Dynamic LFSRs as an alternative to LFSRs in extended fields - A comparative study

A. Peinado[1 [0000−0003−1183−736X]] and S. Petrovic[2 [0000−0002−4435−2716]]

[1] Universidad de Málaga, ETS Ingeniería de Telecomunicación, 29071 Málaga, Spain
apeinado@ic.uma.es
[2] Norwegian University of Science and Technology (NTNU), P.O. box 191, N-2802 Gjøvik, Norway
slobodan.petrovic@ntnu.no

Abstract. Linear feedback shift registers (LFSRs) with dynamic feedback (DLFSRs) and LFSRs defined over extended fields i.e., over $GF(2^n)$, constitute building blocks of many pseudorandom sequence generators used in stream ciphers. In this work, the advantages and disadvantages of using DLSFR instead of LFSR in $GF(2^n)$ are analyzed. The work is based on the possibility of obtaining a DLFSR in $GF(2)$ equivalent to an LFSR in $GF(2^n)$, given that both structures present equivalent binary models formed by interleaved sequences. Likewise, the possibility of using DLFSR on binary vectors is proposed in order to take advantage of the word lengths of current processors.

Keywords: Dynamic LFSR · stream cipher · pseudorandom sequence

1 Introduction

The linear feedback shift register (LFSR) is a well known primitive used in generators of pseudorandom sequences [7]. Its simplicity and the posibility to determine a priori the length of the generated sequences, as well as its excellent statistical properties, justify its use as a basic element in the construction of generators. From a cryptographic point of view, the sequences generated by the LFSRs are not directly usable due to their high predictability, which is a consequence of the linear feedback. The solutions commonly adopted to fix this inconvenience are based on non-linear combinations of several LFSRs or on non-linear filtering of the generated sequences [1, 11].

Although an LFSR can be defined in any finite field, the most common utilization has always been the binary case, where each of the L cells that compose the LFSR stores the value of one bit. In this way, the maximum length of the generated sequences is $2^L - 1$, obtained when the feedback polynomial is primitive. On the other hand, when operating over the binary field, the sum and product operations that determine the feedback can be implemented using the logical operator XOR, which is the one that must be used in a stream cipher to mix the plaintext message with the pseudorandom cipher sequence [9]. For this

reason, LFSR-based stream ciphers have been especially efficient when implemented in hardware. However, software implementations have always presented an efficiency problem that has worsened as the word length of the processors has increased. The problem is because the processors use instructions with n-bit operands, with $n = 16, 32$ or 64, while the LFSR feedback only requires 1-bit operands [3].

In order to improve the efficiency of these processors and, at the same time, increase the number of bits generated in each iteration of the LFSR, stream ciphers were designed using LFSRs defined over $GF(2^n)$. In this way, each clock cycle of the LFSR generates n bits, so the generation speed increases. However, the speed improvement does not reach the value n since the product operation in $GF(2^n)$ is much more computationally expensive. In spite of that, the LFSRs in extended fields are used in stream ciphers although the implementations are usually restricted to certain feedback polynomials [4, 6].

Another way to improve the sequences generated by the LFSR is to use the dynamic LFSR (DLFSR). Initially proposed in [10], it is based on the dynamic modification of the feedback polynomial. The most relevant improvement is the increase of the period length of the sequences and their linear complexity. In this work, we show the equivalence between certain types of binary DLFSRs and the LFSRs defined over $GF(2^n)$. The advantages and disadvantages of both devices are presented. In Section 2, the details of the LFSR in extended fields and their binary equivalent model are presented. Section 3 describes the features, types and the equivalent model of the DLFSRs. The comparison and equivalence between both devices are shown in Section 4. Conclusions are provided in Section 5.

2 Linear feedback shift registers in extended fields

2.1 Definitions and notation

An LFSR can be defined as a register of L cells $b_0, \ldots, b_{L-1}$, with $b_i \in GF(q)$, $q = p^n$ being a positive power of a prime p. The contents are synchronously updated applying a shift from one cell to the one at its side, and a linear recurrence to obtain each new element. Hence, the sequence generated by the LFSR is composed by the elements b_i computed as

$$b_i = b_{i-1} \cdot c_1 + b_{i-2} \cdot c_2 + \cdots + b_1 \cdot c_{L-1} + b_0 \cdot c_L, \tag{1}$$

for $i \geq L$. The elements $b_0, \ldots, b_{L-1}$ constitute the seed of the sequence and the connection polynomial

$$f(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + \cdots + c_{L-1} \cdot x^{L-1} + c_L \cdot x^L, \tag{2}$$

determines the period of the sequence. As we only consider fields of characteristic 2, that is, $p = 2$, the maximal period is $T = (2^{L \cdot n} - 1)$ n-bit elements when $f(x)$ is primitive over $GF(2^n)$. In the binary case, the maximal period is $2^L - 1$. In practical implementations, the LFSRs in extension fields are defined using n as the word length of the processor, with typical values of 16 and 32.

The main advantage of such an LFSR with respect to the classical LFSR defined in $GF(2)$ is the improvement in the generation rate, since n bits in each iteration are produced. However, the operations are complex. They consume a lot of resources. This inconvenience, in practice, limits its use since a generic implementation cannot be performed for any feedback polynomial. Instead, implementations are deployed for very specific polynomials that make operations easier, such as the SNOW3G algorithm used in 3G and 4G communications [6] and the SNOW-V proposed for 5G [4].

2.2 Binary equivalent model of LFSR in $GF(2^n)$

Recently, in [5], an equivalent binary model has been proposed that allows an LFSR of L cells defined in $GF(2^n)$ to be represented by n binary LFSRs of length $L \cdot n$ allowing faster implementations. The equivalent model is based on n binary LFSRs, all with the same feedback polynomial working in parallel and generating the same m-sequence as the LFSR in the extension field.

The implementation based on n binary LFSRs is faster because it is not necessary to have n processors in parallel. With a single n-bit processor ($n = 32$, for example) the iteration of the n binary LFSRs can be performed at once because they all have the same feedback polynomial [2]. This is what we have called *n-grouped operations*. A 32-bit data can store one stage of each of the 32 binary LFSRs, and the XOR operation can be applied directly to 32-bit data. Furthermore, this equivalent model allows the implementation of any primitive feedback polynomial. On the contrary, a greater memory size is needed, $n \cdot L \cdot n$ bits, instead of $L \cdot n$ bits, and the initial seed must be extended to intialize each of the n binary LFSRs.

3 Dynamic LFSR

3.1 Features and classification

The Dynamic LFSR (DLFSR) is an LFSR, whose feedback polynomial changes dynamically controlled by some parameter. In [13], a classification of the DLFSR is presented and in [12], the mathematical model that allows expressing a DLFSR as a set of interleaved sequences, which can be generated by the same LFSR in parallel is given. In practice, most DLFSRs are made up of a main LFSR, in charge of generating the sequence, and a secondary LFSR (or other pseudorandom generator), in charge of controlling the dynamic changes in the feedback. For this reason, these devices are often identified as DLFSR(n,m), where n and m are the number of cells in the main and control LFSR, respectively.

A binary DLFSR of $n = L$ stages allows the length of the generated sequence and its linear complexity to be extended by a factor of N_s, where N_s is the number of interleaved sequences, whose value depends on various parameters and whose calculation depends on the type of configuration, such as the one described in [13]. The DLFSR behaviour is modeled by means of the following matrix

$$M = \prod_{t}^{t+N_s} A_t, \tag{3}$$

A_t being the connection matrix of the feedback polynomial $p_t(x)$ applied at time instant t. The characteristic polynomial $c(x)$ of M determines the period, following the same rule of LFSR. That means, the maximal sequence is obtained when $c(x)$ is primitive, and the length is $T = N_s \cdot (2^L - 1)$ [12]. The classification presented in [13] distinguishes four types of DLFSR, identified as Class1 to Class 4, as a function of the order the polynomials are applied, sequentially or randomly, and the amount e_i of consecutive bits generated by each polynomial $p_i(x)$, constant or random. Table 1 shows the general descripction of all the types.

Table 1. Classification of DLFSR

DLFSR Type	Polynomial utilization Order	e_i
Class 1	Sequential	Constant
Class 2	Sequential	Random
Class 3	Random	Constant
Class 4	Random	Random

The main advantage of DLFSR is that the sequences have a greater length and linear complexity than those generated by a LFSR. Although the exact length depends on the type of DLFSR, on average a DLFSR(n,m) can generate sequences of period $T = (2^L - 1)(2^m - 1)$. On the other hand, the most important restriction is the bit generation rate, that remains at 1 bit per iteration.

3.2 Binary equivalent model of DLFSR

As indicated in [12], the sequence generated by the DLFSR is composed of N_s interleaved sequences, which allows it to be implemented with N_s binary LFSRs that run in parallel, all with the same feedback polynomial but with different seeds. The generation rate can be increased up to a theoretical maximum of N_s bits for each iteration. In some types of DLFSR, those that have been used in practical ciphers [8], N_s is a very large value and in some types of DLFSR the feedback polynomial cannot be calculated. But, instead of a single L-stage register (LFSR), $N_s \cdot L$-stage registers would be needed, that is, $L \cdot N_s$ bits.

4 DLFSR versus LFSR in extended fields

4.1 DLSR-LFSR equivalence

As it can be deduced from the previous sections, both devices have similar equivalent models. The sequences can be generated in both cases from a set of inter-

leaved sequences, all generated from an LFSR with a unique feedback polynomial: n sequences, in the case of the LFSRs in $GF(2^n)$; and, N_s sequences, in the case of the DLFSR. Consequently, the possibility of implementing an LFSR in $GF(2^n)$ in terms of a DLFSR in $GF(2)$ arises, where the operations are much simpler. In both cases, we start from a register with $L \cdot n$ bits, which corresponds to $L \cdot n$ stages of the binary DLFSR and L stages in the LFSR over $GF(2^n)$. Device equivalence is reached when the DLFSR generates $N_s = n$ interleaved sequences and when such sequences have the same minimal polynomials.

The generated sequences, in both cases, will have a period $T = n \cdot (2^{L \cdot n} - 1)$ bits since they will be composed of n sequences, each with a period of $2^{L \cdot n} - 1$. Taking into account the four fundamental types in which the DLFSRs can be classified [13], those of Class 1 are the most suitable to achieve the equivalence with the LFSRs in $GF(2^n)$, due to the patterns that govern the feedback changes. In class 1 DLFSRs, the feedback polynomials are always applied in a sequential order and each polynomial $p_i(x)$ always generates e_i consecutive bits. Hence, the number N_s of interleaved sequences is determined as follows:

$$N_s = \sum_{i=1}^{N_p} e_i, \tag{4}$$

where N_p is the total amount of different polynomials applied.

As an example, we consider the LFSR defined in $GF(2^4)$ with the feedback polynomial $f(x) = (\alpha^3+1)\cdot x^3+x+1 \in GF(2^4)[x]$, which is primitive over $GF(2^4)$ and where α is a root of x^4+x+1. The sequence generated by the concatenation of all bits of each element has a period of $(2^{12}-1)\cdot 4 = 16380$, a linear complexity $LC = 48$ and minimal polynomial $f(x) = x^{48} + x^{24} + x^{20} + x^{12} + 1$. The binary equivalent model is composed of 4 binary LFSRs with the feedback polynomial $p(x) = x^{12} + x^6 + x^5 + x^3 + 1$, initialized at different seeds.

In order to obtain the DLFSR equivalent to that LFSR, we first consider a Class 1 DLFSR with $N_s = 4$ interleaved sequences, in such a way that the connection matrices A_i, corresponding to the N_p feedback polynomials $p_i(x)$, determine a matrix M, with characteristic polynomial $c(x) = p(x)$. In order to find those polynomials, we have to consider $2 \leq N_p \leq 4$. Since the behaviour of the Class 1 DLFSR is controlled by the equations (3) and (4), different configurations are considered as a function of N_p. Note that $N_p = 1$ is not considered since it involves only one polynomial and consequently no dynamic modification is applied. When $N_p = 2$, three configurations are considered, since two polynomials are used and different combinations of exponents e_i satisfy the condition in Eq. (4). The three configurations correspond to the following forms:

$$\begin{aligned} M &= A_i^3 \cdot A_j \\ M &= A_i^2 \cdot A_j^2 \\ M &= A_i \cdot A_j^3 \end{aligned} \tag{5}$$

This configurations are identified, in short, as $< 3, 1 >$, $< 2, 2 >$ and $< 1, 3 >$ due to the exponents applied to the respective connection matrices. With the same notation, $N_p = 3$ and $N_p = 4$ generate the configurations $< 1, 1, 2 >$, $< 1, 2, 1 >$, $< 2, 1, 1 >$ and $< 1, 1, 1, 1 >$. Table 2 shows all the Class 1 solutions for $N_p = 2$, containing 8 for $< 3, 1 >$ configuration, 4 for $< 2, 2 >$ and 8 for $< 1, 3 >$. The polynomials are represented in hexadecimal format, where the coefficient of the highest degree corresponds to the most significant bit. Each polynomial in the table has its associated exponent on the right. All the solutions are composed by two polynomials $p_1(x)$ and $p_2(x)$, with their correspondent exponents e_1 and e_2. Table 3 shows only a fraction of the 966 solutions for the $N_p = 3$ configuration $< 1, 1, 2 >$. Only one solution for each valid polynomial $p_1(x)$ has been included in the table.

Table 2. Class 1 DLFSR with equivalent binary polynomial $p(x) = x^{12}+x^6+x^5+x^3+1$ for $N_p = 2$.

$p_1(x)$	e_1	$p_2(x)$	e_2	$p_1(x)$	e_1	$p_2(x)$	e_2	$p_1(x)$	e_1	$p_2(x)$	e_2	$p_1(x)$	e_1	$p_2(x)$	e_2
1E19	3	1C87	1	12CB	3	1D23	1	1CC9	2	1609	2	1897	1	1053	3
1E456	3	1339	1	1A2B	3	1D43	1	131D	2	1B91	2	14AD	1	198B	3
1EE3	3	1C9F	1	114F	3	17BF	1	1C87	1	1E19	3	1D23	1	12CB	3
1053	3	1897	1	1B91	2	131D	2	1339	1	1E45	3	1D43	1	1A2B	3
198B	3	14AD	1	1609	2	1CC9	2	1C9F	1	1EE3	3	17BF	1	114F	3

Using the same seed $0xF06$, all the sets of N_p polynomials identified as solutions generate sequences with the same parameters as those generated by the LFSR in $GF(2^4)$, that is, a period 16380, a linear complexity 48 and a minimal polynomial $x^{48} + x^{24} + x^{20} + x^{12} + 1$.

Thus, we can consider that the binary Class 1 DLFSR is equivalent to an LFSR defined in $GF(2^n)$, provided that $N_s = n$ and the decimated sequences of both devices have all the same minimal polynomial.

As one can observe from the previous example, there are many combinations of primitive polynomials that satisfy the equivalent condition. In this case, the solutions have been obtained using exhaustive search by means of a Python script. In the general case, for typical values i.e., $n = 16$ or $n = 32$, and LFSR of approximately 512 bits, the exhaustive search is not an option. However, the huge number of solutions made possible a random search among the primitive polynomial to obtain a valid polynomial set. The main drawback of equivalent DLFSR is that the sequences are generated one bit per clock cycle.

4.2 *n*-grouped DLSFR

Although the equivalent DLFSR has the drawback that the bits are generated one by one, the sequences produced by the DLFSR can be much longer. Instead of restricting by n interleaved sequences, the DLFSR can be implemented using

Table 3. Feedback polynomials of binary Class 1 DLFSR with equivalent binary polynomial $p(x) = x^{12} + x^6 + x^5 + x^3 + 1$ for $N_p = 2$ and configuration $< 1, 1, 2 >$.

$p_1(x)$	$p_2(x)$	$p_3(x)$	$p_1(x)$	$p_2(x)$	$p_3(x)$	$p_1(x)$	$p_2(x)$	$p_3(x)$	$p_1(x)$	$p_2(x)$	$p_3(x)$
1053	1C11	1C05	13A3	1F71	1659	17C1	1F71	114F	1C17	17C1	1775
1069	1609	1099	13A9	1941	1069	1857	1D51	1C27	1C27	1A69	120D
107B	12C1	11DF	1407	1965	19CF	185D	15DD	1609	1C87	1D89	1AE1
107D	1941	1FBD	1431	1C4D	1D5B	1891	1161	1A1B	1C9F	1339	1AD1
1099	15DD	17B3	1437	1E51	1D43	1897	1941	1CC9	1CA5	12C1	17BF
10D1	1F71	15C5	144F	1BC1	185D	18B9	1069	1273	1CBB	1D89	1A2B
10EB	17AD	150F	145D	1CC9	15C5	18EF	1D51	1E19	1CC5	1A69	1D07
1107	1D51	1CC5	1467	1185	1F11	191B	1D89	197B	1CC9	1789	1099
111F	1609	1593	1475	16BD	1F99	1935	12C1	1D85	1CCF	1161	1F1B
1123	1321	116B	14A7	1339	1719	1941	1745	198B	1CF3	1161	16BD
113B	1371	116B	14AD	1609	1467	1965	1941	1743	1D23	1161	13A3
114F	1B91	1475	14D3	1941	1399	197B	1891	1A33	1D43	1069	16A5
1157	1E15	1467	150F	134D	11B3	198B	1069	1AD1	1D51	1C11	1B57
1161	1321	1107	151D	145D	1C27	19B1	1745	14AD	1D5B	1AD1	1475
116B	12B9	1B0B	154D	1099	19CF	19BD	11B3	15C5	1D75	12C1	113B
1185	1891	1FBD	1593	1F71	1CC5	19C9	1891	150F	1D85	1161	1AD1
11B3	1B91	1BC1	1593	18EF	1F1B	19CF	12C1	1273	1D89	1161	11D9
11D9	1F11	1F99	15C5	1941	10D1	19E7	1D85	10D1	1E15	17C1	16A5
11DF	1D43	1E51	15D7	12C1	1609	1A1B	15DD	1407	1E19	1941	1A8B
120D	1F1B	1BBF	15DD	12C1	1F1B	1A2B	19B1	150F	1E2F	1161	1B91
1237	1CC9	11D9	15EB	13A9	1BA7	1A33	1B91	1D5B	1E45	1CC5	10EB
123D	1BC1	1857	1609	1655	1743	1A69	12C1	1CC5	1E51	1F11	1BD3
1267	1941	12CB	1647	1321	1593	1A8B	1B91	1897	1E67	1AE1	1371
1273	1AD1	1D51	1655	1161	151D	1AD1	1BC1	1B0B	1E73	1321	19CF
127F	1FC9	1A2B	1659	1F11	1655	1AE1	19B1	133F	1E8F	17B3	1F11
12B9	1099	1475	16A5	1371	1745	1AF5	1F71	1E73	1EE3	1C11	111F
12C1	1D75	1EE3	16BD	1AD1	11DF	1B0B	1655	19CF	1F11	13A9	1B57
12CB	1CA5	107D	1715	1C05	1F27	1B13	12C1	1099	1F1B	1659	14A7
130F	1321	134D	1719	1B91	1D23	1B1F	12CB	114F	1F27	1941	1F71
131D	1E19	1C87	1743	1161	1123	1B57	19B1	1743	1F71	18B9	127F
1321	1AE1	18B9	1745	17C1	1123	1BA7	1161	1053	1F99	12B9	18EF
1339	18B9	17AD	1775	1941	1185	1BBF	1431	1B13	1FBB	1AE1	1069
133F	1F11	11B3	1789	1BC1	1655	1BC1	1891	144F	1FBD	1F99	1437
134D	1C05	1D51	17AD	17C1	18B9	1BD3	12C1	1B1F	1FC9	19B1	114F
1371	1719	1C4D	17B3	1609	1A1B	1C05	1D51	1E45			
1399	151D	1D89	17BF	1B91	1E67	1C11	1AE1	130F			

higher values of N_s; typical values are $N_s = 2^m - 1$, for an integer m smaller than the DLFSR length. On the other hand, as an alternative to the 1 by 1 bit generation, the DLFSR can be be applied over n-bit vectors, by means of the n-bit instructions of the processors. To do this, n DLFSRs can be used in parallel, all controlled by the same dynamic feedback. The cost of performing feedback in a DLFSR is the same as in n DLFSR. In this way, n bits would be generated in each iteration.

5 Conclusions

The comparison between an LFSR defined in $GF(2^n)$ and a binary DLFSR has shown that they can generate equivalent sequences i.e., with the same length, linear complexity and minimal polynomial, provided that the internal DLFSR parameter N_s coincides with n. It has also been shown, by exhaustive search, that many combinations of primitive polynomials satisfy the equivalence conditions, leading us to conclude that a given LSFR can be implemented by means of many different DLSFRs. Additionally, the DLFSR can be applied on n-bit vectors, instead of binary digits, in order to fix the 1 bit per iteration limitation.

Acknowledgements This work is part of the R+D+i grant P2QProMeTe (PID2020-112586RB-I00), funded by MCIN/AEI/10.13039/501100011033.

References

1. Biham, E., Dunkelman, O.: Cryptanalysis of the A5/1 GSM stream cipher. In: Proceedings of the International Conference on Cryptology in India, pp. 43–51. Calcutta, India, (2000)
2. Cotrina, G., Peinado, A., Ortiz, A.: Efficient implementation of stream cipher SNOW3G for resource-constrained devices. In: Porceedings of 15th International Conference on Computational Intelligence in Security for Information Systems, Salamanca, Spain (2022)
3. Delgado-Mohatar, O., Fúster-Sabater, A., Sierra, J.M.: Performance evaluation of highly efficient techniques for software implementation of LFSR. Comput. Electr. Eng., **37**, 1222–1231 (2011)
4. Ekdahl, P., Johansson, T., Maximov, A., Yang, J.: A new SNOW stream cipher called SNOW-V. IACR Trans. Symmetr. Cryptol. **3**, 1–42 (2019)
5. Espinosa, J., Cotrina, G., Peinado, A., Ortiz, A.: Security and efficiency of linear feedback shift registers in $GF(2^n)$ using n-bit grouped operations, Mathematics, **10**(6), 996 (2022)
6. ETSI/SAGE. Specification of the 3GPP, Confidentiality and Integrity Algorithm UEA2 and UIA2; Document 2: SNOW 3G Specification. ETSI, Sophia Antipolis, France, (2006)
7. Golomb, S.W.: Shift register sequences. 3rd edn. Aegean Park Press, Laguna Hills, CA, USA (2017)
8. Kiyomoto, S., Tanaka, T., Sakurai, K,: K2: A stream cipher algorithm using dynamic feedback control. In: Proceedings of the International Conference on Security and Cryptography, pp. 204—213. INSTICC Press, Lisboa, Portugal, (2007)

9. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Ratón, FL, USA, (2020)
10. Mita, R., Palumbo, G., Pennisi, S., Poli, M.: Pseudorandom bit generator based on dynamic linear feedback topology. Electronic Letters, **38**, 1097–1098 (2002)
11. Padgette, J., Bahr, J., Batra, M., Holtmann, M., Smithbey, R., Lily, C., Scarfone, K.: Guide to Bluetooth Security; NIST: Gaithersburg, MD, USA, (2017)
12. Peinado, A., Fúster-Sabater, A.: Generation of pseudorandom binary sequences by means of LFSRs with dynamic feedback, Mathematical and Computer Modelling, **57**(11–12), 2596–2604 (2013)
13. Peinado, A., Munilla, J., Fúster-Sabater, A.: Optimal Modes of Operation of Pseudorandom Sequence Generators based on DLFSRs, Logic Journal of the IGPL, **24**, 933–943 (2016)