

where f denotes some combinatorial logic. We call the set of all feedback functions as ruleset and express as $\mathcal{F}$. The state transition of one iteration of a CA is expressed as $\mathcal{S}_{t+1} = \mathcal{F}(\mathcal{S}_t)$ where $\mathcal{S}_t$ is the set of all cells in the CA at tth time step.

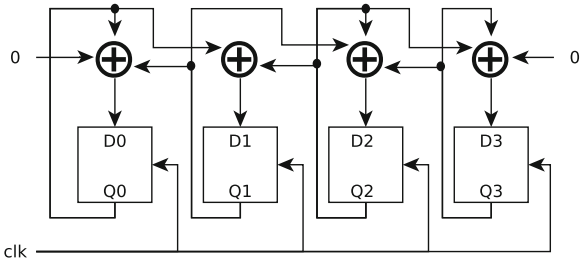

Fig. 1 4-cell 3-neighborhood null boundary LHCA with ruleset 1011

Since, a three-neighborhood CA having two states (0 or 1) can have $2^3 = 8$ possible binary states, there are a total of $2^{2^3} = 256$ possible rules. Each rule can be represented as an decimal integer from 0 to 255. If the combinatorial logic for the rules have only Boolean XOR operation, then it is called *linear* or *additive* rule. Some of the three-neighborhood additive CA rules are 0, 60, 90, 102, 150 etc. Moreover, if the combinatorial logic contains AND/OR operations, then it is called *nonlinear* rule.

An n cell CA with cells $\{x_0, x_1, \cdots, x_{n-1}\}$is called *null boundary* CA if $x_n = 0$ and $x_{-1} = 0$. Similarly, for a *periodic boundary* CA $x_n = x_0$.

A CA is called *uniform*, if all its cells follow the same rule. Otherwise, it is called *nonuniform* or *hybrid* CA. For a hybrid CA, the sequence of the rules followed by the cells in a particular order (MSB to LSB or vise versa). If all the ruleset of a hybrid CA is linear, then we call the CA linear hybrid cellular automata (LHCA), otherwise it is called nonlinear hybrid cellular automata (NHCA). In Fig. 1, a four-cell null boundary LHCA is shown.

The *shifting operation* [9] on an NHCA is defined as follows.

Definition 7 The one-cell shifting operation, denoted by $f_i \xrightarrow{P} f_{i\pm1}$ moves a set of ANF monomials P from ith cell of an NHCA to all the cells from $(i-1)$ to $(i+1)$-th cell, according to the dependency of the affected cells upon the ith cell. Each variable in P is changed by its previous state. Similarly, a k cell shifting is obtained by applying the one-cell shifting operation for k times upon the initial NHCA and symbolized as $f_i \xrightarrow{P} f_{i\pm k}$.

For example, we have a 5-bit 3 neighborhood CA with the following initial ruleset:

$$
\begin{aligned}
f_0 &= (x_1 \oplus x_0) \\
f_1 &= (x_2 \oplus x_1 \oplus x_0) \\
f_2 &= (x_3 \oplus x_2 \oplus x_1) \oplus x_4 \\
f_3 &= (x_4 \oplus x_3 \oplus x_2) \\
f_4 &= x_3
\end{aligned}
$$

It is clear from the equations that x_3 is the previous state of x_4. Now applying the shifting $f_2 \xrightarrow{x_4} f_{2\pm1}$, the new ruleset becomes:

$$
\begin{aligned}
f_0 &= x_1 \oplus x_0 \\
f_1 &= (x_2 \oplus x_1 \oplus x_0) \oplus x_3 \\
f_2 &= (x_3 \oplus x_2 \oplus x_1) \oplus x_3 \\
f_3 &= (x_4 \oplus x_3 \oplus x_2) \oplus x_3 \\
f_4 &= x_3
\end{aligned}
$$

3 Design of Scalable a Stream Cipher

The design of the proposed stream cipher consists of three parts, a maximum length sequence generator, a nonlinear sequence generator, and a final combiner function. Their design rationale and construction are discussed below. The overall scheme is depicted in Fig. 2.

3.1 Maximum Length Sequence Generator

The necessity of maximum length sequence is to prevent low period attack. We designed the maximum length sequence using linear hybrid cellular automata(LHCA) as it is widely known that linear functions provide good diffusion properties. To synthesize a maximum length LHCA rule a primitive polynomial is needed. From that primitive polynomial, a ruleset is generated using the algorithm described in [8]. In our design, the polynomial $x^{128} + x^{29} + x^{27} + x^2 + 1$ is used. We call this LHCA as $\mathcal{L}$. The individual bits of $\mathcal{L}$ is denoted by s_i where $i \in \{0, 127\}$. It is trivial that $\mathcal{L}$ is a null boundary CA.

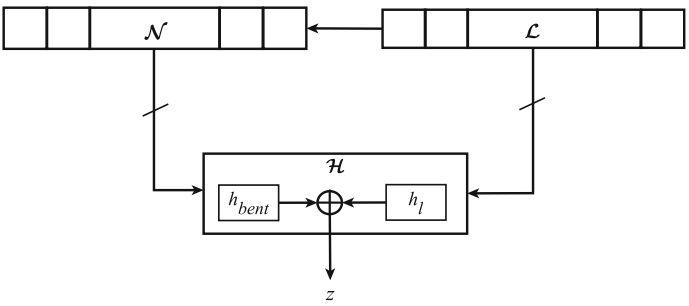

Fig. 2 Operations of cipher

3.2 Nonlinear Sequence Generator

Linear functions alone cannot provide cryptographic security as they can be easily cryptanalyzed. To introduce nonlinearity, a nonlinear sequence generator is needed. The design the nonlinear sequence generator should be in such a way that it has a long period. In this case, we use Algorithm 1 depicted in [9]. This algorithm takes a maximum length LHCA as input and injects required nonlinearity into some given positions of the CA while retaining the period of $2^n - 1$. In our design we synthesized the nonlinear sequence generator from an LHCA, the same as $\mathcal{L}$ and then injected nonlinearity at positions $\{20, 42, 79, 117\}$. For each position i, the nonlinear function $\mathbf{f_N}$, injected at ith position is, $(x_{i+2}.x_{i-2})$. We will call this nonlinear sequence generator as $\mathcal{N}$. The individual bits of $\mathcal{N}$ is denoted by b_i where $i \in \{0, 127\}$. The bit b_0 is bounded by null value, whereas b_{127} is bounded by s_0.

Algorithm [1]: NHCA Synthesize Algorithm

Input: A maximum length LHCA with ruleset $\mathcal{F}_L$, A position j to inject nonlinearity and the set of cells of the LHCA $\mathcal{S}$
Output: A maximum length NHCA ruleset $\mathcal{F}_N$

1: $\mathcal{F}_N \leftarrow \mathcal{F}_L$
2: Let $\mathcal{F}_N = \{f_{n-1}, \cdots, f_0\}$
3: $\mathcal{X} \subset \mathcal{S} : \forall x \in \mathcal{X}, x \notin \mathbf{N}(j)$ ▷ select a subset from $\mathcal{S}$
4: $P \leftarrow \mathbf{f_N}(\mathcal{X})$ ▷ $\mathbf{f_N}$ is non-linear function
5: $f_j \leftarrow f_j \oplus P$
6: $(f_j \xrightarrow{P} f_{j+1})$ ▷ Apply shifting operation
7: $f_j \leftarrow f_j \oplus P$
8: **return** $\mathcal{F}_N$

3.3 Final Combiner Function

Some suitable tap bit positions are chosen from both $\mathcal{L}$ and $\mathcal{N}$. We call the set of these tap bit positions as $\mathcal{T}$. The final combiner function $\mathcal{H}$ is defined as $\mathcal{H} : 0, 1^{\{|\mathcal{T}|\}} \rightarrow \{0, 1\}$ where the input to $\mathcal{H}$ is $\mathcal{T}$. The construction of $\mathcal{H}$ has two primary parts, a bent function h_{bent} and a linear part h_l. The bent function provides high nonlinearity,[1] whereas the linear part increases the correlation immunity and resiliency of $\mathcal{H}$. The Boolean xor of l and b generates the required value of $\mathcal{H}$. The function b is defined as

[1]The nonlinearity of a bent function is the highest possible value among all Boolean functions of the same number of variables.

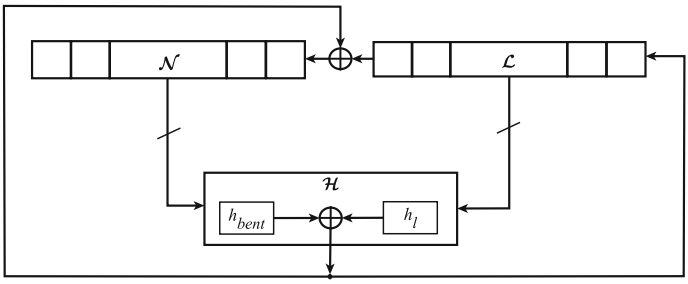

Fig. 3 Initialization of cipher

$$h_{bent}(x) = x_0x_2 + x_0x_6 + x_1x_3 + x_1x_7 + x_2x_4 + x_3x_5 + x_4x_6 + x_5x_7 + x_0x_2x_5 + x_0x_3x_5 + x_0x_3x_6 + x_1x_3x_6 + x_1x_4x_6 + x_1x_4x_7 + x_2x_4x_7 + x_2x_5x_7$$

where $x = \{b_{17}, s_{12}, s_{35}, s_{58}, s_{78}, s_{97}, s_{117}, b_{97}\}$. Similarly, l is defined as

$$h_l = \sum_{k \in A} b_k$$

where $A = \{21, 43, 80, 118\}$.

4 Scalability and Key Initialization

Before generating any keystream, the cipher is initialized with a key k and initial vector IV . The number of bits in IV is 96 and we denote the bits of IV as IV_i, $0 \leq i \leq 95$. The size of k is variable and can vary from 80 and 128. The size is chosen by the user according to the security parameter. Let n be the size of the key for a particular scheme where i^{th} bit of the key is denoted as k_i, $0 \leq i \leq n$. The 96 LSB bits of $\mathcal{L}$ is initialized with IV, $s_i = IV_i, 0 \leq i \leq 95$. The rest of the bits are set at 1. This ensures that $\mathcal{L}$ cannot be initialized with all in case of a chosen IV attack. Similarly, the first n bits of $\mathcal{N}$ is filled with k, $b_i = k_i, 0 \leq i \leq n$ and the remaining bits (if any) are set to 1. Next the cipher is clocked for 128 cycles without producing any keystream and the keystream is XORed with both the MSB of $\mathcal{L}$ and $\mathcal{N}$ as shown in Fig. 3.

5 Security

The principal design criteria of a stream cipher is to be secured against all existing cryptanalysis techniques. The best possible algorithm to recover the secret key is to be no less than exhaustive search of the key space. In this section we discuss some possible attacks and the corresponding design criteria in our cipher against them.

(a)

Itn.	b_{21}	b_{43}	b_{80}	b_{118}
1	4	2	4	2
2	16	0	16	0
3	0	4	0	16
4	0	0	32	64
5	256	0	128	128
6	0	256	64	128
7	256	0	256	0
8	4096	512	512	512
9	1024	1024	0	0
10	2048	2048	4096	2048

(b)

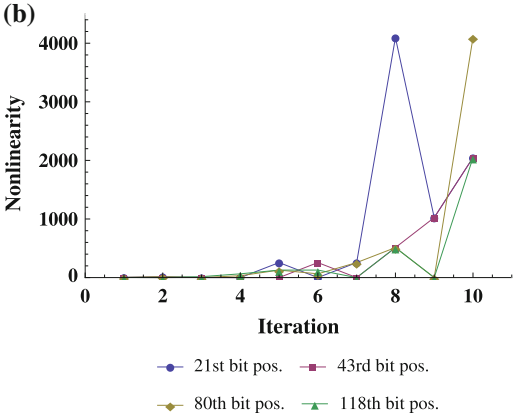

Fig. 4 Nonlinearity of the tap bits. **a** Nonlineartiy with iteration. **b** Nonlineartiy graph

5.1 Linear Cryptanalysis

Linear cryptanalysis tries to formulate a linear approximation of the cipher. High nonlinear value of the cipher protects the cipher against this attack. Table 4a shows the nonlinear values of the tap points with each iteration. After a few initial rounds the nonlinearity reaches a high value. So the cipher is expected to be secured against linear cryptanalysis techniques (Fig. 4).

5.2 Algebraic Attacks

Algebraic cryptanalysis techniques are very efficient in terms of finding loopholes in the design. Weak or careless design principal can cause such kinds of attacks. In our design, the $\mathcal{H}$ function provides a boolean function of degree three. With each iteration this degree increases if the output bit is expressed as a function of only the initial state bits. Hence, solving algebraic equations to cryptanalyze the system is computationally infeasible.

5.3 Correlation Attacks

Correlation immunity is an important aspect of designing stream ciphers which prevents the chosen *IV* attacks. The idea of the attack is to find any correlation between the *IV* and the output stream. Unbalanced output helps an adversary to find a correlation. Using only bent functions in $\mathcal{H}$ may cause vulnerability for finding correlation. Thus a linear function h_l is needed. The $\mathcal{H}$ function has correlation immunity 3

Table 1 Results of NIST statistical test suite

Test name	Status
Frequency (Monobit) test	Pass
Frequency test within a block	Pass
Runs test	Pass
Discrete fourier transform (Spectral) test	Pass
Non-overlapping template matching test	Pass
Overlapping template matching test	Pass
Serial test	Pass
Approximate entropy test	Pass
Cumulative sums (Cusum) test	Pass

and nonlinearity 1664. It is a balanced function, so it is also a 3 resilient function. If the input bits of $\mathcal{H}$ are represented as functions of the initial state bits, then with each iteration the nonlinearity as well as resiliency increases very fast. So correlation attack against CASca will not be faster than a brute force attack.

5.4 Statistical Analysis

Statistical analysis of the cipher is carried out using NIST Randomness Test Suite and Table 1 summarizes the result. The tests are performed by taking 10,000 bit keystream from a fixed 128 bit key and IV pair.

5.5 Fault Attacks

Fault attacks are the most powerful and popular cryptanalysis techniques. The easier way to inject fault into the cryptographic devices makes it highly feasible. Designing a cryptographic scheme that is fault attack resistant is a challenging task for researchers. Initially, an attacker injects single or multibit fault into the state of the cipher. The output difference of the fault-free and faulty ciphertext leaks some information about the state of the cipher. This leakage is indicated with some equation. This method is repeated multiple times until a probabilistic polynomial time algorithm can recover the secret key (or the state of the cipher). The fault locations are chosen by the attacker in such a fashion that the set of equation can be solved easily. In our design, the state of the cipher is implemented as CA. The high diffusion property of CA infects the state with the fault within a very few iteration. Hence, the algebraic degree and the number of unknown variables in the set of equations becomes so high that it is a hard problem to solve. Moreover, a fault attack based on the decomposition of the final combiner function in Grain v1 is discussed in [2]. This attack was possible as the

combiner function can be decomposed into a form of $s.u + v$ where u is a function of only the nonlinear state bits and v is a function of the linear state bits. In our design, the h_{bent} function is a rotational symmetric bent function. This function hardly reveals any information about the fault positions for their symmetric property and cannot be decomposed like the above-mentioned attack. Thus the design is expected to be robust against fault attacks.

6 Conclusion

A new stream cipher CASca is proposed in this paper. The choice of design rationale of the cipher considers the existing cryptanalysis techniques. The size of *IV* is 96 bits but the key length has been kept as a variable one. The reason behind this is to provide scalability for the applications where the security parameter can vary. The design of CASca is very hardware efficient. It suitable for applications where low power consumption and low area overhead are required, such as mobile devices.

References

1. Babbage, S., Dodd, M.: The mickey stream ciphers. In: Robshaw, M., Billet O. (eds.) New Stream Cipher Designs, Lecture Notes in Computer Science, vol. 4986, pp. 191–209. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68351-3_15. http://dx.doi.org/10.1007/978-3-540-68351-3_15
2. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the grain family of stream ciphers. In: Prouff, E., Schaumont P. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2012, Lecture Notes in Computer Science, vol. 7428, pp. 122–139. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33027-8_8. http://dx.doi.org/10.1007/978-3-642-33027-8_8
3. Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: Sosemanuk, a fast software-oriented stream cipher. In: Robshaw, M.J.B., Billet, O. (eds.) The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986, pp. 98–118. Springer, Berlin (2008). http://dblp.uni-trier.de/db/series/lncs/lncs4986.html#BerbainBCCGGGLMPS08
4. Bernstein, D.J.: Notes on the ecrypt stream cipher project (estream). http://cr.yp.to/streamciphers.html
5. Bernstein, D.J.: New stream cipher designs. In: The Salsa20 Family of Stream Ciphers, pp. 84–97. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68351-3_8. http://dx.doi.org/10.1007/978-3-540-68351-3_8
6. Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., Scavenius, O.: Rabbit: a new high-performance stream cipher. In: Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, 24–26 Feb 2003, Revised Papers, Lecture Notes in Computer Science, vol. 2887, pp. 307–329. Springer, Heidelberg (2003). doi:10.1007/978-3-540-39887-5_23. http://www.iacr.org/cryptodb/archive/2003/FSE/3049/3049.pdf
7. Cannire, C.: Trivium: a stream cipher construction inspired by block cipher design principles. In: Katsikas, S., Lpez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) Information Security, Lecture Notes in Computer Science, vol. 4176, pp. 171–186. Springer, Heidelberg (2006). doi:10.1007/11836810_13. http://dx.doi.org/10.1007/11836810_13

8. Cattell, K., Muzio, J.C.: Synthesis of one-dimensional linear hybrid cellular automata. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **15**(3), 325–335 (1996). doi:10.1109/43.489103
9. Ghosh, S., Sengupta, A., Saha, D., Chowdhury, D.R.: A scalable method for constructing non-linear cellular automata with period $2^n - 1$. In: Cellular Automata—11th International Conference on Cellular Automata for Research and Industry, ACRI 2014, Krakow, Poland, 22–25 Sept 2014. Proceedings, pp. 65–74 (2014). doi:10.1007/978-3-319-11520-7_8. http://dx.doi.org/10.1007/978-3-319-11520-7_8
10. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. Int. J. Wire. Mob. Comput. **2**(1), 86–93 (2007). doi:10.1504/IJWMC.2007.013798. http://dx.doi.org/10.1504/IJWMC.2007.013798
11. Wu, H.: The stream cipher hc-128. In: Robshaw, M.J.B., Billet, O. (eds.) The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986, pp. 39–47. Springer, Heidelberg (2008). http://dblp.uni-trier.de/db/series/lncs/lncs4986.html#Wu08