Cryptanalysis of a cascade of non-uniformly clocked linear feedback shift registers

Lelai Shi

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2010

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Cryptanalysis of a cascade of non-uniformly clocked linear feedback shift registers

Lelai Shi

2010/06/29

Abstract

In this thesis, we study ciphertext-only cryptanalysis of a cascade of pseudorandom sequence generators employing linear feedback shift registers (LFSRs) with so-called irregular clocking. The cascade of LFSRs is a well known pseudorandom generator scheme that produces sequences with good cryptographic characteristics (long period, high linear complexity, good statistical properties, etc.) A method of cryptanalysis of cascades containing two such LFSRs is well known. We generalize this method to cryptanalysis of a cascade with an arbitrary number of LFSRs. We reconstruct a set of candidate clock control sequences at each stage of the cascade, instead of enumerating all the possible initial states of the corresponding subcascade. The reconstruction is performed by means of an independent search through the edit distance matrix associated with every stage of the cascade. The experimental results show that such a generalized method of cryptanalysis is feasible.

Acknowledgements

I would like to thank my supervisor, Professor Slobodan Petrović, for his fascinating guidance, patient encouragement and careful feedback on my master thesis. I would also like to thank to my family for their support throughout the course.

Contents

List of Figures

1 Introduction

1.1 Topic covered by the thesis

In this thesis, we study ciphertext-only cryptanalysis of a cascade of pseudorandom sequence generators employing linear feedback shift registers (LFSRs) with so-called irregular clocking. The cascade structure represents a generalization of a primitive irregular clocking structure with 2 LFSRs, of which one generates clock pulses for the other (see figures 1 and 2). There are several ways of irregular clocking, which determine the type of the irregular clocking- based pseudorandom sequence generator. Examples of irregular clocking schemes include the stop- and- go generator [1], the Binary Rate Multiplier [2], the shrinking generator [3] and the alternating step generator [4]. It is well-known (see for example [5]) that the two most important cryptographic quality criteria for stream ciphers are the length of the period of the output sequence and the linear complexity of the output sequence. Long periods and high linear complexities with structures employing LFSRs can be achieved in several ways. We can use non-linear filters [6], non-linear combiners [7] or irregular clocking. Non-linear filters and non-linear combiners were shown to be vulnerable to various kinds of attacks such as algebraic attacks (see for example [8], [9] and correlation attacks (see for example [10]). Besides, with irregular clocking it is possible to achieve longer periods and higher linear complexities than with non-linear filters and combiners [11]. Because of that, it is of particular interest to investigate the difficulties of cryptanalysis of such schemes.

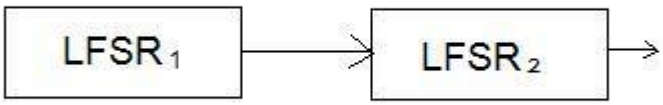

Figure 1. Elementary cascade of two LFSRs

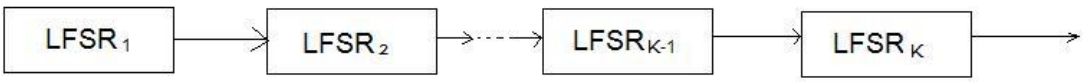

Figure 2. The cascade of irregularly clocked LFSRs

In cryptanalysis of irregularly clocked LFSRs in general and their cascades in particular, it is not possible to directly implement correlation attack methods as used by Siegenthaler [10]. The reason for this is the need to compare the output sequence of the generator with internal sequences that are in general longer than the output sequence. Because of that, the Hamming distance measure, used by Siegenthaler has to be replaced by a more convenient measure, with which we can compare sequences

of different lengths. One possibility is to use the constrained edit distance (or constrained Levenshtein distance) [12, 13]. Thus, it is possible to use the ideas from [10] in the cryptanalysis of the pseudorandom generator schemes that employ irregularly clocked LFSRs.

In this thesis, we generalize the ideas from [14] to cryptanalysis of cascades of irregularly clocked LFSRs.

1.2 Keywords

Cryptanalysis, Linear Feedback Shift Registers (LFSRs), Cascade, Irregular Clocking, Constrained Edit Distance.

1.3 Problem description

In the secret communication procedure (Figure 3), the plaintext is expected to be sent securely from the side A to the side B. To ensure security of transmission, A enciphers the plaintext using the method shared between A and B. The enciphered plaintext (ciphertext) is sent to B and deciphered by B. The task of cryptanalysis in this case is to obtain information about plaintext from the analysis of the eavesdropped ciphertext.

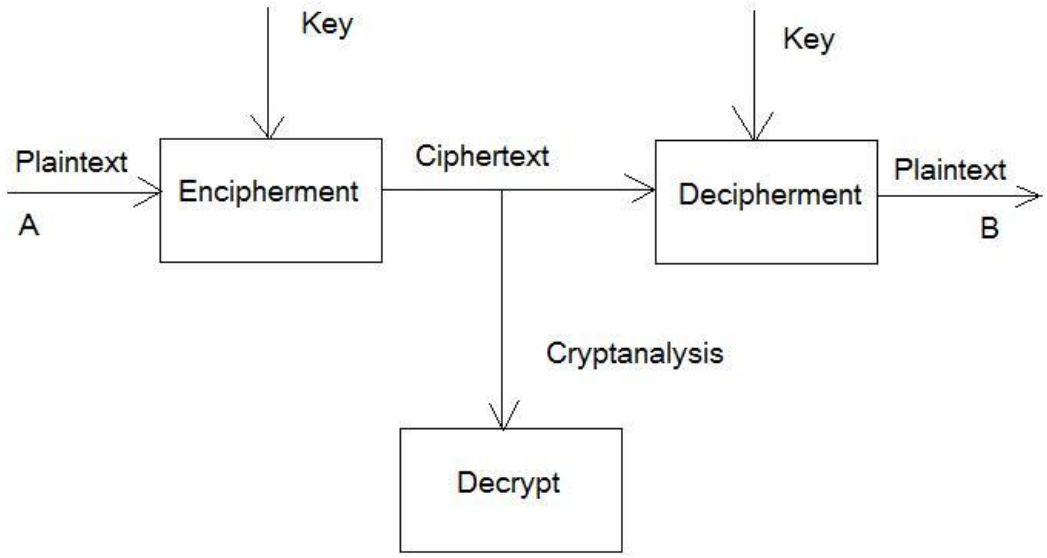

Figure 3. The secret communication procedure

In this thesis, we consider stream ciphers, a large class of so-called symmetric ciphers, i.e. the ciphers in which the same key is present at the transmitting and the receiving side before the beginning of the communication (Figure 4).

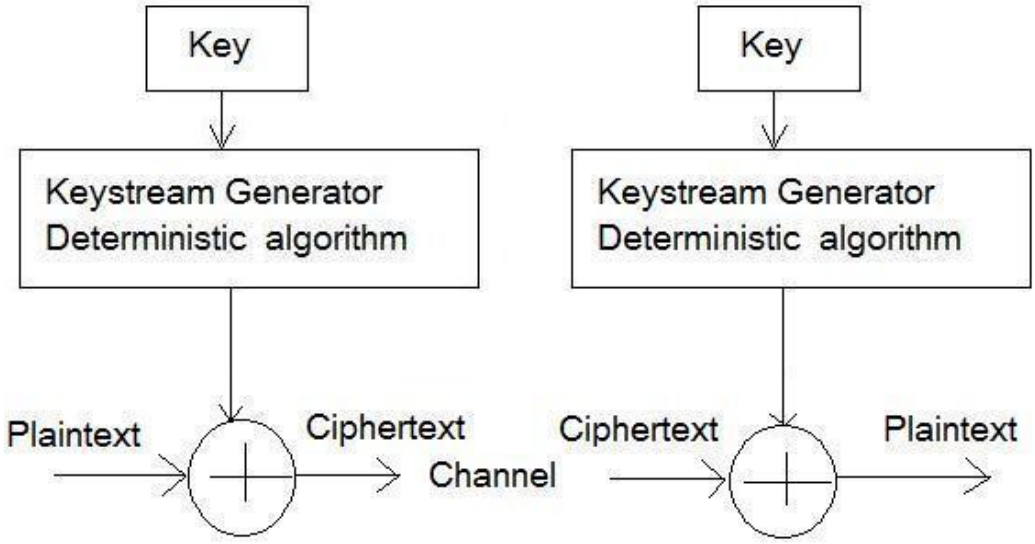

Figure 4. A stream cipher

We concentrate on cryptanalysis of cascades of irregularly clocked LFSRs (Fig. 2) realized through the Binary Rate Multiplier (BRM) [2]. We now formalize the task of cryptanalysis of such a scheme:

Given the prefix of the intercepted output sequence of length M, determine the initial states of all the LFSRs in the cascade.

The goal of the thesis is to determine how this task could be solved.

1.4 Justification, motivation and benefits

Stream ciphers represent the backbone of security of today's most important digital communications (such as Internet, mobile telephony, mobile ad hoc networks (MANETs) etc.) With irregularly clocked LFSRs in general and especially with their cascades, it is possible to achieve extremely long periods and large linear complexities of keystream sequences. Because of that, the most of today's high-grade stream ciphers are based on irregular clocking, in one form or another. Therefore, it is very important to study the conditions under which these systems can operate in a secure way.

By studying possibilities of cryptanalysis of such pseudorandom generator schemes, it is possible to determine the potential of securing modern digital communications with this type of devices and to reduce the exploitation of their vulnerabilities to a minimum.

1.5 Research questions

In this thesis, we are going to answer the following research questions:

1. Is it possible to generalize the correlation attack against a scheme with 2 LSFRs, of which one irregularly clocks another, [14] to a cascade of irregularly clocked LFSRs?
2. What phases such an attack would consist of?
3. What methods of cryptanalysis are applicable in each phase of the attack?

1.6 Methodology

We use "classical" research methodology in this thesis: literature study – in order to find out what was the success of cryptanalysis against similar pseudorandom generator schemes and what methods of cryptanalysis have been used against irregularly clocked LFSRs; logical reasoning - to develop generalized cryptanalytical method applicable in breaking cascades of irregularly clocked LFSRs; experimental analysis - to confirm our theoretical considerations by effectively cryptanalyzing a particular pseudorandom generator scheme.

1.7 Irregularly clocked LFSRs

We start the description of pseudorandom sequence generators employing irregularly clocked LFSRs by giving a short survey of linear feedback shift register theory.

1.7.1 Definition of a linear feedback shift register (LFSR)

A general feedback shift register is defined as a kind of logic unit in a digital system. Such a register stores data, calculates results, receives data, transmits data, and shifts digits. The model of a feedback shift register is shown in Figure 5.

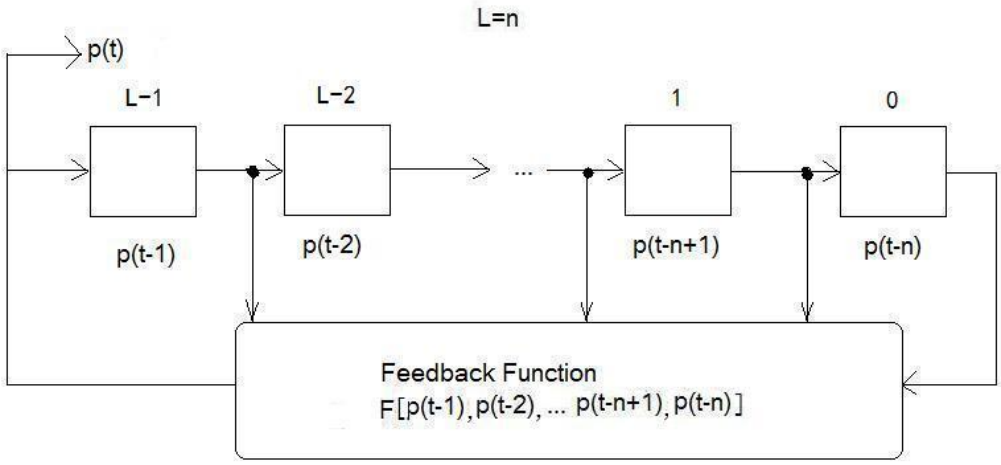

Figure 5. A general feedback shift register

A special case of feedback shift register is the one in which the feedback function is linear [5].

According to [5]:
A linear feedback shift register (LFSR) of length L consists of L stages (or delay elements) numbered $0, 1, \ldots, L-1$, each capable of storing one bit and having one input and one output; and a clock which controls the movement of data. During each unit of time the following operations are performed:

(i) the content of a stage (typically stage 0) is output and forms part of the output sequence;
(ii) the content of stage i is moved to stage $i-1$ for each $i, 1 \leq i \leq L-1$; and
(iii) the new content of stage $L-1$ is the feedback bit sj which is calculated by adding together modulo 2 the previous contents of a fixed subset of stages $0, 1, \ldots, L-1$.

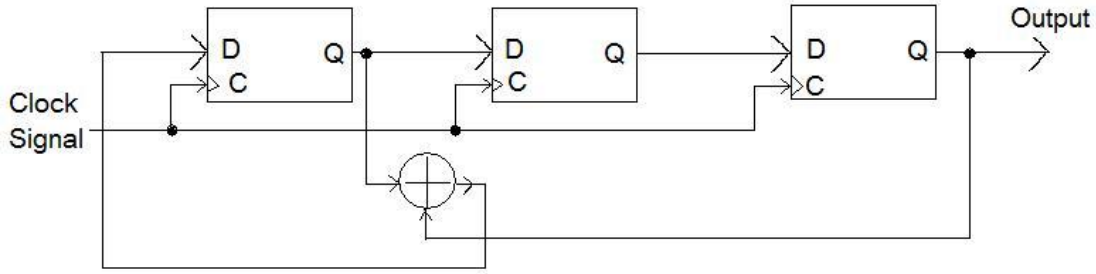

Figure 6. A realization of an LFSR with D flip-flops

A realization of an LFSR with D flip-flops is shown in Figure 6. In Figure 6, this linear feedback shift register consists of three D flip-flops. All these flip-flops (stages) are clocked by the clock signal. When a clock pulse triggers the linear feedback shift register, the digits of the flip-flops are shifted by one position to the right. In Figure 6, the LFSR has 3 stages, which means that the length $L = 3$. The feedback polynomial realized by this LFSR is $f(x) = 1 + x + x^3$, see below.

The state of an LFSR is defined as the content of each flip-flop (stage) between two consecutive clock pulses. The initial state of the LFSR is defined as the state of the register before the first digit of the output sequence is generated. If the initial state has contents all zeros, the output sequence will be all zeros. In practice, in order to avoid this situation, the initial state should not be set to the all-zero state. Besides the all-zero state, there are at most $2^n - 1$ different states.

An LFSR can be uniquely determined by its feedback function. Every new digit of the output sequence is expressed as a function of the n previous digits in n stages. The feedback function is a linear recurrence of order n:

$$F(t) = p_1 a(t-1) \oplus p_2\, a(t-2) \oplus \ldots p_n a(t-n) \qquad p_i \in \{0,1\} \qquad (1)$$

In order to analyze each register more easily, the linear recurrence of the register is often mapped to a formal polynomial, which is called feedback polynomial of the register. The feedback polynomial corresponding to the linear recurrence (1) is shown below:

$$f(x) = 1 + a_1x + a_2x^2 + \ldots + a_nx^n$$

Very often, the reciprocal of the feedback polynomial is also used, which is called characteristic polynomial: $g(x) = f^*(x) = f\left(\frac{1}{x}\right)$.

1.7.2 LFSRs used in cryptography

In order to obtain pseudorandom sequences from linear feedback shift registers, it is crucial for the designer to choose an appropriate feedback polynomial for the register. Not all the feedback polynomials are adequate for use in cryptography. We now introduce a kind of classification of linear feedback shift registers and discuss which linear feedback shift registers can be used in cryptography.

The feedback polynomial of a linear feedback shift register can be classified into three categories: reducible polynomials, irreducible polynomials and primitive polynomials. It is a good strategy to use LFSRs with primitive feedback polynomials in cryptography. The linear feedback shift register with a reducible polynomial has the property that the length of the period T of the output sequence depends on the initial state. The linear feedback shift register with irreducible feedback polynomial has the property that the length of the period T of the output sequence does not depend on the initial state, but is a factor of $2^L - 1$ where L is the length of the LFSR. Neither of these two types of LFSRs is adequate for use in cryptography because the statistical properties of the sequences they produce are bad. Instead, only LFSRs with primitive feedback polynomials satisfy all the Golomb's postulates that define desirable statistical properties of LFSRs (see for example [5]). Thus, they are adequate for use in cryptography.

Although in cryptography primitive polynomials are more suitable for use in LFSRs than the other types of feedback polynomials, not all the primitive polynomials are suitable. A particular kind of LFSR, which contains a primitive feedback polynomial with degree n such that $2^n - 1$ is so- called Mersenne prime (a prime of the form $2^n - 1$), can be used in LFSRs. Other LFSRs, which contain primitive feedback polynomials with degree n such that $2^n - 1$ has a small number of large prime factors, can also be taken into account to be used.

The key of the linear feedback shift register is usually the initial state, but the polynomial corresponding to the linear feedback shift register can also be kept secret.

1.7.3 Correlation attacks on non-linear combiners

Non-linear combining function with regularly clocked LFSRs was considered to be used in producing pseudo-noise sequence to avoid cryptanalytic attack with Berlekamp-Massey shift register synthesis algorithm [15] since such functions, if adequately selected, produce sequences with high linear complexity. In such a non-linear combiner, the inputs of the non-linear function are from regularly clocked LFSRs, and the output sequence of the function is used as the running key in a stream cipher. The running key is used to transform the plaintext to the ciphertext. The initial state of each LFSR can be regarded as a part of the initial key K of such a cryptographic system. The statistical model of such

a generator is presented in Fig. 7.

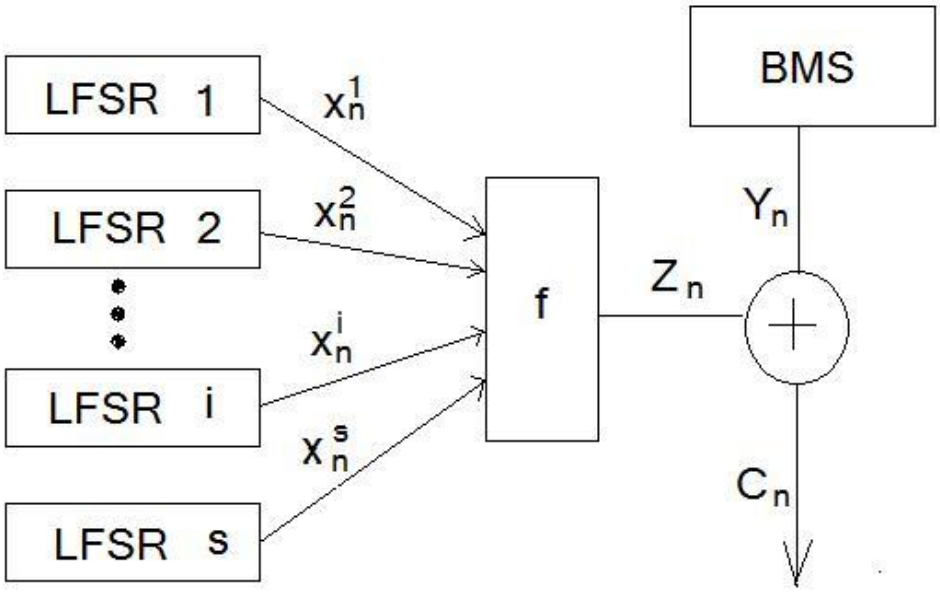

Figure 7. Statistical model of non-linear combiner (BMS – Binary Memoryless Source)

In theory, several attacks can be applied in order to cryptanalyze such a non-linear combiner system. One of them is so-called brute force attack. In the brute force attack, the combination of all the possible primitive feedback connections, all the initial states of each LFSR and all LFSRs should be tried. This attack is, of course, impractical for any reasonably large scheme, since no computational power can enumerate all these possibilities.

The number of trials can be reduced by correlation attack [10] if the non-linear function is such that there is correlation between the output sequence of the generator and a linear combination of the input sequences that does not involve all of them. Using this attack, a variable, which represents the measure of correlation between the ciphertext and each part of the input of the correlation function, is defined. Given N bits of ciphertext and a designated part of the input of the correlation function, the value of that variable is calculated. Finally, by comparing this obtained value of the variable and a threshold T for every possible primitive feedback connection combining every possible initial state of the corresponding LFSR, we determine whether the correct initial state and feedback connection is used.

The Geffe's generator is an example of Non-linear combiners (see Fig. 8). The Geffe's generator output function is balanced since its truth table has equal number of 0s and 1s. The algebraic normal form (ANF) of that Boolean function is:

$$F(x_1, x_2, x_3) = x_1x_2 \oplus (1 \oplus x_2)x_3 = x_3 \oplus x_1x_2 \oplus x_2x_3 \ .$$

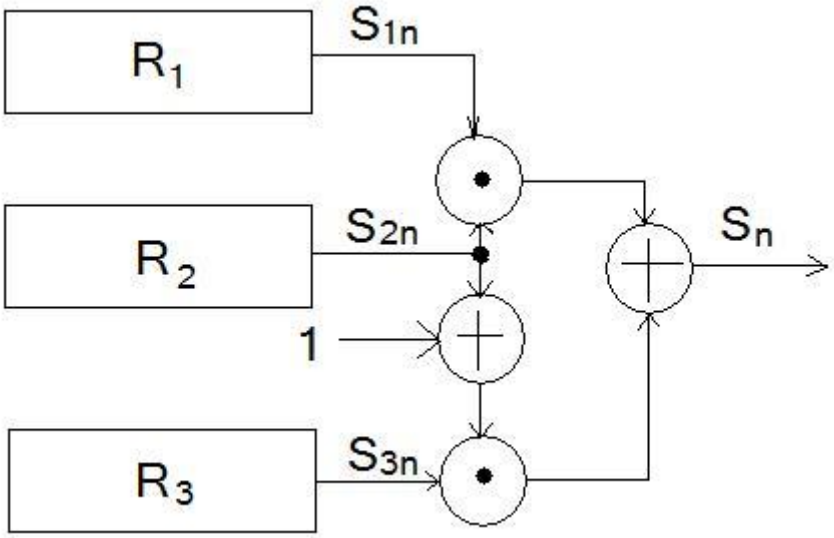

Figure 8. The Geffe's generator

The Geffe's geneator is vulnerable to correlation attack. The analysis is presented below:

$$\Pr(s_n = s_{1n} \mid s_{2n} = 1) = 1$$

$$\Pr(s_n = s_{1n} \mid s_{2n} = 0) \approx 1/2$$

So,

$$\Pr(s_n = s_{1n}) \approx 3/4$$

Using similar analytical method, we get:

$$\Pr(s_n = s_{2n}) \approx 3/4$$

The advantage of the correlation attack [10] over the brute force attack is in the fact that in the best case (the correlation of the output sequence with every single input sequence) the complexity (i.e. the number of initial states of the LFSRs to be guessed in order to find the correct one) of the correlation attack is $\sum_{i=1}^{n} 2^{L_i}$ and with the brute force attack the complexity is $\prod_{i=1}^{n} 2^{L_i}$. The goal of this thesis is to devise a correlation attack of the type [10] in the case when the LFSRs of the generator are irregularly clocked and are elements of a cascade. For this to be achieved, a special generalized measure of correlation of sequences of different lengths has to be defined.

1.7.4 Decimation of sequences and irregular clocking

The decimation of sequences happens when the output sequence of a subgenerator is fed into the clock control input of one or more other subgenerators. The minimal example of such a scheme is a scheme in which one LFSR clocks another.

In §1.1 we mentioned some types of irregular clocking: the stop-and-go clocking, shrinking and the Binary Rate Multiplier. By stop-and-go clocking, each LFSR will be clocked once if the clocking bit from another generator is one, which means to go. Otherwise, it will not be clocked (stop). By shrinking, when the controlled LFSR is clocked by zero, the generated bit is discarded. And when clocked by one, the generated bit will be sent to the output.

In our research, we concentrate on the Binary Rate Multiplier (BRM), a classical irregular clocking primitive used in many popular pseudorandom sequence generators (see, for example, [16]). Let $\{x_n\}$ be the binary sequence produced by a shift register R. Let $\{d_n\}$ be a sequence of integers, named decimation sequence. In the decimation process, the sequence $\{z_n\}$ is obtained in the following way (Figure 9):

$$z_n = x_{f(n)}, \quad f(n) = n + \sum_{i=0}^{n} d_i \, , n = 0,1,2, \ldots$$

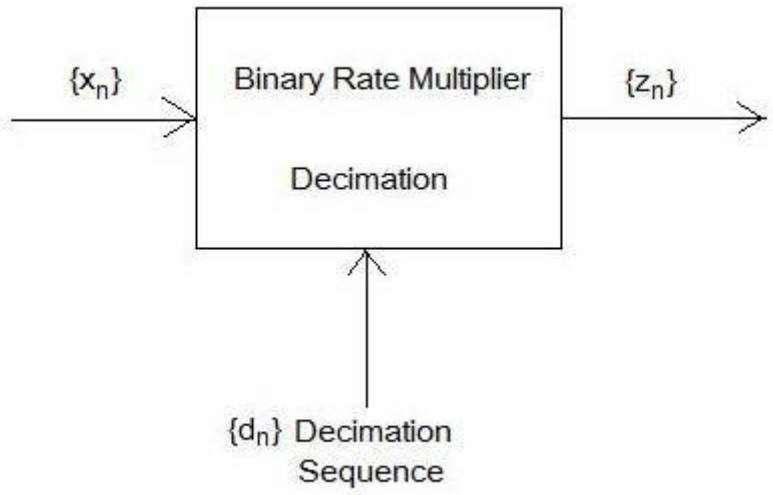

Figure 9. The Binary Rate Multiplier Decimation Model

Physical realization of decimation by means of BRM is the case in which one LFSR clocks another in the following way:

1. K positions of the clocking LFSR determine the integer d_n of the decimation sequence.
2. If $K = 0$, then the clocked LFSR is shifted only once and the bit produced by the clocked LFSR is sent to the output.
3. If $K > 0$, then the clocked LFSR is shifted $K + 1$ times, and the last produced bit is sent to the output.

A special case is when K takes only the value of 0 or 1 (so-called 0/1 clocking). In that case, only 1 bit of the clocking register is used to determine K. In this thesis, we limit ourselves only to this simplest case.

2 Literature survey

Several approaches to the problem of clock control sequence reconstruction in this kind of generators have been described so far. First, for every candidate initial state of the clocked LFSR, it is obvious that the "brute force method" can be used, i.e. all the possible initial states of the subgenerator that produce the corresponding clock control sequences can be enumerated and tested. In [17], obvious inefficiency of the brute force method is overcome by using a probabilistic coding theory approach for the reconstruction of the clock control sequence in the shrinking generator. In [18], the possibility of clock control sequence reconstruction by backtracking through the edit distance matrix is mentioned in the context of cryptanalysis of the alternating step generator. In [19], a MAP (Maximum Aposteriori Probability) decoding technique is used for reconstructing both candidate initial states of the clocked LFSR(s) and the clock control sequence. Molland [20] developed an attack against schemes based on irregular clocking, in which an improved linear consistency test (LCT, see for example [5]) was used.

The attacks mentioned above, except of the brute force attack, are so-called known-plaintext attacks. That means that it is supposed that the output sequence of the generator is known to the cryptanalyst. However, in any realistic setting, the sequence that the cryptanalyst can obtain is noise-degraded, i.e. plaintext should be taken into account, which in this context should be considered as noise. Thus, in the process of clock control sequence reconstruction, the influence of noise on the effectiveness of the procedure is decisive. Namely, the noise can either prevent a clock control sequence reconstruction procedure from functioning or significantly reduce its effectiveness.

The ciphertext-only attack on stream ciphers, as the most difficult one, is the most realistic attack scenario. Unlike the most of other attacks, the attack on a scheme with 2 LFSRs applying directed search through the constrained edit distance matrix [21] was developed with ciphertext-only scenario in mind. It was shown in [21] that it is possible for such an attack to be efficient even at relatively high levels of noise (probability of "1" in the noise sequence up to 0.3). In this thesis, we try to generalize those ideas to cascades of arbitrary length. Further in this literature survey, we concentrate on the elements of that cryptanalytic technique.

2.1 A particular statistical model applied in cryptanalysis of cascade of LFSRs

The statistical model that we are interested in is based on the binary rate multiplier (Figure 9). This model was originally mentioned in [14]. The model is applied in [22] for the cryptanalysis of pseudorandom sequence generators. It is shown in Figure 10. In this thesis, we analyze a stage of the cascade using this model.

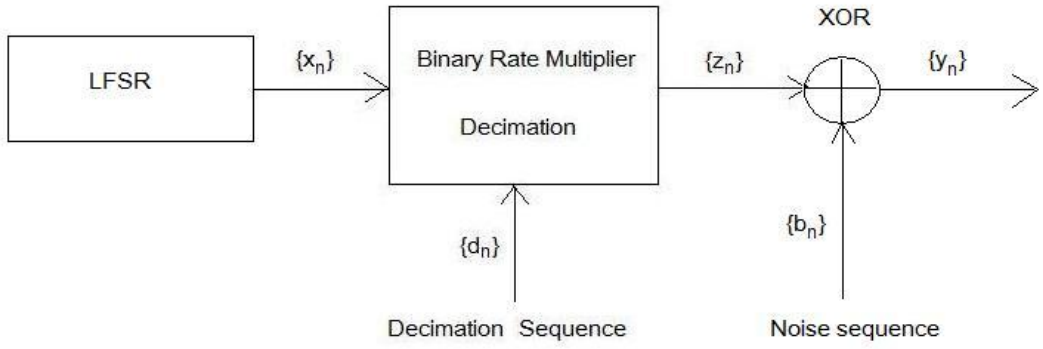

Figure 10. The statistical model of a stage of the cascade

According to [14], the feedback polynomial of the LFSR in this model is given as $f(x) = 1 + a_1x + a_2x^2 + \ldots + a_nx^L$, where L is the given length of this LFSR. $\{x_n\}$ is the output sequence of the LFSR. $\{d_n\}$, serving as the decimation sequence, is the output sequence from another LFSR. $\{z_n\}$ is the output sequence after the decimation processor. $\{b_n\}$ is the binary noise sequence, for example, the plaintext. $\{y_n\}$ is produced by the sum modulo 2 of the decimated sequence $\{z_n\}$ and the noise sequence $\{b_n\}$.

$$y_n = z_n \oplus b_n, \quad z_n = x_{f(n)}, \quad f(n) = n + \sum_{i=0}^{n} d_i \, , n = 0,1,2, \ldots$$

In this statistical model, it is supposed that $\{d_n\}$ is the realization of the sequence $\{D_n\}$ of independent and identically distributed random variables, with a certain probability $\Pr(D_n = i)$, which should be chosen in our research as it is described in Chapter 3.

The binary noise sequence $\{b_n\}$ is the realization of the sequence of random independent and identically distributed variables $\{B_n\}$ with probability $\Pr(B_n = 1) = p < 0.5, \forall n$.

2.2 Constrained edit distance

2.2.1 Definition of edit distance

In the paper [23] published in 1966, Vladimir Levenshtein considered error-correcting codes for channels with insertions and deletions. A review of matching methods as well as their relative mathematical analysis is introduced in [24] in which Levenshtein distance is mentioned, too. Levenshtein distance between two sequences (or strings) is defined as the minimum number of edit operations, including insertions, deletions and substitutions needed to transform one sequence (or string) into another. Due to the wide use of Levenshtein distance, it is often regarded as if it had the same meaning with the term “edit distance”.

2.2.2 Definition of constrained edit distance

The basic edit distance is used extensively with all kinds of constraints. For example, in papers [12],

[13] the algorithms, with which the constrained edit distance with the constraints on the total number of deletions, insertions and substitutions respectively, were developed.

2.2.3 Constrained edit distance for application in cryptanalysis

The cryptanalysis of a cascade of LFSRs with use of the statistical model given in Figure 8 requires a proper definition of constrained edit distance with special constrains. In this thesis, the required constrained edit distance is defined based on the edit operations of deletions and substitutions. The constraints are on the maximum number of consecutive deletions. This was discussed in [14]. In the following several paragraphs, we discuss the definition of constrained edit distance and its computation. They are mainly based on [14]. However, some additional explanations are given.

Let $X = \{x_i\}_{i=1}^{N}$ and $Y = \{y_i\}_{i=1}^{M}$ be two binary sequences with lengths N and M, respectively, over a finite alphabet A. x_i and y_i are elements of X and Y, respectively. The goal is to define a definition of constrained edit distance assigned to transforming the sequence X to the sequence Y.

First of all, let us define a nonnegative real function $d(x, y)$ as the elementary distance assigned to transforming X to Y. We introduce another alphabet $A^* = A \cup \{\emptyset\}$ where the symbol $\emptyset$ is so-called "empty element" needed for representing deletions. Then: $d(x, \emptyset)$ is the elementary distance associated with the deletion of a symbol, where $x \in A$; $d(x, y)$ denotes the elementary distance associated with the substitution of the symbol x with the symbol y, where $x, y \in A$.

Next, the definition of the constrained edit distance is given. The constrained edit distance $D(x, y)$ is defined as the minimum sum of elementary edit distances associated with the edit operations of deletion and substitution required to transform X to Y, under the constraint that the maximum number of consecutive deletions is E. Obviously, by transforming X to Y, the total numbers of deletions and substitutions are $N - M$ and M, respectively. We should bear in mind that there is no ordering of elementary edit operations of the edit transformation. The fact that only deletions and substitutions are used in transforming X to Y leads to the following inequality that must hold:

$$M \leq N \tag{2}$$

Since the constraint is on the maximum number of consecutive deletions and there is no ordering of the elementary edit operations, the following inequality must hold:

$$N \leq M + (M + 1)E \tag{3}$$

We now present a method to compute the constrained edit distance as defined above. Let $Y' = \{y_i'\}_{i=1}^{N}$ be an arbitrary finite sequence over A^* with the property that by removing all the empty elements from it we get Y (see [12]). Each transformation from X to Y can be expressed by a two-dimensional sequence (X, Y') called edit sequence. For $(X, Y') = \{(x_i, y_i')\}_{i=1}^{N}$, the following holds:

1. $(X, Y') \in A \times A^*$;
2. If $y_i' = \emptyset$, then x_i is deleted and if $y_i' \neq \emptyset$, then x_i is substituted with y_i'.

By G_{XY} we denote the set of all the so-called permitted edit sequences that transform the string X to the string Y. Permitted means that it holds that there are at most E consecutive empty symbols in Y'. The constrained edit distance is expressed in the following way:

$$D(X,Y) = \{\sum_{i=1}^{N} d(x_i, y_i') \mid (X,Y') \in G_{XY}\} \tag{4}$$

Finally, in order to calculate the constrained edit distance effectively, we express (4) in an iterative way by a new, iterative formula. This requires a new definition named partial constraint edit distance, which is also used in [22]. The partial constraint edit distance $W(e,s)$ is defined as the constrained edit distance between the prefix $X_{e+s} = \{x_i\}_{i=1}^{e+s}$ of the sequence X and the prefix $Y_s = \{y_i\}_{i=1}^{s}$ of the sequence Y with the same properties where the edit operations are deletions and substitutions and there is a constraint on the maximum number of consecutive deletions. Then,

$$W(e,s) = \min\{\sum_{i=1}^{e+s} d(x_i, y_i') | (X_{e+s}, Y_s') \in G_{es}\} \tag{5}$$

$$G_{es} = G_{X_{e+s}Y_s}$$

From (4), we get $D(X,Y) = W(N-M, M)$. The value $W(e,s)$ can be calculated by varying e and s in such a way that $0 \le s \le M$ and $0 \le e \le N-M$. From (3), we get $N-M \le (M+1)E$. This means that in (5), $e \le (s+1)E$. So, the set of all the values of $W(e,s)$ is obtained by varying

$$0 \le s \le M \tag{6}$$

$$0 \le e \le \min\{N-M, (s+1)E\} \tag{7}$$

According to the theorem from [14], when the elementary edit distance associated with deletion $d(x, \emptyset) \equiv d_e$ holds, (5) can be expressed as a recursion given below:

$$W(e,s) = \min\left\{W(e-e_1, s-1) + e_1 d_e + d(x_{e-e_1+s}, y_s) \middle| \max\{0, e-\min\{N-M, sE\}\} \le e_1 \le \min\{e, E\}\right\} \tag{8}$$

All the values of $W(e,s)$, depending on whether or not s=0, are divided into two sets:

For $1 \le s \le M$, $0 \le e \le \min\{N-M, (s+1)E$, and, for $s = 0$, and $0 \le e \le \min\{N-M, E\}$,
$W(e,0) = ed_e$ (9)

2.3 The phases of cryptanalysis of cascade with two LFSRs

Two phases of cryptanalysis of cascade with two LFSRs (Fig. 1) are discussed in [22]. In this cascade, there are two LFSRs. The first register LFSR1 is regularly clocked while the second one LFSR2 is irregularly clocked with the clock signals from the output sequence of LFSR1 (Fig.1). The clocking

satisfies the model described in §2.1 (Fig.10).

As it can be seen in Figure 10, the output sequence of LFSR1 $\{d_n\}$ serves as the decimation sequence. The output sequence of LFSR2 $\{x_n\}$ is decimated by $\{d_n\}$. The output sequence of the binary rate multiplier $\{z_n\}$ and the noise sequence $\{b_n\}$ are summed modulo 2 to produce the final output sequence $\{y_n\}$.

There are two main phases in the cryptanalysis of the cascade with two LFSRs. The two phases are briefly described in the paper [22], in which the emphasis is put on the reconstruction of suboptimal paths in the constrained edit distance array. We will discuss the outline of these two phases, based on [22], in the subsequent paragraphs.

The first phase of the cryptanalysis of a cascade with two LFSRs is that the candidate initial states of LFSR2 should be determined. [14] describes a correlation attack based on the statistical model given in §2.1. Firstly, certain number of initial states are chosen at random from all the non-zero initial states of LFSR2. Then, these selected initial states are used to generate $\{x_n\}$, whose length depends on the length of the intercepted sequence. After that, the constrained edit distance between the intercepted sequence and each generated $\{x_n\}$ is calculated. The threshold is chosen to be an integer greater than or equal to the maximum of the results of computation of the constrained edit distance. In the next step, we use all the remaining initial states of LFSR2 to generate $\{x_n\}$, between which and the intercepted sequence, a new set of constrained edit distances is calculated. The states producing $\{x_n\}$, whose constrained edit distance to the intercepted output sequence is less than the threshold represent the candidate initial states for LFSR2.

The second phase of the attack is to reconstruct the clock control sequence of LFSR2. In 2007, Petrovic and Fuster-Sabater [22] proposed an attack, in which a new algorithm was used to search for the optimal and suboptimal paths through the matrix of constrained edit distances, each corresponding to a possible clock control sequence for LFSR2. This search is directed by increasing the tolerance for weight discrepancy between the weight (i.e. constrained edit distance) of a reconstructed suboptimal path and the weight of the optimal path. In this thesis, we extend the ideas from [22] to cryptanalysis of a cascade of irregularly clocked LFSRs containing an arbitrary number of LFSRs.

3 Cryptanalysis of the cascade with an arbitrary number of LFSRs

3.1 The method of splitting the cascade into subcascades

The scheme of a cascade of LFSRs presented in Figure 11 (a) can be divided into two parts: the first one is the subcascade that generates the clock control sequence for the last LFSR, and the second one is the last LFSR in the cascade itself (Figure 11 (b))

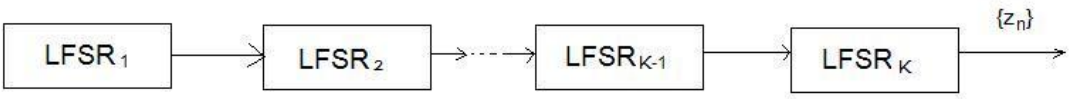

Figure 11 (a). The scheme of a cascade of irregularly clocked LFSRs

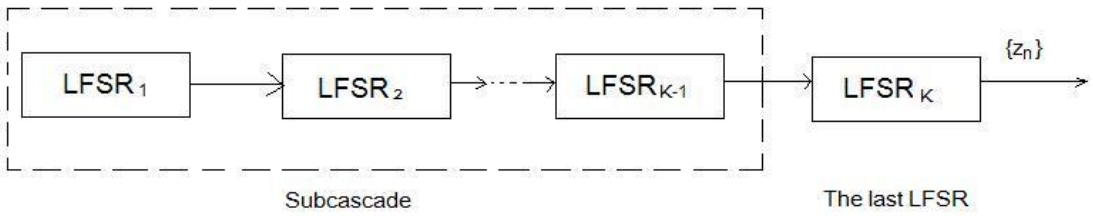

Figure 11 (b). The scheme from the Figure 11 (a) divided into two parts

If the prefix of the output sequence of the last LFSR is known to the attacker, it is possible to reconstruct the initial state of the last LFSR with a generalized correlation attack. It is possible to obtain a set of candidate initial states of that LFSR, which could generate the intercepted output sequence. This can be realized by using the model described in §2.1.

The next step is to determine which initial state, among all the candidate initial states obtained, can generate the intercepted output sequence. So the control sequence that, together with one of the candidate initial states of the last LFSRs in the cascade, could generate the prefix of the intercepted sequence has to be determined. We use the constrained edit distance matrix associated with each candidate initial state to reconstruct the paths, each of which maps to a possible control sequence.

After the possible clock control sequences of the last LFSR in the cascade are reconstructed, we can treat the rest of the scheme as a cascade of lower dimension. We can use the possible clock control sequences of the last LFSR as the output sequence of the subcascade. Then the same process can be used in the subcascade. The selection of the number of digits of the possible clock control sequence will be discussed later.

3.2 Reconstruction of candidate initial state

The first step is to reconstruct the candidate initial state of each LFSR in the cascade (except of the first one). We use the statistical model described in §2.1. For convenience, we copy Figure 10 to Figure 12 here.

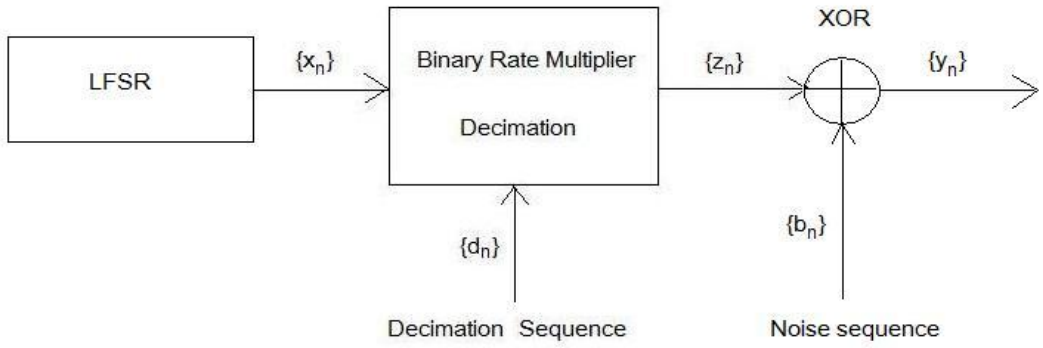

Figure 12. The statistical model of a stage in the cascade

$\{x_n\}$ is the output binary sequence of LFSR. $\{d_n\}$, serving as the decimation sequence, is the output sequence from the previous subcascade of LFSRs, $0 \le d_n \le E$ where E is given in advance. $\{b_n\}$ is the binary noise sequence including plaintext information, $\{z_n\}$ is the output sequence after the decimation processor that is obtained in the following way :

$$z_n = x_{f(n)}, \qquad f(n) = n + \sum_{i=0}^{n} d_i \ , n = 0,1,2, \ldots$$

In this statistical model, $\{d_n\}$ is the realization of the sequence $\{D_n\}$ of independent and identically distributed random variables, with the probability $\Pr(D_n = i) = \frac{1}{E+1}, 0 \le i \le E, \forall n$. E is the maximum number of consecutive deletions. The binary noise sequence $\{b_n\}$ is the realization of the sequence of random independent and identically distributed variables $\{B_n\}$ with probability $\Pr(B_n = 1 = p \neq 0.5, \forall n$. Here it is supposed that

$$\Pr(B_n = 1) = p < 0.5.$$

$\{y_n\}$ is produced by the sum modulo 2 of the decimated sequence $\{z_n\}$ and the noise sequence $\{b_n\}$.

$$y_n = z_n \oplus b_n.$$

In the process of reconstruction of candidate initial states, it is supposed that the cryptanalyst possesses M consecutive bits of $\{y_n\}$. The task for the cryptanalyst is to determine the initial state of the generator that produced the M intercepted bits of the sequence $\{y_n\}$.

The first phase of the attack on a stage of the cascade makes use of the edit distance with the constraint on the maximum length of the runs of deletions, which was discussed in §2.2.3. However, an additional constraint is added here to conform to the real situation in the pseudorandom generator, as presented in Figure 12.

Let us consider two binary sequences X and Y. The length of the binary sequence X is N, and the length of the binary sequence Y is M. The elementary edit operations transforming from X to Y contain deletions and substitutions. We define the constrained edit distance between X and Y as the minimum number of elementary edit operations needed to transform X to Y. There are two constraints in the constrained edit distance. The first constraint is that the number of consecutive deletions is less than or equal to E. The second one is that the elementary edit operations are ordered in the sense that first deletions are executed and then the substitutions are performed.

With the additional constraint, the formulas (2) and (3) are modified to the following (10) and (11) respectively:

$$M \leq N \tag{10}$$

$$N \leq M + ME \tag{11}$$

Making use of the similar deduction process as in §2.2.3, the formulas (6) and (7) are modified to the following formulas (12) and (13) respectively:

$$0 \leq s \leq M \tag{12}$$

$$0 \leq e \leq \min\{N - M, sE\} \tag{13}$$

Finally, the formulas (8) and (9) will be modified to the formulas (14) and (15):

$$W(e,s) = \min\left\{W(e - e_1, s-1) + e_1 d_e + d(x_{e+s}, y_s) \middle| \max\{0, e - \min\{N-M, (s-1)E\}\} \leq e_1 \leq \min\{e, E\}\right\} \tag{14}$$

For $1 \leq s \leq M, 1 \leq e \leq \min\{N - M, sE\}$, and, for $e = 0$, and for all s between 1 and M, we have

$$W(0,s) = W(0, s-1) + d\,(x_s, y_s) \tag{15}$$

From now on, we assume that $d(x,y) = 0$ if $x = y$; $d(x,y) = 1$ if $x \neq y$; $E = 1$.

In the first phase of the attack, we determine the candidate initial states of the last LFSR in the cascade. It was described in [14] and contains three steps.

The first step: the length N of the output sequence of the last LFSR without decimation has to be estimated. Given the string Y with the length M, the length of N of the string X which generates Y is unknown. So the aim is to choose N given M. N depends on the maximum number of consecutive deletions E. There are two ways of estimating the value of N. First, N is chosen close to its mathematical expectation. For example, in our case, $E = 1$, then $N = M/2 + 2M/2 = 3M/2$. Second, we can specify N such that the probability that the sequence X is longer than N is close to zero. This can be realized by modifying the definition of constrained edit distance such that an arbitrary number of consecutive deletions is permitted at the beginning or at the end of X [14].

The second step: the threshold T necessary for the classification of the initial states of the last LFSR in the cascade should be estimated. For this purpose, the probabilities of "missing the event" P_m and "false alarm" should be selected in advance. Normally, P_m is chosen close to zero (e.g. 10^{-3}) and P_f is picked very close to zero [14]. Then threshold is computed by checking $1/P_m$ initial states selected at random. For each of the initial states, the constrained edit distance defined above is calculated. The threshold T is chosen to be greater than or equal to the maximum distance value obtained in the process.

The third step: for each possible initial state, which is not used in the second step, the constrained edit distance between the output sequence $\{x_n\}$ and the intercepted sequence $\{y_n\}$ is calculated. All the initial states with the constrained edit distance less than or equal to the threshold T are determined to be the candidate initial states.

Reconstruction of the clock control sequence of the last LFSR will be discussed in the next section §3.3.

3.3 Clock control sequence reconstruction

From now on, by the last LFSR in the cascade we mean the last LFSR in the subcascade that is currently analyzed (see Figure 11(b)).

The purpose of our reconstruction is that we reconstruct the clock control sequence corresponding to every candidate initial state of the last LFSR. The clock control sequence can be obtained by determining the optimal and/or suboptimal paths of adequate length over the constrained edit distance matrix.

In this thesis, we apply the method under the following constraints:

1. The maximum length of runs of deletions is E
2. The elementary edit operations are ordered in the sense that first the deletions are performed and then the substitutions.

These particular constraints were discussed in §3.2. And the corresponding edit distance is defined and expressed by formulas (14) and (15). In the following paragraphs, we follow the analysis method from [22] and use it in correspondence with this particular definition of edit distance containing the

particular constraints. However, all the definitions will be updated to serve the new constrained edit distance.

Optimal paths [21] are defined as the paths which start at $W[N-M, M]$ and go through the whole edit distance matrix. The value pl is defined as the length of the clock control sequence needed to reconstruct the initial state of the sub-generator described above. It holds that $pl \leq M$. The optimal paths go through the cells $W[e_{p_1}, pl], \dots W[e_{p_n}, pl]$. However, due to the presence of the noise sequence, the clock control sequences corresponding to the optimal paths do not necessarily generate the intercepted output sequence. So we also need to reconstruct the suboptimal paths as the supplements of the optimal paths. The suboptimal paths [21] are the paths, whose weight-difference from the optimal ones is not larger than the discrepancy value D. D is given according to the noise level in the statistical model, which is based on the binary rate multiplier.

In the column pl, there are several points, which the optimal paths that begin from $W[N-M, M]$ go through. In order to determine these particular points, each cell $W[e, s]$ in the matrix is assigned a value c as the edit distance, and four associated vectors [21]:

1. vp is the vector of 'primary' pointers. These pointers correspond to particular cells $W[vp(1), s-1], \dots, W[vp(k), s-1]$. From these cells, it is possible to arrive to the cell $W[e, s]$ with the minimum cost of edit operations.
2. vu is the vector of 'updated' pointers. These pointers correspond to particular cells $W[vu(1), pl], \dots, W[vu(l), pl]$. From these cells, it is possible to arrive to the cell $W[e, s]$.
3. ve is the vector of 'updated' pointers. These pointers correspond to the cells $W[ve(1), s-1], \dots, W[ve(j), s-1]$. From these cells, it is possible to arrive to the cell $W[e, s]$, regardless of the cost of edit operations.
4. The vector of values vj of the edit distances corresponding to the elements of the vector ve.

ve contains the whole of vp. In this thesis, we keep them separated for easier understanding of the algorithms.

The numbers k, l, and j depend on E and pl.

The matrix W is filled by means of the following algorithm [21]:

Algorithm 1 [21]

Input:

1. The sequences X and Y of lengths N and M, respectively.
2. The length pl of the clock control sequence necessary to reconstruct the initial state of the subgenerator that generates it
3. The maximum length E of runs of deletions.
4. The elementary distance d_e associated with the deletion of a symbol.
5. The elementary edit distance $d(x, y)$ associated with the substitution of the symbol x by the

symbol y, $\forall x, y$.

Output:

The matrix W of edit distances with the vectors $vp, vu, vj,$ and ve associated with every cell.

Initialization: $W[e,s].c = \infty, e = 0, \ldots, N - M, s = 0, \ldots, M;$

The vectors vp, vu, vj, and ve associated with every cell $W[e,s]$ are empty.

$W[0,0].c = 0;$

// The row 0 of the matrix W:

for $s = 1$ to M {

$W[0,s].k = 1;$

$W[0,s].c = W[0,s-1].c + d(X[s], Y[s]);$

$W[0,s].vp[1] = 0;$

}

Main loop

for $s = 1$ to M {

for $e = 1$ to $\min\{N - M, s * E\}$ {

Let q be the minimum value of the expression

$$W[e - e1, s - 1].c + e1 * de + d(X[e + s], Y[s]), \quad (16)$$

$$e1 = \max\{0, e - \min\{N - M, (s-1) * E\}\} \ldots \min\{e, E\}.$$

Let nq be the number of values of $e1$ for which the expression (16) takes the value q.

Then

$W[e,s].c = q;$

$W[e,s].k = nq;$

The vector $W[e,s].vp$ is filled with nq values of the expression $e - e1$ corresponding to the values

e1 for which the expression (16) takes the value q.

}

The vector $W[e,s].vj$ is filled with all the values (not necessarily the minimum ones) of the expression (16).

The vector $W[e,s].ve$ is filled with the values $e - e1$ corresponding to the values of

$W[e,s].vj$.

Determining updated pointers

// In the column $pl + 1$, $vp = vu$

if $s = pl + 1$ then {

for $e = 0$ to $\min\{N - M, s * E\}$

$W[e,s].vu = W[e,s].vp$;

}

else if $s > pl + 1$ {

For every element of the vector $W[e,s].vp, e = 0 \ldots \min\{N - M, s * E\}$ the corresponding set of updated pointers is determined. These are placed into the vector $W[e,s].vu$ in the following way:

All the elements of $W[W[e,s].vp[i], s - 1].vu$, are placed into the vector $W[e,s].vu$, deleting those that are repeated, $i = 1 \ldots W[e,s].k$.

}

}

After the edit distance matrix has been constructed, the next step is reconstructing the candidate clock control sequences. In the literature, especially in that related with computational biology, this is often carried out by building a search tree corresponding to a special kind of graph emerging from the edit distance matrix (see for example [25, 26, 27, 28]. However, in that case two edit distance matrices are necessary: one corresponding to the forward edit transformation of X and Y and the other corresponding to the edit transformation of the sequence X′ to the sequence Y′ with the order of elementary edit operations interchanged (first the substitution takes place and then the run of deletions), where X′ and Y′ are the sequences X and Y in reverse order, respectively. These methods do not generally consider the type of the constraints used in this thesis. Because of that, we developed another method for reconstructing paths in the matrix W suitable for the purpose of cryptanalysis.

From now on, by paths we mean fragments of paths that start in the column pl of the matrix W.

There are three sets of paths to be reconstructed. The first one consists of optimal paths that start at the points $e_{p_i} = W[N - N, M], vu[i], i = 1, \ldots W[e, s].l$. The second one consists of suboptimal paths, whose weight-difference from the optimal ones is $\leq D$, that start at $e_{p_i} = W[N - N, M], vu[i], i = 1, \ldots W[e, s].l$. The third set consists of suboptimal paths, whose weight-difference from the optimal ones is $\leq D$, that start at other points in the column pl.

The elements of the vector $W[N - M, M].vu$ at the end of the execution of the Algorithm 1 represent the initial points of the "depth-first" search for the elements of the first and second set mentioned above. As for the third set, if $| W[e_{p_i}, pl].c - W[e, pl].c | \leq D, e = 0, \ldots \min\{N - M, sE\}, e \neq e_{p_i}$ for at least one i, then the point $W[e, pl]$ is an initial point of the depth first search for the paths of the third set.

For every initial point ε of any set, the following algorithm is executed in order to determine all the optimal and necessary suboptimal paths that start in it:

Algorithm 2 [21]:

Input:

1. The matrix W of edit distances, obtained by means of the Algorithm 1.
2. The value of ε .

Output:

All the paths that start at the point $W[\varepsilon, pl]$ that belong to the corresponding set(s) mentioned above. Each of these paths determines an output sequence of the subgenerator that generates the clock control sequence.

Initialization:

// The stack consists of the elements of the matrix W

$t = 0;\ e = \varepsilon\ ;\ s = pl;\ nsp = 0;$

Main loop:

repeat

badpath = false ; // Overweight indicator

while (($e > 0$) or ($s > o$)) and not badpath {

// Detect a branching point

```
   If ( W[e, s]. j > 1) and ((e <> stack [nsp]. e ) or
   (s <> stack [nsp]. s)) then {
nsp + + ;
 // Put e, s and W[e, s] on the stack, at the position nsp.
 stack [nsp] . W = W[e, s];
 stack [nsp] . e = e ;
 stack [nsp] . s = s ;
}
 // Process a branching point
 if ( stack [nsp]. e = e) and (stack [nsp] . s = s) then {
 badpath = false;
 repeat
```

Consider the possibility of branching from the current branching point to one of the possible successors, i.e. the point j. If this possibility is chosen, and after that only the branchings to the points that lead to the optimal subpaths are followed, then the total weight of the chosen subpath is

```
aw = stack [nsp] . W . vj [stack [nsp]. W . j] ;
```

and the total weight of the corresponding path is

```
tw = weight (α , β , t) + aw ;
```

where weight is the function that returns the weight of the path before the branching and (α, β) is the prefix of the edit sequence of length t.

```
   // eprev is the value of e that corresponds to the previous path element
   If tw <= W [ε , pl]. c + D then
      eprev = stack [nsp]. w. ve[stack[nsp]. w. j];
   stack [nsp]. W . j − − ;
```

```
   if stack [nsp] . W . j = 0 then nsp-- ;
   until (eprev was initialized from the stack) or (all the successors were examined) ;
   if (eprev was not initialized from the stack ) then
      badpath = true ;
}
// Process a non- branching point
if (eprev was not initialized from the stack) and
(not badpath) then {
   aw = W[e, s]. vj [W[e, s]. j];
   tw = weight (α , β , t) + aw ;
   if tw <= W [ε , pl]. c + D then
     eprev = W[e, s]. ve[W[e, s]. j] ;
   else badpath = true ;
}
// Reconstruct the current path element
if not badpath then{
   if s > 0 then {
  t + + ;
 α [t] = X[e + s];
 β [t] = Y[s];
  }
for ii = 1 to e – eprev {
   t + + ;
```

α [t] = X[e + s − ii],

β [t] = Ø ;

}

e = eprev ; s − − ;

}

}

if not badpath then

Store the output sequence of the subgenerator obtained in such a way;

// Back to the current branching point

If nsp > 0 then {

t = t – stack [nsp]. e – stack [nsp] . s ;

e = stack [nsp] . e ;

s = stack [nsp] . s ;

}

until nsp = 0

Example:

Suppose that we have the sequences X and Y. Let X=1010110100, Y=1101011, N=10, M=7, E=1, D=1, pl=5. Fig. 13 shows the constrained edit distance matrix obtained by the Algorithm 1. The figure contains pointers vp, ve and vj. In the matrix W, the symbol & means that the value in the corresponding cell does not exist due to the particular constraints:

1. The maximum length of runs of deletions is $E = 1$
2. The elementary edit operations are ordered in the sense that first the deletions are performed and then the substitutions.

Taking the cell W [1,2] for example, the value of cell W [1,2] can either obtained from the cell W [0,1] or from the cell W[1,1] , under the constraints described.

If the value of cell W [1,2] is from cell W [0,1], both deletion and substitution are performed. The

deletion operation costs 1.The substitution operation from $X_{e+s} = X_{1+2} = X_3$ to $Y_s = Y_2$costs 0. Then, the value of cell W [1,2] is the sum of W [0,1], the cost of deletion and that of substitution . The sum is $W[0,1] + 1 + 1 = 0 + 1 + 0 = 1$.

If the value of cell W [1,2] is from cell W [1,1], only substitutions are performed. The operation of substitution from $X_{e+s} = X_{1+2} = X_3$ to $Y_s = Y_2$ costs 0. Then, the value of cell W [1,2] is the sum of W [1,1] and the cost of substitution. The sum is $W[1,1] + 0 = 2 + 0 = 2$

As a result, there are two possible values of cell W [1,2]: 1 and 2. Since 1<2, W [1,2].c=1. W [1,2].vp[1]=0 (index), W [1,2].ve[1]=1 (underlined index), and W [1,2].vj[1]= 2 (underlined).

e \ s	0	1	2	3	4	5	6	7
0	0_0	0_0	1_0	2_0	3_0	4_0	4_0	5_0
1	&	2_0	$1;\underline{2}_{0;1}$	$1;\underline{2}_{1;0}$	$1;\underline{3}_{1;0}$	$2;\underline{5}_{1;0}$	$3;\underline{6}_{1;0}$	$3;\underline{5}_{1;0}$
2	&	&	4_1	$3;\underline{5}_{1;2}$	$2;\underline{3}_{1;2}$	$2_{1,2}$	$2;\underline{3}_{2;1}$	$3;\underline{5}_{2;1}$
3	&	&	& 6_2	$5;\underline{7}_{2;3}$	$4;\underline{6}_{2;3}$	$4;\underline{5}_{2;3}$	$4;\underline{5}_{2;3}$	

Figure 13. Edit distance matrix obtained by the Algorithm 1, with the pointers vp, ve and vj

Fig. 14 shows the pointer updating, including the calculation of the pointers vu. All the pointers vu are presented in boldface letters. The pointers vu only exist in the cells in the columns 6 and 7 of the edit distance matrix because the value of pl is 5. In particular, the pointers vu in the column 6 are equal to the pointers vp.

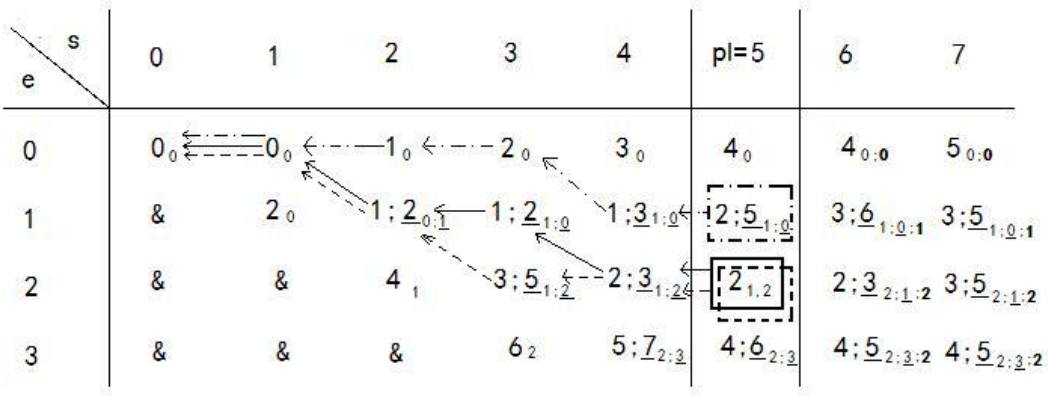

Figure 14. Updating pointers; reconstructed paths from the first, second and third set: the solid line corresponds to the first set, the dashed line to the second set and the dotdashed line to the third set

Let us consider the element W[3,7]. At this cell, W[3,7].c=2, W[3,7].vp[1]=2 ,W[3,7].ve[1]=3 and W [3,7].vj[1]= 5 . Since W[2,6].vu[1]=2 then W[3,7].vu[1]=2.

Fig. 14 also illustrates examples of reconstructed paths from the first, second and third set. In Fig.14, the solid line corresponds to the first set, the dashed line to the second set and the dotdashed line to the third set. The cell W[2,5] is the starting cell for the paths of the first and second set. The cell W[1,5] is the only possible starting point for the paths from the third set, because it is the only cell in the column pl for which $W[e, pl].c \leq W[2, pl].c + D$.

In this particular case, D is equal to 1, which is given in advance. In the experiment, the parameter D can be adjusted. For example, if D=2, not only the cell W[1,5], but also the cells W[0,5] and W[3,5] are the starting points for the paths from the third set, as well.

4 Experimental Work

4.1 Objective of the experiment

The purpose of our experiment was to verify that the methods discussed in the previous chapters can be used in cryptanalysis of a cascade of LFSRs. A similar experiment with two LSFRs was implemented in [22]. In our experiment, by using the theory discussed in the previous chapters, we carry out cryptanalysis of a cascade with three LSFRs. By doing this, the method is extended from the cascade with two LFSRs to that with three LFSRs. Then the same procedure can be applied to extending to the cascade with an arbitrary number of LSFRs in the future work.

We will choose three LFSRs constituting the cascade (see Fig. 15). These LFSRs are clocked in the following way:

1. $LFSR_1$ is regularly clocked.
2. If the output of $LFSR_1$ is 0, $LFSR_2$ is clocked once, and the output bit of $LFSR_2$ is valid and is used to clock $LFSR_3$.
3. If the output of $LFSR_1$ is 1, LFSR2 is clocked twice. The first output bit of $LFSR_2$ is discarded. The second output bit is valid and is sent to clock $LFSR_3$.
4. The same procedure is applied in clocking $LFSR_3$ by using the valid output bit of $LFSR_2$.

In our experiment, as a matter of convenience, we chose three LFSRs with the same feedback polynomials:

$$f(x) = 1 + x^2 + x^7 + x^6 + x^{10} \quad (17)$$

Each LFSR had 4 feedback taps and 1 output tap. The output tap was the content of the leftmost cell of each LFSR. In addition, all the LFSRs had the length of 10.

Figure 15. A cascade of three LFSRs

4.2 Simulation of generating the intercepted sequence

The aim of the attack was, given the intercepted sequence (the output sequence of $LFSR_3$ with a certain level of noise), to determine the initial states of $LFSR_3$, $LFSR_2$ and $LFSR_1$ successively

as well as the correct clocking sequences of $LFSR_3$ and $LFSR_2$. In order to demonstrate our attack procedure to be feasible, we needed to compare the initial states and the clocking sequences we obtained from our attack to the correct ones, which were known in advance. This could be realized by the following logical steps:

1. The initial states of each LFSR in the cascade were chosen at random. They were recorded.
2. With the initial states obtained in the step 1 and the clocking algorithm, we could start the simulation procedure. By simulating the irregular clocking procedure, the intercepted sequence was generated.
3. The corresponding output sequences of $LFSR_2$ and $LFSR_1$, which were used to generate the intercepted sequence, were recorded.
4. The attack (cryptanalysis) procedure was started bearing in mind the fact that the only known information was the intercepted sequence and the feedback polynomials of each LFSR.
5. Compare the results of the attack, including the initial states of the three LFSRs and the output sequences of $LFSR_3$ and $LFSR_2$, to the recorded ones in step 1 and 3.
6. If the values of each comparison are identical, the attack method is proved to be feasible.

In the simulation procedure of our experiment, the initial states of $LFSR_1$, $LFSR_2$ and $LFSR_3$ were set to 0101111111, 0011100100 and 1011100000, respectively. The length of the output intercepted sequence was set to 200. The LFSRs had feedback polynomials (17). Then the simulation was executed and we generated 200 bits of the intercepted sequence Y as well as the corresponding output sequence of $LFSR_1$ (clock 1) and $LFSR_2$ (clock 2), and the intercepted sequence (Y_1) without noise.

4.3 Cryptanalysis processes and results

The cryptanalysis (attack) procedure consists of the following steps:

1. Determine the candidate initial states of $LFSR_3$.
2. Determine the correct clocking sequence of $LFSR_3$.
3. Determine the candidate initial states of $LFSR_2$.
4. Determine the correct clocking sequence of $LFSR_2$.
5. Determine the correct initial state of $LFSR_1$.

In this cryptanalysis procedure, it is supposed that the feedback polynomials are known in advance as shown in (17). In the first place, the method used in the realization of step 1 and step 3 was described in [14]. Furthermore, after we obtained the clocking sequence of $LFSR_2$ in the step 4, we can calculate the initial state of $LFSR_1$ by solving the system of linear equations in the step 5. So, in our experiment, we shall only focus on the step 2 and the step 4.

In the step 1, with the knowledge of the intercepted sequence, we can determine a set of candidate initial states of $LFSR_3$ (see the correlation attack described in [14]). In our experiment, we supposed that we obtained the correct initial state of $LFSR_3$ which, together with the corresponding clocking sequence of LFSR3, could produce the intercepted sequence.

In the step 2, the initial state of $LFSR_3$ was set to 1011100000, which was equal to the value used in the simulation process. And the intercepted sequence used to establish the constrained edit distance was also set to be equal to the value obtained by the simulation process. The length of the intercepted sequence was set to 200. Besides, the length pl out of the clocking sequence of $LFSR_3$ was set to 40, which means only the first 40 bits of clock 2 were used in the comparison process. Then we generated the corresponding constrained edit distance matrix using Algorithm 1, and reconstructed all the optimal and sub-optimal paths using Algorithm 2. The result shows that, after searching 7612 paths, the correct clock sequence is found equal to clock 2.

The same process was applied in the step 4, where the initial state of $LFSR_2$ was set to 0011100100. As the feedback polynomial of $LFSR_1$ was known, the length pl was set to 10, which was equal to the length of $LFSR_1$. The length of clock 2 needed was set to 40. Then we generated the corresponding constrained edit distance matrix using Algorithm 1 again, and reconstructed the optimal and sub-optimal paths using Algorithm 2 again. The result shows that, after searching only 20 paths, the correct clock-control sequence was found equal to clock 1.

The experiment results proved that the theory discussed in previous chapters and the methods described in this chapter were feasible.

5 Conclusion

In this thesis, the cryptanalysis of the cascade of irregularly clocked linear feedback shift registers is described. The attack is essentially a method for reconstruction of the initial state of the subcascade that generates the clock control sequence. The statistical model based on binary rate multiplier, which employs the constrained edit distance is used. Instead of checking all the possible initial states of the subcascade, all the possible optimal paths in the edit distance matrix as well as the suboptimal paths, whose weight-difference from the optimal ones dose not overcome the discrepancy D given in advance, are reconstructed by depth-first search. Experimental results show that the methods described in this thesis, which is used to cryptanalyze a cascade with an arbitrary number of LFSRs, are applicable.

The research questions raised in §1.5 have been answered. It is feasible to generalize the correlation attack against a scheme with 2 LFSRs, of which one irregularly clocks another, to a cascade of irregularly clocked LFSRs. First of all, the cascade with an arbitrary number of LFSRs was split into subcascades. Then, the possible candidate initial states of the last LFSR were reconstructed with a generalized correlation attack. Once the set of candidate initial states is known, the element that generates the intercepted output sequence has to be determined. The attack continued by determining the clock control sequence that, together with one of the candidate initial states of the last LFSR, could generate the intercepted sequence. This can be achieved by searching through the constrained edit distance matrix associated with every candidate initial state obtained in the previous step of attack. The optimal and suboptimal paths were reconstructed in the searching procedure. Once the possible clock control sequences of the last LFSR in the cascade were reconstructed, we treated the rest of the scheme as a cascade of lower dimension utilizing the reconstructed clock sequences as its output sequence. Then, the same analytical methods were applied into the cascade of lower dimension. This process continues until the possible initial states of the first LFSR in the cascade were reconstructed.

Bibliography

[1] T. Beth, F. Piper. 1984. The stop-and-go-generator, in Proceeding of EUROCRYPT 84, LNCS 209, Springer Verlag, Berlin, pp. 88-92

[2] Chambers, W.G., Jennings, S.M. 1984. Linear Equivalence of Certain BRM Shift-register Sequences. Electronics Letters 10(24), 1018-1019.

[3] D. Coppersmith, H. Krawczyk, Y. Mansour. 1993. The shrinking generator, in Proceedings of CRYPTO 2003, LNCS 773, Springer Verlag, pp. 22-39

[4] C.Günther. 1987. Alternating Step Generators Controlled by de Bruijn Sequences, in proceedings of EUROCRYPT 87, LNCS 304, Springer Verlag, pp. 5-14.

[5] A. Menezes, P. van Oorschot, and S. Vanstone. 1997. Handbook of Applied Cryptography, CRC Press.

[6] R.A.Rueppel. 1986. Analysis and Design of stream Ciphers. Springger-Verlag.

[7] H.Beker, F.Piper. 1982. Cipher systems-The protection of communications. Northwood Books.

[8] M. Krause, F. Armknecht. 2003. Algebraic Attacks on Combiners with Memory, in Proceedings of Crypto 2003, LNCS 2729, Springer Verlag, pp. 162-176.

[9] N. Courtois. 2003. Fast algebraic attacks on stream ciphers with linear feedback, in proceedings of Crypto 2003, LNCS 2729, Springer Verlag, pp. 177-194.

[10] T. Siegenthaler. 1985. Decrypting a class of stream ciphers using ciphertext only, IEEE Trans. Comput. Vol. 34, pp. 81-85.

[11] D. Gollman, W.G. Chambers. May 1989. Clock-controlled shift registers: a review, IEEE J. Select. Areas Comm., vol. 7, pp. 525-533.

[12] B.J. Oommen. 1986. Constrained string editing, Inform. Sci., vol. 40, pp. 267-284.

[13] B.J. Oommen. Sept. 1987. Recognition of noisy subsequences using constrained edit distance, IEEE Trans. Pattern Anal. Mach. Intell., vol. 9, pp. 676-685.

[14] Golić, J., Mihaljević, M. 1991. A Generalized Correlation Attack on a Class of Stream Ciphers Based on the Levenshtein Distance. Journal of Cryptology 3(3), 201-212.

[15] E. R. Berlekamp. 1968. Algebraic Coding Theory. New York: McGraw-Hill. Cha7, 10.

[16] Z. Al-Hinai, E. Dawson, M. Henricksen, L. Simpson. 2007. On the security of the LILI family of

stream ciphers against algebraic attacks, in proceedings of ACISP 2007, LNCS 4586, Springer Verlag, pp. 11-28

[17] Fast Reconstruction of Clock-Control Sequence, Electronics Letters, Vol. 38, No. 20 (2002), pp. 1174-1175.

[18] J. Golić and R. Menicocci, Edit Distance Correlation Attack on the Alternating Step Generator, in: Kaliski B. (Ed.), Advances in Cryptology: Proceedings of CRYPTO 97, Lecture Notes in Computer Science 1294, Springer-Verlag, New York, 1997, pp. 499-512.

[19] T. Johansson, Reduced Complexity Correlation Attacks on Two Clock-Controlled Generators, in: Ohta K. (Ed.), Advances in Cryptology: Proceedings of ASIACRYPT '98, Lecture Notes in Computer Science 1514, Springer-Verlag, New York, 1998, pp. 342-356.

[20] H. Molland, Improved Linear Consistency Attack on Irregular Clocked Keystream Generators, in: B. Roy and W. Meier (Eds.): FSE 2004, LNCS 3017, 2004, pp. 109–126.

[21] S. Petrović and Amparo Fúster Sabater, Clock Control Sequence Reconstruction in the Ciphertext Only Attack Scenario, in: J. López, S. Qing, and E. Okamoto (Eds.): ICICS 2004, LNCS 3269, Springer-Verlag, Berlin Heidelberg, 2004, pp. 427–439.

[22] S. Petrović and A. Fuster. 2007. Reconstruction of Suboptimal Paths in the Constrained Edit Distance Array with Application in Cryptanalysis, in Proceedings of the 2007 International Conference on Computational Science and its Applications, ICCSA 2007, Kuala Lumpur, Malaysia, Lecture Notes in Computer Science, LNCS 4707, Part III, pp. 597-610.

[23] A. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals, Soviet Phys. Dokl., vol. 10, pp. 707-710.

[24] D. Sankoff, J.B. Kruskal. 1983. Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, Reading, MA.

[25] E.L. Lawler. 1972. A Procedure for Computing the K-Best Solutions to Discrete Optimization Problems and its Applications to the Shortest Paths Problem, Manag. Sci., No. 18 pp. 401-405.

[26] D. Naor and D. Brutlag. 1994. On Near-Optimal Alignments of Biological Sequences, J.Comput. Biology, Vol. 1, No. 4 pp. 349-366

[27] M. Vingron and P. Argos. 1990. Determination of Reliable Regions in Protein Sequence Alignmentss, Prot. Eng., No.3 pp. 565-569.

[28] M.S. Waterman and T.H. Byers. 1985. A Dynamic Programming Algorithm to Find All Solutions in a Neighbourhood of the Optimum, Math. Biosci. Vol. 66 pp. 179-188