On the Use of Cellular Automata in Symmetric Cryptography

A. Fúster-Sabater\(^{(1)}\) and P. Caballero-Gil\(^{(2)}\)

\(^{(1)}\) Institute of Applied Physics, C.S.I.C., Serrano 144, 28006 Madrid, Spain
amparo@iec.csic.es

\(^{(2)}\) DEIOC, University of La Laguna, 38271 La Laguna, Tenerife, Spain
pcaballe@ull.es

Abstract

In this work, pseudorandom sequence generators based on finite fields have been analyzed from the point of view of their cryptographic application. In fact, a class of nonlinear sequence generators has been modelled in terms of linear cellular automata. The algorithm that converts the given generator into a linear model based on automata is very simple and is based on the concatenation of a basic structure. Once the generator has been linearized, a cryptanalytic attack that exploits the weaknesses of such a model has been developed. Linear cellular structures easily model sequence generators with application in stream cipher cryptography.

Keywords: automata, finite fields, cryptography, sequence generator.
Classification: 11T71, 14G50, 94A60, 40B05

1 Introduction

Confidential information must be encrypted by means of a mathematical function currently called cipher that converts the original information (plaintext) into the ciphered information (ciphertext). Symmetric cryptography is usually divided into two large classes [21]: stream ciphers and block-ciphers. Stream ciphers encrypt each data symbol into a ciphertext symbol under a time-varying transformation. Block-ciphers divide the plaintext into blocks of symbols and by means of a specially constructed function mix the block of plaintext with the secret key in order to produce the block of ciphertext.

Stream ciphers are very fast (in fact, the fastest among the encryption procedures) so they are implemented in many technological applications e.g. algorithms A5 in GSM communications or the encryption system E0 used in the [work supported by Ministerio de Educación y Ciencia (Spain), Projects SEG2004-02418 and SEG2004-04352-C04-03. Acta Applicandae Mathematicae. Volume 93, Numbers 1-3, pp. 215-236. Sept 2006. Springer. DOI:10.1007/s10440-006-9041-6]
Bluetooth specifications or the RC4 function for the application Excel of Microsoft. Stream ciphers try to imitate the ultimate one-time pad cipher \[21\] and are supposed to be good pseudorandom generators capable of stretching a short secret seed (the secret key) into a long sequence of seemingly random bits (the keystream sequence). This sequence is then bit-wise XORed with the plaintext in order to obtain the ciphertext. Finite fields are used in most of the constructions of pseudorandom sequences either under the form of Cellular Automata (CA) or under the form of traditional Linear Feedback Shift Registers (LFSRs).

Cellular Automata (CA) are particular forms of finite state machines that can be investigated by the usual analytic techniques \([10\], \[18\], \[20\], \[26\]). CA have been used in application areas so different as physical systems simulation, biological process, species evolution, socio-economical models or test pattern generation. They are defined as arrays of identical cells in an \(n\)-dimensional space and characterized by different parameters \[27\]: the cellular geometry, the neighborhood specification, the number of contents per cell and the transition rule to compute the successor state. Their simple, modular and cascadable structure makes them very attractive for VLSI implementations.

On the other hand, LFSRs \[11\] are linear structures currently used in the generation of pseudorandom sequences. The inherent simplicity of LFSRs, their ease of implementation and the good statistical properties of their output sequences turn them into natural building blocks for the design of pseudorandom sequence generators with applications in spread-spectrum communications, circuit testing, error-correcting codes, numerical simulations or cryptography.

In recent years, one-dimensional CA have been proposed as an alternative to LFSRs \([2\], \[3\], \[20\], \[27\]) in the sense that every sequence generated by a LFSR can be obtained from one-dimensional CA too. In cryptographic applications, pseudorandom sequence generators currently involve several LFSRs combined by means of nonlinear functions or irregular clocking techniques (see \[19\], \[21\]). Moreover in \[22\], it is proved that one-dimensional linear CA are isomorphic to conventional LFSRs. Thus, the latter structures can be simply substituted by the former ones in order to accomplish the same goal: generation of keystream sequences.

The above class of linear CA has been found to satisfy randomness properties with application in the testing of digital circuits and self-checking \[28\]. The current interest of these CA stems from the lack of correlation between the bit sequences generated by adjacent cells, see \[9\]. In this sense, linear CA are superior to the more common LFSRs \[11\] that have been traditionally used in stream ciphers. Nevertheless, the main advantage of CA is that multiple generators designed as nonlinear structures in terms of LFSRs preserve the linearity when they are expressed under the form of CA.

This paper considers the problem of finding one-dimensional CA that reproduce the output sequence of a particular LFSR-based generator. More precisely, in this work a wide class of LFSR-based nonlinear generators, the so-called Clock-Controlled Shrinking Generators (CCSGs) \[15\], can be described in terms of one-dimensional CA configurations. Indeed, the well known Shrinking Generator \[8\] is just an element of such a class. The automata here presented
unify in a simple structure the above mentioned class of sequence generators. The algorithm that converts a given CCSG into a CA-based linear model is very simple and can be applied to CCSGs in a range of practical interest. The underlying idea of this modelling procedure is the concatenation of a basic automaton. Once the generators have been linearized, a cryptanalytic approach to reconstruct the generated sequence is also presented.

The paper is organized as follows: in section 2, the basic structures considered, e.g. one-dimensional CA and CCSGs, are introduced. A simple algorithm to determine the pair of CA corresponding to a particular shrinking generator and its generalization to Clock-Controlled Shrinking Generators are given in sections 3 and 4, respectively. A method of reconstructing the generated sequence that exploits the linearity of the CA-based model is presented in section 5. Finally, conclusions in section 6 end the paper.

2 Basic Structures

In the following subsections, we introduce the general characteristics of the basic structures we are dealing with: linear feedback shift registers, one-dimensional cellular automata, the shrinking generator and the class of clock-controlled shrinking generators. The work is restricted to binary structures, that is the contents of CA as well as those of LFSRs belong to $GF(2)$.

2.1 Linear Feedback Shift Registers

A binary LFSR is an electronic device with $L$ memory cells (stages), numbered $0, 1, ..., L - 1$, each of one capable of storing one bit. The binary content of the $L$ stages at each unit of time is the state of the LFSR at that instant. In addition, a clock controls the shift of data. At each unit of time the following operations are performed: (i) The content of stage 0 is output; (ii) the content of stage $i$ is moved to stage $i - 1$ for each $i$, $1 \leq i \leq L - 1$; (iii) The new content of stage $L - 1$ is the exclusive-OR of a subset of stages given by $P(x)$, that is the LFSR connection polynomial. If $P(x)$ is a primitive polynomial of degree $L$ [17], then the LFSR is called a maximum-length LFSR and its output sequence is a PN-sequence. Period, balancedness, run distribution and correlation properties of PN-sequences have been exhaustively studied in the literature, see [11] and [19]. In the sequel, only maximum-length LFSRs will be considered.

2.2 One-Dimensional Cellular Automata

One-dimensional cellular automata can be described as $L$-cell registers [4], whose cell contents are updated at the same time instant according to a particular $k$-variable function (the transition rule) denoted by $\Phi$. If the function $\Phi$ is a linear function, so is the cellular automaton. In addition, for cellular automata with binary contents there can be up to $2^{2^k}$ different mappings to the next state. Moreover, if $k = 2r + 1$, then the binary content of the $i$-th cell at time $t + 1$
depends on the contents of \( k \) neighbor cells at time \( t \) in the following way:

\[
x_{i}^{t+1} = \Phi(x_{i-r}^t, \ldots, x_{i-r}^t, \ldots, x_{i+r}^t) \quad (i = 1, \ldots, L).
\]

(1)

The number of cells \( L \) (numbered from left to right) is the length of the automaton. CA are called uniform whether all cells evolve under the same rule while CA are called hybrid whether different cells evolve under different rules. At the ends of the array, two different boundary conditions are possible: null automata when cells with permanent null contents are supposed adjacent to the extreme cells or periodic automata when extreme cells are supposed adjacent.

In this paper, only transition rules with \( k = 3 \) will be considered. Thus, there are \( 2^8 \) of such rules among which just two (rule 90 and rule 150) lead to non trivial machines. Such rules are described as follows:

**Rule 90**

\[
x_{i}^{t+1} = \Phi_{90}(x_{i-1}^t, x_{i}^t, x_{i+1}^t) = x_{i-1}^t + x_{i+1}^t
\]

\[
\begin{array}{cccccccc}
111 & 110 & 101 & 100 & 011 & 010 & 001 & 000 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0
\end{array}
\]

**Rule 150**

\[
x_{i}^{t+1} = \Phi_{150}(x_{i-1}^t, x_{i}^t, x_{i+1}^t) = x_{i-1}^t + x_{i}^t + x_{i+1}^t
\]

\[
\begin{array}{cccccccc}
111 & 110 & 101 & 100 & 011 & 010 & 001 & 000 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
\]

Remark that the names rule 90 and rule 150 derive from the decimal values of their next-state functions: 01011010 (binary) = 90 (decimal) and 10010110 (binary) = 150 (decimal). Indeed, \( x_{i}^{t+1} \) the content of the \( i \)-th cell at time \( t+1 \) depends on the contents of either two different cells (rule 90) or three different cells (rule 150) at time \( t \). The symbol + denotes addition modulo 2 among cell contents. Remark that both transition rules are linear. This work deals exclusively with one-dimensional linear null hybrid CA with rules 90 and 150. A natural way of specifying such CA is an \( L \)-tuple \( M = [R_1, R_2, \ldots, R_L] \), called rule vector, where \( R_i = 0 \) if the \( i \)-th cell satisfies rule 90 while \( R_i = 1 \) if the \( i \)-th cell satisfies rule 150. A sub-automaton of the previous automata class consisting of cells 1 through \( i \) will be denoted by \( R_1 R_2 \ldots R_i \).

For a cellular automaton of length \( L = 10 \) cells, configuration rules ( \( R_1 = 0, R_2 = 1, R_3 = 1, R_4 = 1, R_5 = 0, R_6 = 0, R_7 = 1, R_8 = 1, R_9 = 1, R_{10} = 0 \) ) and initial state \((0,0,0,1,1,1,0,1,1,0)\), Table 1 illustrates the formation of its output sequences (binary sequences read vertically) and the succession of states (binary configurations of 10 bits read horizontally). For the above mentioned rules, the different states of the automaton are grouped in closed cycles. The number of different output sequences for a particular cycle is \( \leq L \) as the same sequence (although shifted) may appear simultaneously in different cells. At
Table 1: An one-dimensional linear null hybrid cellular automaton of 10 cells with rules 90/150 starting at a given initial state

|    | 90 | 150 | 150 | 150 | 90 | 90 | 150 | 150 | 150 | 90 |
|----|----|-----|-----|-----|----|----|-----|-----|-----|----|
| 0  | 0  | 0   | 1   | 1   | 1  | 0  | 1   | 1   | 0   | 0  |
| 0  | 0  | 1   | 0   | 0   | 1  | 0  | 0   | 0   | 0   | 1  |
| 0  | 1   | 1   | 1   | 0   | 1  | 0  | 1   | 0   | 1   | 0  |
| 1   | 0   | 1   | 1   | 0   | 1  | 0  | 1   | 0   | 1   | 0  |
| 0   | 0   | 0   | 1   | 1   | 0  | 1  | 0   | 1   | 0   | 0  |
| 0   | 0   | 1   | 0   | 1   | 0  | 1  | 1   | 1   | 0   | 0  |
| 0   | 1   | 1   | 0   | 0   | 0  | 0  | 1   | 0   | 1   | 0  |
| 1   | 0   | 0   | 1   | 0   | 0  | 1  | 1   | 0   | 0   | 0  |
| 0   | 1   | 1   | 1   | 1   | 1  | 0  | 1   | 0   | 1   | 1  |
| 1   | 0   | 1   | 1   | 0   | 1  | 1  | 1   | 1   | 1   | 1  |

On the other hand, linear finite state machines are currently represented and analyzed by means of their transition matrices. The form and characteristics of these matrices for the CA under consideration can be found in [4]. In fact, such matrices are tri-diagonal matrices with the rule vector on the main diagonal, 1’s on the diagonals below and above the main one and all other entries being zero. Every automaton is completely specified by its characteristic polynomial, that is the characteristic polynomial of its transition matrix. Such a characteristic polynomial can be computed in terms of the characteristic polynomials of the previous sub-automata according to the recurrence relation [4]:

\[ P_i(x) = (x + R_i)P_{i-1}(x) + P_{i-2}(x), \quad 0 < i \leq L \quad (2) \]

being \( P_{-1}(x) = 0 \) and \( P_0(x) = 1 \). Next, the following definition is introduced:

**Definition 2.1** A Multiplicative-Polynomial Cellular Automaton is defined as a cellular automaton whose characteristic polynomial is a reducible polynomial of the form \( P_M(x) = (P(x))^p \) where \( p \) is a positive integer. If \( P(x) \) is a primitive polynomial, then the automaton is called a Primitive Multiplicative-Polynomial Cellular Automaton.

The class of binary sequence generators we are dealing with is described in the following subsections.

### 2.3 The Shrinking Generator

The shrinking generator is a binary sequence generator [8] composed by two LFSRs: a control register \( SR_1 \) that decimates the sequence produced by the
other register \( SR_2 \). We denote by \( L_j (j = 1, 2) \) their corresponding lengths with \((L_1, L_2) = 1\) as well as \( S_1 < L_2 \). Then, we denote by \( C_j(x) \in GF(2)[x] \) \((j = 1, 2)\) their corresponding characteristic polynomials of degree \( L_j \) \((j = 1, 2)\), respectively.

The sequence produced by \( SR_2 \), denoted by \( \{a_i\} \), controls the bits of the sequence produced by \( SR_2 \), that is \( \{b_i\} \), which are included in the output sequence \( \{z_j\} \) (the shrunken sequence), according to the following rule \( P \):

1. If \( a_i = 1 \Rightarrow z_j = b_i \)
2. If \( a_i = 0 \Rightarrow b_i \) is discarded.

A simple example illustrates the behavior of this structure.

**Example 2.2** Let us consider the following LFSRs:

1. Register \( SR_1 \) of length \( L_1 = 3 \), characteristic polynomial \( C_1(x) = 1 + x^2 + x^3 \) and initial state \( IS_1 = (1, 0, 0) \). The PN-sequence generated by \( SR_1 \) is \( \{1, 0, 1, 0, 0, 1\} \) with period \( T_1 = 2^{L_1} - 1 = 7 \).
2. Register \( SR_2 \) of length \( L_2 = 4 \), characteristic polynomial \( C_2(x) = 1 + x + x^4 \) and initial state \( IS_2 = (1, 0, 0, 0) \). The PN-sequence generated by \( SR_2 \) is \( \{1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1\} \) with period \( T_2 = 2^{L_2} - 1 = 15 \).

The output sequence \( \{z_j\} \) is given by:

- \( \{a_i\} \rightarrow 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 \ldots \)
- \( \{b_i\} \rightarrow 1 0 0 0 0 1 0 0 1 1 0 1 1 0 0 1 0 \ldots \)
- \( \{z_j\} \rightarrow 1 0 1 0 1 1 0 1 1 0 0 1 0 \ldots \)

The underlined bits 0 or 1 in \( \{b_i\} \) are discarded.

In brief, the sequence produced by the shrinking generator is an irregular decimation of \( \{b_i\} \) from the bits of \( \{a_i\} \). According to [8], the period of the shrunken sequence is

\[
T = (2^{L_2} - 1) 2^{(L_1 - 1)}
\] 

and its linear complexity [21], notated \( LC \), satisfies the following inequality

\[
L_2 2^{(L_1 - 2)} < LC < L_2 2^{(L_1 - 1)}.
\]

A simple calculation, based on the fact that every state of \( SR_2 \) coincides once with every state of \( SR_1 \), allows one to compute the number of 1’s in the shrunken sequence. Such a number is constant and equal to

\[
No. \ 1's = 2^{(L_2 - 1)} 2^{(L_1 - 1)}.
\]

Comparing period and number of 1’s, it can be concluded that the shrunken sequence is a quasi-balanced sequence.

In addition, it can be proved [8] that the output sequence has good distributional statistics too. Therefore, this scheme is suitable for practical implementation of stream ciphers and pattern generators.
2.4 The Clock-Controlled Shrinking Generators

The Clock-Controlled Shrinking Generators constitute a wide class of clock-controlled sequence generators \[15\] with applications in cryptography, error correcting codes and digital signature. An CCSG is a sequence generator composed of two LFSRs notated \(SR_1\) and \(SR_2\). The parameters of both registers are defined as those of subsection 2.3. At any time \(t\), \(SR_1\) (the control register) is clocked normally while the second register \(SR_2\) is clocked a number of times given by an integer decimation function notated \(X_t\). In fact, if \(A_0(t), A_1(t), \ldots, A_{L_1-1}(t)\) are the binary cell contents of \(SR_1\) at time \(t\), then \(X_t\) is defined as

\[
X_t = 1 + 2^0 A_{i_0}(t) + 2^1 A_{i_1}(t) + \ldots + 2^{w-1} A_{i_{w-1}}(t)
\]

where \(i_0, i_1, \ldots, i_{w-1} \in \{0, 1, \ldots, L_1 - 1\}\) and \(0 < w \leq L_1 - 1\).

In this way, the output sequence of an CCSG is obtained from a double decimation:

1. The output sequence of \(SR_2\), \(\{b_i\}\), is decimated by means of \(X_t\) giving rise to the sequence \(\{b'_i\}\).

2. The same decimation rule \(P\), defined in subsection 2.3 is applied to the sequence \(\{b'_i\}\).

Remark that if \(X_t \equiv 1\) (no cells are selected in \(SR_1\)), then the proposed generator is just the shrinking generator. Let us see a simple example of CCSG.

**Example 2.3** For the same LFSRs defined in the previous example and the function \(X_t = 1 + 2^0 A_0(t)\) with \(w = 1\), the decimated sequence \(\{b'_i\}\) is given by:

\[
\begin{align*}
\{b_i\} & \rightarrow 1\underline{0}0\underline{1}00\underline{1}01101011111111000010101100101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101
In brief, the sequence produced by an CCSG is an irregular double decimation of the sequence generated by \( SR_2 \) from the function \( X_t \) and the bits of \( SR_1 \). This construction allows one to generate a large family of different sequences by using the same LFSR initial states and characteristic polynomials but modifying the decimation function. Period, linear complexity and statistical properties of the generated sequences by CCSGs have been established in [15].

2.5 Cattel and Muzio Synthesis Algorithm

The Cattell and Muzio synthesis algorithm [5] presents a method of obtaining two CA (based on rules 90 and 150) corresponding to a given polynomial. Such an algorithm takes as input an irreducible polynomial \( Q(x) \in GF(2)[x] \) defined over a finite field and computes two linear reversal CA whose output sequences have \( Q(x) \) as characteristic polynomial. Such CA are written as binary strings with the previous codification: 0 = rule 90 and 1 = rule 150. The theoretical foundations of the algorithm can be found in [7]. The total number of operations required for this algorithm is listed in [5] (Table II, page 334). It is shown that the number of operations grows linearly with the degree of the polynomial, so the method does not suffer from any sort of exponential blow-up. The method is efficient for all practical applications (e.g. in 1996 finding a pair of length 300 CA took 16 CPU seconds on a SPARC 10 workstation). For cryptographic applications, the degree of the irreducible (primitive) polynomial is \( L_2 \approx 64 \), so that the consuming time is negligible.

Finally, a list of One-Dimensional Linear Hybrid Cellular Automata of Degree Through 500 can be found in [6].

3 CA-Based Linear Models for the Shrinking Generator

In this section, an algorithm to determine the pair of CA corresponding to a given shrinking generator is presented. Such an algorithm is based on the following results:

**Lemma 3.1** The characteristic polynomial of the shrunk sequence is of the form \( P(x)^N \), where \( P(x) \in GF(2)[x] \) is a \( L_2 \)-degree primitive polynomial and \( N \) is an integer satisfying the inequality \( 2^{(L_1-2)} < N \leq 2^{(L_1-1)} \).

**Proof:** The shrunk sequence can be written as an interleaved sequence [12] made out of an unique PN-sequence starting at different points and repeated \( 2^{(L_1-1)} \) times. Such a sequence is obtained from \( \{b_i\} \) taking digits separated a distance \( 2^{L_1}-1 \), that is the period of the sequence \( \{a_i\} \). As \( 2^{L_2-1}, 2^{L_1-1} = 1 \) due to the primality of \( L_2 \) and \( L_1 \), the result of the decimation of \( \{b_i\} \) is a PN-sequence of primitive characteristic polynomial \( P(x) \) of degree \( L_2 \). Moreover, the number of times that this PN-sequence is repeated coincides with the number of 1’s in \( \{a_i\} \) since each 1 of \( \{a_i\} \) provides the shrunk sequence with \( 2^{L_2-1} \) digits.
of \{b_i\}. Consequently, the characteristic polynomial of the shrunken sequence will be \(P(x)^N\) with \(N \leq 2^{(L_1-1)}\). The lower limit follows immediately from equation (4) that defines the linear recurrence relationship.

**Lemma 3.2** Let \(C_2(x) \in GF(2)[x]\) be the characteristic polynomial of \(SR_2\) and let \(\lambda\) be a root of \(C_2(x)\) in the extension field \(GF(2^{L_2})\). Then, \(P(x) \in GF(2)[x]\) is of the form

\[
P(x) = (x + \lambda^E)(x + \lambda^{2E}) \ldots (x + \lambda^{2^{L_2-1}E})
\]

being \(E\) an integer given by

\[
E = 2^0 + 2^1 + \ldots + 2^{L_1-1}.
\]

**Proof:** As the decimation of the sequence \(\{b_i\}\) is realized taking one out of \(2^{L_1-1}\) digits, the obtained \(PN\)-sequence is nothing but the characteristic sequence associated to the cyclotomic coset \(E = 2^{L_1-1}\), see \([11]\). Hence, the roots of its characteristic polynomial will be \(\lambda^E, \lambda^{2E}, \ldots, \lambda^{2^{L_2-1}E}\). According to the definition of cyclotomic coset, the value of \(E\) is given by equation (8).

Remark that \(P(x)\) depends exclusively on the characteristic polynomial of the register \(SR_2\) and on the length \(L_1\) of the register \(SR_1\). Based on the Cattell and Muzio synthesis algorithm \([5]\), the following result is derived:

**Lemma 3.3** Let \(Q(x) \in GF(2)[x]\) be a polynomial defined over a finite field and let \(s_1\) and \(s_2\) two binary strings codifying the two linear CA obtained from the Cattell and Muzio algorithm. Then, the two CA in form of binary strings whose characteristic polynomial is \(Q(x)^2\) are:

\[
S'_i = S_i \ast S_i^*\quad i = 1, 2
\]

where \(S_i\) is the binary string \(s_i\) whose least significant bit has been complemented, \(S_i^*\) is the mirror image of \(S_i\) and the symbol \(\ast\) denotes concatenation.

**Proof:** The result is just a generalization of the Cattell and Muzio synthesis algorithm. The concatenation is due to the fact that rule 90 (150) at the end of the array in null automata is equivalent to two consecutive rules 150 (90) with identical sequences. The fact of that an automaton and its reversal version have the same characteristic polynomial completes the proof. Proceeding in the same way a number of times, a multiplicative-polynomial cellular automaton \([2, 3]\) is obtained. In this way, the construction of a linear structure from the concatenation of a basic automaton is accomplished.

According to the previous results, an algorithm to linearize the shrinking generator is introduced:

**Input:** A shrinking generator characterized by two LFSRs, \(SR_1\) and \(SR_2\), with their corresponding lengths, \(L_1\) and \(L_2\), and the characteristic polynomial \(C_2(x)\) of the register \(SR_2\).
Step 1 From $L_1$ and $C_2(x)$, compute the polynomial $P(x)$ in $GF(2^{L_2})$ as

$$P(x) = (x + \lambda^E)(x + \lambda^{2E}) \cdots (x + \lambda^{2^{L_2-1}E})$$

with $E = 2^0 + 2^1 + \ldots + 2^{L_1-1}$.

Step 2 From $P(x)$, apply the Cattell and Muzio synthesis algorithm to determine two linear CA (with rules 90 and 150), notated $s_i$, whose characteristic polynomial is $P(x)$.

Step 3 For each $s_i$ separately, proceed:

3.1 Complement its least significant bit. The resulting binary string is notated $S_i$.
3.2 Compute the mirror image of $S_i$, notated $S_i^*$, and concatenate both strings

$$S'_i = S_i * S_i^*.$$  
3.3 Apply steps 3.1 and 3.2 to each $S_i'$ recursively $L_1 - 1$ times.

Output: Two binary strings of length $L = L_2 \cdot 2^{L_1-1}$ codifying two CA corresponding to the given shrinking generator.

Remark 3.4 In this algorithm the characteristic polynomial of the register $SR_1$ is not needed. Thus, all the shrinking generators with the same $SR_2$ but different registers $SR_1$ (all of them with the same length $L_1$) can be modelled by the same pair of one-dimensional linear CA.

Remark 3.5 It can be noticed that the computation of both CA is proportional to $L_1$ concatenations. Consequently, the algorithm can be applied to shrinking generators in a range of practical application.

Remark 3.6 In contrast to the nonlinearity of the shrinking generator, the CA-based models that generate the shrunk sequence are linear.

In order to clarify the previous steps a simple numerical example is presented.

Input: A shrinking generator characterized by two LFSRs $SR_1$ of length $L_1 = 3$ and $SR_2$ of length $L_2 = 5$ and characteristic polynomial $C_2(x) = 1 + x + x^2 + x^4 + x^5$.

Step 1 $P(x)$ is the characteristic polynomial of the cyclotomic coset $E = 7$. Thus,

$$P(x) = 1 + x^2 + x^5 .$$

Step 2 From $P(x)$ and applying the Cattell and Muzio synthesis algorithm, two reversal linear CA whose characteristic polynomial is $P(x)$ can be determined. Such CA are written in binary format as:
Step 3 Computation of the required pair of CA.

For the first automaton:

\[
\begin{array}{cccccc}
0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
\end{array}
\]

For the second automaton:

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

For each automaton, the procedure of concatenation has been carried out \(L_1 - 1\) times.

**Output:** Two binary strings of length \(L = L_2 \cdot 2^{(L_1 - 1)} = 20\) codifying the required pair of CA.

In this way, we have obtained a pair of linear CA able to generate the shrunken sequence corresponding to the given shrinking generator. In addition, for each one of the previous automata there is one state cycle where the shrunken sequence is generated at each one of the cells.

## 4 CA-Based Linear Models for the Clock Controlled Shrinking Generators

In this section, an algorithm to determine the pair of one-dimensional linear CA corresponding to a given CCSG is presented. Such an algorithm is based on the following results:

**Lemma 4.1** The characteristic polynomial of the output sequence of a CCSG is of the form \(P'(x)^N\), where \(P'(x) \in GF(2)[x]\) is a primitive \(L_2\)-degree polynomial and \(N\) is an integer satisfying the inequality \(2^{(L_1 - 2)} < N \leq 2^{(L_1 - 1)}\).

**Proof:** The proof is analogous to that one developed in lemma 3.1.

Remark that, according to the structure of the CCSGs, the polynomial \(P'(x)\) depends on the characteristic polynomial of the register \(SR_2\), the length \(L_1\) of the register \(SR_1\) and the decimation function \(X_t\). Before, \(P(x)\) was the characteristic polynomial of the *cyclotomic coset* \(E\), where \(E = 2^0 + 2^1 + \ldots + 2^{L_1 - 1}\) was a fixed separation distance between the digits drawn from the sequence \(\{b_i\}\). Now, this distance \(D\) is variable as well as a function of \(X_t\). The computation of \(D\) gives rise to the following result:
Lemma 4.2 Let $C_2(x) \in GF(2)[x]$ be the characteristic polynomial of $SR_2$ and let $\lambda$ be a root of $C_2(x)$ in the extension field $GF(2^{L_2})$. Then, $P'(x) \in GF(2)[x]$ is the characteristic polynomial of cyclotomic coset $D$, where $D$ is given by

$$D = 2^{L_1-w} \left( \sum_{i=1}^{2^w} i \right) - 1 = (1 + 2^w) 2^{L_1-1} - 1. \quad (9)$$

**Proof:** The proof is analogous to that one developed in lemma 3.2. In fact, the distance $D$ can be computed taking into account that the function $X_t$ takes values in the interval $[1, 2, \ldots, 2^w]$ and the number of times that each one of these values appears in a period of the output sequence is given by $2^{L_1-w}$. A simple computation, based on the sum of the terms of an arithmetic progression, completes the proof.

From the previous results, it can be noticed that the algorithm that determines the pair of CA corresponding to a given CCSSG is analogous to that one developed in section 3. Indeed, the expression of $E$ in equation (8) must be replaced by the expression of $D$ in equation (9).

In order to clarify the previous steps a simple numerical example is presented.

**Input:** A CCSSG characterized by: Two LFSRs $SR_1$ of length $L_1 = 3$ and $SR_2$ of length $L_2 = 5$ and characteristic polynomial $C_2(x) = 1 + x + x^2 + x^4 + x^5$ plus the decimation function $X_t = 1 + 2^0 A_0(t) + 2^1 A_1(t) + 2^2 A_2(t)$ with $w = 3$.

**Step 1** $P'(x)$ is the characteristic polynomial of the cyclotomic coset $D$. Now $D \equiv 4 \mod 31$, that is we are dealing with the cyclotomic coset 1. Thus, the corresponding characteristic polynomial is:

$$P'(x) = 1 + x + x^2 + x^4 + x^5.$$  

**Step 2** From $P'(x)$ and applying the Cattell and Muzio synthesis algorithm, two reversal linear CA whose characteristic polynomial is $P'(x)$ can be determined. Such CA are written in binary format as:

$$
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

**Step 3** Computation of the required pair of CA.

For the first automaton:

$$
\begin{array}{cccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}
$$

For the second automaton:

$$
\begin{array}{ccccccccccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$
For each automaton, the procedure of concatenation has been carried out \( L_1 - 1 \) times.

**Output:** Two binary strings of length \( L = 20 \) codifying the required CA.

**Remark 4.3** From a point of view of the CA-based linear models, the shrinking generator or any one of the CCGS are entirely analogous. Thus, the fact of introduce an additional decimation function does neither increase the complexity of the generator nor improve its resistance against cryptanalytic attacks. Indeed, both kinds of generators can be linearized by the same class of CA-based models.

## 5 A Cryptanalytic Approach to this Class of Sequence Generators

Since CA-based linear models describing the behavior of CCGSs have been derived, a cryptanalytic attack that exploits the weaknesses of these models has been also developed. It consists in determining the initial states of both registers \( SR_1 \) and \( SR_2 \) from an amount of CCGS output sequence (the intercepted sequence). In this way, the rest of the output sequence can be reconstructed. For the sake of simplicity, the attack will be illustrated for the shrinking generator although the process can be extended to any CCGS. The proposed attack is divided into two different phases:

**Phase 1** From bits of the intercepted sequence and using the CA-based linear models, additional bits of the shrunken sequence can be reconstructed.

**Phase 2** Due to the intrinsic characteristics of the shrinking generator, a cryptanalytic attack can be mounted in order to determine the initial states of the LFSRs. The attack makes use of both intercepted bits as well as reconstructed bits.

Both phases will be considered separately.

### 5.1 Reconstruction of output sequence bits

Given \( r \) bits of the shrunken sequence \( z_0, z_1, z_2, ..., z_{r-1} \), we can assume without loss of generality that this sub-sequence has been generated at the most left extreme cell of any of its corresponding CA. That is \( x_1^i = z_0, \ x_1^{i+1} = z_1, ..., x_1^{r-1} = z_{r-1} \). From \( r \) bits of the shrunken sequence, it is always possible to reconstruct \( r - 1 \) sub-sequences \( \{x_i^j\} \) of lengths \( r - i + 1 \) at the \( i \)-th cell of each automaton such as follows:

\[
x_i^j = \Phi_{i-1}(x_{i-2}^j, x_{i-1}^j, x_{i-1}^{j+1}) \quad (1 < i \leq r),
\]

where \( \Phi_{i-1} \) corresponds to either rule 90 or 150 depending on the value of \( R_{i-1} \).

From \( r \) intercepted bits, the application of equation (10) gives rise to a total of \( (r + (r - 1) + \ldots + 2 + 1) \) bits that constitute the first chained sub-triangle notated
Δ1, see Table 2. Now, if any sub-sequence \{x^t_i\} is placed at the most left extreme cell, then \(r - 2i + 2\) bits are obtained at the \(i\)-th cell in the second chained sub-triangle notated \(Δ2\). Repeating recursively \(n\) times the same procedure, \(r - ni + n\) bits are obtained at the \(i\)-th cell in the \(n\)-th chained sub-triangle notated \(Δn\).

Table 2 shows the succession of 4 chained sub-triangles constructed from \(r = 10\) bits of the shrunken sequence \(\{z_i\} = \{0, 0, 1, 1, 1, 0, 1, 0, 1, 1\}\) and first rules \(R_1 = R_2 = 0\). In fact, the 10 initial bits generate 8 bits at the third cell in \(Δ1\). These 8 bits are placed at the most left extreme cell producing 6 new bits at cell 3 in \(Δ2\). With these 6 bits, we get 4 additional bits in \(Δ3\). Finally, 2 new bits are obtained at cell 3 in the sub-triangle \(Δ4\). Since rules 90 and 150 are additive, the generated sub-sequences will be sum of elements of the shrunken sequence. General expressions can be deduced for the elements of any sub-sequence in any chained sub-triangle. In fact, the \(i\)-th sub-sequence in the \(n\)-th chained sub-triangle includes the bits \(z_j\) corresponding to the exponents of \((P_{i-1}(x))^n\) where \(P_{i-1}(x)\) is the characteristic polynomial of the sub-automaton \(R_1 R_2 ... R_{i-1}\), see equation (2). More precisely, for the previous example the characteristic polynomial of the sub-automaton \(R_1 R_2\) is \(P_2(x) = x^2 + 1\). Then \((P_2(x))^2 = x^4 + 1, (P_2(x))^3 = x^8 + x^4 + x^2 + 1, (P_2(x))^4 = x^8 + 1, \ldots\) Hence, \(x^t_3\) in the different sub-triangles will take the form:

\[
\begin{align*}
  x^t_3 &= z_0 + z_2 & \text{in } Δ1 \\
  x^t_3 &= z_0 + z_4 & \text{in } Δ2 \\
  x^t_3 &= z_0 + z_2 + z_4 + z_6 & \text{in } Δ3 \\
  x^t_3 &= z_0 + z_8 & \text{in } Δ4 \\
\end{align*}
\]

For the successive bits \(x^{t+1}_3, x^{t+2}_3, \ldots\) it suffices to add 1 to the previous subindexes. Table 3 shows the general expressions of the sub-sequence elements in \(Δ1\) and \(Δ2\) for the example under consideration.

On the other hand, Lemmas (3.1) and (3.2) show us that the shrunken sequence is the interleaving of \(2^{\left(L_1 - 1\right)}\) different shifts of an unique PN-sequence of length \(2^{L_2} - 1\) whose characteristic polynomial \(P(x)\) is given by equation (7). Consequently, the elements of the shrunken sequence indexed \(z_{di}\), with \(i \in \{0, 1, \ldots, 2^{L_2} - 2\}\) and \(d = 2^{\left(L_1 - 1\right)}\), belong to the same PN-sequence. Thus, if the element \(x^t_i\) of the \(i\)-th sub-sequence in the \(n\)-th chained sub-triangle takes the general form:

\[
x^t_i = z_{k_1} + z_{k_2} + \ldots + z_{k_j}, \quad (11)
\]

with

\[
k_l \equiv 0 \mod 2^{\left(L_1 - 1\right)} \quad (l = 1, \ldots, j), \quad (12)
\]

then \(x^t_i\) can be rewritten as

\[
x^t_i = z_{k_m}, \quad (13)
\]

with \(z_{k_m}\) satisfying equation (12). Therefore, \(\{x^t_i\}\), the \(i\)-th sub-sequence in the \(n\)-th chained sub-triangle, is just a sub-sequence of the shrunken sequence shifted a distance \(\delta\) from the \(r\) bits of the intercepted sequence. The value of \(\delta\) depends on the extension field \(GF(2^{L_2})\) generated by the roots of \(P(x)\). In
Table 2: Reconstruction of 4 chained sub-triangles from 10 bits of the shrunken sequence

| Δ1 | R₁ | R₂ | R₃ | ... | Δ2 | R₁ | R₂ | R₃ | ...
|----|----|----|----|-----|----|----|----|----|-----|
| 0  | 0  | 1  | ... |     | 1  | 1  | 1  | ... |     |
| 0  | 1  | 1  |     |     | 1  | 0  | 0  |     |     |
| 1  | 1  | 0  |     |     | 0  | 1  | 0  |     |     |
| 1  | 1  | 1  |     |     | 1  | 0  | 1  |     |     |
| 1  | 0  | 0  |     |     | 0  | 0  | 0  |     |     |
| 0  | 1  | 0  |     |     | 0  | 0  | 1  |     |     |
| 1  | 0  | 0  |     |     | 0  | 1  |     |     | 1    |
| 0  | 1  | 1  |     |     | 1  |     |     |     |     |
| 1  | 1  |     |     |     | 1  |     |     |     |     |

| Δ3 | R₁ | R₂ | R₃ | ... | Δ4 | R₁ | R₂ | R₃ | ...
|----|----|----|----|-----|----|----|----|----|-----|
| 1  | 0  | 1  | ... |     | 1  | 1  | 1  | ... |     |
| 0  | 0  | 1  |     |     | 1  | 0  | 1  |     |     |
| 0  | 1  | 0  |     |     | 0  | 0  |     |     |     |
| 1  | 0  | 0  |     |     | 0  |     |     |     |     |
| 0  | 1  |     |     |     | 1  |     |     |     |     |

Table 3: General expressions for different sub-sequences in Δ1 and Δ2 with $R₁ = R₂ = 0$

| Δ1 | R₁ | R₂ | R₃ | ... | Δ2 | R₁ | R₂ | R₃ | ...
|----|----|----|----|-----|----|----|----|----|-----|
| $z₀$ | $z₁$ | $z₀ + z₂$ | ... |     | $z₀ + z₂$ | $z₁ + z₃$ | $z₀ + z₄$ | ... |
| $z₁$ | $z₂$ | $z₁ + z₃$ |     |     | $z₁ + z₃$ | $z₂ + z₄$ | $z₁ + z₅$ |     |
| $z₂$ | $z₃$ | $z₂ + z₄$ |     |     | $z₂ + z₄$ | $z₃ + z₅$ | $z₂ + z₆$ |     |
| $z₃$ | $z₄$ | $z₃ + z₅$ |     |     | $z₃ + z₅$ | $z₄ + z₆$ | $z₃ + z₇$ |     |
| $z₄$ | $z₅$ | $z₄ + z₆$ |     |     | $z₄ + z₆$ | $z₅ + z₇$ | $z₄ + z₈$ |     |
| $z₅$ | $z₆$ | $z₅ + z₇$ |     |     | $z₅ + z₇$ | $z₆ + z₈$ | $z₅ + z₉$ |     |
| $z₆$ | $z₇$ | $z₆ + z₈$ |     |     | $z₆ + z₈$ | $z₇ + z₉$ |     |     |
| $z₇$ | $z₈$ | $z₇ + z₉$ |     |     | $z₇ + z₉$ |     |     |     |     |
| $z₈$ | $z₉$ |     |     |     |     |     |     |     |     |
| $z₉$ |     |     |     |     |     |     |     |     |     |
brief, the chained sub-triangles enable us to reconstruct additional bits of the shrunken sequence from bits of the intercepted sequence.

The number of reconstructed bits depends on the amount of intercepted bits. Indeed, if we know \( N_l \) bits in each one of the \( PN \)-sequence shifts, then the total number of reconstructed bits is given by:

\[
2^{(L_1 - 1)} \sum_{l=1}^{N_l} \sum_{k=2}^{N_l} \binom{N_l}{k}
\]

The required amount of intercepted sequence is \( 2^{L_1 - 1} \) that is exponential in the length of the shortest register \( SR_1 \). Remark that in this reconstruction process both reconstructed bits as well as their positions on the shrunken sequence are known with absolute certainty.

### 5.2 Reconstruction of LFSR Initial States

We denote by \( IS_1 = (a_0, a_1, a_2, \ldots, a_{L_1-1}) \) the initial state of \( SR_1 \) and by \( IS_2 = (b_0, b_1, b_2, \ldots, b_{L_2-1}) \) the initial state of \( SR_2 \). In order to avoid ambiguities on the initial states, it is assumed that \( a_0 = 1 \), thus the first element of the shrunken sequence is \( z_0 = b_0 \). In this way, the goal of this attack is to determine the sub-vectors \( (a_1, a_2, \ldots, a_{L_1-1}) \) as well as \( (b_1, b_2, \ldots, b_{L_2-1}) \).

According to equation (13), the period of the shrunken sequence is \( T = (2^{L_2 - 1}) 2^{(L_1 - 1)} \), so that such a sequence can be written as an \( (2^{L_2 - 1}) \times (2^{(L_1 - 1)}) \) matrix whose elements are the bits of the shrunken sequence. Its columns are denoted by \( C_1, C_2, \ldots, C_{2^{(L_1 - 1)}} \), respectively. Each column of the matrix is the \( PN \)-sequence above referenced starting at different points. In addition, the first column \( C_1 \) corresponds to the decimation of the sequence \( \{b_i\} \) from \( SR_2 \) by a factor \( (2^{L_1 - 1}) \). Thus, we can compute the position of the bits \( b_1, b_2, \ldots, b_{L_2-1} \) on such a column. Indeed, the \( i \)-th bit, \( b_i \), is at the \( j_i = \frac{i}{2^{L_2 - 1}} \mod 2^{L_2 - 1} \) position of \( C_1 \) where \( j_i \) is solution of the equation:

\[
j_i (2^{L_1 - 1}) \equiv i \mod 2^{L_2 - 1} \quad (i = 1, \ldots, L_2 - 1).
\]

Moreover, the bits of \( IS_1 \) determine the initial bits of the subsequent columns \( C_i \) such as follows:

**Hypothesis 1** If the first bits of \( IS_1 \) are \( (a_0 = 1, a_1 = 1) \), then \( C_2 \) will start at the \( j_1 \)-th position of \( C_1 \) given by equation (15).

**Hypothesis 2** If the first bits of \( IS_1 \) are \( (a_0 = 1, a_1 = 0, a_2 = 1) \), then \( C_2 \) will start at the \( j_2 \)-th position of \( C_1 \) given by equation (15).

**Hypothesis n** If the first bits of \( IS_1 \) are \( (a_0 = 1, a_1 = 0, \ldots, a_{n-1} = 0, a_n = 1) \), then \( C_2 \) will start at the \( j_n \)-th position of \( C_1 \) given by equation (15).
We can formulate different hypothesis covering the first bits of \( IS_1 \) as well as each new hypothesis determines the initial bit of the following column. As we have intercepted and reconstructed bits in the columns \( C_i \), we can check the previous hypothesis until getting a contradiction. In that case, all the \( IS_1 \) starting with the wrong configuration must be rejected. The search continues through the configurations of \( a_i \) free of contradiction by formulating new hypothesis. In brief, the attacker has not to traverse an entire search tree including all the initial states of \( SR_1 \), but the search is concentrated exclusively on the configurations not exhibiting contradiction with regard to the available bits. In this sense, the proposed attack reduces considerably the exhaustive search over the initial states of \( SR_1 \) as many contradictions occur at the first levels of the tree. On the other hand, the bits of the register \( SR_2 \) are easily determined as the starting bits of \( C_2, C_3, C_4, \ldots \) in each one of the non-rejected branches. An illustrative example of Phases 1 and 2 is presented in the next subsection.

### 5.3 An Illustrative Example

Let us consider a shrinking generator with the following parameters: \( L_1 = 4 \), \( L_2 = 5 \), \( C_1(x) = 1 + x^3 + x^4 \) and \( C_2(x) = 1 + x + x^3 + x^4 + x^5 \). According to equation (7), we can compute the polynomial \( P(x) = 1 + x + x^2 + x^4 + x^5 \) while the two basic automata \( 1 \ 0 \ 0 \ 0 \ 0 \) and \( 0 \ 0 \ 0 \ 0 \ 1 \) are obtained from the algorithm of Cattell and Muzio. The corresponding \( CA \) of length \( L = 40 \) are computed via the algorithm developed in section 3. Indeed, they are \( CA_1 = 0060110600 \) and \( CA_2 = 8C0300C031 \) in hexadecimal notation. In addition, let \( \alpha \) be a root of \( P(x) \) that is \( \alpha^5 = \alpha^4 + \alpha^2 + \alpha + 1 \) as well as a generator element of the extension field \( GF(2^{L_2}) \). The period of the shrunken sequence is \( T = (2^{L_2} - 1) \cdot 2^{(L_1-1)} = 248 \) and the number of interleaved \( PN \)-sequences is \( 2^{(L_1-1)} = 8 \). Finally, the intercepted sequence of length \( r = 24 \) is: \( \{ z_0, z_1, \ldots, z_{23} \} = \{ 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1 \} \). With the previous premises, we accomplish Phases 1 and 2.

**Phase 1:**

For \( CA_1 \) The chained sub-triangles provide the following reconstructed bits.

For \( i = 3 \), sub-automaton \( R_1 R_2 \) and \( P_2(x) = x^2 + 1 \).

- In \( \Delta 4 \), \( x_3^4 = z_0 + z_8, \ x_3^{i+1} = z_1 + z_9, \ldots, x_3^{i+15} = z_{15} + z_{23} \). Considering \( z_0, z_8 \) as the first and second element of the \( PN \)-sequence and keeping in mind that in \( GF(2^{L_2}) \) the equality \( 1 + \alpha = \alpha^{19} \) holds, we get \( x_3^4 = z_{19} + z_8, \ x_3^{i+1} = z_{15} + z_9, \ldots, x_3^{i+15} = z_{167} \). Thus, 16 new bits of the shrunken sequence have been reconstructed at positions \( 152, 153, \ldots, 167 \).

- In \( \Delta 8 \), \( x_3^4 = z_0 + z_{16}, \ x_3^{i+1} = z_1 + z_{17}, \ldots, x_3^{i+7} = z_7 + z_{23} \). As \( 1 + \alpha^2 = \alpha^7 \), we get \( x_3^4 = z_{27} + z_8, \ x_3^{i+1} = z_{57} + z_9, \ldots, x_3^{i+7} = z_{63} \). Thus, 8 new bits of the shrunken sequence have been reconstructed at positions \( 56, 57, \ldots, 63 \).
For CA2 The chained sub-triangles provide the following reconstructed bits. For \( i = 3 \), sub-automaton \( R_1 R_2 \) and \( P_2(x) = x^2 + x + 1 \).

- In \( \Delta 8 \), \( x^t_3 = z_0 + z_8 + z_{16} \), \( x^{t+1}_3 = z_1 + z_9 + z_{17} \), \( \ldots \), \( x^{t+7}_3 = z_7 + z_{15} + z_{23} \). As \( 1 + \alpha + \alpha^2 = \alpha^{23} \), we get \( x^t_3 = z_{23} = z_{184}, \ x^{t+1}_3 = z_{185}, \ldots, x^{t+7}_3 = z_{191} \). Thus, 8 new bits of the shrunken sequence have been reconstructed at positions 184, 185, \ldots, 191.

After Phase 1, the known bits of the shrunken sequence are depicted in Table 4. Rows 0, 1, 2 correspond to intercepted bits while rows 7, 19, 20 and 23 correspond to reconstructed bits. The symbol \(-\) represents the unknown bits. In brief, from 24 intercepted bits a total of 32 bits have been reconstructed.

Table 4: The shrunken sequence produced by the shrinking generator described in subsection 5.3.

| \( C_1 \) | \( C_2 \) | \( C_3 \) | \( C_4 \) | \( C_5 \) | \( C_6 \) | \( C_7 \) | \( C_8 \) |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 2 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 3 | - | - | - | - | - | - | - |
| 4 | - | - | - | - | - | - | - |
| 5 | - | - | - | - | - | - | - |
| 6 | - | - | - | - | - | - | - |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 8 | - | - | - | - | - | - | - |
| 9 | - | - | - | - | - | - | - |
| 10 | - | - | - | - | - | - | - |
| 11 | - | - | - | - | - | - | - |
| 12 | - | - | - | - | - | - | - |
| 13 | - | - | - | - | - | - | - |
| 14 | - | - | - | - | - | - | - |
| 15 | - | - | - | - | - | - | - |
| 16 | - | - | - | - | - | - | - |
| 17 | - | - | - | - | - | - | - |
| 18 | - | - | - | - | - | - | - |
| 19 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 20 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 21 | - | - | - | - | - | - | - |
| 22 | - | - | - | - | - | - | - |
| 23 | - | - | - | - | - | - | - |
| 24 | - | - | - | - | - | - | - |
| 25 | - | - | - | - | - | - | - |
| 26 | - | - | - | - | - | - | - |
| 27 | - | - | - | - | - | - | - |
| 28 | - | - | - | - | - | - | - |
| 29 | - | - | - | - | - | - | - |
| 30 | - | - | - | - | - | - | - |

18
Phase 2: According to equation (15), the bits $b_1, b_2, b_3, b_4$ are placed at positions 29, 27, 25, 23 of column $C_1$, respectively (see the first column of Table 4). On the other hand, Table 5 shows the sequences corresponding to the following hypothesis.

**Hypothesis 1** If the first bits of $IS_1$ are $(a_0 = 1, a_1 = 1)$, then $C_2$ will start at the $29^{th}$ position of $C_1$ given rise to the column $H_1$. In row 2, $H_1$ and $C_2$ have a common bit without contradiction. The union of both sequences allows us to construct $C^2_2$ the second column of the matrix for this hypothesis. A total of 13 bits are then known in $C^2_2$.

**Hypothesis 2** If the first bits of $IS_1$ are $(a_0 = 1, a_1 = 0, a_2 = 1)$, then $C_2$ will start at the $27^{th}$ position of $C_1$ given rise to the column $H_2$. In row 23, $H_2$ and $C_2$ have a common bit with contradiction (starred bits). Thus, the initial states of $SR_1$ starting with bits 101 must be rejected.

**Hypothesis 3** If the first bits of $IS_1$ are $(a_0 = 1, a_1 = 0, a_2 = 0, a_3 = 1)$, then $C_2$ will start at the $25^{th}$ position of $C_1$ given rise to the column $H_3$. In row 7, $H_3$ and $C_2$ have a common bit without contradiction. The union of both sequences allows us to construct $C^3_2$ the second column of the matrix for this hypothesis. A total of 13 bits are then known in $C^3_2$.

**Hypothesis 4** If the first bits of $IS_1$ are $(a_0 = 1, a_1 = 0, a_2 = 0, a_3 = 0, a_4 = 1)$, then $C_2$ will start at the $23^{th}$ position of $C_1$ given rise to the column $H_4$. In row 0, $H_2$ and $C_2$ have a common bit with contradiction (starred bits). Thus, the initial state of $SR_1$ 1000 must be rejected.

On the hypothesis free of contradiction, we can formulate other ones depicted in Table 6.

**Hypothesis 5** If the first bits of $IS_1$ are $(a_0 = 1, a_1 = 1, a_2 = 1)$, then $C_3$ will start at the $27^{th}$ position of $C_1$ given rise to the column $H_5$. In row 23, $H_5$ and $C_3$ have a common bit with contradiction (starred bits). Thus, the initial states of $SR_1$ starting with bits 111 must be rejected.

**Hypothesis 6** If the first bits of $IS_1$ are $(a_0 = 1, a_1 = 1, a_2 = 0, a_3 = 0, a_4 = 1)$, then $C_3$ will start at the $23^{th}$ position of $C_1$ given rise to the column $H_6$. Bits 24 and 25 of $C_1$ have been deduced from $C^1_2$ in Hypothesis 1. In row 2, $H_6$ and $C_6$ have a common bit with contradiction (starred bits). Thus, the initial state of $SR_1$ 1100 must be rejected.

From Hypothesis 5 and 6, Hypothesis 1 must be rejected. Remark that the configuration $(a_0 = 1, a_1 = 0, a_2 = 0, a_3 = 1)$ in Hypothesis 3 is the only one free of contradiction. Thus, it corresponds to the actual initial state of $SR_1$. The successive bits of $SR_1$, that is the PN-sequence $\{1, 0, 0, 1, 0, 0, 0, 1, \ldots\}$, are checked by the successive columns $C_4, C_5, \ldots, C_8$ of the shrunken sequence. Concerning the initial state of $SR_2$, in Table 6 (column Solution) we can see that bits $b_4, b_3, b_2$ can be obtained from the known bits of $C_1$ in rows 23, 25 and
Table 5: Different hypothesis formulated on the bits of $SR_1$

|   | $C_1$ | $H_1$ | $C_2$ | $C_2^1$ | $C_1$ | $H_2$ | $C_2$ | $C_2^1$ | $C_1$ | $H_3$ | $C_2$ | $C_2^1$ | $C_1$ | $H_4$ | $C_2$ |
|---|------|------|------|--------|------|------|------|--------|------|------|------|--------|------|------|------|
| 0 | 1    | 0    | 1    | 0      | 1    | 1    | 0    | 0      | 1    | 1    | 1    | 1      | 0    | 1    | 0    |
| 1 | 1    | 0    | 0    | 0      | 1    | 1    | 1    | 1      | 1    | 1    | 1    | 1      | 1    | 1    | 1    |
| 2 | -1   | -1   | -1   | -1     | -1   | -1   | -1   | -1     | -1   | -1   | -1   | -1     | -1   | -1   | -1   |
| 3 | 1    | 1    | 1    | 1      | 1    | 1    | 1    | 1      | 1    | 1    | 1    | 1      | 1    | 1    | 1    |
| 4 | -1   | -1   | -1   | -1     | -1   | -1   | -1   | -1     | -1   | -1   | -1   | -1     | -1   | -1   | -1   |
| 5 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 6 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 7 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 8 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 9 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 10 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 11 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 12 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 13 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 14 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 15 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 16 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 17 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 18 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 19 | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    |
| 20 | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    |
| 21 | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    |
| 22 | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    | 0      | 0    | 0    | 0    |
| 23 | 1    | 1    | 1    | 1      | 1    | 1    | 1    | 1      | 1    | 1    | 1    | 1      | 1    | 1    | 1    |
| 24 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 25 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 26 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 27 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 28 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 29 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |
| 30 | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    | -      | -    | -    | -    |

Hypothesis 1 | Hypothesis 2 | Hypothesis 3 | Hypothesis 4

27 respectively. In fact, $b_4 = 1, b_3 = 0, b_2 = 1$. The bit $b_1$ in row 29 satisfies the equality

$$b_1 = z_{29.8} = z_{1.8} + z_{2.8} + z_{4.8},$$

(16)
as $\alpha + \alpha^2 + \alpha^4 = \alpha^{29}$ in the extension field $GF(2^{L_2})$. We know that $z_8 = 1, z_{16} = 1$ while $z_{32}$ can be easily deduced from the equality $z_{14.8} = z_{1.8} + z_{4.8}$ as $1 + \alpha^4 = \alpha^{14}$. Thus, $z_{32} = 1 + 1 = 0$ and substituting in $b_1$ we get $b_1 = 1 + 1 + 0 = 0$.

The final issues of Phases 1 and 2 are the initial states of both LFSRs $IS_1 = (a_0, a_1, \ldots, a_3) = (1, 0, 0, 1)$ and $IS_2 = (b_0, b_1, \ldots, b_4) = (1, 0, 1, 0, 1)$. From the knowledge of both initial states the whole shrunken sequence can be reconstructed.
Table 6: Different hypothesis formulated on the bits of $SR_1$

|   | $C_1$ | $H_5$ | $C_3$ | $C_1$ | $H_6$ | $C_3$ | $C_1$ | $C_2$ |
|---|-------|-------|-------|-------|-------|-------|-------|-------|
| 0 | 1     | -     | 1     | 1     | 1     | 1     | 1     | 0     |
| 1 | 1     | -     | 0     | 1     | 0     | 0     | 1     | 0     |
| 2 | 1     | -     | 0     | 1     | 1*    | 0*    | 1     | 1     |
| 3 | -     | -     | -     | -     | -     | -     | -     | -     |
| 4 | -     | 1     | -     | -     | 0     | -     | -     | -     |
| 5 | -     | 1     | -     | -     | 0     | -     | -     | -     |
| 6 | -     | 1     | -     | -     | 0     | -     | -     | 1     |
| 7 | 0     | -     | 1     | 0     | -     | 1     | 0     | 1     |
| 8 | -     | -     | -     | -     | 1     | -     | -     | 1     |
| 9 | -     | -     | -     | -     | 1     | -     | -     | -     |
| 10| -     | -     | -     | -     | 1     | -     | -     | -     |
| 11| -     | 0     | -     | -     | -     | -     | -     | -     |
| 12| -     | -     | -     | -     | -     | -     | -     | -     |
| 13| -     | -     | -     | -     | -     | 0     | 0     | -     |
| 14| -     | -     | -     | -     | -     | 1     | -     | -     |
| 15| -     | -     | -     | -     | 0     | -     | -     | -     |
| 16| -     | -     | -     | -     | -     | -     | -     | -     |
| 17| -     | -     | -     | -     | -     | 1     | -     | -     |
| 18| -     | -     | -     | -     | -     | -     | -     | -     |
| 19| 0     | -     | 1     | 0     | -     | 1     | 0     | 0     |
| 20| 0     | -     | 0     | 0     | -     | 0     | 0     | 1     |
| 21| -     | -     | -     | -     | -     | -     | -     | -     |
| 22| -     | -     | -     | -     | -     | -     | -     | -     |
| 23| 1     | 0*    | 1*    | 1     | -     | 1     | 1     | 1     |
| 24| -     | 0     | 0     | -     | -     | -     | -     | -     |
| 25| -     | -     | -     | 1     | -     | -     | 0     | 0     |
| 26| -     | -     | -     | -     | -     | 0     | 0     | 0     |
| 27| -     | 1     | -     | -     | 0     | -     | 1     | -     |
| 28| -     | -     | -     | -     | 0     | -     | -     | -     |
| 29| -     | -     | -     | -     | -     | 1     | 0     | 0     |
| 30| -     | -     | -     | -     | -     | -     | -     | -     |

|   | Hypothesis 5 | Hypothesis 6 | Solution |
|---|--------------|--------------|----------|

5.4 Computational Features

The computational complexity of the previous cryptanalytic attack can be considered in two different phases: off-line and on-line complexity.

**Off-line computational complexity:** This phase is to be executed before intercepting sequence. It includes:

- Computation of the characteristic polynomials $P_i(x)$ of the sub-automata $R_1 R_2 \ldots R_l$ ($1 < i \leq l$) by means of equation (2) where $l$ is related to the amount of intercepted sequence ($l \cdot 2^{L_1-1} \sim r$). This computation is necessary in order to obtain general expressions for the elements of the chained sub-triangles in the reconstruction procedure.

- Computation of the positions of the bits $b_i$ ($i = 1, 2, \ldots, L_2 - 1$) on $C_1$ the first column of the shrunken sequence matrix by means of equation (15). This computation is necessary in order to determine the bits of the initial state of $SR_2$. 

21
• Computation of different elements of the extension field $GF(2^{L_2})$ such as $1 + \alpha$, $1 + \alpha^2$, \ldots, $1 + \alpha^{N_l}$ and linear combinations of them by means of the Zech log table method \cite{1} for arithmetic over $GF(2^m)$. This computation is necessary in order to determine the distance between the intercepted sequence and the portions of reconstructed shrunken sequence.

On-line computational complexity: This phase is to be executed after intercepting sequence. According to the previous subsections, the computational method consists in the comparison of series of bits coming from formulated hypothesis and from intercepted/reconstructed bits. The comparison is realized by means of bit-wise logical operations so the computational complexity is rather low. Occasionally, the computation of the any element of $GF(2^{L_2})$ must be realized in order to determine additional elements of the $PN$-sequences. The most consuming time of this cryptanalytic attack is the search over the $2^{L_1-1}$ possible initial states of $SR_1$ (supposed $a_0 = 1$). Due to contradictions found in the first levels of the search tree, the exhaustive search can be dramatically improved. On average, we can say that in the worst case the search can be reduced to the half, so that the computational complexity of this attack is $O(2^{L_1-2})$. In addition, several considerations must be kept in mind:

1. The improved exhaustive search is carried out over the state space of the shortest register $SR_1$.

2. Every checking of hypothesis is realized only over the 1’s of the configuration under consideration, then the procedure speeds for configurations with a low number of 1’s.

Finally, comparing the proposed attack with those ones found in the literature we get that all of them are exponential in the lengths of the registers. In particular, the complexity of the divide-and-conquer attack proposed in \cite{23} is $O(2^{L_2})$. The probabilistic correlation attack described in \cite{13} has a computational complexity of $O(L_2^2 \cdot 2^{L_2})$. Also the probabilistic correlation attack introduced in \cite{14} is exponential in $L_2$. In this work a deterministic attack has been proposed that improves the complexity of the previous cryptanalytic approaches.

6 Conclusions

This paper considers the linearization of pseudorandom sequence generators based on finite fields. More precisely, a wide family of traditional LFSR-based sequence generators, the so-called Clock Controlled Shrinking Generators, has been analyzed and modelled in terms of linear cellular automata. In this way, sequence generators conceived and designed as complex nonlinear models can be written in terms of simple linear models. An easy algorithm to compute the pair of one-dimensional linear hybrid cellular automata that generate the CCSG output sequences has been derived. The key idea of this modelling is just the concatenation of a basic structure repeated a number of times. In addition, a
cryptanalytic attack that reconstructs the output sequence of such generators has been proposed too. The cryptanalytic approach is deterministic and improves an exhaustive search over the states of the shortest register. Computing the initial state of the longest register is a direct consequence of the previous step. The attack exploits the linearity of these CA-based models as well as the characteristics of this class of generators. Applying the same schemes, we can develop linear cellular automata-based models to analyze/cryptanalyze wider classes of clock-controlled generators.

References

[1] F. Assis and C. Pedreira, An Architecture for Computing Zech’s Logarithms in GF(2^m), IEEE Transactions on Computers 49 (5) (2000), 519-524.

[2] F. Bao, Crytanalysis of a New Cellular Automata Cryptosystem, 8th Australasian Conference on Information Security and Privacy- - ACISP 2003. Lecture Notes in Computer Science, Springer Verlag 2727 (2003), 416-427.

[3] S. Blackburn, S. Merphy and K. Paterson, Comments on 'Theory and Applications of Cellular Automata in Cryptography', IEEE Transactions on Computers 46 (1997), 637-638.

[4] K. Cattell and J. Muzio, Analysis of One- Dimensional Linear Hybrid Cellular Automata over GF(q), IEEE Transactions on Computers 45 (7) (1996), 782-792.

[5] K. Cattell and J. Muzio, Synthesis of One- Dimensional Linear Hybrid Cellular Automata, IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems 15 (3) (1996), 325-335.

[6] K. Cattell and Z. Shujian, Minimal Cost One- Dimensional Linear Hybrid Cellular Automata of Degree Through 500. J. of Electronic Testing: Theory and Applications 6 (1995), 255-258.

[7] K. Cattell and J. Muzio, A Linear Cellular Automata Algorithm: Theory, Dept. of Computer Science. University of Victoria, Canada, Tech. Rep. DCS-161-IR, 1991.

[8] D. Coppersmith, H. Krawczyk and Y. Mansour, The Shrinking Generator, Advances in Cryptology- - CRYPTO’93. Lecture Notes in Computer Science, Springer Verlag, 773 (1994), 22-39.

[9] S. Cho, C. Un- Sook and H. Yoon- Hee, Computing Phase Shifts of Maximum-Length 90/150 Cellular Automata Sequences, Proc. of ACRI 2004. Lecture Notes on Computer Science, Springer- Verlag, 3305 (2004), 31-39.
[10] A.K. Das, A. Ganguly, A. Dasgupta, S. Bhawmik and P.P. Chaudhuri, *Efficient Characterisation of Cellular Automata*, IEE Proc., Part E. 1 (1990), 81- 87.

[11] S. Golomb, *Shift- Register Sequences (revised edition)*, Aegean Press (1982).

[12] G. Gong, *Theory and Applications of q- ary interleaved sequences*, IEEE Transactions on Information Theory 41 (1995), 400- 411.

[13] J. Golic, L. O’Connors, *A Cryptanalysis of Clock-Controlled Shift Registers with Multiple Steps*, Cryptography: Policy and Algorithms 41 (1995), 174-185.

[14] T. Johansson, *Complexity Correlation Attacks on Two Clock-Controlled Generators*, Proc. of Asiacrypt’98. Lecture Notes in Computer Science, Springer Verlag 1426 (1998), 342- 356.

[15] A. Kanso, *Clock- Controlled Shrinking Generator of Feedback Shift Registers*, 8th Australasian Conference on Information Security and Privacy--ACISP 2003. Lecture Notes in Computer Science, Springer Verlag 2727 (2003), 443- 451.

[16] E.L. Key, *An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators*, IEEE Transactions on Information Theory 22 (6) (1976), 732- 736.

[17] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, Cambridge: Cambridge University Press, (1986).

[18] O. Martin, A.M. Odlyzko and S. Wolfram, *Algebraic Properties of Cellular Automata*, Commun. Math. Phys. 93 (1984), 219- 258.

[19] A.J. Menezes, P. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.

[20] S. Nandi, B.K. Kar and P.P. Chaudhuri, *Theory and Applications of Cellular Automata in Cryptography*, IEEE Transactions on Computers 43 (1994), 1346- 1357.

[21] R.A. Rueppel, *Stream Ciphers*, in Gustavus J. Simmons (Ed), Contemporary Cryptology, The Science of Information. IEEE Press (1992), 65- 134.

[22] M. Serra, T. Slater, J. Muzio and D.M. Miller *The Analysis of One- dimensional Linear Cellular Automata and Their Aliasing Properties*, IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems 9 (7) (1990), 767- 778.

[23] L. Simpson et al., *Clock- A probabilistic Correlation Attack on the Shrinking Generator*, Proc. of Australasian Conference on Information Security and Privacy--ACISP 1998. Lecture Notes in Computer Science, Springer Verlag 1438 (1998), 147- 158.
[24] H.S. Stone, *Discrete mathematical Structures and Their Applications*, Chicago, IL. Science Research (1973).

[25] X. Sun, E. Kontopidi, M. Serra and J. Muzio, *The Concatenation and Partitioning of Linear Finite State Machines*, Int. J. Electronics. **78** (1995), 809-839.

[26] S. Wolfram, *Random Sequence generation by Cellular Automata. Advances in Applied Mathematics*, **7** (123) (1986).

[27] S. Wolfram, *Cryptography with Cellular Automata*, Advances in Cryptology- - CRYPTO’85. Lecture Notes in Computer Science, Springer Verlag **218** (1994), 22-39.

[28] S. Zhang, *Quantitative Analysis for Linear Hybrid CA and LFSR as BIST Generators for Sequential Faults*, J. Electronic Testing, **7** (3) (1995), 209-221.