Journal of Algebra, Number Theory: Advances and Applications
Volume 9, Number 1-2, 2013, Pages 1-28

A NEW APPROACH TO COMBINE LFSRS: α-GENERATORS

PIERRE DUSART[1], ABDELKADER NECER[1] and SINALY TRAORÉ[2]

[1]XLIM UMR CNRS 7252
123 avenue Albert Thomas
87060 Limoges
France
e-mail: pierre.dusart@unilim.fr

[2]Faculté des Sciences et Techniques
USTTB
BP E 32 06 Bamako
Mali
France

Abstract

The pseudo-random generators based on LFSR are widely used in stream cipher cryptosystems. In this paper, we propose a new family of pseudo-random generators based on LFSR: The family of the "α-generators". We named them α-generators, because to define these generators, we use a primitive element α of the multiplicative group of a finite field. We show that these generators produce sequences of very large period and we find that they have very large linear complexity. Moreover, we find that they resist correlation attacks and algebraic attacks.

2010 Mathematics Subject Classification: 94A55, 11B83, 94A60, 11B80.

Keywords and phrases: random generator, LFSR, finite field, *m*-sequences.

Received May 1, 2013

1. Introduction

A linear feedback shift register (LFSR) allows the production of linear recurring sequences [5] having significant statistical properties for their application in cryptography. It is sufficient to exchange the initial state of the generator to produce the same output sequence on each side of the communication. However, these properties are not good enough to directly use LFSR for security goals. Indeed, the algorithm of Berlekamp-Massey ([1], [6]) makes possible to find all the initial parameters of LFSR from the knowledge of a small number of successive bits of an output sequence. In order to be able to use LFSR anyway, we have two methods which mask their linear structure by adding a component. These methods consist in either combining several LFSRs or to filter only one LFSR by a nonlinear function. For the combination method, one distinguishes: combination of generators by a nonlinear boolean function, summation generators [10], and generators by clock controlling (see [4], for example). Many attacks were proposed against these generators, in particular, correlation [12] and algebraic [2] attacks.

In this paper, we propose a new family of generators obtained by combination which we call the family of the α-generators. After a short recall on the LFSR, we recall some concepts on the finite fields $\mathbb{F}_{2^m}$, which will be useful for our construction. Then we give the description of an α-generator over a finite field $\mathbb{F}_{2^m}$, then we make a security analysis of these generators before concluding.

2. Linear Feedback Shift Register and *m*-Sequences

In this section, we recall briefly the description of the LFSR over the binary field $\mathbb{F}_2$. A linear feedback shift register (LFSR) consists in a collection of flip-flops (single-bit memories) set up in a linear fashion and clocked. The length k of the LFSR is determined by the number of flip-

flops. The initial value of the register is called *seed*. At each cycle of the circuit clock, the LFSR outputs a bit u_i. The output sequence $U = (u_i)_{i\in\mathbb{N}}$ of the LFSR is completely determined by the seed and satisfies a relation of the form

$$u_{i+k} = \sum_{n=1}^{k} f_n u_{i+k-n} \bmod 2, \tag{1}$$

where the sequence $(f_1, \ldots, f_k)$ is a binary vector with $f_k = 1$. Moreover, U is a periodic sequence of period to most equal to $2^k - 1$. If its period is equal to $2^k - 1$, then it is called a maximal length sequence, m-sequence for short, and the LFSR is known as maximal (see [5]).

The sequence $(f_1, \ldots, f_k)$ is related to the feedback polynomial defined by $f(x) = 1 \oplus f_1 x \oplus \cdots \oplus f_k x^k$. To produce a maximal LFSR which produces m-sequences, we must choose a primitive feedback polynomial (see [5, 8]). The linear complexity of a periodic sequence is the length of smallest LFSR being able to generate it.

The following result is an interesting property of the m-sequences:

Proposition 1 ([5, 8])**.** *If U is a sequence generated by a maximal LFSR of length k, then in every period of U, zeros occur $2^{k-1} - 1$ times and ones occur 2^{k-1} times.*

The m-sequences have many significant properties. However, one should not directly use such a sequence for cryptographic applications. One can retrieve its secret parameters (feedback polynomial and initial state of registers) thanks its linearity. Indeed, thanks the algorithm of Berlekamp-Massey [5], if a sequence has a small linear complexity k, then the knowledge of $2k$ consecutive bits of the sequence makes possible the discovery of the LFSR parameters of the length k, with only $O(k^2)$ binary operations.

Hence, this leads the idea to combine several distinct LFSRs or to filter only one LFSR by a nonlinear function, in order to produce sequences having a sufficient linear complexity to be out of range of the Berlekamp-Massey algorithm. In the family of the generator combinations, one distinguishes: combination of generators through a nonlinear boolean function, combination of generators with a function with memory and the generators with clock control. All generators with the clock control have an equivalent generator among the generators combined by a nonlinear boolean function. Hence, they are identical in terms of security. However, we will present these generators and propose, in the fourth section, a new approach to combine LFSRs.

2.1. Combination generator with nonlinear boolean function

A boolean function of n variables is a function with n binary inputs and one binary output.

A combination generator with nonlinear boolean function is a nonlinear boolean function f of n variables, which takes the outputs of parallel LFSRs as input. The combination generator outputs y_i computed from each LFSR output as shown in Figure 1(a), i.e.,

$$y_i = f(x_{1,i}, \ldots, x_{n,i}),$$

where $(x_{i,j})$, $1 \le i \le n$, is the output sequence of i-th LFSR. Let L_i be the length of the i-th LFSR. The period and linear complexity of the output sequence $Y = (y_i)_i$ of combination generator with nonlinear boolean function are as follows:

Theorem 1 ([5]). *The linear complexity of Y is $f(L_1, \ldots, L_n)$ evaluated over $\mathbb{Z}$. The period of Y is equal to* $\operatorname{lcm}(L_1, \ldots, L_n)$.

2.2. Summation generators (Rueppel's generator)

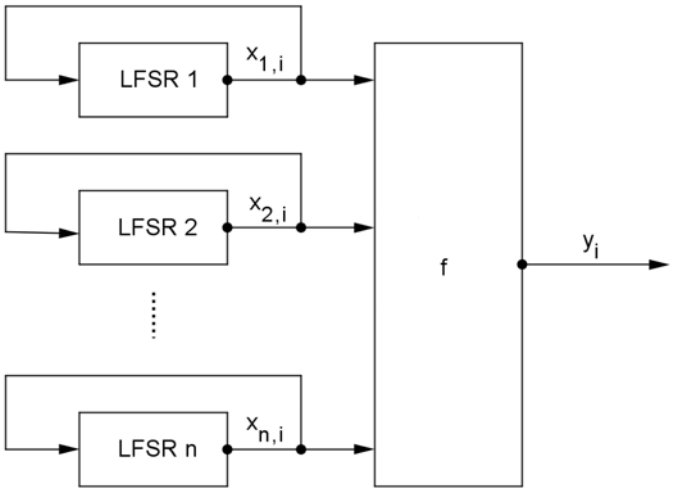
Rueppel's summation generator [10] (see Figure 1(b)) is composed of n maximal LFSRs and a memory m of $(\log_2(n)-1)$ bits. Let $(x_{i,j}), 1 \le i \le n,$ be the output sequence of LFSR i. The summation generator output sequence $Y = (y_i)_{i\ge 0}$ is defined by the following relation:

$$\text{for} \quad j \ge 0, \quad \begin{cases} y_j = x_{1,j} + x_{2,j} + \cdots + x_{n,j} + m_{j-1} \bmod 2, \\ m_j = x_{1,j} + x_{2,j} + \cdots + x_{n,j} + m_{j-1} \operatorname{div} 2, \end{cases} \tag{2}$$

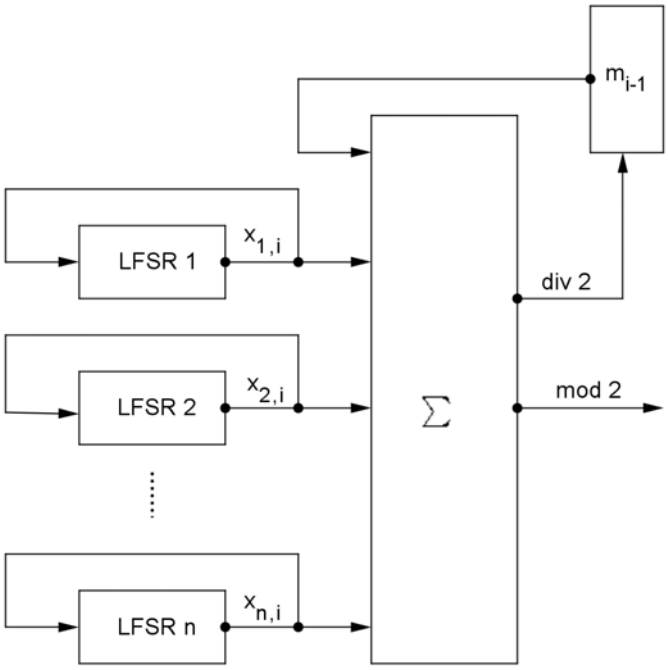
where "mod 2" and "div 2" denote, respectively, the remainder and quotient of integer division by 2.

The important property of the summation generator output sequence Y is the following result:

Theorem 2 ([10]). *The period of the sequence Y is* $\operatorname{lcm}_{1\le i\le n}(2^{L_i}-1)$ *and its linear complexity is approximately equal to this value.*



(a) Combination with nonlinear boolean function.



(b) Summation generator.

Figure 1. Classical methods of combination of LFSR.

3. Finite Field $\mathbb{F}_{2^m}$

Let m be a positive integer and $\mathbb{F}_{2^m}$ be the finite field with order 2^m. Let $f(x) = x^m \oplus f_{m-1}x^{m-1} \oplus \cdots \oplus f_0$ be a primitive polynomial over $\mathbb{F}_2$ and let α be a root of $f(x)$, i.e., $f(\alpha) = 0$. We have the following isomorphism:

$$\mathbb{F}_{2^m} = \mathbb{F}_2(\alpha) \approx \mathbb{F}_2[x] / (f(x)),$$

to represent $\mathbb{F}_{2^m}$.

To construct this structure, we need to choose α as a primitive element of the multiplicative group of $\mathbb{F}_{2^m}$, i.e., $\alpha^{2^m-1} = 1$, where $2^m - 1$ is the smallest positive integer, which satisfies this identity. In particular, every element a of $\mathbb{F}_{2^m}$ satisfies the relation $a^{2^m-1} = 1$. If $a \neq 0$, then $a = \alpha^k$ for some integer k $(1 \leq k \leq 2^m - 1)$.

Moreover, $\mathbb{F}_{2^m}$ is a vector space of dimension m over $\mathbb{F}_2$, whose the polynomial basis is $(1, \alpha, \alpha^2, \ldots, \alpha^{m-1})$. Any element a of $\mathbb{F}_{2^m}$ can be written uniquely in the form

$$a = a_{m-1}\alpha^{m-1} \oplus \cdots \oplus a_1\alpha \oplus a_0, \quad \text{where } a_i \in \mathbb{F}_2.$$

This form is called polynomial representation. We can also represent the same element a in binary notation by $a = \{a_{m-1}a_{m-2} \ldots a_0\}$.

For $a = \{a_{m-1}a_{m-2} \ldots a_0\} \in \mathbb{F}_{2^m}$, $\text{lsb}(a) = a_0$ and $\text{msb}(a) = a_{m-1}$ are, respectively, the least significant bit and most significant bit of a.

We will now describe the addition in $\mathbb{F}_{2^m}$ and the multiplication by α. These two operations are carried out in a very simple way. Indeed, the

addition (also denoted by $\oplus$) of two elements of $\mathbb{F}_{2^m}$ corresponds to an XOR between the coefficients in the polynomial representation of the two terms. Clearly, the addition corresponds with the simple bitwise XOR at the m-bit level.

The multiplication of $a \in \mathbb{F}_{2^m}$ by α corresponds to a simple left shift by one bit of its binary notation, followed by an XOR with the fixed mask $\{f_{m-1}f_{m-2}\cdots f_0\}$, if $\mathrm{msb}(a) = 1$.

4. The α-Generators

Let $m \geq 2$ be an integer and $\mathbb{F}_{2^m}$ be the finite field of order 2^m. Let us agree that α is a primitive element of $\mathbb{F}_{2^m}$.

In the rest of paper, n denotes the number of LFSRs (integer greater than 1) and we choose always maximal LFSRs (LFSRs which produce m-sequences).

Definition 1. An α-generator over $\mathbb{F}_{2^m}$ is composed of n maximal LFSRs and $(m+1)$-bit memory divided in a binary memory c and an m-bit memory $\beta \in \mathbb{F}_{2^m}$. Let $\gamma = \alpha^{m-1}$ and $x_k(x_{k,i})_{i\geq 0}$ be the output sequence of the k-th LFSR $(1 \leq k \leq n)$. Then, the output sequence $Y = (y_i)_{i\geq 0}$ of an α-generator is given by

$$\text{for} \quad i \geq 0, \quad \begin{cases} \beta_i = \beta_{i-1}\alpha^{x_{1,i}+x_{2,i}+\cdots+x_{n,i}}, \\ c_i = \mathrm{msb}(\beta_i \oplus \gamma^{c_{i-1}}), \\ y_i = \mathrm{lsb}(\beta_i \oplus \gamma^{c_{i-1}}), \end{cases} \tag{3}$$

with initializations of memory state: $c_{-1} \in \{0, 1\}$ and β_{-1} a nonzero power of α ($\beta_{-1} = \alpha^l$, where $l \in \{1, \ldots, 2^m - 2\}$).

The initial state of the generator thus defined is the concatenation of the initial states of n LFSRs and the $m+1$ bits of the initial memory (the bits of β_{-1} and c_{-1}).

The Figure 2 represents an α-generator over $\mathbb{F}_{2^m}$.

Note that $\alpha^{x_{1,i}+x_{2,i}+\cdots+x_{n,i}} = \alpha^{x_{1,i}}\alpha^{x_{2,i}}\cdots\alpha^{x_{n,i}}$, so we can compute $\beta_{i-1}\alpha^{x_{1,i}}\alpha^{x_{2,i}}\cdots\alpha^{x_{n,i}}$ as follows:

$$\beta_{i-1}\alpha^{x_{1,i}}\alpha^{x_{2,i}}\cdots\alpha^{x_{n,i}} = \alpha^{x_{n,i}}(\cdots(\alpha^{x_{2,i}}(\alpha^{x_{1,i}}\beta_{i-1}))).$$

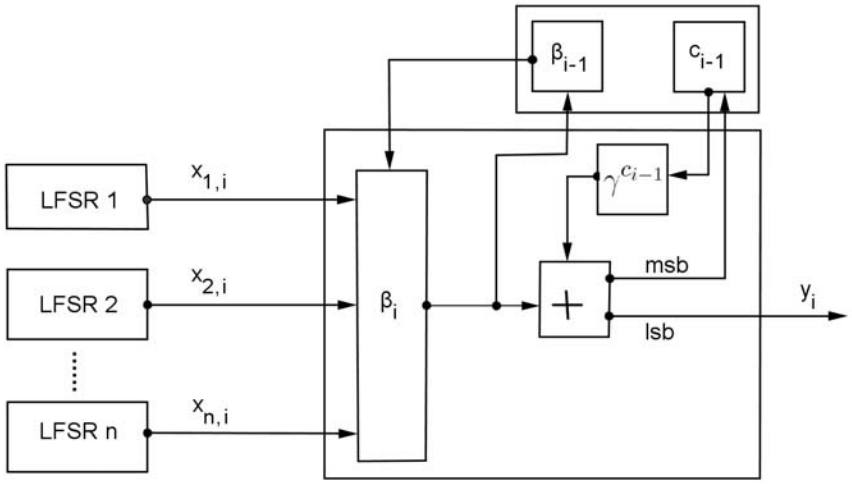


Figure 2. α-generator over $\mathbb{F}_{2^m}$.

This implementation performs a series of not more than n multiplications by α. In binary notation, this can be done by a series of left shift eventually followed by a bitwise XOR with a fixed mask (at most n times).

It should be noted also that the proposed choice of γ allows that the expression $\gamma^{c_{i-1}}$ be equal to $\{0\ldots01\}$, if $c_{i-1}=0$ or $\{10\ldots0\}$, if $c_{i-1}=1$ in binary notation.

Define as usual

$$N(q,\ m) = \frac{\varphi(q^m - 1)}{m},$$

where φ is the Euler's totient function ([5]).

Let us notice that, from n given LFSRs, one can construct $N(2,\ m)$ α-generators on $\mathbb{F}_{2^m}$ using these n LFSRs, where $N(2,\ m)$ is the number of primitive polynomial of degree m over $\mathbb{F}_2[x]$.

More precisely, to construct a representation of the field $\mathbb{F}_{2^m}$, we have to choose a primitive polynomial of degree m on $\mathbb{F}_2[x]$. The coefficients of this polynomial are involved in the implementation of an α-generator, because they are used in the multiplication of an $\mathbb{F}_{2^m}$-element by one of its primitive elements. So, given n LFSRs, each choice of a primitive polynomial of degree m allows the construction of an α-generator over $\mathbb{F}_{2^m}$ based on these n LFSRs.

4.1. Period and linear complexity

In the rest of paper, the notation per(s) denotes the period of the periodic sequence s.

Lemma 1. *Let $L_1, \ldots, L_n$ be the lengths of n LFSRs of an α-generator over $\mathbb{F}_{2^m}$. Let*

$$h_i = \frac{\mathrm{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)}{2^{L_i} - 1}, \quad \sigma = \sum_{i=1}^{n} h_i 2^{L_i - 1},$$

and

$$\tau = \min_{\substack{3 \le \mu \le 2^m - 1 \\ \gcd(\mu,\, 2^m - 1) \ne 1}} \{\mu \mid 2^m - 1 \textit{ divides } \mu\sigma\}.$$

Then the sequence $(\beta_i)_{i\geq 0}$ *of m-bit* β *memory of* α*-generator is a periodic sequence and its period satisfies one of the two following assertions*:

(1) *if* $2^m - 1$ *divides* σ, *then* $per(\beta) = \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)$;

(2) *if* $2^m - 1$ *does not divides* σ, *then* $per(\beta) = \tau \times \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)$.

Proof. The period of the sequence $(x_{1,i} + \cdots + x_{n,i})_{i\geq 0}$ is

$$T = \text{lcm}(\text{per}(x_1), \ldots, \text{per}(x_1)) = \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1).$$

In addition, according to Proposition 1, we have

$$\sum_{j=0}^{2^{L_i}-2} x_{i,j} = 2^{L_i - 1}, \quad \text{for any } i \in \{1, \ldots, n\}.$$

Let $l \in \{1, \ldots, m-1\}$ and $\beta_{-1} = \alpha^l$. First, we give the general expression of the sequence β. We have

$$\beta_0 = \alpha^l \alpha^{x_{1,0}+\cdots+x_{n,0}} = \alpha^{(x_{1,0}+\cdots+x_{n,0})+l};$$

$$\beta_1 = \beta_0 \alpha^{x_{1,0}+\cdots+x_{n,0}} = \alpha^{(x_{1,0}+\cdots+x_{n,0})+(x_{1,1}+\cdots+x_{n,1})+l};$$

$$\vdots$$

$$\beta_i = \beta_{i-1} \alpha^{x_{1,0}+\cdots+x_{n,0}} = \alpha^{(x_{1,0}+\cdots+x_{n,0})+(x_{1,1}+\cdots+x_{n,1})+\cdots+(x_{1,i}+\cdots+x_{n,i})+l}.$$

Thus for any $i \geq 0$, $\beta_i = \alpha^{\sum_{j=0}^{i}(x_{1,j}+\cdots+x_{n,j})+l}$. Hence, the β_T term can be expressed as follows:

$$\beta_T = \alpha^{\sum_{j=0}^{T}(x_{1,j}+\cdots+x_{n,j})+l}$$

$$= \alpha^{\sum_{j=0}^{T-1}(x_{1,j}+\cdots+x_{n,j})+(x_{1,p}+\cdots+x_{n,p})+l}.$$

The calculation of the exponent of α in the expression β_T gives

$$\begin{aligned}
&\sum_{j=0}^{T-1}(x_{1,j}+\cdots+x_{n,j})+(x_{1,0}+\cdots+x_{n,0})+l\\
&\quad=\sum_{i=1}^{n}\left(\sum_{j=0}^{T-1}x_{i,j}\right)+(x_{1,0}+\cdots+x_{n,0})+l\\
&\quad=\sum_{i=1}^{n}\left(\sum_{j=0}^{h_i\mathrm{per}(x_i)-1}x_{i,j}\right)+(x_{1,0}+\cdots+x_{n,0})+l\\
&\quad=\sum_{i=1}^{n}\left(h_i\sum_{j=0}^{\mathrm{per}(x_i)-1}x_{i,j}\right)+(x_{1,0}+\cdots+x_{n,0})+l\\
&\quad=\sum_{i=1}^{n}h_i2^{L_i-1}+(x_{1,0}+\cdots+x_{n,0})+l\\
&\quad=\sigma+(x_{1,0}+\cdots+x_{n,0})+l.
\end{aligned}$$

Thus, we have $\beta_T=\alpha^{\sigma+(x_{1,0}+\cdots+x_{n,0})+l}$.

Now, let us proceed by case analysis.

- If $\sigma=k(2^m-1)$ with $k\in\mathbb{N}-\{0\}$, then we have

$$\begin{aligned}
\beta_T&=\alpha^{\sigma+(x_{1,0}+\cdots+x_{n,0})+l}\\
&=\alpha^{k(2^m-1)}\alpha^{(x_{1,0}+\cdots+x_{n,0})+l}\\
&=\alpha^{(x_{1,0}+\cdots+x_{n,0})+l}=\beta_0.
\end{aligned}$$

This implies that $\beta_{T+i}=\beta_i$ for any $i\geq 0$, because

$$x_{1,T+i}+\cdots+x_{n,T+i}=x_{1,i}+\cdots+x_{n,i},\quad \text{for any}\quad i\geq 0.$$

Therefore, we obtain $\text{per}(\beta) = T$ if $2^m - 1$ divides σ.

- Now, let us suppose that $2^m - 1$ does not divide σ. Continuing to calculate the terms of the sequence β after the term β_T, one can easily see that

$$\text{for any } k \geq 1, \quad \beta_{kT} = \alpha^{k\sigma + (x_{1,0} + \cdots + x_{n,0}) + l}.$$

This implies that the period of β is necessarily equal to τT. Indeed, as $2^m - 1$ divides $\tau\sigma$, then $\alpha^{\tau\sigma} = 1$, therefore, we have

$$\begin{aligned} \beta_{\tau T} &= \alpha^{\tau\sigma + (x_{1,0} + \cdots + x_{n,0}) + l} \\ &= (\alpha^{\tau\sigma})\alpha^{(x_{1,0} + \cdots + x_{n,0}) + l} \\ &= \alpha^{(x_{1,0} + \cdots + x_{n,0}) + l} = \beta_0. \end{aligned}$$

We can deduce that $\beta_{\tau T + i} = \beta_i$ for any $i \geq 0$, because

$$x_{1,\tau T+i} + \cdots + x_{n,\tau T+i} = x_{1,i} + \cdots + x_{n,i}, \quad \text{for any} \quad i \geq 0.$$

In this case, the period of the sequence β is equal to

$$\tau \times \text{lcm}(\text{per}(x_1), \cdots, \text{per}(x_n)).$$

Lemma 2. *The sequence $c = (c_i)_{i \geq 0}$ of binary memory of an α-generator over $\mathbb{F}_{2^m}$ is a periodic sequence of period equal to that of the m-bit memory sequence β.*

Proof. By definition, we have $c_i = \text{msb}(\beta_i \oplus \gamma^{c_{i-1}})$. The expression $\gamma^{c_{i-1}}$ can be written in the form $\gamma^{c_{i-1}} = c_{i-1}\gamma \oplus \bar{c}_{i-1}$. Replacing β_i by $b_{m-1,i}\alpha^{m-1} \oplus b_{m-2,i}\alpha^{m-1} \oplus \cdots \oplus b_{0,i}$, we have for $i \geq 0$,

$$\begin{aligned} c_i &= \text{msb}((b_{m-1,i}\alpha^{m-1} \oplus b_{m-2,i}\alpha^{m-1} \oplus \cdots \oplus b_{0,i}) \oplus (c_{i-1}\gamma \oplus \bar{c}_{i-1})) \\ &= b_{m-1,i} \oplus c_{i-1}. \end{aligned}$$

Since β is a periodic sequence, $(b_{m-1,i})_{i\geq 0}$ is also periodic. We deduce that the sequence c is periodic and has the same period as β.

Now, we focus on the periodicity of a output sequence of α-generator over $\mathbb{F}_{2^m}$. The following result follows from Lemmas 1 and 2.

Theorem 3. *Let $L_1, \ldots, L_n$ be the lengths of all n LFSRs of an α-generator over $\mathbb{F}_{2^m}$. Let*

$$h_i = \frac{\mathrm{lcm}(2^{L_1}-1, \cdots, 2^{L_n}-1)}{2^{L_i}-1}, \quad \sigma = \sum_{i=1}^{n} h_i 2^{L_i-1},$$

and

$$\tau = \min_{\substack{3\leq \mu \leq 2^m-1 \\ \gcd(\mu, 2^m-1)\neq 1}} \{\mu \mid 2^m-1 \text{ divides } \mu\sigma\}.$$

An output sequence Y of the α-generator over $\mathbb{F}_{2^m}$ is periodic sequence and its period is given by

$$per(Y) = \begin{cases} \mathrm{lcm}(2^{L_1}-1, \cdots, 2^{L_n}-1), & \text{if } 2^m-1 \text{ divides } \sigma, \\ \tau \times \mathrm{lcm}(2^{L_1}-1, \cdots, 2^{L_n}-1), & \text{if } 2^m-1 \text{ does not divides } \sigma. \end{cases} \tag{4}$$

Proof. Since the sequence of memory β and c are periodic with common period, and $y_i = \mathrm{lsb}(\beta_i \oplus \gamma^{c_i-1})$, then the periodicity of the sequence Y follows. In addition, the period is the same as the memory ones.

If 2^m-1 is prime, then we can refine the conditions for the determination of per(Y). To this end, we will first recall a classical result (Lemma 3).

Lemma 3. *Let r and s be two positive integers with $r < s$. Then 2^r-1 divides 2^s-1, if and only if r divides s.*

Corollary 1. *Let m be a positive integer such that $2^m - 1$ is prime. Let $L_1, \ldots, L_n$ be the lengths of n LFSRs of an α-generator over $\mathbb{F}_{2^m}$. Then the period of an output sequence Y from the α-generator satisfies*

$$per(Y) = \begin{cases} \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1), & \text{if } m \mid L_i \text{ for any } 1 \le i \le n, \\ (2^m - 1) \times \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1), & \text{if } \exists i, \ m \nmid L_i. \end{cases}$$

Proof. Let

$$h_i = \frac{\text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)}{2^{L_i} - 1} \quad \text{and} \quad \sigma = \sum_{i=1}^{n} h_i 2^{L_i - 1}.$$

Since $2^m - 1$ is prime, then $\gcd(\mu, 2^m - 1) = 1$ for any positive integer μ less than $2^m - 1$. By Theorem 3, we have $\text{per}(Y) = \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)$, if $2^m - 1$ divides σ, otherwise $\text{per}(Y) = (2^m - 1) \times \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)$. But, $2^m - 1$ divides σ if and only if it divides h_i for all $i \in \{1, \ldots, n\}$. This is equivalent, by Lemma 3, to m divides L_i for all $i \in \{1, \ldots, n\}$.

Example 1. Consider two LFSRs whose characteristics are:

- LFSR1: length $L_1 = 2$, feedback polynomial $= X^2 + X + 1$, generates x_1, $\text{per}(x_1) = 3$.

- LFSR2: length $L_2 = 3$, feedback polynomial $= X^3 + X^2 + 1$, generates x_2, $\text{per}(x_2) = 7$.

We are in the case $\text{lcm}(\text{per}(x_1), \text{per}(x_2)) = 21$, $h_1 = 7$, $h_2 = 3$, and $\sigma = 7 \times 2 + 3 \times 4 = 26$.

The period of the α-generator based on these two LFSRs will be discuss according to the choice of the finite field:

(1) Let $\mathbb{F}_{2^2} = \mathbb{F}_4$ be the finite field with 4 elements and $\alpha \in \mathbb{F}_4$ be a primitive element. We have gcd(3, 26) = 1. Then, the period of the output sequence from the α-generator over $\mathbb{F}_4$ is 63 for a given nonzero initialization of the LFSRs.

(2) Let $\mathbb{F}_{2^3} = \mathbb{F}_8$ be the finite field with 8 elements and $\alpha \in \mathbb{F}_8$ be a primitive element. Here $2^m - 1 = 7$. For the α-generator over $\mathbb{F}_8$ formed by the two above LFSRs, we have $\gcd(\sigma, 7) = 1$. So, for a given nonzero initialization of the LFSR, this α-generator over $\mathbb{F}_8$ produces sequences of period $7 \times 21 = 147$.

(3) Let $\mathbb{F}_{2^8} = \mathbb{F}_{256}$ be the finite field of order 256 and $\alpha \in \mathbb{F}_{256}$ be a primitive element. Since 255 does not divide σ, then the α-generator over $\mathbb{F}_{256}$ that combines the above two LFSRs produces sequences of period $255 \times 21 = 5355$.

Remark 1. For the periodicity of a sequence Y produced by an α-generator over $\mathbb{F}_{2^m}$, there are two cases in terms of length. The first case is when all the n LFSRs have the same length. The second case is when the lengths of all LFSRs are relatively prime. For both cases, we have the following result:

Theorem 4. *Let $L_1, \ldots, L_n$ be the lengths of n LFSRs of an α-generator over $\mathbb{F}_{2^m}$. Let Y be an output sequence from this α-generator. We have the following assertions*:

(1) *If* $\gcd(L_1, \ldots, L_n) = 1$, *then*

$$per(Y) = (2^m - 1) \times \operatorname{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1).$$

(2) *If* $L_1 = \ldots = L_n = L$, *then by putting*

$$\tau = \min_{1 \le \mu \le 2^m - 1} \{\mu \mid 2^m - 1 \text{ divides } \mu n\},$$

we have

$$per(Y) = \tau(2^L - 1).$$

Proof. Let

$$h_i = \frac{\text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)}{2^{L_i} - 1} \quad \text{and} \quad \sigma = \sum_{i=1}^{n} h_i 2^{L_{i-1}}.$$

(1) If $\gcd(L_1, \ldots, L_n) = 1$, then

$$\gcd(2^{L_1} - 1, \cdots, 2^{L_n} - 1) = 1, \qquad h_i = \prod_{j=1,\, j\neq i}^{n} (2^{L_j} - 1),$$

$$\text{and} \quad \gcd(h_1, \ldots, h_n) = 1,$$

then, by Theorem 3, we have $\text{per}(Y) = \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)$, if $2^m - 1$ divides σ. This is equivalent to say that $2^m - 1$ divides h_i for any $i \in \{1, \ldots, n\}$. This is possible if and only if m divides L_i for any $i \in \{1, \ldots, n\}$ (Lemma 3). Since $\gcd(L_1, \ldots, L_n) = 1$, we have thus, $\text{per}(Y) > \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1)$.

Now, let $2^m - 1 = p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k}$ as its decomposition into prime factors greater than 2. Let $P_i = (2^m - 1) / p_i^{r_i} = \prod_{j=1,\, j\neq i}^{k} p_j^{r_j}$.

For $i = 1$ to n, we have

$$P_i\sigma = \sum_{t=1}^{n} \prod_{j=1,\ j\neq i}^{k} p_j^{r_j} h_t 2^{L_{t-1}}.$$

Since $\gcd(h_1, \ldots, h_n) = 1$, then there exists a term of the sum in which the p_i factor does not appear. Thus, the product $P_i\sigma$ is not divisible by $2^m - 1$. Consequently, the period of a sequence Y at the output of α-generator is equal to

$$(2^m - 1) \times \text{lcm}(2^{L_1} - 1, \cdots, 2^{L_n} - 1).$$

(2) If $L_1 = \ldots = L_n = L$, then $h_1 = \ldots = h_n = 1$ and $\sigma = n(2^{L-1})$. So, the divisibility of σ by $2^m - 1$ is equivalent to the divisibility of n by $2^m - 1$ because $\gcd(2^{L-1}, 2^m - 1) = 1$. Therefore, we have $\text{per}(Y) = \tau \times (2^L - 1)$.

Now, let us study the linear complexity of an α-generator over $\mathbb{F}_{2^m}$. It is not unreasonable to continue to talk of linear complexity of an α-generator over $\mathbb{F}_{2^m}$, although its combination function uses exponentiation in the multiplicative group $\mathbb{F}_{2^m}^*$, which is far from being a linear function. Since it is periodic and that any periodic sequence is considered linear recurring, it would be interesting to get an idea of its linear complexity. To this purpose, we conducted experiments whose results are given in Tables 1, 2, and 3. Since, α-generators over $\mathbb{F}_{2^m}$ have very great period, we considered those who consist of LFSR of small length L_i. To conduct these simulation examples of linear complexity (LC), we choose $m \in \{3, 4, 5, 6, 7, 8\}$ and $L_i \in \{2, 3, 4, 5, 6\}$. We denoted per and LCM, respectively, the period of an α-generator and the least common multiple of LFSRs periods component it. Based on these experiences, we formulate the following result we could not give a theoretical demonstration for now.

Theorem 5. *Let $L_1, \ldots, L_n$ be the lengths of n LFSRs of an α-generator over $\mathbb{F}_{2^m}$. Let Y be an output sequence from this α-generator. The linear complexity LC of Y satisfies*

$$\text{lcm}(2^{L_1} - 1, \ldots, 2^{L_n} - 1) \leq LC \leq m \times \text{lcm}(2^{L_1} - 1, \ldots, 2^{L_n} - 1) + 1.$$

Table 1.

Basis field	(L_1, L_2)								
	(3, 4)			(3, 5)			(3, 6)		
	LCM	per	LC	LCM	per	LC	LCM	per	LC
$\mathbb{F}_{2^3}$	105	735	315	217	1519	631	63	441	190
$\mathbb{F}_{2^4}$	105	1575	420	217	3255	840	63	315	158
$\mathbb{F}_{2^5}$	105	3255	526	217	6727	1086	63	1953	316
$\mathbb{F}_{2^6}$	105	6615	631	217	13671	1302	63	1323	378
$\mathbb{F}_{2^7}$	105	13335	735	217	27559	1520	63	8001	441
$\mathbb{F}_{2^8}$	105	26775	841	217	55335	1736	63	5355	126

Comparison of complexity between generators: Table 4 provides a comparison of periods and linear complexities of α-generator over $\mathbb{F}_{2^m}$, generators by combining boolean function and summation generators, using n maximal LFSRs with lengths L_1, L_2, …, and L_n, where LCM = $\text{lcm}(2^{L_1} - 1, \ldots, 2^{L_n} - 1)$ and LC is the linear complexity.

4.2. Statistical tests

In order to test pseudo-random property of sequences generated by α-generators, we have taken an example of α-generators using LFSRs whose characteristics are:

Table 2.

Basis field	(L_1, L_2)					
	(4, 5)			(4, 6)		
	LCM	per	LC	LCM	Per	LC
$\mathbb{F}_{2^3}$	465	3255	1350	315	2205	946
$\mathbb{F}_{2^4}$	465	6975	1801	315	1575	787
$\mathbb{F}_{2^5}$	465	14415	2325	315	9765	1575
$\mathbb{F}_{2^6}$	465	29295	2791	315	6615	1891
$\mathbb{F}_{2^7}$	465	59055	3256	315	40005	2206
$\mathbb{F}_{2^8}$	465	118575	3721	315	26775	2521

Table 3.

Basis field	(L_1, L_2, L_3)					
	(2, 3, 5)			(2, 3, 6)		
	LCM	per	LC	LCM	per	LC
$\mathbb{F}_{2^3}$	651	4557	1891	63	441	189
$\mathbb{F}_{2^4}$	651	9765	2520	63	189	94
$\mathbb{F}_{2^5}$	651	20181	3255	63	1953	315
$\mathbb{F}_{2^6}$	651	39711	3907	63	3969	379
$\mathbb{F}_{2^7}$	651	82677	4557	63	8001	441
$\mathbb{F}_{2^8}$	651	166005	5209	63	3213	505

Table 4. Comparison of period and complexity

Generator	Period	Linear complexity
Combining boolean function	LCM	$LC \ll LCM$
Combining by summation	LCM	$LC \approx LCM$
α-generator	$\tau \times LCM$ with $1 \leq \tau \leq 2^m - 1$	$LCM \leq LC \leq m \times LCM + 1$

– LFSR1: length $L_1 = 11$, feedback polynomial $= X^{11} + X^8 + X^5 + X^2 + 1$,

– LFSR2: length $L_2 = 13$, feedback polynomial $= X^{13} + X^9 + X^7 + X^2 + 1$,

– LFSR3: length $L_3 = 17$, feedback polynomial $= X^{17} + X^{12} + X^8 + X^4 + 1$,

– LFSR4: length $L_4 = 19$, feedback polynomial $= X^{19} + X^{13} + X^9 + X^4 + 1$.

In order to construct these generators, we have the following combination of LFSRs over $\mathbb{F}_{2^m}$ for $m \in 16, 32$: (LFSR1, LFSR2, LFSR3, and LFSR4).

The statistical tests of these α-generators were performed by using the NIST Test Suite [9]. For each selected generator, we sampled 100 sequences of size 10^6. For each statistical test, a set of p-values (corresponding to the set of sequences) is produced. If p-value ≥ 0.01, the tested sequence is considered to be random. For each statistical test, the proportion of sequences that pass the test is computed. The expected proportion of sequences that pass a test is equal to $96/100$. The results of these tests, given in Table 5, show that generators produce sequences of very good quality statistics.

Table 5. NIST statistical tests

	(LFSR1, LFSR2, LFSR3, LFSR4)	
	On $\mathbb{F}_{2^{16}}$	On $\mathbb{F}_{2^{32}}$
Test	Proportion	Proportion
Frequency	99/100	99/100
Block Frequency	100/100	100/100
Cumulative Sums	100/100	98/100
Cumulative Sums	99/100	100/100
Runs	100/100	99/100
Longest Run	98/100	98/100
Rank	100/100	98/100
FFT	98/100	100/100
Maurer	100/100	99/100
Approximate Entropy	95/100	96/100
Serial	100/100	100/100
Serial	98/100	100/100
Linear Complexity	100/100	100/100

4.3. Security analysis of α-generators

For our security analysis, only the initialization of the generator is not known, all other parameters of the system are known. An attack is aimed at recovering the initial state of a generator from several output terms read after the end of the initialization phase. These attacks attempt to exploit either the algebraic structure of the generator or its statistical quality. First, we try to exploit the structure of our generator to provide a security argument. Then, we focus on algebraic attacks and correlation attacks.

An argument of security

In this part, we examine the possibility to find a breach of security through the exponentiation method used by our generator.

Consider the expression of an output sequence Y from an α-generator over $\mathbb{F}_{2^m}$. We have

$$\text{for any} \quad i \geq 0, \quad y_i = \text{lsb}(\beta_i \oplus \gamma^{c_{i-1}}),$$

where $\beta_i = \beta_{i-1}\alpha^{x_{1,i}+\cdots+x_{n,i}}$. This expression of β_i can be written as $\beta_i = \alpha^{z_i}\alpha^{x_i}$, with $z_i \in \{0, 1, \ldots, 2^m - 2\}$ and $x_i = x_{1,i} + \cdots + x_{n,i} \in \{0, 1, \cdots, n\}$ (the sum is performed in $\mathbb{Z}$), i.e., x_i is the integer sum of the i-th output of LFSR. Thus, $\beta_i = \alpha^{t_i}$, with $t_i = x_i + z_i \in \{0, 1, \cdots, 2^m + n - 2\}$. Therefore, the calculation of β_i consists in calculating powers of α, where successive exponents t_i are not ordered.

Since we cannot predict the state of a LFSR at a clock cycle (one iteration) without knowing at least one of its preceding states, its output becomes unpredictable. Thus, it is the same for the sequence (x_i). Therefore, we cannot predict the exponents t_i. As a result, it is impossible to predict the sequences β and c (c being related to β).

Since Y is a function of the sequences β and c, and thus with the argument above, it is impossible to predict a term of the output sequence of the α-generator, by trying to exploit the fact that α-generator uses the exponentiation in the group $\mathbb{F}_{2^m}^*$.

Algebraic attacks

Assume that we have k bits $y_0, \ldots, y_{k-1}$ produced by an α-generator over $\mathbb{F}_{2^m}$ from a given initialization.

Denote by $(x_{1,i}, \ldots, x_{n,i})$ the n-tuple representing the i-th output from n LFSRs. We denote the function that takes $x_{1,i}, \ldots, x_{n,i}, \beta_{i-1}$ and c_{i-1} to give the i-th output y_i from the α-generator by ω. Then, we must try to form the following system:

$$y_i = \omega(x_{1,i}, \ldots, x_{n,i}, \beta_{i-1}, c_{i-1}), \quad 0 \le i \le k-1,$$

where the unknowns are the bits of the initializations of all the n LFSRs and those of the initial memory (bits β_{-1} and c_{-1}).

The ability to form and solve this system depends on the function ω. An improved variant of this basis algebraic attack can be found in [2]. According to the relation (3), we have

$$y_i = \mathrm{lsb}(\beta_{i-1}\alpha^{x_{1,i}+\cdots+x_{n,i}} \oplus \gamma^{c_{i-1}}) \quad \text{for} \quad 0 \le i \le k-1.$$

So, there are $(m-2)$ unknown elements $a_{1,i}, \ldots, a_{m-2,i}$ of $\mathbb{F}_2$ such that

$$\beta_{i-1}\alpha^{x_{1,i}+\cdots+x_{n,i}} \oplus \gamma^{c_{i-1}} = c_i\alpha^{m-1} \oplus a_{m-2,i}\alpha^{m-2} \oplus \cdots \oplus a_{2,i}\alpha^2 \oplus a_{1,i}\alpha \oplus y_i.$$

This is equivalent to

$$\beta_{i-1}\alpha^{x_{1,i}+\cdots+x_{n,i}} \oplus \gamma^{c_{i-1}} \oplus c_i\gamma \oplus a_{m-2,i}\alpha^{m-2} \oplus \cdots \oplus a_{2,i}\alpha^2 \oplus a_{1,i}\alpha = y_i.$$

Let $l \in \{1, \ldots, m-1\}$ and $\beta_{-1} = \alpha^l$.

So using the fact that $\beta_{i-1}\alpha^{x_{1,i}+\cdots+x_{n,i}} = \alpha^{\sum_{j=0}^{i}(x_{1,j}+\cdots+x_{n,j})+l}$, then the above system becomes: For $0 \le i \le k-1$,

$$\alpha^{\sum_{j=0}^{i}(x_{1,j}+\cdots+x_{n,j})+l} \oplus \gamma^{c_{i-1}} \oplus c_i\gamma \oplus a_{m-2,i}\alpha^{m-2} \oplus \cdots \oplus a_{2,i}\alpha^2 \oplus a_{1,i}\alpha = y_i. \tag{5}$$

We do not see how can we exploit the system (5) to find the bits of the initial states of LFSRs and the bits of initial memory (c_{-1} and β_{-1}).

Now, let us look if we can form a polynomial system using the fact that α^a, where a is a bit, can be written as $a\alpha \oplus \overline{a}$.

We can write $\beta_i \oplus \gamma^{c_{i-1}}$ as

$$\beta_i \oplus \gamma^{c_{i-1}} = \alpha^{\sum_{j=0}^{i}(x_{1,j}+\cdots+x_{n,j})+l} \oplus \gamma^{c_{i-1}}$$

$$= \alpha^l \prod_{j=0}^{i}\prod_{r=1}^{n}(x_{r,j}\alpha \oplus \overline{x}_{r,j}) \oplus c_{i-1}\gamma \oplus \overline{c}_{i-1}. \quad (6)$$

Considering relation (6), we see that if we try to form a system of polynomial equations with unknowns, the bits of the initial states of the n LFSRs and initial memory (bits of β_{-1} and c_{-1}), then the degree of the t-th equation is $t(n+1)$. It quickly becomes infeasible to obtain such equations. Moreover, the complexity of techniques for solving nonlinear systems increases exponentially with the degree of the equations. We also see that, after the initial state of the generator, the degree of the bits of the memory (m-bit memory and binary memory) expressed as function of bits of the previous state is $n+1$. Because of this, the attack in [2] may not apply.

Thus, we see that the output sequence Y from the α-generator has an algebraic description more difficult to exploit. With these arguments, we believe that the α-generator is resistant to algebraic attacks.

In addition, based on the observation that the degree of the t-th output from the α-generator is $t(n+1)$ (so the degree quickly increases), cubes attacks described in [3] by Itai Dinur and Adi Shamir are impracticable against the α-generator.

Correlation attacks

The original correlation attack is due to Siegenthaler [12, 11]. He brought against generators based on a combination of LFSR with a nonlinear boolean function f (it applies to generators of a filtered LFSR).

It consists to consider the correlation between the output of the function f with n variables and t fixed elements of its inputs with $t < n$. If by fixing these t elements of the entries of f changes the distribution of the output of f, then we can plan a correlation attack. However, if f remains balanced, then there is no correlation of order t (uncorrelated with respect to its t terms). Clearly, the aim is to find a linear function g of $\mathbb{F}_2^t$ into $\mathbb{F}_2$ such that

$$\text{Prob}(\{(z_1, \ldots, z_n) \in \mathbb{F}_2^n \,/\, f(z_1, \ldots, z_n) = g(z_{i_1}, \ldots, i_t)\}) > \frac{1}{2}.$$

If this linear function exists, then we can consider conducting a correlation attack by an exhaustive search on the t fixed inputs. For a large value of t, this attack becomes ineffective because the complexity of the exhaustive search will be very large. Subsequently, a variation of this attack, called fast correlation attack, is proposed by Meier and Staffelbach [7]. This alternative approach uses decoding techniques instead of exhaustive search, which reduces the complexity of the attack.

In this part, we will analyze the feasibility of conducting a correlation attack on an α-generator over $\mathbb{F}_{2^m}$.

Let δ and ϑ be the exponential and summation function, respectively, defined by

$$\delta : \{0, 1, \ldots, 2^m + n - 2\} \to \mathbb{F}_{2^m}^*,$$

$$z \mapsto \alpha^z,$$

$$\vartheta : \{0, 1\}^n \times \{0, 1, \ldots, 2^m - 2\} \to \{0, 1, \ldots, 2^m + n - 2\},$$

$$(x_{1,i}, \cdots, x_{n,i}, \lambda_{i-1}) \mapsto x_{1,i} + \cdots + x_{n,i} + \lambda_{i-1}.$$

The computation of $\beta_i = \beta_{i-1}\alpha^{x_{1,i}+x_{2,i}+\cdots+x_{n,i}} = \alpha^{x_{1,i}+x_{2,i}+\cdots+x_{n,i}+\lambda_{i-1}}$, with $\alpha^{\lambda_{i-1}} = \beta_{i-1}$, corresponds to calculate the image of the vector $(x_{1,i}, x_{2,i}, \cdots, x_{n,i}, \lambda_{i-1})$ by the composite function $\delta \circ \vartheta$.

The function ϑ is nonlinear at the binary level and has the additional property that fixing some of its inputs that does not change in the distribution of its outputs. In fact, the function ϑ is n-resilient and is also an exponential function, then there is no linear relation between the bits of the binary notation of $\beta_i = \delta(\vartheta(x_{1,i}, \cdots, x_{n,i}, \lambda_{i-1}))$ and $x_{j,i}$, where $1 \leq j \leq n$. Furthermore, we notice that the result of the addition of $\gamma^{c_{i-1}} = \overline{c}_{i-1} \oplus \gamma^{c_{i-1}}$ to β_i occurs in y_i. Then y_i became a linear function of c_{i-1} and the most significant bit of the binary representation of β_i. This cannot be used to perform an attack.

With the above arguments, we conclude that the α-generators resist to correlation attacks.

5. Implementation Properties

In this section, we give implementation properties of α-generators.

An α-generator over $\mathbb{F}_{2^m}$ based on n LFSRs use the multiplication by α^k, with $0 \leq k \leq n$, in the polynomial basis. Therefore, we must build a circuit capable of multiplication by α in the polynomial basis. It is well known that a Galois LFSR of length m can do this. So to perform a multiplication by α^k, we must perform k cycles with this Galois LFSR. Thus, the total gate count for different parts of the generator is approximately the number of gates required to implement $n+1$ LFSR (with an LFSR of length m).

6. Conclusion

In this paper, we introduced a new family of generators by combining LFSRs, which we called the family of the α-generators. These generators produce sequences of very large periods and rather high linear complexity. They resist both to correlation and algebraic attacks. Moreover, they can be implemented easily by using simple operations such as “exclusive or” and “shift”.

References

[1] R. Elwyn Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

[2] Nicolas Courtois, Fast algebraic attacks on stream ciphers with linear feedback, CRYPTO (2003), 176-194.

[3] Itai Dinur and Adi Shamir, Cube Attacks on Tweakable Black Box Polynomials, IACR Cryptology ePrint Archive, 2008:385, 2008.

[4] C. G. Günther, Alternating Step Generators Controlled by de Bruijn Sequences, In Proceedings of the 6th Annual International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'87, pages 5-14, Berlin, Heidelberg, Springer-Verlag, 1988.

[5] R. Lidl and H. Niederreiter, Finite Fields Encyclopedia of Mathematics and its Applications, Addison-Wesley Publ. Co., Advanced Book Program/World Science Division, 1983.

[6] James L. Massey, Shift-register synthesis and BCH decoding, IEEE Transactions on Information Theory 15 (1969), 122-127.

[7] Willi Meier and Othmar Staffelbach, Fast Correlation Attacks on Stream Ciphers (extended abstract), In C. G. Günther, editor, EUROCRYPT, Volume 330 of Lecture Notes in Computer Science, pages 301-314, Springer, 1988.

[8] H. Niederreiter, The multiple-recursive matrix method for pseudo random number generation, Finite Fields and their Applications 1 (1995), 3-30.

[9] NIST, A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications, http://csrc.nist.gov/groups/ST/toolkit/rng/documentation software.html, 2010.

[10] Rainer A. Rueppel, Correlation Immunity and the Summation Generator, In Hugh C. Williams, editor, CRYPTO, Volume 218 of Lecture Notes in Computer Science, pages 260-272, Springer, 1985.

[11] Thomas Siegenthaler, Cryptanalysts Representation of Nonlinearly Filtered *ml*-sequences, In Franz Pichler, editor, EUROCRYPT, Volume 219 of Lecture Notes in Computer Science, pages 103-110, Springer, 1985.

[12] Thomas Siegenthaler, Decrypting a class of stream ciphers using ciphertext only, IEEE Trans. Computers 34(1) (1985), 81-85.

■