Fault Attacks against a Cellular Automata Based Stream Cipher

José Carrijo [1], Anderson C. A. Nascimento [1], Rafael Tonicelli [1], Vinícius de Morais Alves[1]

[1]Departamento de Engenharia Elétrica, Universidade de Brasília
Campus Darcy Ribeiro, 70910-900, Brasilia, DF, Brazil

carrijo@cepesc.gov.br,andclay@ene.unb.br

{tonicelli, vmalves}@redes.unb.br

Abstract. *This paper presents fault attacks against a cellular automata based stream cipher. A fault attack assumes that the adversary is able to physically operate the cryptographic device and insert some errors into it. As a consequence, the adversary can induce faulty results into the device and use them to recover the stored secret key. By using this approach we provide extremely efficient and practical cryptanalytic methods: by injecting $n/2 + n^2/32$ faults we recover the n-bit secret key from a stream cipher based on cellular automaton rule 30. To the best of our knowledge this is the first application of fault attacks against cellular automata based stream ciphers.*

1. Introduction

1.1. Overview

Traditionally, cryptanalytic methods have been concentrated on exploiting vulnerabilities in the algorithmic structure of the target cryptosystem. The conventional approach often ignores the inherent physical properties of the implementation. In this context, the fault analysis model emerged as an alternative that allowed the development of a variety of realistic attacks against symmetric and asymmetric cryptographic protocols. The fault analysis model relies on the principle that the adversary can physically control the cryptographic device, and induce it to abnormally operate, making it to output faulty results. These faulty results can later on be used by the adversary to derive secret information, such as recovering a shared secret key. Depending on the implementation, an attacker can perform fault injections into the device by several ways: exposing it to heat or radiation, changing its power supply voltage, increasing its external clock, provoking temperature variations, among other possibilities.

Fault analysis was introduced by Boneh, DeMillo, and Lipton [2], who used this technique to attack public key cryptosystems, such as digital signature and identification schemes. Their results stimulated a progressive research in the area, and since then other significant results have been obtained. Remarkably, Biham and Shamir [1] used differential fault analysis to attack block ciphers, such as DES. More recently, Hoch and Shamir [4] developed a systematic study about the vulnerabilities of stream ciphers in the fault analysis setting. Despite all the active research regarding the field, there are no published results about the use of fault attacks against cellular automata based stream ciphers. One of the goals of this paper is to fill this gap by presenting some practical fault attacks against

the aforementioned cryptosystem. The effectiveness of the proposed attacks demonstrates that fault analysis represents a major threat for cellular automata based ciphers.

1.2. Cellular Automata Based Stream Ciphers

Wolfram [9, 10] was the first to notice the usefulness of cellular automata as stream ciphers major building blocks. Wolfram pointed out that some rules used to define the temporal evolution of one-dimensional cellular automata generated seemingly pseudo-random behaviors. The proposed family of stream ciphers was very fast yielding efficient implementations in hardware and software. Later analysis [5] showed that the cipher's parameters initially proposed by Wolfram were too optimistic, however for appropriate key sizes all the known attacks against the cipher proposed in [9, 10] have exponential complexity.

Later, many other ciphers based on cellular automata were proposed [3, 6, 7, 8] but the overall goal was the same: to exploit the apparently simple rules and architecture of cellular automata for obtaining efficient ciphers.

1.3. The Attack Model

Roughly speaking, in the fault analysis model, the adversary is focused on attacking the physical implementation rather than the cryptographic algorithm.

We assume that the adversary has physical access to the device, but no previous knowledge about the key. The adversary is allowed to run the device several times while provoking faults into chosen memory areas of the same device. Specifically, we consider that the adversary is able to apply bit flipping faults to either the RAM or the internal registers of the device. Moreover, he/she can arbitrarily reset the cryptographic device and later induce other randomly chosen faults into it.

We also assume that the adversary knows small parts of the plaintext (thus obtaining also parts of the keystream). This is a widely used assumption in cryptography (known as chosen plaintext attacks) and is quite realistic as parts of the message (such as headers) are usually predictable by an attacker.

1.4. Rule 30 Stream Cipher

Cellular automata theory describes, among other things, how simple and well-defined rules can lead to complex structures. It is claimed that the random properties of cellular automata could be used to implement secure, simple, fast, and low cost cryptosystems. In his seminal paper [9], Wolfram proposed the use of cellular automaton rule 30 as a keystream generator for a stream cipher. The resultant encryption scheme is the so-called Rule 30 Stream Cipher.

A cellular automaton consists of a one-dimensional circular register of n cells, where each cell can present one of two possible states (values), 0 or 1. These cells are updated in parallel in discrete time steps according to a next state function (or rule). Let a_i^t denote the value of the i-th cell at time t. The rule 30 is given by:

$$a_i^t = a_{i-1}^{t-1} \, XOR \, \left(a_i^{t-1} \, OR \, a_{i+1}^{t-1} \right) \tag{1}$$

2. Fault Analysis of the Rule 30 Stream Cipher

This section introduces our fault attack on the Rule 30 Stream Cipher, described earlier in section 1.4. For sake of feasibility, the following assumptions are made:

- The adversary knows a sequence of $n/2+1$ bits extracted from the register of the cryptographic device. I.e., he/she knows a_i^t, for $t = 1, \ldots, n/2+1$. This sequence is stored in the central column of a matrix **A**
- The adversary has the capability of changing the content of chosen areas of the register, i.e., of flipping their stored value. He/she can also reset the device.

This cryptanalytic technique is divided into 3 steps (or phases). In the first step, we determine the bits of the two columns adjacent to the central column. In the second step, we proceed with the determination of the bits on the right side of the central column. In step 3, we determine the bits on the left side of the central column.

As will be shown, as the attack goes on, the actions taken by the cryptanalyst will depend on the observed configuration of the cells. The method is described below.

STEP 1: Determination of the bits of the two columns adjacent to the central column.

This step consists of provoking a fault into the register for each known pair of bits. It is possible to determine the two pairs of bits adjacent to the central column.

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix}$$

Configuration 1.1

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} a_{i-1}^{t-1} & 0 & a_{i+1}^{t-1} \\ x & 0 & x \end{pmatrix}$$

This configuration is only possible if $a_{i-1}^{t-1} = a_{i+1}^{t-1} = 1$ or $a_{i-1}^{t-1} = a_{i+1}^{t-1} = 0$. If the attacker flips the bit a_i^{t-1} and then recomputes the bit a_i^t, there will be two possibilities:

- If $a_i^t = 0$, then $a_{i-1}^{t-1} = a_{i+1}^{t-1} = 1$
- If $a_i^t = 1$, then $a_{i-1}^{t-1} = a_{i+1}^{t-1} = 0$

Configuration 1.2

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} a_{i-1}^{t-1} & 0 & a_{i+1}^{t-1} \\ x & 1 & x \end{pmatrix}$$

This configuration is only possible if $a_{i-1}^{t-1} = 0$ and $a_{i+1}^{t-1} = 1$ or $a_{i-1}^{t-1} = 1$ and $a_{i+1}^{t-1} = 0$. If the attacker flips the bit a_i^{t-1} and then recomputes the bit a_i^t, there will be two possibilities:

- If $a_i^t = 1$, then $a_{i-1}^{t-1} = 0$ and $a_{i+1}^{t-1} = 1$
- If $a_i^t = 0$, then $a_{i-1}^{t-1} = 1$ and $a_{i+1}^{t-1} = 0$

Configuration 1.3

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} a_{i-1}^{t-1} & 1 & a_{i+1}^{t-1} \\ x & 0 & x \end{pmatrix}$$

This configuration allows one to immediately determine $a_{i-1}^{t-1} = 1$. However, a_{i+1}^{t-1} remains undefined. If the attacker flips the bit a_i^{t-1} and then recomputes the bit a_i^t, there will be two possibilities:

- If $a_i^t = 1$, then $a_{i-1}^{t-1} = 1$ and $a_{i+1}^{t-1} = 0$
- If $a_i^t = 0$, then $a_{i-1}^{t-1} = 1$ and $a_{i+1}^{t-1} = 1$

Configuration 1.4

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} a_{i-1}^{t-1} & 1 & a_{i+1}^{t-1} \\ x & 1 & x \end{pmatrix}$$

This configuration allows one to immediately determine $a_{i-1}^{t-1} = 0$. However, a_{i+1}^{t-1} remains undefined. If the attacker flips the bit a_i^{t-1} and then recomputes the bit a_i^t, there will be two possibilities:

- If $a_i^t = 1$, then $a_{i-1}^{t-1} = 0$ and $a_{i+1}^{t-1} = 1$
- If $a_i^t = 0$, then $a_{i-1}^{t-1} = 0$ and $a_{i+1}^{t-1} = 0$

STEP 2: Determination of the bits on the right side of the central column.

Assume the following notation.

$$\begin{pmatrix} \alpha & \beta \\ x & \delta \end{pmatrix} = \begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_i^t \end{pmatrix}$$

One can easily see that there are 8 different possibilities for the bits $\{\alpha, \beta, \delta\}$. Basing on equation 1, an adversary in possession of the bits $\{\alpha, \beta, \delta\}$ can determine the bit a_{i+1}^{t-1} by applying one fault into the register.

We shall now analyze these 8 possibilities.

Configuration 2.1

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 0 & 0 & a_{i+1}^{t-1} \\ x & 0 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i+1}^{t-1} = 0$.

Configuration 2.2

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 0 & 0 & a_{i+1}^{t-1} \\ x & 1 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i+1}^{t-1} = 1$.

Configuration 2.3

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 0 & 1 & a_{i+1}^{t-1} \\ x & 0 & x \end{pmatrix}$$

- This configuration is not possible.

Configuration 2.4

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 0 & 1 & a_{i+1}^{t-1} \\ x & 1 & x \end{pmatrix}$$

If the attacker flips the bit a_i^{t-1} and then recomputes the bit a_i^t, there will be two possibilities:

- If $a_i^t = 1$, then $a_{i+1}^{t-1} = 1$.
- If $a_i^t = 0$, then $a_{i+1}^{t-1} = 0$.

Configuration 2.5

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 1 & 0 & a_{i+1}^{t-1} \\ x & 0 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i+1}^{t-1} = 1$.

Configuration 2.6

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 1 & 0 & a_{i+1}^{t-1} \\ x & 1 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i+1}^{t-1} = 0$.

Configuration 2.7

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 1 & 1 & a_{i+1}^{t-1} \\ x & 0 & x \end{pmatrix}$$

If the attacker flips the bit a_i^{t-1} and then recomputes the bit a_i^t, there will be two possibilities:

- If $a_i^t = 1$, then $a_{i+1}^{t-1} = 0$.
- If $a_i^t = 0$, then $a_{i+1}^{t-1} = 1$.

Configuration 2.8

$$\begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} & a_{i+1}^{t-1} \\ x & a_i^t & x \end{pmatrix} = \begin{pmatrix} 1 & 1 & a_{i+1}^{t-1} \\ x & 1 & x \end{pmatrix}$$

- This configuration is not possible.

STEP 3: Determination of the bits on the left side of the central column.

Assume the following notation.

$$\begin{pmatrix} \alpha & \beta \\ \delta & x \end{pmatrix} = \begin{pmatrix} a_{i-1}^{t-1} & a_i^{t-1} \\ a_{i-1}^{t} & x \end{pmatrix}$$

One can easily see that there are 8 different possibilities for the bits $\{\alpha, \beta, \delta\}$. Basing on equation 1, an adversary in possession of the bits $\{\alpha, \beta, \delta\}$ can determine the bit a_{i-2}^{t-1} without applying any fault into the register.

We shall now analyze these 8 possibilities.

Configuration 3.1

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 0 & 0 \\ x & 0 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 0$.

Configuration 3.2

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 0 & 0 \\ x & 1 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 1$.

Configuration 3.3

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 1 & 0 \\ x & 0 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 1$.

Configuration 3.4

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 1 & 0 \\ x & 1 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 0$.

Configuration 3.5

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 0 & 1 \\ x & 0 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 1$.

  Configuration 3.6

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 0 & 1 \\ x & 1 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 0$.

  Configuration 3.7

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 1 & 1 \\ x & 0 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 1$.

  Configuration 3.8

$$\begin{pmatrix} a_{i-2}^{t-1} & a_{i-1}^{t-1} & a_i^{t-1} \\ x & a_{i-1}^{t} & x \end{pmatrix} = \begin{pmatrix} a_{i-2}^{t-1} & 1 & 1 \\ x & 1 & x \end{pmatrix}$$

- This configuration is only possible for $a_{i-2}^{t-1} = 0$.

3. Further Explanation on the Rule 30 Stream Cipher Fault Analysis

In this part, we explain in-depth why our fault attack on the rule 30 stream cipher works.

The main idea is simple and quite intuitive. We recall that the attacker knows a sequence of $n/2+1$ cells, which are located on the central column of a matrix **A**. We also recall equation 1.

$$a_i^t = a_{i-1}^{t-1}\, XOR\, \left(a_i^{t-1}\, OR\, a_{i+1}^{t-1}\right)$$

In the **first step** of our attack, the cryptanalyst intends to discover the values a_{i-1}^{t-1} and a_{i+1}^{t-1}, given the values that he/she knows, i.e., a_i^t and a_i^{t-1}. However, he/she has two variables and only a boolean equation (this initial configuration is displayed in figure 1).

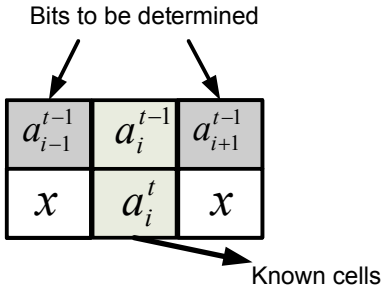

Figure 1. Initial configuration.

Our fault analysis capabilities allow the cryptanalyst to obtain another boolean equation by flipping the bit a_i^{t-1} and observing the new value assumed by a_i^t after the fault injection. So, at the end of this action, the cryptanalyst will have a simple system of the form:

$$\begin{cases} [a_i^t]_{flipping}^{before} = a_{i-1}^{t-1}\ XOR\ \left(b\ OR\ a_{i+1}^{t-1}\right) \\ [a_i^t]_{flipping}^{after} = a_{i-1}^{t-1}\ XOR\ \left(\overline{b}\ OR\ a_{i+1}^{t-1}\right) \end{cases} \tag{2}$$

where $[a_i^t]_{flipping}^{before}$ and $[a_i^t]_{flipping}^{after}$ denote the values observed at cell a_i^t before and after the fault injection, respectively. Before, the fault injection $a_i^{t-1} = b$ and after this, $a_i^{t-1} = \overline{b}$, where $\overline{b}$ is the complementary value of b. It is easy to find the solution for this system. The action performed is illustrated in figure 2.

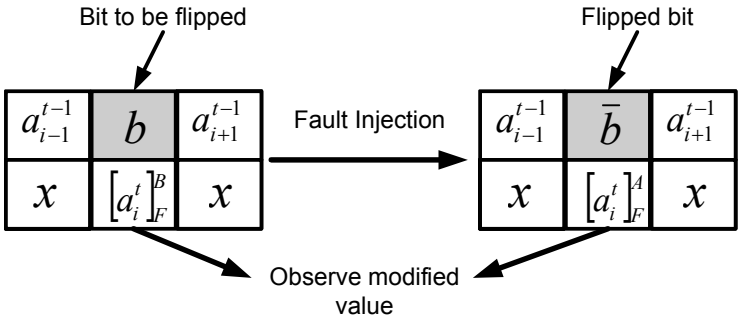

Figure 2. Illustration of the main idea involved in the first step of our attack.

Steps 2 and **3** are trivial, and we hardly have to perform a flip action, because, usually, we have equation with only one variable to be determined. Figure 3 suggests that.

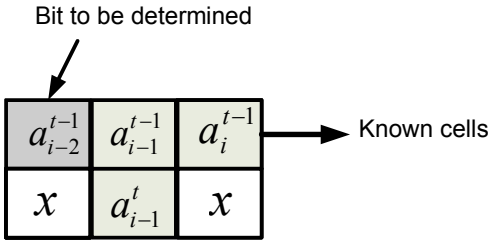

Figure 3. Illustration of step 3 configuration.

Regarding the complexity of the attack, it is easy to obtain an estimation on the number of faults required to break the cryptosystem. At **step 1**, we provoke $n/2$ fault injections to determine the two pairs of bits adjacent to the central column. At **step 2**, there are $(1/2) \times [(n/2) \times (n/2)]$ bits, and, on average, we provoke 0.25 fault for each bit to be determine. So, **step 2** requires $n^2/32$ faults. At **step 3**, as explained previously, no faults need to be realized. Thus,

$$\text{Number of Faults }_{\text{Rule30}} = \frac{n^2}{32} + \frac{n}{2}$$

3.1. Fault Analysis Effect on Rule 30 Stream Cipher

One of the most known cryptanalytic techniques against the rule 30 stream cipher was proposed by Meier and Staffelbach [5]. Their statistical technique allows one to determine secret keys with lengths varying between 300 and 500 bits by using personal computers. However, it is claimed that the recovery of secret keys of 1,000 bits would demand the use of large scale parallel computers.

On the other hand, the attack based on fault analysis assumptions has proven to be efficient and feasible for a large set of key lengths. For instance, to obtain a secret key of 256 bits, the cryptanalyst should inject 2^7+2^{11} faults and know (256/2)+1 = 129 bits of the generated sequence. To obtain a key of 1,024 bits, 2^9+2^{15} fault injections and 512 known bits are needed. This amount of operations can be performed by any personal computer equipped with current technology. Besides that, the operations required to implement the attack are simple (comparisons and fault injections) and can be performed by any unsophisticated computational system.

4. Conclusions

Cellular automata based stream ciphers were designed to respond to the increasing need of fast and low-cost cryptographic mechanisms. However, we have shown how devastating fault analysis can be when applied to some of those cryptosystems.

Our fault attack against the rule 30 stream cipher needs, in order to determine a secret key of length n, only $n/2 + n^2/32$ fault injections and a sequence of $n/2 + 1$ bits.

It is an interesting future research direction to see how well the results introduced here apply to other ciphers designed for low-cost, high performance cryptographic devices.

References

[1] Eli Biham, Adi Shamir. A New Cryptanalytic Attack on DES: Differential Fault Analysis. Preprint, October 1996.

[2] Dan Boneh, Richard A. DeMillo and Richard J. Lipton. On the Importance of Checking Cryptografic Protocols for Faults. Advances in Cryptology – EUROCRYPT 1997, Lecture Notes in Computers Science vol.1233, Springer-Verlag, pp. 37–51, May 1997.

[3] A. Fuster-Sabater, P. Caballero-Gil and M.E. Pazo-Robles, Application of Linear Hybrid Cellular Automata to Stream Ciphers, EUROCAST 2007, Lecture Notes in Computer Science vol. 4739, pp. 564-571, 2007

[4] Jonathan J. Hock and Adi Shamir. Faut Analysis of Stream Ciphers. CHES 2004, Lecture Notes in Computer Science vol. 3156, Springer-Verlag, pp. 240–253, 2004.

[5] Willi Meier and Othmar Staffelbach. Analysis of Pseudo Random Sequences Generated by Cellular Automata. Advances in Cryptology – EUROCRYPT 1991, Lecture Notes in Computer Science vol. 547, Springer-Verlag, pp. 186–199, 1991.

[6] S. Nandi, B.K. Par, P. Pal Chaudhuri. Theory and Application of Cellular Automata in Cryptography, IEEE Transactions on Computers, vol 43, Issue 12, pp.1346-1357, 1994

[7] F. Seredynsky, P. Bouvry and A. Zomaya. Cellular Automata Computations and Secret Key Cryptography. Parallel Computing, Vol. 30, Issues 5-6, pp. 753-766, 2004.

[8] M. Tomassini and M Perrenoud. Cryptography with Cellular Automata, Applied Soft Computing, vol 1, Issue 2, pp. 151-160, 2001.

[9] S. Wolfram. Cryptography with Cellular Automata. Advances in Cryptology - CRYPTO 1985, Proceedings, Springer-Verlag, pp. 429–432, 1986.

[10] S. Wolfram. Random Sequence Generation by Cellular Automata. Advances in Applied Mathematics 7, pp. 123–169, 1986.