Applications of the Galois Model LFSR in Cryptography

Gardner, David. 2019. “Applications of the Galois Model LFSR in Cryptography”. figshare. https://hdl.handle.net/2134/21932.

Applications of the Galois Model LFSR in Cryptography

by

David Gareth Gardner

A Doctoral Thesis

Submitted in partial fulfilment
of the requirements for the award of

Doctor of Philosophy
of
Loughborough University

30th November 2015

Abstract

The linear feedback shift-register is a widely used tool for generating cryptographic sequences. The properties of the Galois model discussed here offer many opportunities to improve the implementations that already exist. We explore the overall properties of the phases of the Galois model and conjecture a relation with modular Golomb rulers. This conjecture points to an efficient method for constructing non-linear filtering generators which fulfil Golić's design criteria in order to maximise protection against his *inversion attack*. We also produce a number of methods which can improve the rate of output of sequences by combining particular distinct phases of smaller elementary sequences.

Acknowledgements

The research in this thesis was undertaken as a joint work with Ana Sălăgean and Raphael Phan and I would like to offer great appreciation to them for providing me with the opportunity to pursue this work and the encouragement and expert advice required to reach its completion. Without first introducing me to the field of symmetric-key cryptography none of this work would have been possible and without the initial push I do not think I would have thought myself capable of achieving all that I have. Also to my fellow PhD students Alex Burrage and Richard Winter I give my thanks for their helpful comments, advice and support.

I also wish to give my full gratitude to my wife Lehanne Gardner for her indispensable and enduring support throughout my period of study. In the absence of her continued faith in me and my ability to overcome all the difficulties that faced me I truly believe that I would not have been able to complete this work.

Contents

List of Figures

List of Tables

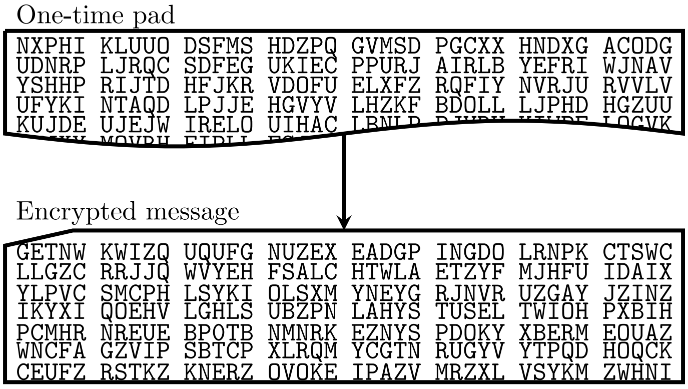
Chapter 1

Introduction

The goal of cryptography, from the Greek words *kryptós* and *graphein*, literally meaning 'secret writing', is to establish a secure line of communication between parties such that conversations retain their confidentiality, integrity and authenticity in the presence of an adversary who is trying to intercept passing messages. Of particular interest is the retention of the confidentiality of conversations across the line and the systems that may be utilised in order to arrest any such attempt on the conversation by an adversary. Many and various are the numbers of systems proposed with the intention of addressing this issue through the methods of encryption and decryption and as can be expected, each system or cipher, has an application best suited to its particular properties.

The one-time pad (OTP) is an example of a symmetric-key algorithm as demonstrated in Figure 1.1, used repeatedly throughout the 20th century, to fulfil the purpose of such a cipher. The greatest property of the OTP, if used correctly, is that it will be impossible for an adversary to decrypt an intercepted encrypted message. It was derived from the Vernam cipher of Gilbert Vernam, patented in 1919 [34], a simple cipher which details the method of combining a message or *plaintext* with a key using the operation of exclusive or (XOR) to obtain a ciphertext which could in turn be combined again with the key in the same manner to reacquire the plaintext. In the example shown the OTP uses addition modulo 26 to combine the characters of the plaintext with the contents of the pad to obtain the ciphertext, with each letter of the alphabet assigned a value between 0 and 25. Attributed to Frank Miller for its first description in 1882 [23] the OTP consists of a pad of paper with a randomly generated key written on each page. If one used the key written on a page of the pad to encrypt a message, as in the spirit of the Vernam cipher, and then destroyed the page, only someone with an exact duplicate of the original pad could decrypt the message. It follows that in situations where the use of a OTP was known, the pads became greatly sought after and the strength of the cipher then relied upon the care its participants took to

Figure 1.1: A one-time pad used to encrypt the first paragraph of this chapter.


One-time pad
NXPHI KLUUO DSFMS HDZPQ GVMSD PGCXX HNDXG ACODG
UDNRP LJRQC SDFEG UKIEC PPURJ AIRLB YEFRI WJNAV
YSHHP RIJTD HFJKR VDOFU ELXFZ RQFIY NVRJU RVVLV
UFYKI NTAQD LPJJE HGVYV LHZKF BDOLL LJPHD HGZUU
KUJDE UJEJW IRELO UIHAC
Encrypted message
GETNW KWIZQ UQUFG NUZEX EADGP INGDO LRNPK CTSWC
LLGZC RRJJQ WVYEH FSALC HTWLA ETZYF MJHFU IDAIX
YLPVC SMCPH LSYKI OLSXM YNEYG RJNVR UZGAY JZINZ
IKYXI QOEHV LGHLS UBZPN LAHYS TUSEL TWIOH PXBIH
PCMHR NREUE BPOTB NMNRK EZNYS PDOKY XBERM EOUAZ
WNCFA GZVIP SBTCP XLRQM YCGTN RUGYV YTPQD HOQCK
CEUFZ RSTKZ KNERZ OVOKE IPAZV MRZXL VSYKM ZWHNI

secure their pads.

Claude Shannon proved in 1949 [28] that the OTP has a property known as *perfect secrecy*, which details that the ciphertext provides no extra information to an adversary about the plaintext other than its maximum possible length. This result holds true even to an adversary with infinite computational power, a statement of enormous weight for sure. The strength of the OTP relied upon two key points:

- The extreme extent of Shannon's proof could only be realised if each of the keys were truly random,
- the use of the OTP can only be as secure as the method used to exchange the pads themselves.

In practice there are many very good approximations to truly random variables but few of these lend themselves to generating the number of sequences which would be required for large scale use of the OTP, this alone is a difficult, if not impossible, goal to achieve. Also, in order that a pad does not fall into an adversary's hands it is imperative that the exchange take place in a totally controlled environment, which again raises various issues in practicality.

The OTP is not suitable for many applications, but the notion of perfect secrecy is a useful one and practical ciphers can be judged by this yardstick. In the area of telecommunications when a party wishes to transmit data across a public network securely, they can use a cipher that mimics a close approximation of the OTP. Some such ciphers will generate pseudorandom sequences (PRSs) which are determined to adequately approximate a truly random sequence and at a reasonable speed which can be used as a key to encrypt the data. PRSs in this context are referred to as *keystreams* to reflect their use as keys in the cipher. In most applications these PRSs may be generated from another relatively smaller key which can be more readily exchanged between clients along another channel. This key exchange is generally studied under the field of public-key cryptography and we shall make the assumption in this thesis that whatever key exchange system is used that it is adequately secure and the secret key remains unknown to an adversary.

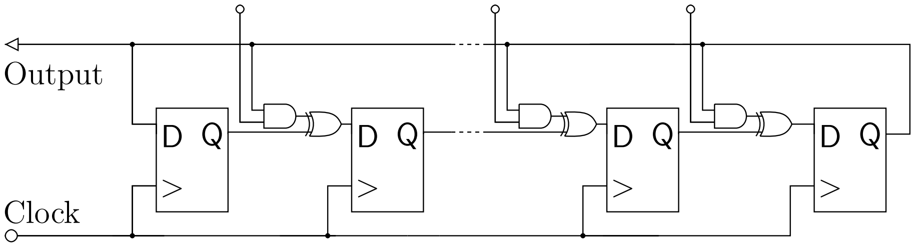
Stream ciphers, another class of symmetric-key algorithm which are discussed herein, are designed to generate a keystream for encryption which simulates the operation of the OTP. Due to the nature of the generation of the keystreams within most stream ciphers, the resulting PRS is periodic. This is a cryptographically undesirable property of stream ciphers since it can leak information to an adversary which can in turn be used to break the cipher. This situation can be reduced in severity by ensuring that the period of the PRS is much greater than the length of the data that is to be encrypted, thus avoiding a repetition in the keystream within the limits of the encryption of our data. It is primarily for this reason that one of the prime requirements of most stream ciphers is that they can efficiently produce PRSs with very large periods given relatively small secret keys.

The LFSR shown in Figure 1.2, is constructed from a collection of memory cells whose values are allowed to change over time according to a predefined set of rules, is a widely used and well documented component of many stream ciphers. The *taps* of the LFSR shown in this diagram may be turned on and off by supplying a signal to the connections across the top. In the figure the implementation uses D flip-flops, a latch circuit which captures the input value D and presents the last captured value onto the output Q at each clock cycle. The LFSRs ubiquity is due to the fact that it can be used to produce PRSs with very large periods given a relatively low number of components. Apart from being able to produce PRSs with very large periods, an LFSR may be constructed such that the sequences will also have a number of other cryptographically desirable properties such as good randomness properties as described by Golomb [13], balancedness, large period and high linear complexity. However, used by themselves or as standalone cryptographic systems, LFSRs can be broken relatively easily using a myriad of differing techniques [35] and it is for this reason that they will generally appear only as a core element of a larger stream cipher along with other components that each provide some other desired properties to reinforce the system's security.

Many design criteria exist to ensure that a cipher meets some set of standard requirements and generally each of the design criteria must be met for the system to resist current cryptographic attacks. One such design principle that has not received a lot of attention until recently is the interaction between an LFSR and the rest of the cipher.

Following our own examination of this area it is clear that not only the methods used to access an LFSR can affect the cipher on a larger scale but also the physical

Figure 1.2: A realisation of an LFSR using an array of D flip-flops, AND gates and XOR gates



model used to describe the LFSR itself. These physical models shall be presented in Chapter 2 of this thesis along with an overview of the main mathematical tools used to study LFSRs and some important definitions and concepts that are used throughout this thesis. Included in Chapter 2 under Section 2.2 are two new notations devised by ourselves in order to ease the manipulation of sequences.

Following this overview, Chapter 3 goes on to cover our own exploration of the nature of LFSRs under different conditions, such as when restrictions are applied to the polynomial used to drive the LFSR or linking numerous LFSRs in various ways using XOR. The consideration here being specifically when non-primitive irreducible polynomials are used in conjunction with the LFSRs and the generation of elementary sequences.

Implementing some points described in Chapter 3, Chapter 4 draws up a newly discovered property of LFSRs involving modular Golomb rulers (MGRs). Here we are using primitive polynomials to generate sequences and we are observing the state of the LFSR over time to derive a set of sequences. We explore the properties of the sequences produced by the LFSR and conjecture on whether the shifts of these sequences themselves form a MGR.

Chapter 5 contains some examinations of biases within existing stream ciphers and how the interaction between an LFSR and the rest of the cipher can cause undesired side effects upon the generated keystream. This chapter also focuses on Golić's inversion attack [11]. The attack is examined both as described in the current literature and in relation to the work in this thesis. We observe how the attack performs against one of the models introduced in this thesis and also how the attack itself can be tweaked to handle this model. Chapter 6 contains another interesting property that was noticed during experimentation that did not provide itself to immediate application involving work in this thesis. Finally Chapter 7 provides a summary of the findings of this work in relation to the use of certain LFSRs within cryptographic systems and speculates on directions for further research in this area.

Chapter 2

Background

Stream ciphers are symmetric-key algorithms that combine a plaintext with a keystream. This keystream is usually a PRS generated from some secret key or seed value. Most stream ciphers operate in the field of $\mathbb{F}_2$ of two values $\{0, 1\}$ and thus most computer based stream ciphers manipulate binary values. The plaintext is encoded usually one bit at a time using XOR, although some of the latest stream ciphers operate in a manner similar to block ciphers in that groups of bits or blocks are encoded at once. This newer style of operation can be attributed to the growing processing speeds of small scale electronics and the tendency of electronics to employ faster and more efficient parallelization techniques.

This chapter describes the existing fundamental concepts of mathematics upon which much of cryptographic theory is built. We shall first define a particular type of sequence which will become the model of the keystreams mentioned earlier and we shall describe a handful of methods of manipulating and analysing these sequences. To end this chapter we will define one of the ubiquitous constructions that is implemented as the core of many stream ciphers to generate sequences and keystreams.

2.1 Sequences

The sequences we study in this thesis are known as *linear recurrence sequences* and are inherently deterministic. This means that given some distinct initial parameters we can generate identical sequences at any time, in any place and given most any software or hardware.

Recalling the definition of the OTP in Figure 1.1 this property of time and location independence is an obvious advantage to one who wishes to send and receive secret messages.

Let us begin by introducing some notation that will allow us to express sequences of values of varying length in their entirety.

Definition 2.1. We shall represent a sequence s as a tuple of elements of a set S and we will write the tuple within the usual parentheses but with additional limits a and b which represent the bounds of the index i of the sequence, $s = (s_i)_{i=a}^{b} = s_a, s_{a+1}, \ldots, s_{b-1}, s_b$ where $s_a, \ldots, s_b \in S$ are the elements of the sequence.

This allows us to separate the actual values of the sequence and concentrate on the relations between elements at certain positions of the sequence.

2.1.1 Linear recurrence relations

We now define some concepts of linear relations and introduce these ideas to our notion of sequences. These concepts form the cornerstone of the study of determinant sequences.

Definition 2.2. [15] Given an ordered list of k elements we refer to a predecessor of an element as all those that appear topologically previous to it. A *k-th order recurrence relation* is then a functional relation between an element and k of its predecessors of the form

$$a_{i+k} = f(a_{i+k-1}, \ldots, a_{i+1}, a_i). \tag{2.1}$$

We shall study recurrence relations in which the function f as shown above is linear and homogeneous. Explicitly, f is of the form

$$f(a_{i+k-1}, \ldots, a_{i+1}, a_i) = c_{k-1}a_{i+k-1} + \cdots + c_1 a_{i+1} + c_0 a_i. \tag{2.2}$$

Example 2.3. A common example of a simple recurrence relation is $a_{i+2} = a_{i+1} + a_i$, the relation that defines the Fibonacci numbers. Here, each element is the sum of the two elements immediately previous to itself.

Now we apply this relation to the ordered elements of a sequence either by finding a relation to fit a sequence or by generating a sequence from the relation.

Definition 2.4. An infinite sequence $s = (s_i)_{i=0}^{\infty}$ with elements in a field F is called a *linear recurrence sequence* if there exists a *linear homogeneous recurrence relation*, i.e. a relation of the form

$$s_{i+n} = c_{n-1}s_{i+n-1} + \cdots + c_1 s_{i+1} + c_0 s_i \tag{2.3}$$

for all $i = 0, 1, \ldots$ where $c_0, c_1, \ldots, c_{n-1} \in F$ are constants.

Example 2.5. Given the relation from Example 2.3 $a_{i+2} = a_{i+1} + a_i$ and using it to generate a sequence $(a_i)_{i=0}^{\infty}$ we obtain

$$x, y, x+y, x+2y, 2x+3y, 3x+5y, 5x+8y, \ldots. \tag{2.4}$$

Substituting $x = y = 1$ we arrive at the familiar Fibonacci sequence. Note that the actual values of the elements of the sequence are uniquely determined by our choice of x and y.

There are many ways to encapsulate the relational function of a sequence but the one we shall use in this thesis is known as the *characteristic polynomial* of the linear recurrence relation of the sequence. This representation has many uses one of which is the linear complexity of the sequence which will be important in the last section of this chapter.

Definition 2.6. Given a linear recurrence relation of the form described above we associate to it a *characteristic polynomial*

$$f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1 x - c_0. \tag{2.5}$$

If n is minimal for the given sequence we call n the *linear complexity* of the sequence and we call f the *minimal polynomial* of s. In this case we call s an *n-th order linear recurrence sequence.*

Example 2.7. Again drawing from the previous examples we can see that the Fibonacci relation may be associated with its characteristic polynomial $f(x) = x^2 - x - 1$. From this and an argument showing that this is also the minimal polynomial of the sequence we see that the linear complexity of the Fibonacci sequence is 2 and that such a sequence is a 2nd order linear recurrence sequence.

A k-periodic sequence for some k has the property that $s_i = s_{i+k}$ for all $i = 0, 1, \ldots$. Thus a k-periodic sequence $(s_i)_{i=0}^{\infty}$ may be represented by any finite sequence $(s_i)_{i=a}^{a+k-1}$, where a is usually chosen to be 0.

Theorem 2.8. *[19, Theorem 6.11] If $(s_i)_{i=0}^{\infty}$ is a linear recurrence sequence in a finite field satisfying the linear recurrence relation $s_{i+n} = c_{n-1}s_{i+n-1} + \cdots + c_1 s_{i+1} + c_0 s_i$ for $n = 0, 1, \ldots$ and if the coefficient c_0 is nonzero, then the sequence $s_0, s_1, \ldots$ is* periodic.

Sadly our Fibonacci sequence examples are not defined over a finite field but over the naturals and thus are not necessarily periodic. Examples such as these may be interpreted to have a period of ∞.

The period and related stability of linear recurrence sequences in regard to linear complexity has a very rich and broadly studied background [5].

Theorem 2.9. *[19, Theorem 6.33] A linear recurrence sequence s over a finite field $\mathbb{F}_2$ with linear complexity n has a maximum possible period of* $2^n - 1$.

Definition 2.10. A sequence which has maximum period for given n is called a maximal length sequence or *m-sequence*, the period of such sequences is $2^n - 1$.

All PRSs used in computer based cryptography can be represented as *linear recurrence sequences* and generally these sequences have extremely large linear complexities. The linear complexity of a sequence is used as a statistical measure of randomness; a sequence with higher linear complexity is more desirable to others. The Berlekamp-Massey algorithm [22], originally designed for use in the field of coding theory can be used to synthesise a characteristic polynomial of a sequence. This property has led to the common use of the Berlekamp-Massey algorithm as a tool to calculate the linear complexity of sequences.

It may be noted that the order or naming of the coefficients c in the definitions above are in reverse order to most standard definitions of linear recurrence relations from the study of discrete mathematics [7, 2, 3]. It appears that those with intent to use linear recurrence relations in a computational manner tend to prefer the ordering we use here [19, 13, 24] This difference is purely cosmetic and our choice allows some expressions to be handled more naturally later on.

2.1.2 Manipulating sequences

Now that we understand what sequences are and how we may represent them we move on to describing some important methods of manipulating them that will be used throughout this thesis. We begin by describing the shifting of sequences, in essence moving relatively the positions of the elements of a sequence.

Definition 2.11. A shift of a sequence s is represented by $(s \ll h)$ or $(s \gg h)$ using parentheses to avoid confusion with the notation for much-greater/less-than. Given $s = (s_i)_{i=0}^{\infty}$ we denote by $(s \ll k)$ the sequence obtained by shifting s by k positions to the left, i.e. the sequence $(s_{i+k})_{i=0}^{\infty}$. If s is n-periodic we denote by $(s \gg k)$ the sequence obtained by shifting s by k positions to the right, i.e. the concatenation of the sequences $(s_{i+n-k})_{i=0}^{k}$ and $(s_i)_{i=0}^{\infty}$. Note that since $(s \gg k) = (s \ll (n-k))$ we may view the shift on a periodic sequence as a rotation or as a *cyclic shift*.

Example 2.12. Let us take the the 8-periodic sequence $s =$ `01101001` and apply a shift to it, $(s \ll 3)$. The sequence we have after the shift, takes the form `01001011`.

A number of further representations of sequences are described in the following definitions and for these we make an assertion about the sequences we shall examine.

We are interested in those recurrence sequences with irreducible characteristic polynomials over $\mathbb{F}_2$ since it is these sequences which model the sequences used in cryptographic systems. By Theorem 2.8 we know that such sequences are periodic since a requirement of irreducibility causes the coefficient c_0 to be nonzero.

Definition 2.13. [19, Definition 2.22] For an element $a \in F = \mathbb{F}_{q^m}$ and a field $K = \mathbb{F}_q$ the trace of a over K, denoted $\mathrm{Tr}_{F/K}(a)$ is defined by

$$\mathrm{Tr}_{F/K}(a) = a + a^q + a^{q^2} + \cdots + a^{q^{m-1}}. \tag{2.6}$$

For any irreducible polynomial f over $\mathbb{F}_2$ we may apply the trace function $\mathrm{Tr}(a)$ as defined above with $q = 2$ to powers of a root of f in order to produce a representation of the linear recurrence relation with characteristic polynomial f.

Theorem 2.14. *[19, Theorem 6.24] Let $(s_i)_{i=0}^{\infty}$ be an n-th order homogeneous linear recurring sequence in $\mathbb{F}_2$ whose characteristic polynomial f is irreducible over $\mathbb{F}_2$. Let β be a root of f in the extension field $\mathbb{F}_{2^n}$. Then there exists a uniquely determined $a \in \mathbb{F}_{2^n}$ such that*

$$s_i = \mathrm{Tr}(a\beta^i) = \sum_{k=0}^{n-1} a^{2^k}(\beta^{2^k})^i = a\beta^i + (a\beta^i)^2 + (a\beta^i)^4 + \cdots + (a\beta^i)^{2^{n-1}} \tag{2.7}$$

for all $i = 0, 1, \ldots$.

The a here uniquely determines the initial n values of the sequence and since each shift of the sequence changes these initial values, each a may represent a particular shift of the sequence. Given the initial n values of an n-th order linear recurring sequence we may find a by solving the set of n linear equations constituting the trace representation of the first n elements of the sequence.

Definition 2.15. An *elementary sequence* is a sequence that has an irreducible minimal polynomial. m-sequences are therefore elementary sequences, but each non-primitive irreducible polynomial is itself the minimal polynomial of a non-maximal length elementary sequence.

2.2 New Notation

2.2.1 The *Seq* notation

In order to facilitate the manipulation of multiple sequences later in this text we introduce the following new short hand notation to refer to elementary sequences in terms of their trace representation and a collection of important results that follow from it:

Definition 2.16. We define $\mathrm{Seq}_\beta(a)$ (or just $\mathrm{Seq}(a)$ if β is clear from the context) as the sequence s whose i-th element is represented by

$$s_i = \mathrm{Tr}(a\beta^i) = \sum_{k=0}^{n-1} a^{2^k}(\beta^{2^k})^i. \tag{2.8}$$

Since this notation is our own we briefly outline some of the properties of the function as it relates to our work. The shifting or rotating of sequences defined by the Seq notation is related as follows:

Lemma 2.17. *Let β be a primitive element of $\mathbb{F}_{2^n}$. For $a \in \mathbb{F}_{2^n}^*$ and $h \in \mathbb{Z}$ we then have the following equalities,*

$$(\mathrm{Seq}_\beta(a) \gg h) = \mathrm{Seq}_\beta(a\beta^{-h}) \tag{2.9}$$

$$(\mathrm{Seq}_\beta(a) \ll h) = \mathrm{Seq}_\beta(a\beta^{h}) \tag{2.10}$$

Proof. By the definition of the shift operators $\ll$ and $\gg$, the i-th element of $(\mathrm{Seq}_\beta(a) \gg h)$ is the $(i-h)$-th element of $\mathrm{Seq}_\beta(a)$,

$$s_{i-h} = \sum_{k=0}^{n-1} a^{2^k}(\beta^{2^k})^{i-h} = \sum_{k=0}^{n-1} (a\beta^{-h})^{2^k}(\beta^{2^k})^{i} \tag{2.11}$$

which by Definition 2.16 is the i-th element of $\mathrm{Seq}_\beta(a\beta^{-h})$, similarly for $(\mathrm{Seq}_\beta(a) \ll h)$. □

2.2.2 The *log* notation

Although not entirely new we now introduce a new concept of the familiar log notation to describe specifically the discrete logarithm over a field:

Definition 2.18. Let β be a primitive element of $\mathbb{F}_{2^n}$ then $\log_\beta(a)$ denotes the discrete logarithm of a in the base of β such that $k = \log_\beta(a) \Leftrightarrow a = \beta^k$. Note that when β is not a primitive element of the field $\log_\beta$ will not always be defined.

Using the definition of the discrete logarithm we readdress the notation seen in Lemma 2.17:

Lemma 2.19. *Again, let* β *be a primitive element of* $\mathbb{F}_{2^n}$*. For* $a_1, a_2 \in \mathbb{F}_{2^n}^*$ *and* $h \in \mathbb{Z}$ *we then have the following relation,*

$$(\mathrm{Seq}_\beta(a_1) \gg h) = \mathrm{Seq}_\beta(a_2) \Leftrightarrow h = \log_\beta(a_1 a_2^{-1}). \tag{2.12}$$

Proof. From Lemma 2.17 we have $(\mathrm{Seq}_\beta(a_1) \gg h) = \mathrm{Seq}_\beta(a_1\beta^{-h}) = \mathrm{Seq}_\beta(a_2)$ and thus $a_1\beta^{-h} = a_2$ or equivalently by Definition 2.18 $h = \log_\beta(a_1 a_2^{-1})$. □

We note that Seq is linear and thus for any $a, b \in \mathbb{F}_{2^n}$ and $c \in \mathbb{F}_2$ we have the following equalities:

$$\mathrm{Seq}_\beta(a) + \mathrm{Seq}_\beta(b) = \mathrm{Seq}_\beta(a+b) \tag{2.13}$$

$$c\,\mathrm{Seq}_\beta(a) = \mathrm{Seq}_\beta(ca) \tag{2.14}$$

Again since a uniquely determines the shift of the sequence, we can deduce a relation between the set of all elements of $\mathbb{F}_{2^n}$ and the n-th order sequences over $\mathbb{F}_2$.

Theorem 2.20. *Let* $a_0, a_1, \ldots, a_{n-1}$ *be elements of the field* $\mathbb{F}_{2^n}$*. Viewing* $\mathbb{F}_{2^n}$ *as an* n*-dimensional vector space over* $\mathbb{F}_2$ *we have the following equivalence: For some* $\beta \in \mathbb{F}_{2^n}$ *with minimal polynomial* g *the values* $a_0, a_1, \ldots, a_{n-1}$ *form a basis of* $\mathbb{F}_{2^n}$ *if and only if* $\mathrm{Seq}_\beta(a_0), \mathrm{Seq}_\beta(a_1), \ldots, \mathrm{Seq}_\beta(a_{n-1})$ *is a basis of the set of sequences with a shared minimal polynomial* g*.*

Proof. Let $a \in \mathbb{F}_{2^n}$, then $a_0, a_1, \ldots, a_{n-1}$ is a basis of $\mathbb{F}_{2^n}$ if and only if there are unique $c_0, c_1, \ldots, c_{n-1} \in \mathbb{F}_2$ such that $a = \sum_{i=0}^{n-1} c_i a_i = c_0 a_0 + c_1 a_1 + \cdots + c_{n-1} a_{n-1}$. This is equivalent to the requirement of unique $c_0, c_1, \ldots, c_{n-1} \in \mathbb{F}_2$ such that $\mathrm{Seq}_\beta(a) = \mathrm{Seq}_\beta(\sum_{i=0}^{n-1} c_i a_i) = \sum_{i=0}^{n-1} c_i \,\mathrm{Seq}_\beta(a_i)$ which is true if and only if $\mathrm{Seq}_\beta(a_0), \mathrm{Seq}_\beta(a_1), \ldots, \mathrm{Seq}_\beta(a_{n-1})$ is a basis of the set of sequences with some minimal polynomial g. □

Definition 2.21. If two periodic sequences are such that one can be obtained from the other by a cyclic shift, then we say that the two sequences are equivalent under cyclic shifts. The different cyclic shifts of a sequence are sometimes called *phases*, especially in engineering contexts.

2.3 Comparing and combining sequences

2.3.1 Polynomials

We now briefly recall the following definition of the order of a polynomial:

Definition 2.22. The least positive integer d such that a polynomial $g \in \mathbb{F}_2[x]$ divides $x^d - 1$ is known as the *order of* g and is denoted as $\mathrm{ord}(g) = d$.

Theorem 2.23. *Given an irreducible polynomial* $g \in \mathbb{F}_2[x]$ *of degree* n *and order* d*, there are exactly* $2^n - 1$ *distinct non-zero elementary sequences with minimal polynomial* g*. There are* $q = (2^n - 1)/d$ *classes of equivalence (under cyclic shifts), each having* d *elements.*

If β *is a root of* g *and* α *is a primitive element of* $\mathbb{F}_{2^n}$ *such that* $\beta = \alpha^q$ *then each of the classes above is of the form* $\{\mathrm{Seq}_\beta(\alpha^{i+jq}) | j = 0, 1, \ldots, d-1\}$ *for* $i = 0, 1, \ldots, q-1$.

Proof. The sequences can be written as $\mathrm{Seq}_\beta(a)$ with each of the $2^n - 1$ elements $a \in \mathbb{F}_{2^n}^*$ uniquely identifying a sequence. By Lemma 2.17, two sequences $\mathrm{Seq}_\beta(b_1)$ and $\mathrm{Seq}_\beta(b_2)$ are equivalent under cyclic shifts if and only if there is an integer h such that $b_1 = b_2\beta^h$. □

2.3.2 Decimation and interleaving

Definition 2.24. Given a sequence $s = s_0, s_1, \ldots$, its q*-decimation* starting at position j is the sequence r such that $r_i = s_{j+iq}$. If j is not specified, it is by default zero, the decimation begins at the first element of the sequence s. Given a sequence of length m, a q-decimation in which $\gcd(m, q) = 1$ is known as a *proper* decimation. If there exists some k such that $m = qk$ then the decimation is called *improper.*

It is known that decimating an m-sequence produces elementary sequences, see [31] and [20, Theorem 11, Ch. 8, §4]. We recall this result here and tailor it to better fit our needs:

Theorem 2.25. *Let* $s = s_0, s_1, \ldots$ *be an* m*-sequence with composite period* $2^n - 1 = dq$*. Let* α *be a root of the minimal polynomial of* s *and let* $a \in \mathbb{F}_{2^n}$ *be such that* $s = \mathrm{Seq}_\alpha(a)$*. Then the* q*-decimation of* s *starting at positions* $0, 1, \ldots q-1$ *will result in the* q *sequences* $s^{(0)}, \ldots, s^{(q-1)}$ *such that* $s^{(j)} = \mathrm{Seq}_\beta(a\alpha^j)$ *for* $j = 0, 1, \ldots q-1$ *and* $\beta = \alpha^q$.

Proof. The i-th element of $s^{(j)}$ is $s_i^{(j)} = s_{j+iq} = \mathrm{Tr}(a\alpha^{j+iq}) = \mathrm{Tr}(a\alpha^j(\alpha^q)^i) = \mathrm{Tr}(a\alpha^j\beta^i)$, which is exactly the i-th element of the sequence $\mathrm{Seq}_\beta(a\alpha^j)$. □

If we now consider the inverse of the above result and explore the outcome of interleaving elementary sequences together we see the following:

Definition 2.26. The *interleaving* of q sequences $r^{(0)}, r^{(1)}, \ldots, r^{(q-1)}$ is the sequence s defined as $s_{j+iq} = r_i^{(j)}$ with $i = 0, \ldots$ and $j = 0, \ldots, q-1$.

Note that if all the sequences $r^{(0)}, r^{(1)}, \ldots, r^{(q-1)}$ have period d, then their interleaving will have as period a factor of dq.

Theorem 2.27. *Let s be an m-sequence of linear complexity n and minimal polynomial f. Assume $2^n - 1 = dq$ is a non-trivial factorisation. Let g be an irreducible polynomial of order d, degree n and let β be a root of g. Let $t^{(0)}, \ldots, t^{(q-1)}$ be elementary sequences with minimal polynomial g.*

If there is a primitive root α of f such that $\beta = \alpha^q$, and there is an $a \in \mathbb{F}_{2^n}$ such that $s = \mathrm{Seq}_\alpha(a)$ and $t^{(j)} = \mathrm{Seq}_\beta(a\alpha^j)$ for $j = 0, 1, \ldots, q-1$ then s can be obtained by interleaving $t^{(0)}, \ldots, t^{(q-1)}$.

Proof. For each $t^{(j)}$ we can choose some $b \in \mathbb{F}_{2^n}$ such that $\mathrm{Seq}_\beta(b\beta^j)$. Since α is a primitive root in $\mathbb{F}_{2^n}$ we have unique $\alpha^{qj} = \beta^j$. Substituting α and choosing an appropriate order in which to interleave the sequences $t^{(0)}, \ldots, t^{(q-1)}$ we can create the sequences $\mathrm{Seq}_\alpha(a\alpha^{jq})$ $j = 0, 1, \ldots, q-1$, which we can see are precisely the sequences produced by a q-decimation of the sequence s. □

Note that if we start from g and construct the finite field $\mathbb{F}_{2^n}$ as the algebraic extension by β, then there are several primitive elements $\alpha \in \mathbb{F}_{2^n}$ which are solutions for $\beta = \alpha^q$. Moreover these α need not all have the same minimal polynomial.

Example 2.28. Let us take as example the field $\mathbb{F}_{64}$, $n = 6$ and the sequence s with minimal polynomial $f(x) = x^6 + x^4 + x^3 + x + 1$. We note that f is primitive and thus s is an m-sequence with period 63.

The factors of 63 are 3, 7, 9 and 21 and it happens that there are irreducible polynomials of degree 6 with only orders 9 and 21. Let us take $g(x) = x^6 + x^5 + x^4 + x^2 + 1$ with order 21, and let β be a root of g. If we now set $\beta = \alpha^q$ which in this case $q = 3$ where α is a primitive root of f we find that $g(\alpha) = \alpha^{18} + \alpha^{15} + \alpha^{12} + \alpha^6 + 1$. Performing the necessary calculations we see that $g(\alpha) = 0$.

From here if we generate and interleave the sequences $t^{(j)} = \mathrm{Seq}_\beta(a\alpha^j)$, the resulting sequence will be $\mathrm{Seq}_\alpha(a)$ for some $a \in \mathbb{F}_{2^n}$.

If we arrange these sequences as shown below it should be clear that interleaving them produces the correct result.

Table 2.1: Interleaving sequences

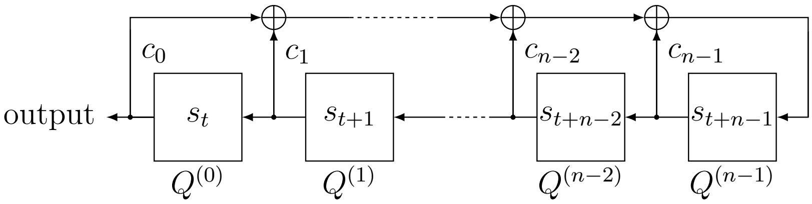
$t^{(0)} =$ 000111001111110110001
$t^{(1)} =$ 001100100101000001101
$t^{(2)} =$ 011101010110111100001
000001011111100101010001100111101110101101001101100010010000111

2.4 Linear feedback shift registers

2.4.1 Fibonacci LFSRs

The LFSR can produce linearly recurring sequences with large periods and good element distributions relatively easily. The Fibonacci model LFSR or just Fibonacci LFSR, named after the 12th century Italian mathematician Leonardo Bonacci, is the most common model of LFSR used in systems that require a PRS to be generated. It is these qualities along with its lightweight hardware implementation that make it so popular in cryptographic systems. A Fibonacci LFSR consists of a number of memory cells that shift their values with each clock interval and a feedback function that feeds new values into the first cell, see Figure 2.1. The memory cells whose values are evaluated by the feedback function are known as the taps or tapped positions.

Figure 2.1: A Fibonacci model LFSR



Firstly it shall be useful to note that the output sequence s of the LFSR shown in Figure 2.1 satisfies the linear recurrence,

$$s_{t+n} = c_{n-1}s_{t+n-1} + c_{n-2}s_{t+n-2} + \cdots + c_1 s_{t+1} + c_0 s_t. \tag{2.15}$$

In the diagram the presence of a connection between the lower track and an XOR on the upper track represents a value of 1 in the corresponding coefficient c_i.

The feedback polynomial of the LFSR is equivalent to the reciprocal of the characteristic polynomial of the linear recurrent sequence above:

$$f(x) = 1 - c_{n-1}x - \cdots - c_1 x^{n-1} - c_0 x^n. \tag{2.16}$$

Example 2.29. Let us take for example a Fibonacci LFSR composed of four memory cells. Naming the cells from left to right as in Figure 2.1, $Q^{(0)}, Q^{(1)}, Q^{(2)}$ and $Q^{(3)}$, and noting connections or taps at memory cells $Q^{(0)}$ and $Q^{(3)}$ we assign the coefficients as such: $c_0 = c_3 = 1$, $c_1 = c_2 = 0$ and thus the feedback polynomial of the LFSR is $f(x) = 1 - x - x^3$.

In the Fibonacci model the non-zero coefficients of the feedback polynomial describe the tapped memory cells of the LFSR. In the literature studied there does not seem to be a global naming convention for the memory cells of LFSRs. While some authors choose to number the memory cells in order, from the input cell to the output cell, there is also a discrepancy on choice of using 0 or 1 based indexing. In this thesis the memory cells of the length n LFSR will be denoted by $Q^{(0)}, Q^{(1)}, \dots, Q^{(n-1)}$.

The letter Q is chosen to reflect the output of a D flip-flop as it is this point in most hardware implementations which holds the analog measurement of the state of the memory cell. When designing what came to be known as Turing machines this Q was introduced by Alan Turing as a reference to *quanta* in order to emphasise the value's discrete nature.

In this thesis the numbering of the memory cells is chosen to be in order from output to input with 0 based indexing in order to provide an intuitive correlation between the contents of the cells and elements of the resultant sequence. Note that the indices of the coefficients c align with those of the memory cells Q. At time t the content of memory cell $Q^{(i)}$ will be denoted $Q_t^{(i)}$. The contents of the complete set of memory cells at time t denoted by the n-tuple

$$Q_t = \left(Q_t^{(0)}, Q_t^{(1)}, \dots, Q_t^{(n-1)}\right), \tag{2.17}$$

is known as the internal state of the LFSR at time t. The initial state of the LFSR is the state at time $t = 0$, $Q_0 = (Q_0^{(0)}, Q_0^{(1)}, \dots, Q_0^{(n-1)})$. The contents of the memory cell $Q^{(i)}$ over time equates to the sequence $(Q_t^{(i)})_{t=0}^{\infty} = Q_0^{(i)}, Q_1^{(i)}, \dots$.

Definition 2.30. A Fibonacci LFSR of length n, see Figure 2.1, with feedback polynomial $f(x) = 1 - c_{n-1}x - \dots - c_1 x^{n-1} - c_0 x^n$ will update itself at each clock interval of t according to the following

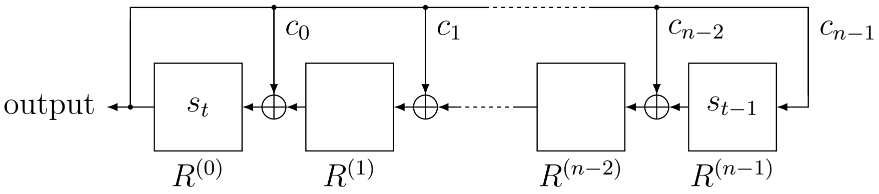
$$Q_{t+1}^{(i)} = \begin{cases} c_{n-1}Q_t^{(n-1)} + \dots + c_1 Q_t^{(1)} + c_0 Q_t^{(0)} & \text{if } i = n-1 \\ Q_t^{(i+1)} & \text{otherwise.} \end{cases} \tag{2.18}$$

When only a single output is required of the Fibonacci LFSR it is most commonly taken from memory cell $Q^{(0)}$. Note that this is also not a global convention but is adopted for use within this thesis. When describing a set of taps of an LFSR many authors wisely use an explicit definition by way of a diagram or by stating the linear recurrence relation of the output sequence to avoid any unnecessary ambiguity.

2.4.2 Galois LFSRs

Another less commonly used implementation of the LFSR and the one this thesis shall concentrate on is the Galois model, named after the 19th century French mathematician Évariste Galois. The function and basic idea behind this model is the same as the Fibonacci model. The critical difference is in how the feedback polynomial is interpreted.

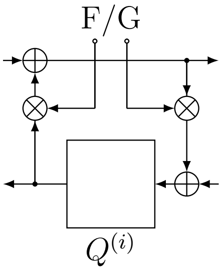
Figure 2.2: A Galois model LFSR



Here the memory cells will be denoted $R^{(i)}$ to avoid confusion with the Fibonacci model. We do not reorder the indices of the memory cells since both models can be seen as equivalent when representing each cell as in Figure 2.3.

This result shows the modular nature of the LFSR as each configuration may be assembled using only these cells. We observe that using the F/G switches an LFSR may be produced by daisy chaining these cells together. In each case, regardless of the model being Fibonacci or Galois, the left most cell will be set to Fibonacci mode and the right most cell will be set to Galois mode. Although this idea is not used to a great extent in this thesis it has been included here as an interesting observation that was made during the research.

Figure 2.3: A generic LFSR memory cell



The F/G input allows the cell to activate and deactivate its own Fibonacci and Galois operating modes.

Note that the coefficients c are also numbered in the same order as its Fibonacci counterpart. It is this choice that gives rise to the fact that given the same feedback polynomial and a suitable initial state the Galois model will produce the reverse sequence of its Fibonacci counterpart. The other naming conventions for sequences and memory cell contents by time are the same as in the previous section. As before the presence of a connection between an XOR on the lower track and the the upper track represents a value of 1 in the corresponding coefficient c_i.

Example 2.31. Assuming a similar setup to the one described in Example 2.29 except now our memory cells are named $R^{(i)}$ instead of $Q^{(i)}$, according to Figure 2.3 we may arrive at the new connection state by flipping all cells from Fibonacci mode to Galois mode except for $R^{(0)}$ which now operates under both modes and $R^{(3)}$ which previously operated in both modes but now only operates in Galois mode.

This equivalence allows us to retain not only the numbering of the memory cells but also the numbering of the coefficients derived from each connection or tap. This LFSR has a feedback polynomial of $f(x) = 1 - x - x^3$, which in this particular case is the same polynomial stated in Example 2.29.

The sequences produced by examining a single memory cell of the Galois LFSR over time have characteristic polynomial of the form $x^n - c_0x^{n-1} - \cdots - c_{n-2}x - c_{n-1}$. Reversing the step taken to arrive at a feedback polynomial from a characteristic polynomial we define the feedback polynomial of the above Galois LFSR as $f(x) = 1 - c_0x - \cdots - c_{n-2}x^{n-1} - c_{n-1}x^n$.

There exists a relation between characteristic and feedback polynomials of both Galois and Fibonacci LFSRs and these relationships are demonstrated relative to the characteristic polynomial of the Fibonacci model in Table 2.2.

Table 2.2: Relationship between characteristic and feedback polynomials of LFSRs

Polynomial	Fibonacci	Galois
Characteristic	$x^n f(x^{-1})$	$x^{-1}(f(x) - 1) + 1$
Feedback	$f(x)$	$x^n(f(x^{-1}) - 1) + 1$

Definition 2.32. A Galois LFSR of length n, see Figure 2.2, with feedback polynomial $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1x - c_0$ will update itself at each clock interval of t according to the following

$$R_{t+1}^{(i)} = \begin{cases} c_{n-1}R_t^{(0)} & \text{if } i = n-1 \\ R_t^{(i+1)} + c_iR_t^{(0)} & \text{otherwise.} \end{cases} \tag{2.19}$$

The state $R_t = (R_t^{(0)}, R_t^{(1)}, \ldots, R_t^{(n-1)})$ at time t of a Galois LFSR with feedback polynomial f primitive over $\mathbb{F}_2$ can be interpreted as the coefficients of the element $R_t^{(0)}\alpha^{n-1} + \cdots + R_t^{(n-2)}\alpha + R_t^{(n-1)} \in \mathbb{F}_{2^n}$ where α is a primitive root of f. If the initial state is interpreted as α^k then at time t we have the state α^{k+t}. It is this property along with a particular primitive polynomial that some authors use to generate the elements of finite fields.

2.4.3 Increasing the rate of output

Some authors have explored various ways to artificially increase the rate of output of cryptographic systems that employ LFSRs to generate PRSs. Robshaw [26] described a method of interleaving 2^k phases of the desired m-sequence, with all these phases being generated synchronously by separate LFSRs. Here k is a value less than or equal to the complexity of the sequence n, see also [18, 31] whose work is similar but places greater requirements on the properties of the sequences. Robshaw's method does not differentiate between Fibonacci and Galois LFSRs since only one output bit is considered. Here the rate of output of offset by the hardware complexity of the system since each increase by a factor of 2 to the rate of output also requires an increase by a factor of 2 of the number of LFSRs. However it must be noted that Robshaw's method is designed to compromise computational power for speed.

Blackburn [4] then extended this idea by using a smaller number of Galois LFSRs in order to produce the required synchronous sequences, exploiting the fact that a Galois LFSR with a primitive feedback polynomial can produce a greater range of phases of a single m-sequence than an equivalent Fibonacci LFSR. This method allows us to gain an increase in the rate of output without needing to suffer the extra hardware costs. The main difficulty here is one of efficiently identifying a primitive feedback polynomial that will produce sequences in the required phases. The former of these methods claims to increase the rate of output by a factor of 2^k using 2^k registers to generate a number of distinct phases and interleaving them. The latter method increases the rate of output by a factor of k by interleaving k phases in some situations generated by fewer than k registers.

Surböch and Weinrichter [31] explored interleaving so called *elementary sequences* instead of m-sequences. We look more closely at elementary sequences in Section 3.2 where we shall use them almost exclusively to efficiently generate the longer, more cryptographically robust m-sequences.

Chapter 3

Generating Elementary Sequences

This chapter examines first those Galois LFSRs which have specifically primitive feedback polynomials, or equivalently generate maximal elementary sequences, m-sequences, as output. In particular this chapter shall examine the relations between the sequences which would be produced by viewing each memory cell of the LFSR as it changes over time. Thus an LFSR of length n produces n such sequences and in the Fibonacci model these sequences are each a shift by one position of its neighbour. The Galois model however breaks this relation and calculating the shifts becomes more complex. Later in the chapter we move our attention to Galois LFSRs with irreducible non-primitive feedback polynomials. By examining these shifts and the relations between them we hope to be able to find some set of the sequences that we can use in some way to increase the rate of output of m-sequences or other elementary sequences. The majority of the content of this chapter has been presented in [8].

3.1 The shifts of the Galois LFSR

3.1.1 Index tables

Let $f(x) = c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0 \in \mathbb{F}_2[x]$ be a polynomial of degree n giving us $c_n = 1$. If we assume that f is irreducible then we have the case in which $c_0 = 1$ since $c_0 = 0$ would allow the trivial factorisation $f(x) = x(f(x)x^{-1}))$. For now, we shall also assume that f is primitive and we shall define α to be a primitive root of f and define $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/\langle f\rangle$ or equivalently $\mathbb{F}_2(\alpha)$.

Consider now the table with $2^n - 1$ rows and n columns, see Table 3.1, further examples in Appendix C, equivalent to the index table of powers of α in $\mathbb{F}_{2^n}$ described in the polynomial basis

$$\alpha^i = b_{n-1}\alpha^{n-1} + b_{n-2}\alpha^{n-1} + \cdots + b_1\alpha + b_0. \tag{3.1}$$

We shall denote the polynomial entries of the i-th row of the table as follows,

$$r_i^{(0)}, r_i^{(1)}, \ldots, r_i^{(n-2)}, r_i^{(n-1)} \tag{3.2}$$

such that we may define the sequence $r^{(j)}$ as the elements of the j-th column $(r_i^{(j)})_{i=0}^{2^n-2} = r_0^{(j)}, r_1^{(j)}, \ldots, r_{2^n-3}^{(j)}, r_{2^n-2}^{(j)}$.

This table is equivalent to the $2^n - 1$ consecutive states of a Galois LFSR of length n with feedback polynomial f and initial state $(0, 0, \ldots, 0, 1)$. This equivalence is apparent when we note that $\mathbb{F}_{2^n}$ is built from the equivalence classes of $\mathbb{F}_2[x]$ modulo f and that it is this modulus operation that the Galois LFSR emulates in its feedback function.

Table 3.1: Index table of powers of α in $\mathbb{F}_2[x]/\langle x^4 + x^3 + 1\rangle$

α^3	α^2	α^1	α^0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
1	0	0	1
1	0	1	1
1	1	1	1
0	1	1	1
1	1	1	0
0	1	0	1
1	0	1	0
1	1	0	1
0	0	1	1
0	1	1	0
1	1	0	0

Example 3.1. This example references Table 3.2 where we see on the left the consecutive internal states of a Galois LFSR with feedback polynomial $f(x) = x^4 + x^3 + 1$ and initial state $(0, 0, 0, 1)$. On the right is the index table of powers of α in $\mathbb{F}_2[x]/\langle x^4 + x^3 + 1\rangle$ or equivalently the polynomial representation of powers of α modulo $f(\alpha)$. It is clear to see that the coefficients $r_i^{(j)}$ of the polynomials $r_i^{(0)}\alpha^3 + r_i^{(1)}\alpha^2 + r_i^{(2)}\alpha + r_i^{(3)}$ correspond directly to the states of the LFSR chosen.

When we refer to this table we refer to the context of the table of states of the Galois LFSR but all results may readily be applied in the context of the index table of powers of α.

Since f is primitive the sequence $s = r^{(j)}$ is a $(2^n - 1)$-periodic sequence and so we can use the shift notation $(s \gg h)$ to describe a rotation to the right of all

Table 3.2: Interpretation of table Table 3.1

State	Polynomial	Power
`0001`	1	α^0
`0010`	α	α^1
`0100`	α^2	α^2
`1000`	α^3	α^3
`1001`	α^3+1	α^4
`1011`	$\alpha^3+\alpha+1$	α^5
`1111`	$\alpha^3+\alpha^2+\alpha+1$	α^6
`0111`	$\alpha^2+\alpha+1$	α^7
`1110`	$\alpha^3+\alpha^2+\alpha$	α^8
`0101`	α^2+1	α^9
`1010`	$\alpha^3+\alpha$	α^{10}
`1101`	$\alpha^3+\alpha^2+1$	α^{11}
`0011`	$\alpha+1$	α^{12}
`0110`	$\alpha^2+\alpha$	α^{13}
`1100`	$\alpha^3+\alpha^2$	α^{14}

elements of the sequence such that for each unit shift the new value of s_0 becomes the old value of s_{2^n-2}.

3.1.2 Powers of primitive roots of f

In the case of the index table of $\mathbb{F}_{2^n}$, successive entries are generated by multiplying the previous entry by α and performing a modulus operation relating to the equality $f(\alpha) = 0$, with the first entry of the table set to 1. This equivalence is demonstrated in Table 3.1.

As for generating the table of states of a Galois LFSR, an equivalent method exists as described earlier in Section 2.4.2. We assume that our Galois LFSR has a primitive feedback polynomial $f \in \mathbb{F}_2[x]$ of degree n and that α is a primitive root of f.

Recall the definition of an m-sequence from Definition 2.4. We note that all m-sequences have maximal period $2^n - 1$ where n is the linear complexity of the sequence and we can use this property to draw an equivalence between the shift $(s \gg h)$ by h elements of the infinite sequence and the rotation of the sub-sequence of an m-sequence with length equal to its period, i.e. the finite sequence $(s_i)_{i=0}^{2^n-2}$.

If we now choose an a such that $r^{(0)} = \mathrm{Seq}_\alpha(a)$ we can use the shift notation to describe the remaining sequences following the state update function of the Galois

LFSR as described in Definition 2.32 as follows

$$r^{(j)} = \begin{cases} (r^{(n-1)} \gg 1) & \text{if } j = 0 \\ ((r^{(j-1)} + c_j r^{(n-1)}) \gg 1) & \text{otherwise.} \end{cases} \tag{3.3}$$

An examination of the shifts or phases of the sequences produced by the memory cells of the Galois LFSR has also been used by Blackburn in [4] where interleaving of LFSR sequences is used to increase the rate of output of the system. Blackburn's approach was focused on finding a primitive polynomial f such that the shifts of the LFSR contained certain prerequisite values and concluded by stating that

> 'Given a primitive polynomial f, it is not difficult to determine whether any fixed value is contained in $\Sigma(f)$. However, it seems difficult to say much in general about $\Sigma(f)$.'

Here $\Sigma(f)$ is a variation on the set of shifts of the Galois LFSR. Using the Seq notation we can easily deal with the manipulations required to explore this area further.

Since we previously set $r^{(0)} = \mathrm{Seq}(a)$, we know that $r^{(n-1)} = \mathrm{Seq}(a\alpha)$. If we let $r^{(j)} = \mathrm{Seq}(a\alpha^{h_j})$, such that h_j is the relative shift of $r^{(j)}$ to $r^{(0)}$, then using (3.3) and Lemma 2.17 we can show that $\alpha^{h_j} = \alpha^{h_{j-1}-1} + c_j$, $j = 1, \ldots, n-1$. By induction on j and using Equations (2.13) and (2.14) we get

$$r^{(j)} = \mathrm{Seq}\left(a \sum_{k=0}^{j} c_k \alpha^{k-j} \right), \tag{3.4}$$

showing that

$$r^{(j)} = \left(r^{(0)} \ll h_j \right) \Leftrightarrow \sum_{k=0}^{j} c_k \alpha^{k-j} = \alpha^{h_j}. \tag{3.5}$$

Note that we can use the equality $f(\alpha) = 0$ to form the following equality

$$\sum_{k=0}^{j} c_k \alpha^{k-j} = \sum_{k=j+1}^{n} c_k \alpha^{k-j-1}. \tag{3.6}$$

Definition 3.2. The ordered list $H = (h_{n-1}, \ldots, h_0)$ contains the relative shift modulo $2^n - 1$ of each memory cell of a Galois LFSR such that $r^{(j)} = (r^{(0)} \ll h_j)$.

Example 3.3. Using Example 3.1 as a basis and examining the states shown in Table 3.1 we can see that the sequences contained within each column are indeed shifts of the first. We then construct the list $H = (1, 13, 14, 0)$.

Proposition 3.4. *In the list* $H = (h_i | r^{(i)} = (r^{(0)} \ll h_i))$ *each* h_i *is unique.*

Proof. Let us assume that there exists a $h_i = h_j$ then in the table of states the two sequences $r^{(i)}$ and $r^{(j)}$ are identical. This is not possible since every possible state except the all zero state will appear. □

Corollary 3.5. *Given the list* $H = (h_i | r^{(i)} = (r^{(0)} \ll h_i))$ *we have* $h_j - h_i \neq h_k - h_i$ *for all* $j \neq k$.

Theorem 3.6. *Given a Galois LFSR with feedback polynomial* $f(x) = c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ *we may produce an ordered list* H *of the relative shifts of the sequences* $r^{(j)}$ *produced over time from each memory cell. The values* h_j *of this list* $H(h_{n-1}, \ldots, h_0)$ *may be calculated by solving particular instances of the discrete logarithm problem (DLP),*

$$h_j = \log_\alpha \left(\sum_{k=0}^{j} c_k \alpha^{k-j} \right). \tag{3.7}$$

Proof. From (3.4) we have each $r^{(j)} = \mathrm{Seq}(a \sum_{k=0}^{j} c_k \alpha^{k-j})$ and for each shift relative to $r^{(0)}$ in the list H we write $r^{(j)} = (r^{(0)} \ll h_j)$. From (3.5) we deduce that $\sum_{k=0}^{j} c_k \alpha^{k-j} = \alpha^{h_j}$ for each $r^{(j)}$ and h_j. Then we use Definition 2.18 to show that indeed for each $r^{(j)}$ the corresponding entry in H may be calculated by $h_j = \log_\alpha(\sum_{k=0}^{j} c_k \alpha^{k-j})$. □

We may examine Table 3.1 and confirm Theorem 3.6 visually in this case.

It is useful to note that since α is primitive, as in (3.6), the following equality holds:

$$\log_\alpha \left(\sum_{k=0}^{j} c_k \alpha^{k-j} \right) = \log_\alpha \left(\sum_{k=j+1}^{n} c_k \alpha^{k-j} \right) \tag{3.8}$$

The problem here is equivalent to the state-based DLP as defined by Giuliani and Gong in [9, Definition 7]: given the initial state of a Fibonacci LFSR and another state at some time t, determine t. This is shown to be equivalent to the DLP [9, Theorem 3]: given $a \in \langle \beta \rangle$ find k such that $a = \beta^k$ or equivalently compute $\log_\beta(a)$.

Definition 3.7. Given a set $D = \{i_1, \ldots, i_k\} \subseteq \{0, 1, \ldots, n-1\}$ we define the list $H_D = (h_{i_1}, \ldots, h_{i_k})$ as the sublist of shifts corresponding to the subset of indices D.

This will allow us to examine only a particular selection of the list H by applying some restriction on the set of indices D.

3.2 Irreducible polynomials in LFSRs

3.2.1 Elementary sequences

At the start of Section 3.1 regarding our observations of Galois LFSRs we made the assumption that the characteristic polynomial, see Definition 2.4, was primitive. This assumption specifies that the order of such a polynomial shall be equal to $2^n - 1$. We now relax this assumption and allow the order to become a divisor of $2^n - 1$ whilst retaining the irreducibility of the polynomial. Essentially we now wish to observe those Galois LFSRs which have an irreducible non-primitive feedback polynomial. For the sake of readability we denote this new irreducible, non-primitive polynomial of degree n by g and speak of a root $\beta \in \mathbb{F}_{2^n}$. In relation to LFSRs, Definitions 2.30 and 2.32 are still correct when substituting the feedback polynomial f for g and much of fundamental theory remains unchanged.

Since we shall no longer be dealing strictly with primitive polynomials there are a number of properties that arise from this added flexibility which are detailed in the following text.

Firstly the idea of the states of the Galois LFSR having a direct correlation with the index table as demonstrated in Example 3.1 is not entirely relevant in this new context. Previously all results could be readily applied to the index table, now that we no longer have a primitive polynomial we must tighten the definition of the sequences $r^{(i)}$ such that we now only refer to the sequences produced at each memory cell of a Galois LFSR over time.

If g has order d then we may denote a value q such that $qd = 2^n - 1$. If such a polynomial g is used as the feedback polynomial of a Galois LFSR, the sequences $r^{(n-1)}, \dots, r^{(0)}$ will each have period d. Each sequence can then be represented similarly to Definition 2.16,

$$r^{(j)} = \mathrm{Seq}_\beta \left(a \sum_{k=0}^{j} c_k \beta^{k-j} \right) \tag{3.9}$$

for some $a \in \mathbb{F}_{2^n}$.

If we choose some $b_i \in \mathbb{F}_2$ in order to define $\alpha = b_{d-1}\beta^{d-1} + \cdots + b_1\beta + b_0$ in such a way that α^i, $i = 0, \dots, n-1$ are all unique then we may perform a full range of operations over $\mathbb{F}_{2^n}$ including powers of β which do not belong in the same coset. This property is particularly useful for converting elements of $\mathbb{F}_{2^n}$ between forms involving α and β so that we can effectively perform calculations across ‘bases’.

We recall that binary m-sequences have period $2^n - 1$. Decimation and its inverse, interleaving, will play an important role in the following chapter. Note that if s has period N then any q-decimation of s has period $N/\gcd(N, q)$.

3.2.2 Interleaving elementary sequences

Given an irreducible non-primitive polynomial g of degree n over $\mathbb{F}_2[x]$ the aim of this chapter is to compute in parallel all the elementary sequences with minimal polynomial g (i.e. one sequence from each class of equivalence under cyclic shifts). Moreover, they need to each be in a suitable phase such that interleaving them will produce an m-sequence with linear complexity $\deg(g)$; this m-sequence is therefore produced at the rate of $(2^n - 1)/\operatorname{ord}(g)$ bits per clock cycle. Starting from a general method we look at two particular cases:

- Running a small number of Galois LFSRs with suitable seeds and using certain memory cells, possibly with a small amount of buffering.
- Making use of only one Galois or Fibonacci LFSR and computing certain linear combinations of its memory cells.

In the first case we exploit the fact that, unlike a Fibonacci LFSR, each memory cell of a Galois LFSR can produce elementary sequences from different equivalence classes. In the second case we will use XOR operations and exploit the fact that the n sequences obtained from the n memory cells form a basis in the vector space of all elementary sequences with fixed minimal polynomial g as shown in (3.1). For both cases ideally we aim to use less than q LFSRs in order to avoid trivial solutions. Given an m-sequence s of length $2^n - 1$ it is known that for any proper factor q of $2^n - 1$, the improper decimations of s by q are elementary sequences, all having the same irreducible minimal polynomial g.

Taking the reverse approach, we can interleave q elementary sequences in order to obtain an m-sequence. Not any arbitrary collection of q elementary sequences with the same irreducible minimal polynomial will produce an m-sequence by interleaving. Thus, the aim of this chapter is to obtain in an efficient way exactly such a collection of sequences.

We ran experiments for all irreducible polynomials of degree n up to 14, where $2^n - 1$ is not prime and for each n we found that efficient methods exist in both cases for at least one m-sequence.

Whilst these constructions of elementary sequences was targeted at fast generation of m-sequences, there are other possible applications of this construction. In coding theory, elementary sequences are the codewords of minimal cyclic codes, also known as irreducible cyclic codes. Hence this construction will produce the non-equivalent codewords of such a code. In cryptography, one could use the elementary sequences as inputs to a non-linear function in order to construct a filtering generator for a stream cipher.

3.3 Efficient generation of m-sequences from elementary sequences

To recap, the goal of this chapter is to produce q elementary sequences in the correct phase relative to each other such that they may be interleaved to generate an m-sequence. We know from Theorem 2.27 that the sequences we need must be of the form $\mathrm{Seq}_\beta(a\alpha^j)$, $j = 0, \dots, q-1$ where $a \in \mathbb{F}_{2^n}^*$ can be arbitrary.

We can generate these sequences using q (Galois or Fibonacci) LFSRs with suitable chosen initial states. For Galois LFSRs an efficient way of computing these initial states is described in [33] and [17]. Namely the first LFSR has an arbitrary initial state (the impulse response, or in our case a suitable initial state that produces $\mathrm{Seq}_\beta(a)$). The initial state of each following LFSR is obtained as a linear combination of the first n states of the previous LFSR. The coefficients of this combination are exactly the coefficients of α when written in the base $1, \beta, \dots, \beta^{n-1}$.

However, we are interested in obtaining the desired sequences from less than q LFSRs. To this end, we will examine closer what sequences we can obtain from each of the memory cells of an LFSR. We denote as before the sequences produced by a Galois LFSR as $r^{(i)}$ and we now use $q^{(i)}$ to denote those sequences produced by Fibonacci LFSRs.

Theorem 3.8. *Let $g = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0 \in \mathbb{F}_2[x]$ be an irreducible non-primitive polynomial with root $\beta \in \mathbb{F}_{2^n}$.*

Consider a Galois LFSR with feedback polynomial g and Fibonacci LFSR with feedback polynomial $x^n g(x^{-1})$ both with initial states chosen such that they each produce the same output $\mathrm{Seq}_\beta(a)$ for some given $a \in \mathbb{F}_{2^n}$.

The memory cell $Q^{(j)}$ of the Fibonacci LFSR will produce the sequence

$$\left(Q_t^{(j)}\right)_{t=0}^{\infty} = \mathrm{Seq}_\beta(a\beta^j), \tag{3.10}$$

for $j = 0, 1, \dots, n-1$. The j-th memory cell of the Galois LFSR, $R^{(j)}$ produces the sequence

$$\left(R_t^{(j)}\right)_{t=0}^{\infty} = \mathrm{Seq}_\beta\left(av_j\right) \tag{3.11}$$

where

$$v_j = c_{j+1} + c_{j+2}\beta + \dots + c_{n-1}\beta^{n-j-2} + \beta^{n-j-1}, \tag{3.12}$$

for $j = 0, 1, \dots, n-1$. Moreover, if α is a primitive element of $\mathbb{F}_{2^n}$ such that $\beta = \alpha^q$, where $q = (2^n - 1)/\operatorname{ord}(g)$, then the sequences above can be written as $(R_t^{(j)})_{t=0}^{\infty} = \mathrm{Seq}_\beta(a\alpha^{h_j}) = (\mathrm{Seq}_\beta(a\alpha^{k_j}) \ll l_j)$ where $h_j = \log_\alpha v_j = k_j + l_j q$ and $0 \le k_j < q$.

Each of the sets of sequences $\{q^{(j)}|j = 0, 1, \ldots, n-1\}$ *and* $\{r^{(j)}|j = 0, 1, \ldots, n-1\}$ *form a basis for the set of sequences with minimal polynomial* g *when viewed as a* n*-dimensional vector space over* $\mathbb{F}_2$*.*

Proof. The expression for the sequences in each memory cell are derived by (3.9) and to see that the sequences are bases of the vector space, use Theorem 2.20. □

For speeding up the computation of the desired set of sequences

$$A = \{\mathrm{Seq}_\beta(a\alpha^j)|j = 0, \ldots, q-1\} \tag{3.13}$$

we shall employ several Galois or Fibonacci LFSRs, again we aim here to utilize less than q LFSRs. We consider the set B of sequences generated in each memory cell of each of the LFSRs. This set B will contain some of the sequences in A. Any remaining sequences in A can be obtained from the sequences in B by either buffering or by computing a linear combination of sequences in B. The buffering takes place when we have the case in which sequence exists in B but is in the wrong phase. The next two sections shall look at two particular cases of this general method outlined previously.

3.3.1 Using several Galois LFSRs

The first restriction we shall consider here is that of producing our required sequences directly from a number of LFSRs without needing to then linearly combine these sequences. For this we have to use Galois LFSRs, as only a Galois LFSR has the potential of containing sequences from different equivalence classes in different memory cells.

We ran experiments for all irreducible polynomials g of degree n up to 14, where $2^n - 1$ is not prime. For each g we computed all the possible values of α and their minimal polynomial f (which will be the minimal polynomial of the interleaved sequence). Table 3.3 shows a few examples of constructions yielding an elementary sequence in the correct phase from each equivalence class. The n-tuples named *classes* and *shifts* display the values $(k_{n-1}, \ldots, k_0)$ and $(l_{n-1}, \ldots, l_0)$, with the notations from Theorem 3.8. The list named *interleave* specifies which memory cells we need to interleave, with a triple of the form $(R^{(j)}, a\alpha^i, l_j)$ signifying that we have to use a buffer of l_j terms for memory cell $R^{(j)}$ of a Galois LFSR initialised such that the LFSR output is $\mathrm{Seq}_\beta(a\alpha^i)$. The method used to compute such an initial state is as described at the end of Section 2.4.2.

For all lengths in our experiment we were able to determine at least one efficient construction, in the sense that the number of LFSRs needed for the construction is less than q, the total number of sequences required. For each n there is at least one g which will produce 2 of the required sequences, for most n at least 3 sequences can be acquired from one LFSR. If we allow q to increase, such that the elementary sequences become shorter, we generally see that we can obtain elementary sequences from more equivalence classes from a single LFSR, thus lowering the total number of LFSRs needed for the full construction.

Table 3.3: Generating m-sequences by combining the output of multiple Galois LFSRs

$n = 4$, $g = 11111$, $d = 5$, $q = 3$, $\alpha = 1110$, $f = 10011$
classes: $(0, 1, 1, 0)$, shifts: $(4, 4, 1, 0)$, LFSRs: 2
interleave: $(R^{(0)}, a, 0), (R^{(1)}, a, 1), (R^{(0)}, a\alpha, 0)$

$n = 6$, $g = 1010111$, $d = 21$, $q = 3$, $\alpha = 101$, $f = 1100001$
classes: $(0, 2, 1, 1, 0, 0)$, shifts: $(20, 8, 1, 0, 1, 0)$, LFSRs: 2
interleave: $(R^{(0)}, a, 0), (R^{(2)}, a, 0), (R^{(2)}, a\alpha, 0)$

$n = 8$, $g = 111010111$, $d = 17$, $q = 15$, $\alpha = 10011100$, $f = 101110001$
classes: $(0, 3, 13, 13, 13, 13, 3, 0)$, shifts: $(16, 0, 13, 12, 0, 16, 2, 0)$, LFSRs: 6
interleave: $(R^{(0)}, a, 0), (R^{(0)}, a\alpha, 0), (R^{(0)}, a\alpha^2, 0), (R^{(7)}, a, 0),$
$(R^{(7)}, a\alpha, 0), (R^{(7)}, a\alpha^2, 0), (R^{(3)}, a\alpha^8, 0), (R^{(3)}, a\alpha^9, 0),$
$(R^{(0)}, a\alpha^8, 0), (R^{(0)}, a\alpha^9, 0), (R^{(0)}, a\alpha^{10}, 0), (R^{(7)}, a\alpha^8, 0),$
$(R^{(7)}, a\alpha^9, 0), (R^{(3)}, a, 0), (R^{(3)}, a\alpha, 0)$

$n = 9$, $g = 1001100101$, $d = 73$, $q = 7$, $\alpha = 111011111$, $f = 1001101111$
classes: $(0, 0, 5, 5, 5, 4, 0, 0, 0)$, shifts: $(72, 71, 0, 72, 71, 63, 2, 1, 0)$, LFSRs: 4
interleave: $(R^{(0)}, a, 0), (R^{(0)}, a\alpha, 0), (R^{(6)}, a\alpha^4, 0), (R^{(6)}, a\alpha^5, 0),$
$(R^{(0)}, a\alpha^4, 0), (R^{(6)}, a, 0), (R^{(6)}, a\alpha, 0)$

$n = 10$, $g = 10000001111$, $d = 341$, $q = 3$, $\alpha = 1010000110$, $f = 10010000001$
classes: $(0, 1, 2, 0, 0, 0, 0, 0, 0, 0)$, shifts: $(340, 1, 101, 6, 5, 4, 3, 2, 1, 0)$, LFSRs: 2
interleave: $(R^{(0)}, a, 0), (R^{(8)}, a, 1), (R^{(0)}, a\alpha^2, 0)$

3.3.2 Using one LFSR and linear combinations of its memory cells

Next we would like to examine an alternative restriction. We wish to produce the required sequences using only one LFSR, either Fibonacci or Galois, but allow ourselves to linearly combine the output of the memory cells to produce our sequences. This is possible according to Theorem 3.8. Let a be such that the output of our LFSR is $\mathrm{Seq}_\beta(a)$.

To obtain a particular desired sequence $\mathrm{Seq}_\beta(a\alpha^j)$ for $j = 1, \ldots, q-1$ as a linear combination of the memory cells it suffices to represent α^j in the basis $1, \beta, \ldots, \beta^{n-1}$ for a Fibonacci LFSR, or in basis $v_0, v_1, \ldots, v_{n-1}$ for a Galois LFSR. Alternatively we can consider the first n terms of the sequence $\mathrm{Seq}_\beta(a\alpha^j)$, viewed

Table 3.4: Generating m-sequences by combining Galois LFSR output

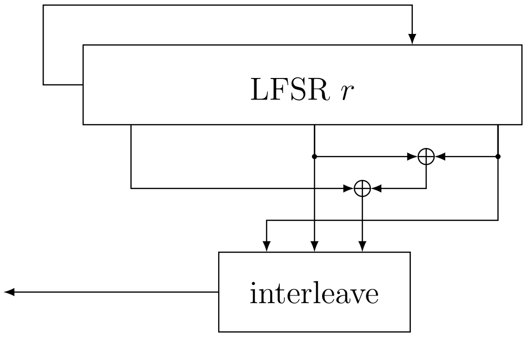
$n = 4$, $g = 11111$, $d = 5$, $q = 3$, $\alpha = 1010$, $f = 10011$ classes: $(0, 2, 2, 0)$, shifts: $(4, 2, 4, 0)$ interleave: $R^{(1)} \oplus R^{(2)} \oplus R^{(3)}, R^{(2)}, R^{(3)}$, XORs: 2
$n = 6$, $g = 1010111$, $d = 21$, $q = 3$, $\alpha = 101$, $f = 1000011$ classes: $(0, 2, 1, 1, 0, 0)$, shifts: $(20, 8, 1, 0, 1, 0)$ interleave: $R^{(5)}, R^{(3)}, R^{(1)} \oplus R^{(3)} \oplus R^{(5)}$, XORs: 2
$n = 8$, $g = 101111011$, $d = 85$, $q = 3$, $\alpha = 100101$, $f = 101011111$ classes: $(0, 0, 0, 2, 2, 0, 0, 0)$, shifts: $(84, 14, 13, 45, 49, 32, 1, 0)$ interleave: $R^{(3)} \oplus R^{(4)} \oplus R^{(6)}, R^{(2)} \oplus R^{(5)} \oplus R^{(7)}, R^{(4)}$, XORs: 4
$n = 8$, $g = 110100011$, $d = 85$, $q = 3$, $\alpha = 10010000$, $f = 110001101$ classes: $(0, 1, 1, 1, 1, 1, 1, 0)$, shifts: $(84, 67, 66, 65, 64, 70, 69, 0)$ interleave: $(A = R^{(3)} \oplus R^{(6)}) \oplus R^{(5)}, (B = R^{(2)} \oplus R^{(3)}), R^{(3)} \oplus A \oplus B$, XORs: 5
$n = 10$, $g = 10010011001$, $d = 341$, $q = 3$, $\alpha = 1110011010$, $f = 10000100111$ classes: $(0, 0, 0, 1, 1, 1, 1, 0, 0, 0)$, shifts: $(340, 339, 338, 85, 91, 90, 89, 2, 1, 0)$ interleave: $R^{(3)} \oplus R^{(6)} \oplus R^{(8)}, R^{(8)} \oplus R^{(9)}, R^{(4)} \oplus R^{(5)} \oplus R^{(9)}$, XORs: 5
$n = 10$, $g = 10010011001$, $d = 341$, $q = 3$, $\alpha = 1111100010$, $f = 10001100101$ classes: $(0, 0, 0, 2, 2, 2, 2, 0, 0, 0)$, shifts: $(340, 339, 338, 312, 318, 317, 316, 2, 1, 0)$ interleave: $R^{(3)} \oplus (A = R^{(7)} \oplus R^{(8)}), (B = R^{(4)} \oplus A), R^{(8)} \oplus R^{(9)} \oplus B$, XORs: 6
$n = 10$, $g = 10000001111$, $d = 341$, $q = 3$, $\alpha = 1010000110$, $f = 10000001001$ classes: $(0, 1, 2, 0, 0, 0, 0, 0, 0, 0)$, shifts: $(340, 1, 101, 6, 5, 4, 3, 2, 1, 0)$ interleave: $R^{(3)}, R^{(5)} \oplus R^{(6)}, R^{(7)} \oplus R^{(9)}$, XORs: 2

as an element in the vector space $\mathbb{F}_2^n$ and write them in the basis consisting of the first n elements of each of the sequences corresponding to the memory cells of the LFSR. Any of these approaches will amount to solving an $n \times n$ system of linear equations over $\mathbb{F}_2$, which is extremely fast.

Again we ran experiments for all irreducible polynomials g of degree n up to 14, where $2^n - 1$ is not prime. We computed, for both Fibonacci and Galois LFSRs, the combinations of memory cells needed to produce the required sequences needed for interleaving. Tables 3.4 and 3.5 show a few examples of possible constructions using Fibonacci and Galois LFSRs respectively. The tables display which memory cells need to be combined in order to produce the sequences required. Interleaving these sequences in order, one will obtain the m-sequence with minimal polynomial f as shown in the table.

The first row in Table 3.5 shows how we may produce the 3 elementary sequences required to produce the sequence with minimal polynomial f. We may combine memory cell $Q^{(0)}$ with each of the memory cells $Q^{(1)}$ and $Q^{(2)}$ to obtain the two sequences A and B and interleave these with the original value in cell $Q^{(0)}$, using the order shown in the table we produce the targeted sequence.

Figure 3.1: Interleaving values from a Galois LFSR to produce an m-sequence.



Some values displayed as assigned inline variables may be used as optimisations to further increase performance when the value may be used to produce more than one required sequence. Such savings are highlighted with parentheses, but a full optimisation is outside the scope of this thesis. The optimisation of these solutions are most likely to described best by some combinatorial problem. For all lengths in our experiment we were able to determine at least one efficient construction, in the sense that for obtaining each of the required sequences the maximum number of sequences that needed to be combined was $\lfloor (n+3)/2 \rfloor$, but for most n this value was as low as $\lfloor (n+1)/2 \rfloor$.

The second row in Table 3.4 is demonstrated in Figure 3.1, here we can see how three values of the LFSR are combined, to make a new elementary sequence and then use this sequence in combination with two of the original sequences to interleave the required m-sequence.

Each of the constructions in these tables are more efficient than naive constructions, meaning that each uses less XORs than using one LFSR to generate each elementary sequence required.

Table 3.5: Generating m-sequences by combining Fibonacci LFSR output

$n = 4$, $g = 11111$, $d = 5$, $q = 3$, $\alpha = 1010$, $f = 10011$ interleave: $Q^{(2)} \oplus Q^{(0)}, Q^{(1)} \oplus Q^{(0)}, Q^{(0)}$, XORs: 2
$n = 6$, $g = 1010111$, $d = 21$, $q = 3$, $\alpha = 101$, $f = 1000011$ interleave: $Q^{(0)}, Q^{(2)} \oplus Q^{(0)}, Q^{(4)} \oplus Q^{(0)}$, XORs: 2
$n = 6$, $g = 1010111$, $d = 21$, $q = 3$, $\alpha = 111000$, $f = 1101101$ interleave: $Q^{(3)} \oplus Q^{(0)}, Q^{(3)} \oplus Q^{(1)}, Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}$, XORs: 4
$n = 8$, $g = 101110111$, $d = 85$, $q = 3$, $\alpha = 1100011$, $f = 100011101$ interleave: $Q^{(3)} \oplus Q^{(1)} \oplus Q^{(0)}, Q^{(5)} \oplus Q^{(2)}, Q^{(4)} \oplus Q^{(0)}$, XORs: 4
$n = 8$, $g = 110001011$, $d = 85$, $q = 3$, $\alpha = 11101011$, $f = 101101001$ interleave: $Q^{(4)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(3)}, Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}$, XORs: 4
$n = 8$, $g = 111011101$, $d = 85$, $q = 3$, $\alpha = 1111110$, $f = 101110001$ interleave: $Q^{(4)} \oplus Q^{(3)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(1)}, Q^{(3)} \oplus Q^{(0)}$, XORs: 4
$n = 8$, $g = 100111111$, $d = 85$, $q = 3$, $\alpha = 1111100$, $f = 110101001$ interleave: $Q^{(4)} \oplus Q^{(3)} \oplus Q^{(1)}, (A = Q^{(5)} \oplus Q^{(2)}), A \oplus Q^{(4)} \oplus Q^{(0)}$, XORs: 5
$n = 10$, $g = 10000001111$, $d = 341$, $q = 3$, $\alpha = 11000110$, $f = 11100011101$ interleave: $(A = Q^{(6)} \oplus Q^{(5)}) \oplus Q^{(1)} \oplus Q^{(0)}, A \oplus Q^{(3)} \oplus Q^{(2)}, Q^{(0)}$, XORs: 5
$n = 10$, $g = 10001000111$, $d = 341$, $q = 3$, $\alpha = 11100011$, $f = 10001101111$ interleave: $Q^{(6)} \oplus Q^{(3)}, Q^{(5)} \oplus Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}, Q^{(5)} \oplus Q^{(1)} \oplus Q^{(0)}$, XORs: 5
$n = 10$, $g = 10000001111$, $d = 341$, $q = 3$, $\alpha = 1010000110$, $f = 10000001001$ interleave: $Q^{(6)}, Q^{(4)} \oplus Q^{(3)}, Q^{(2)} \oplus Q^{(0)}$, XORs: 2

Chapter 4

Modular Golomb Rulers and the Galois LFSR

Here we shall briefly outline the concept of Golomb rulers and some of the properties that we shall be using later in this chapter. The majority of the content of the following two chapters has been presented in [30].

4.1 Golomb rulers

In this section we use the functions $min(A)$ and $max(A)$ to refer to the minimum and maximum elements of a set A respectively.

Golomb rulers, named after the American mathematician Solomon Golomb, also referred to as Sidon sets, after the Hungarian mathematician Simon Szidon, or full positive difference sets are defined as follows:

Definition 4.1. A set of n integers $A = \{a_0, \ldots, a_{n-1}\}$ is a Golomb ruler of order n and length $\max(A) - \min(A)$ if and only if all pairwise sums are different. For all $i, j, k, l \in \{0, \ldots, n-1\}$, $i > j$, $k > l$ we have

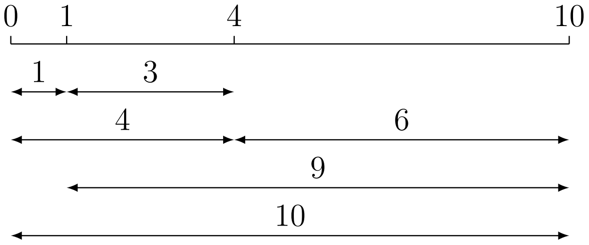
$$a_i - a_j = a_k - a_l \Leftrightarrow i = k \text{ and } j = l. \tag{4.1}$$

Without loss of generality we will always assume that $\min(A) = 0$.

Example 4.2. The set $\{0, 1, 4, 10\}$ is a *Golomb ruler* since the positive pairwise differences of all its elements are unique, see Figure 4.1. The differences in this example being 1, 3, 4, 6, 9 and 10.

A small extension of the concept of the Golomb ruler is obtained by also considering the pairwise differences of the elements modulo some chosen value:

Figure 4.1: A Golomb ruler



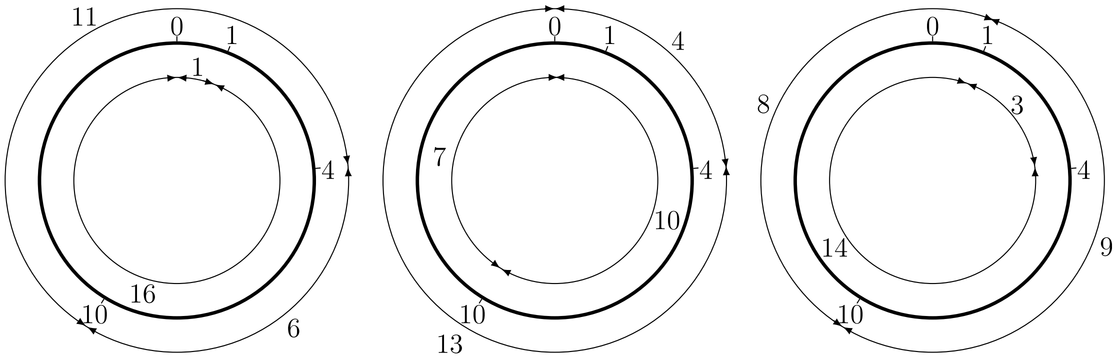
Definition 4.3. A Golomb ruler $A = \{a_0, \ldots, a_{n-1}\}$ with the added property

$$(a_i - a_j) \bmod N = (a_k - a_l) \bmod N \Leftrightarrow i = k \text{ and } j = l \tag{4.2}$$

for all $i, j, k, l \in \{0, \ldots, n-1\}$, $i \neq j$, $k \neq l$, $N > \max(A)$ is a *modular Golomb ruler* modulo N of order n.

Example 4.4. The set $\{0, 1, 4, 10\}$ is an MGR modulo 17 since the pairwise differences mod 17 are all unique.

Figure 4.2: A modular Golomb ruler



For the following Theorem and the remainder of this thesis we assert that we have $a_0 = 0$. Note that we may force this value without loss of generality by substituting $a_j \mapsto a_j - a_0$ for all $j \in \{0, \ldots, n-1\}$.

Theorem 4.5. *Let a set* $A = \{a_0, \ldots, a_{n-1}\}$ *be an MGR modulo* N*, then* $A \cup \{N\}$ *is a Golomb ruler of order* $n+1$ *and length* N*.*

Proof. Since the set A is an MGR modulo N we know also that A is a Golomb ruler. Let us now assume that for some i, j, k we have $N - a_i = a_j - a_k$. It follows from the definition of an MGR and the assertion that $a_0 = 0$ that

$$(N - a_i) \bmod N = (a_j - a_k) \bmod N \tag{4.3}$$

$$(a_0 - a_i) \bmod N = (a_j - a_k) \bmod N. \tag{4.4}$$

Which would imply that $j = 0$ and $i = k$ which is not possible since A is a Golomb ruler. Therefore each difference $N - a_i$ for all i does not already appear in the set of pairwise differences of elements of A and thus $A \cup \{N\}$ meets the criteria of a Golomb ruler. $\square$

Example 4.6. From Example 4.4, we know that the set $\{0, 1, 4, 10, 17\}$ is Golomb ruler.

Definition 4.7. *Optimal Golomb rulers* of order n and length N are either:

- optimally dense, we have maximum n for a given N
- optimally short, we have minimal N for a given n.

Generally the term optimal Golomb ruler refers to the latter.

The notation $v_\beta(k)$ is used by Graham and Sloane [14] to represent the smallest number v such that there exists a k-element set $A = \{a_0, \dots, a_{k-1}\}$, $0 = a_0 < a_1 < \dots < a_{n-1}$ of integers with the property that the sums $a_i + a_j$ for $i \leq j$ belong to $[0, v]$ and represent each element of $[0, v]$ at most once, the set A is an optimal MGR modulo v.

Lemma 4.8. *Let* $A = \{a_0, \dots, a_{n-1}\}$ *with* $\max(A) < N$ *be a Golomb ruler. If* $\max(A) - \min(A) < N/2$ *then* A *is also an MGR modulo* N.

Proof. For all $i < j$ we consider all the pairwise differences $a_j - a_i$ and since A is a Golomb ruler these differences are unique and $b_j - b_i \leq \max(A) - \min(A) < N/2$. Next we consider all the differences $b_i - b_j = N - (b_j - b_i) > N/2$ which are also all unique and also distinct from the first set of differences. $\square$

This result will be used to show that when we are dealing with Fibonacci LFSRs and Golomb rulers of length n we can assume that the lists H as defined in Definition 3.2 are also MGRs mod $2^n - 1$.

4.2 Experimental observations concerning modular Golomb rulers in Galois LFSR shifts

Noticing that the shifts H of Definition 3.2 of the Galois LFSR could be calculated from the coefficients of the feedback polynomial in x, see Theorem 3.8 for preliminaries, brute force experimentation was performed on all Galois fields $\mathbb{F}_{2^n}$ with n from 2 to 23 examining all primitive polynomials for each n. In each case the full states of the Galois LFSR were produced and the shifts $h_{n-1}, h_{n-2}, \ldots, h_1, h_0$ were computed by direct examination of the states. Some examples are described in Table 4.1, with the primitive polynomial f represented as $\texttt{1}c_{n-1}c_{n-2}\ldots c_1\texttt{1}$. The limit on degree n to 23 was due to software and time restrictions.

It was then verified by automated checking that removing those h_j for which $c_j = 0$ leaves a sublist which is an MGR. The values for which $c_j = 0$ are shown in brackets in Table 4.1

Table 4.1: A selection of primitive polynomials f and the corresponding shifts H.

$n = 7$, $f = \texttt{11111101}$, $\mathrm{wt}(f) = 7$, $H = (1, 110, 9, 74, 89, (126), 0)$
$n = 9$, $f = \texttt{1111000111}$, $\mathrm{wt}(f) = 7$, $H = (1, 186, 51, (45), (46), (47), 48, 184, 0)$
$n = 15$, $f = \texttt{1100000111100111}$, $\mathrm{wt}(f) = 9$, $H = (1, (28797), (28798), (28799), (28800), (28801), 28802, 2677,$ $20311, 4439, (8144), (8145), 8146, 28831, 0)$
$n = 21$, $f = \texttt{1010101011110110001101}$, $\mathrm{wt}(f) = 13$, $H = ((1), 2, (300594), 300595, (763444), 763445, (901780),$ $901781, 588446, 1734126, 1277120, (560527), 560528,$ $1553314, (1962686), (1962687), (1962688), 1962689,$ $300591, (2097150), 0)$
$n = 23$, $f = \texttt{111111110111111111111111}$, $\mathrm{wt}(f) = 23$, $H = (1, 7515189, 4958548, 5768351, 6854486, 655069,$ $5077177, (2274675), 2274676, 892313, 5080355, 436706,$ $8162489, 4446935, 3675906, 2274667, 5077170, 655063,$ $6854481, 5768347, 4958545, 7515187, 0)$

Where the coefficient c_i of f is equal to zero the member of the list relating to the shift has been parenthesized.

Further experimental results are available in Appendix F. Recall briefly the ordered list $H = (h_{n-1}, \ldots, h_0)$ containing the relative shift modulo $2^n - 1$ of each memory cell of a Galois LFSR and the sublist H_D with which we place restrictions on the elements of H through the set of indices D. See Definitions 3.2 and 3.7 for further details. The preceding results have led to the following conjecture and are in support of the following proposition:

Conjecture 4.9. *The sublist* H_D *where* $D = \{i | c_i \neq 0\}$, *is an MGR modulo* $2^n - 1$.

Later, in section Section 4.3, we shall attempt to prove this conjecture by examining specific sublists of H_D and showing that in all cases these sublists form MGRs. It is now our objective to find a set of specific cases that together prove that the entire list H_D forms an MGR.

Proposition 4.10. *For all primitive polynomials f of degree less than or equal to 23, H_D with $D = \{i | c_i \neq 0\}$ is an MGR modulo $2^n - 1$.*

To determine if some sublist of shifts H_D of a Galois LFSR is an MGR we may use Algorithm 4.1 with time complexity $\mathcal{O}(n^2)$ and a requirement of $\mathcal{O}(2^n)$ storage space.

The v_i used in the algorithm are those as defined in (3.12) such that $(R_t^{(j)})_{t=0}^{\infty} = \mathrm{Seq}_{\beta}(av_j)$. The array L of the algorithm shall use the polynomial representations of v_i as indices. This is accomplished by taking the coefficients of the polynomial and producing a binary string which in turn may be interpreted as an integer to index the array as normal.

We note that at each iteration of i the array L contains a non-zero entry at select points. The values of the indices at these points describe the differences between pairs of H_D, $h_k - h_j$ and $h_j - h_k$ where $k \leq i$ and these differences will all be unique.

On initialisation the array L contains all zeroes and thus the indices marked by non-zero entries form the empty set, which trivially contains only unique elements and we shall refer to this property as L-unique. Each iteration adds non-zero values to the array L only after checking that the position does not already contain a non-zero value. As such each iteration preserves the L-unique property on the entries of the array L. If the algorithm finds a non-zero value in L whilst trying to add another value it terminates returning False and thus the L-unique property does not hold when $k = 0, 1, \ldots, n-1$. When the algorithm returns True all values of $k = 0, 1, \ldots, n-1$ have been tested and the check that occurs on insertions into L has not failed. Therefore we see that the L-unique property holds on the final state of L. Since all differences $h_i - h_j$ and $h_j - h_i$ with $i = 0, 1, \ldots, n-1$ and $j = i+1, \ldots, n-1$ are unique the set of values $\{h_0, h_1, \ldots, h_{n-1}\}$ is by definition an MGR.

A trivial time-memory trade-off exists whereby the array L is replaced by a binary tree and lines 8 and 11 are replaced by binary searches and insertions respectively. This reduces the memory requirement to $\mathcal{O}(n^3)$ and increases the time complexity to $\mathcal{O}(n^4)$.

We may also iterate over i and j completely with $j = 0, 1, \ldots, n-1$ and check and update L with only one of either $v_i v_j^{-1}$ or $v_i^{-1} v_j$ on each iteration.

Algorithm 4.1 GolombRulerDecision(f, D)

```
Input: $f$ a primitive polynomial of degree $n$; $D \subseteq \{0, 1, \ldots, n-1\}$.
Output: True/False signifying whether $\{h_j | j \in D\}$ is an MGR.
begin
Initialise $L$ to an all-zero array of length $2^n - 1$.
for $i = 0, 1, \ldots, n-1$ do
  for $j = i+1, \ldots, n-1$ do
    Compute the polynomial basis representation of $v_i v_j^{-1}$ and of $v_i^{-1} z_j$
    if ($L[v_i v_j^{-1}] = 1$) or ($L[v_i^{-1} v_j] = 1$) then
      return(False)
    else
      Set $L[v_i v_j^{-1}] = 1$ and $L[v_i^{-1} v_j] = 1$
    end if
  end for
end for
return(True)
end
```

4.3 Modular Golomb rulers in the shifts of the Galois LFSR

We can prove that for certain non-trivial restrictions on the set of indices D of H_D we have an MGR modulo $2^n - 1$. With appropriate choices for the feedback polynomial f these sublists can include up to half the elements of H.

Previously we have only set that all the shifts in H are relative to $r^{(0)}$. We now assert that a, as used in Definition 2.16, is chosen so that $r^{(0)} = \mathrm{Seq}(1)$ all shifts in H are relative to Seq(1). Note that a is relative to the initial state of the Galois LFSR and that in the previous assertion it will always have a unique solution. Now by (3.3) we have that $h_{n-1} = 1$ and $h_0 = 0$. We can also easily show that for all i such that $c_i = 0$ we have $h_i = h_{i-1} - 1$. Hence if $f = x^n + x^i + 1$ is a primitive trinomial, $H = \{0, 2^n - 2, \ldots, 2^n - i, n - i, \ldots, 2, 1\}$.

Theorem 4.11. *Let H_D be a list of values of shifts as in Definition 3.7. The list H_D is an MGR modulo $2^n - 1$ if and only if for all distinct pairs (i, j), (k, l) of elements in D with $i < j$, $k < l$, $j - i \leq l - k$ we have both of the following*

$$\alpha^{h_i} \alpha^{-h_j} \neq \alpha^{h_k} \alpha^{-h_l} \tag{4.5}$$

$$\alpha^{h_i} \alpha^{-h_j} \neq \alpha^{-h_k} \alpha^{h_l} \tag{4.6}$$

Proof. Express each $\alpha^y \alpha^{-z}$ as $\log_\alpha(\alpha^y \alpha^{-z})$ and these are the differences $y - z$ between pairs of elements of the list H_D as in Definition 4.3. □

Lemma 4.12. *Assume c_i, c_j, c_k, c_l are all non-zero.*

Each of the following conditions is sufficient for (4.5) *to be satisfied:*

1. $j - i = l - k$
2. $i + l \leq n$
3. $j + k \geq n - 1$

and each of the following conditions is sufficient for (4.6) *to be satisfied:*

4. $j + l \leq n$
5. $i + k \geq n - 1$.

Proof. The general idea of these proofs is that we assume for a contradiction that equality holds in (4.5) or in (4.6), respectively. We then simplify this equation to the point that only powers of α between $\alpha^0 = 1$ and α^{n-1} appear. Since this is a vector space basis of $\mathbb{F}_{2^n}$ an equality holds if and only if for all i the coefficients of the corresponding α^i are identical on the two sides of the equality. We then prove that this is not the case for our equality thus obtaining a contradiction.

1. Note that in this case we have $i \neq k$ since if $i = k$ we would have $j - i = l - i$ which would then imply $j = l$ and therefore $(i, j) = (k, l)$. Assuming equality in (4.5) we can recall (3.5) and rewrite $\alpha^{h_i}\alpha^{-h_j} = \alpha^{h_k}\alpha^{-h_l}$

$$\sum_{m=0}^{i} c_m \alpha^{m-i} \sum_{m=0}^{l} c_m \alpha^{m-l} = \sum_{m=0}^{k} c_m \alpha^{m-k} \sum_{m=0}^{j} c_m \alpha^{m-j}. \tag{4.7}$$

Using the fact that in this case we have $i + l = j + k$ this simplifies to

$$\sum_{m=0}^{i} c_m \alpha^{m} \sum_{m=0}^{l} c_m \alpha^{m} = \sum_{m=0}^{k} c_m \alpha^{m} \sum_{m=0}^{j} c_m \alpha^{m}. \tag{4.8}$$

Since $i < j$ and $k < l$ we have a situation where the indices of the elements we are summing overlap and thus we may rewrite as follows,

$$\begin{aligned}(c_0 + c_1\alpha + \cdots + c_i\alpha^i)(c_0 + c_1\alpha + \cdots + c_k\alpha^k + \cdots + c_l\alpha^l) = \\ (c_0 + c_1\alpha + \cdots + c_k\alpha^k)(c_0 + c_1\alpha + \cdots + c_i\alpha^i + \cdots + c_j\alpha^j).\end{aligned} \tag{4.9}$$

Breaking the sums in appropriate places we may again rewrite as

$$\begin{aligned}(c_0 + c_1\alpha + \cdots + c_i\alpha^i)[(c_0 + c_1\alpha + \cdots + c_k\alpha^k) + (c_{k+1}\alpha^{k+1} + \cdots + c_l\alpha^l)] = \\ (c_0 + c_1\alpha + \cdots + c_k\alpha^k)[(c_0 + c_1\alpha + \cdots + c_i\alpha^i) + (c_{i+1}\alpha^{i+1} + \cdots + c_j\alpha^j)].\end{aligned} \tag{4.10}$$

In summation form the sums that we have broken up equate to

$$\sum_{m=0}^{j} c_m\alpha^m = \sum_{m=0}^{i} c_m\alpha^m + \sum_{m=i+1}^{j} c_m\alpha^m \tag{4.11}$$

and

$$\sum_{m=0}^{l} c_m\alpha^m = \sum_{m=0}^{k} c_m\alpha^m + \sum_{m=k+1}^{l} c_m\alpha^m. \tag{4.12}$$

Substituting (4.11) and (4.12) into (4.8) we get

$$\sum_{m=0}^{i} c_m\alpha^m \left(\sum_{m=0}^{k} c_m\alpha^m + \sum_{m=k+1}^{l} c_m\alpha^m\right) = \tag{4.13}$$

$$\sum_{m=0}^{k} c_m\alpha^m \left(\sum_{m=0}^{i} c_m\alpha^m + \sum_{m=i+1}^{j} c_m\alpha^m\right) \tag{4.14}$$

which simplifies to

$$\sum_{m=0}^{i} c_m\alpha^m \sum_{m=k+1}^{l} c_m\alpha^m = \sum_{m=0}^{k} c_m\alpha^m \sum_{m=i+1}^{j} c_m\alpha^m. \tag{4.15}$$

We can see that this equation, when expanded, will be of the form

$$\alpha^{N(k)} + \cdots + \alpha^{i+l} = \alpha^{N(i)} + \cdots + \alpha^{j+k} \tag{4.16}$$

where $N(m)$ is the smallest value $u > m$ such that $c_u \neq 0$. To bring all the powers of α into the range $0, \ldots, n-1$ we multiply both sides by α^{-i},

$$\alpha^{N(k)-i} + \cdots + \alpha^{l} = \alpha^{N(i)-i} + \cdots + \alpha^{l}. \tag{4.17}$$

To show that this equation holds we must show that the coefficients of α on both sides coincide. We can see this implies that either $N(i) = N(k)$ which is impossible since $i \neq k$. We now have two cases:

(a) $i < k$ so that $N(i) \leq k < N(k)$

(b) $i > k$ so that $N(i) > i \geq N(k)$

both of which lead to a contradiction.

We note that for the following cases Item 2 and Item 3 we can assume $j - i < l - k$ since the case that $j - i = l - k$ has been covered here.

2. Assuming equality in (4.5) we proceed as in Item 1 and rewrite (4.7) as

$$\sum_{m=0}^{i} c_m\alpha^m \sum_{m=0}^{l} c_m\alpha^m = \alpha^{l+i-j-k} \sum_{m=0}^{k} c_m\alpha^m \sum_{m=0}^{j} c_m\alpha^m. \tag{4.18}$$

   Again we examine the expanded form of this equality

$$1 + \cdots + \alpha^{l+i} = \alpha^{l+i-j-k} + \cdots + \alpha^{l+i} \tag{4.19}$$

   and cancel the higher powers of α

$$1 + \cdots + \alpha^{l+i-j-k-1} = 0. \tag{4.20}$$

   We note that $l+i-j-k-1 < l+i-1 < n$ and we have a contradiction since the coefficients of α are not the same on both sides of the equality.

3. Recall the following statement from (3.6) which allows us to switch the coefficients used when they are in the range $0, \ldots, n-1$,

$$\sum_{k=0}^{j} c_k\alpha^{k-j} = \sum_{k=j+1}^{n} c_k\alpha^{k-j-1}. \tag{4.21}$$

   Assuming equality in (4.5) and using (4.21) we rewrite $\alpha^{h_i}\alpha^{-h_j} = \alpha^{h_k}\alpha^{-h_l}$ as

$$\sum_{m=i+1}^{n} c_m\alpha^{m-i} \sum_{m=l+1}^{n} c_m\alpha^{m-l} = \sum_{m=j+1}^{n} c_m\alpha^{m-j} \sum_{m=k+1}^{n} c_m\alpha^{m-k}. \tag{4.22}$$

   Examining the expanded form of this equality we see that

$$\alpha^{-(i+l)}(\alpha^{N(i)+N(l)} + \cdots + \alpha^{2n}) = \alpha^{-(j+k)}(\alpha^{N(j)+N(k)} + \cdots + \alpha^{2n}) \tag{4.23}$$

   Since $i+l > j+k \geq n-1$ all the powers of α are less than n but for this equality to hold we clearly require that $i+l = j+k$ which is a contradiction.

4. Assuming equality in (4.6) and similarly to Item 1 we rewrite $\alpha^{h_i}\alpha^{-h_j} = \alpha^{-h_k}\alpha^{h_l}$ as

$$\sum_{m=0}^{j} c_m\alpha^m \sum_{m=0}^{l} c_m\alpha^m = \alpha^{j+l-i-k} \sum_{m=0}^{i} c_m\alpha^m \sum_{m=0}^{k} c_m\alpha^m. \tag{4.24}$$

Again we examine the expanded form of this equality

$$1+\cdots+\alpha^{j+l}=\alpha^{j+l-i-k}+\cdots+\alpha^{j+l} \tag{4.25}$$

and cancel the higher powers of α

$$1+\cdots+\alpha^{j+l-i-k-1}=0. \tag{4.26}$$

We note that $j+l-i-k-1<j+l-1<n$ and we have a contradiction since the coefficients of α are not the same on both sides of the equality.

5. Assuming equality in (4.6) and using (3.6) we rewrite $\alpha^{h_i}\alpha^{-h_j}=\alpha^{-h_k}\alpha^{h_l}$ as

$$\sum_{m=j+1}^{n} c_m\alpha^{m-j-1}\sum_{m=l+1}^{n} c_m\alpha^{m-l-1}=\sum_{m=i+1}^{n} c_m\alpha^{m-i-1}\sum_{m=k+1}^{n} c_m\alpha^{m-k-1}. \tag{4.27}$$

Examining the expanded form of this equality we see that

$$\alpha^{-(j+l)}(\alpha^{N(j)+N(l)}+\cdots+\alpha^{2n-1})=\alpha^{-(i+k)}(\alpha^{N(i)+N(k)}+\cdots+\alpha^{2n-1}) \tag{4.28}$$

Since $j+l>i+k\geq n-1$ all the powers of α are less than n but for this equality to hold we clearly require that $j+l=i+k$ which is a contradiction.

□

Theorem 4.13. *Let* $D=\{i|c_i\neq 0\}$ *and let* $D_1=\{i\in D, i\leq\frac{n}{2}\}$ *and* $D_2=\{i\in D, i\geq\frac{n-1}{2}\}$. *Then* $H_{D_1}=\{h_i|i\in D_1\}$ *and* $H_{D_2}=\{h_i|i\in D_2\}$ *are both MGRs modulo* 2^n-1.

Proof. For H_{D_1} all indices satisfy conditions Item 2 and Item 4 in Lemma 4.12. For H_{D_2} all indices satisfy conditions Item 3 and Item 5 in Lemma 4.12. □

Chapter 5

Application of Modular Golomb Rulers in Galois LFSRs

We shall now explore some applications of the properties discussed in the previous chapters. These properties will be applied to some existing notions widely used in cryptography.

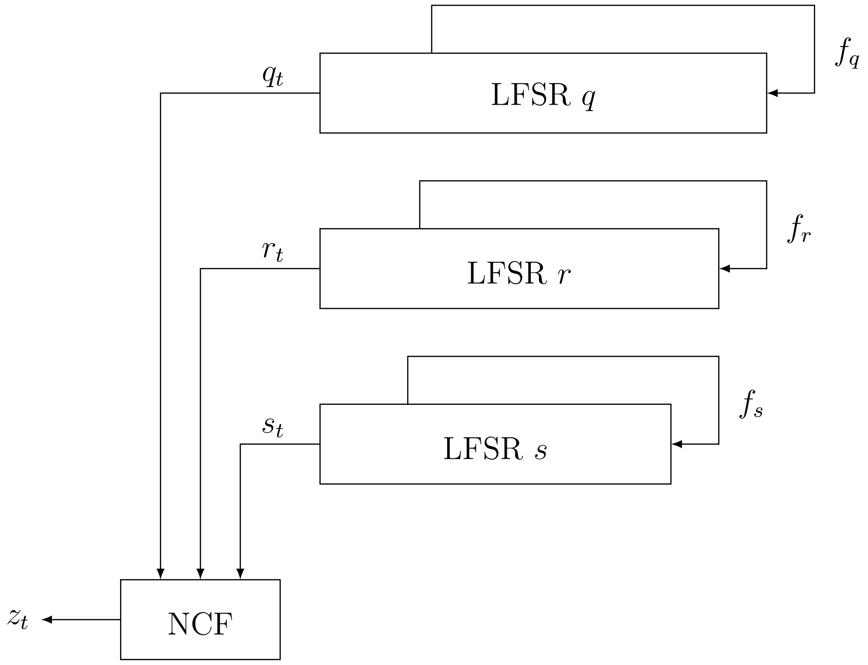
5.1 Combining functions

As mentioned before LFSRs are commonly used in cryptographic systems for their good randomness properties but since many fast methods exist to break LFSRs another layer of security is needed. This additional layer of security usually comes in the form of combining various sequences in particular ways to reduce the risk of attack. The appropriately named method known as *combining* assumes some number of finite state machines (FSMs), sometimes LFSRs, operating in parallel and a *combining function* which takes a single value from each FSM at every time interval to produce an output. These combining functions can be nonlinear and many results exist that show improved properties of the output sequence compared to the input sequence. Such combining functions are known as nonlinear combining function (NCF). The value taken from each FSM is usually the same value that would be chosen for its output if it were treated as a solitary unit. Such a system when regarded as a whole is normally described as a nonlinear combining generator (NCG).

In binary systems a combining function is a boolean function $g : \mathbb{F}_2^k \to \mathbb{F}_2$ that takes as input sequential values from a number of sequences and produces as output a single sequence,

$$z_i = g(s_i^{(0)}, \ldots, s_i^{(k-1)}). \tag{5.1}$$

Figure 5.1: An LFSR fed nonlinear combining function



When the combiner takes values from a number of LFSRs with primitive feedback polynomials the period and linear complexity of the output sequence can be methodically computed [27].

5.2 Filtering functions

One trivial case of combining function is when the input sequences are phase shifts of the same sequence, i.e. the inputs are values of one sequence taken from different points in time see Figure 5.2. This method of utilizing the successive states of sequences produced by FSMs is known as *filtering*.

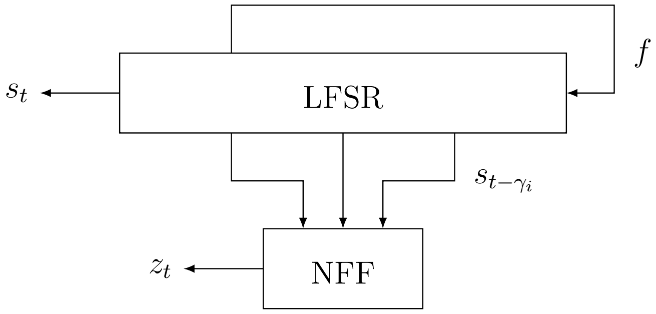
Definition 5.1. We shall use $\gamma = \{\gamma_0, \ldots, \gamma_{k-1}\}$ to denote the *tapping sequence* and shall be used to describe the tapped positions of the sequence feeding a nonlinear filtering function (NFF). We set $\gamma_0 = 0$ and $\gamma_i < N$, $i = 1, \ldots, k-1$ where N is the maximum number of positions that may be tapped. For an unbuffered Fibonacci LFSR this value is the length of the LFSR, n, whereas a Galois LFSR the value of N has the potential to be anything up to $2^n - 1$.

The concept here is that we take a number of values, the tapping sequence defined above, from the state of a FSM at each time interval i. We then perform operations on these values to produce a value to use as the output of the system at that time, known as the *filtering function*, as follows:

$$z_i = g(s_{i-\gamma_0}, \ldots, s_{i-\gamma_{k-1}}). \tag{5.2}$$

It should be noted that the nonlinear filtering generator (NFG) can be viewed as a specific case of an NCGs in which each FSM produces a distinct shift of the same sequence.

Figure 5.2: An LFSR fed nonlinear filtering function



A system that uses a NFF to produce a PRS is known as a NFG.

5.3 Output biases

One small point that has been overlooked in the literature and in practice until very recently is that the properties of these nonlinear combining functions are all proved with the assumption that the inputs are random variables which is not necessarily the case when they are used in cryptographic systems. Certainly when used as filtering functions the inputs are not random variables but can be highly correlated.

In [1] Anderson introduces the *augmented function* $\bar{g} : \mathbb{F}_2^{k+m-1} \to \mathbb{F}_2^m$ which maps a block of input sequence to a block of output sequence of length m.

$$(z_i, \ldots, z_{i+m-1}) = \bar{g}(s_{i-\gamma_0+m-1}, \ldots, s_{i-\gamma_{k-1}}) \tag{5.3}$$

Anderson noticed that the correlations between the input and output frequencies of the augmented function was irregular. Using this imbalance one can determine the best possible local correlations of filtered sequences to use in a correlation attack.

Although some work has been to done to reduce these correlations and to ensure that the output sequence of filtering functions resists many different attacks there are still areas that require a much closer inspection. Teo, Simpson and Dawson show [32] a bias in the output sequences of some high profile balanced nonlinear filters. Of particular interest was Observation 3.3 which stated that in certain cases some m-tuples do not appear at all in the output sequence.

Teo, Simpson and Dawson only considered combinations of three filtering functions, ten LFSRs and two types of tapping sequence. If there exists a large enough bias in the keystream then a possible distinguishing attack may be mounted on it.

Following from this work we performed a brute force test to the same three filtering functions fed by LFSRs with all primitive polynomials of degree 5 to 15, both Fibonacci and Galois models in order to assess the severity of the bias in the keystream. Some results of these tests are included in Appendix G.

An examination of the frequency of different run lengths still present in the output sequence of the filtering function showed that there was indeed a bias. m-sequences, the sequences that are fed into the filtering function have a frequency of 2^{n-l-1} runs of length l and a single run of length n, where n is the linear complexity of the sequence. As shown by Mantin and Shamir [21] a bias in m-tuple distribution allows an attacker to perform statistical analysis and mount distinguishing and ciphertext-only attacks on the keystream. The output of the filtering functions showed a bias towards shorter run lengths. Although there is indeed a bias further examination is required in order to ascertain the amount of statistical analysis would be required to exploit this bias and thus whether or not a successful attack could be carried out.

The number of l-bit patterns that do not appear in the output sequence were also examined and again there is a slight bias, more noticeable in sequences with lower linear complexity. There are patterns of length n and a little shorter that do not appear. These results are not listed here since there are too many to show (approaching 7000 results).

Although these tests do not immediately show how this bias can be used to attack these filtering functions it is interesting nonetheless to note that some properties of the input sequences are not entirely preserved following application of a filtering function.

5.4 Golić's design criteria

Golić describes in [10, 11] methods of attacking NFGs whose tapping sequences are uniformly distributed by some integer δ using decimation, i.e. $\gamma = \{i\delta | i = 0, \ldots, k-1\}$. Also described is an equivalent attack that can exploit taps that are distributed by integer multiples of δ. Golić explains that to render these attacks infeasible one should choose a tapping sequence in order to maximise $\Delta = \max(\gamma) - \min(\gamma)$ preferably so that Δ is as close to the N as described in Definition 5.1. In order to also prevent the uniform decimation attack γ should be chosen such that $\gcd(\gamma_i, \gamma_j) = 1$, $i, j = 0, \ldots, k-1$, $i \neq j$. Golić introduces the *intersection coefficient* which describes for some f and γ the information leakage of the augmented filter function. It is shown [11, Lemma 5] that the intersection coefficient is minimised when γ is a Golomb ruler.

5.4.1 Examination of Golić's inversion attack

Golić's inversion attack, generalized in [12] exploits a property of NFGs whose NFF is of a particular form. Golić proves [11, Theorem 2] that the output sequence of a NFG with a NFF $g(a_0, \ldots, a_{m-1})$, irrespective of the taps chosen, is purely random, uniformly distributed and independent, given that the input sequence is also purely random, uniformly distributed and independent if g may be written with regard to either of the values a_0 or a_{m-1} as follows:

$$g(a_0, \ldots, a_{m-1}) = a_0 + h(a_1, \ldots, a_{m-1}) \tag{5.4}$$

or

$$g(a_0, \ldots, a_{m-1}) = a_{m-1} + h(a_0, \ldots, a_{m-2}). \tag{5.5}$$

In each case the NFF g is linear in either its first or last variable. Golić also conjectures the necessity of this property which is studied further in [29].

The tapping sequence used in the NFG is represented by an increasing sequence of m non-negative integers γ such that $\gamma_{m-1} < n$. Using this notation the NFF g is applied to the LFSR sequence to generate a keystream as follows:

$$z_i = g(s_{i-\gamma_0}, \ldots, s_{i-\gamma_{m-1}}), \tag{5.6}$$

hence the input memory size of the NFG is $\gamma_{m-1} - \gamma_0 = M$. From here we assume without loss of generality that $\gamma_0 = 0$.

If the NFF g is either of the form (5.4) or (5.5) we may write

$$s_i = z_i + h(s_{i-\gamma_1}, \ldots, s_{i-\gamma_{m-1}}) \tag{5.7}$$

or

$$s_{i-\gamma_{m-1}} = z_i + h(s_i, \ldots, s_{i-\gamma_{m-2}}). \tag{5.8}$$

The objective of the inversion attack is to recover the initial state of the LFSR used by a NFG to generate a particular keystream and takes on average 2^{M-1} trials to find a correct initial state. Given known NFF g of the form (5.4), LFSR feedback polynomial f, tapping sequence γ and N bits of the keystream sequence $z_i, \ldots, z_{M-1}$ generated by the NFG the forward inversion attack proceeds as follows:

1. Guess (previously unchecked) M bits of $s_0, \ldots, s_{M-1}$, the unknown initial state.

2. Using (5.7), generate a segment $s_M, \ldots, s_{n-1}$ of the LFSR sequence from a segment $z_M, \ldots, z_{n-1}$ of the keystream sequence.

3. Using f with the initial state $s_0, \ldots, s_{n-1}$ generate a sequence extending $s_n, \ldots, s_{N+M-1}$.
4. Using g, compute a sequence $z'_n, \ldots, z'_{N+M-1}$ and compare with the original $z_n, \ldots, z_{N+M-1}$. If they are the same, stop and accept $s_0, \ldots, s_{M-1}$ as a correct initial state. Otherwise, go to step 1.
5. The entire initial state may be computed from $s_0, \ldots, s_{M-1}$ using the backward LFSR recursion.

If the NFF g was of the form (5.5) then a similar method known as the backward inversion attack may be used. The expected number of false positives for candidate initial states is not expected to exceed 2^{-c} if the length of the keystream is only $N = n + c$.

5.4.2 Golić's attack on the Galois LFSR

We now take Golić's inversion attack and try to apply it directly to a Galois LFSR fed NFG.

The Galois LFSR (see Figure 2.2) updates its state at each clock interval by XOR-ing the value of the memory cell $R^{(n-1)}$ with itself and other memory cells according to the values of the coefficients $c_{n-1}, \ldots, c_0$ in its feedback polynomial. Since the output sequence, taken from $R^{(n-1)}$ of the Galois LFSR, is out of phase with the sequences produced by observing memory cells $R^{(n-2)}, \ldots, R^{(1)}$ by values not necessarily positively increasing or easy to predict we denote these sequences as $r^{(n-1)}, \ldots, r^{(0)}$ to differentiate between them.

Equations (5.7) and (5.8) do not simply transfer to the case of a Galois LFSR since the tapping sequence no longer references values of the LFSR m-sequence. At run time the tapping sequence γ of the NFG references the values of the memory cells of the LFSR, such that in the case of the Galois model we must reformulate equations (5.7) and (5.8) as

$$s_i = z_i + h(s_{i-h_{\gamma_1}}, \ldots, s_{i-h_{\gamma_{m-1}}}) \tag{5.9}$$

or

$$s_{i-h_{\gamma_{m-1}}} = z_i + h(s_i, \ldots, s_{i-h_{\gamma_{m-2}}}), \tag{5.10}$$

where h_i are defined by the list $H = (h_i)$ where $r^{(i)} = (r^{(n-1)} \gg h_i)$. Examining first equation (5.10) we can see that the input memory size M is no longer $\gamma_{m-1} - \gamma_0$, it has increased to $M = \max(h_{\gamma_2}, \ldots, h_{\gamma_{m-1}})$ and immediately we note that using the same steps as listed in the previous section will generally increase our search space far greater than that needed for a brute force attack over all possible initial states.

However, reformulating (5.10) to acknowledge the various sequences of each memory cell gives us

$$r_i^{(n-1-\gamma_{m-1})} = z_i + h(r_i^{(n-1)}, \dots, r_i^{(n-1-\gamma_{m-2})}). \tag{5.11}$$

Using this equation and adjusting the steps of the inversion attack we may proceed as follows:

1. Guess as before (previously unchecked) γ_{m-1} bits of the unknown initial state $r_0^{(n-1)}, \dots, r_0^{(n-1-\gamma_{m-1})}$.
2. Generate the partial states $r_i^{(n-1-j)}$, $i = 1, \dots, n+1-\gamma_m$ using the Galois LFSR function for $j = 0, \dots, \gamma_{m-1}$ and (5.12) for $j = \gamma_{m-1}$.
3. Recover the full states $r_i^{(n-1-j)}$, $i = 1, \dots, n+1-\gamma_m$, $j = 0, \dots, n-1$ from the partial states using the Galois LFSR function.
4. Generate successive states $r_i^{(n-1-j)}$, $i = n+1, \dots, N$, $j = 0, \dots, n-1$ as before using the Galois LFSR function.
5. Proceed from Step 4 as in the previous section.

Although more calculations are needed in this version of the attack it is still possible to recover an initial state of the LFSR with $2^{\gamma_{m-1}}$ trials on average.

By the nature of the Galois LFSR the case in (5.9) cannot be resolved due to the loss of information of $R^{(n-1-\gamma_{m-1})}$. Reformulating as before:

$$r_i^{(n-1)} = z_i + h(r_i^{(n-1-\gamma_1)}, \dots, r_i^{(n-1-\gamma_{m-1})}), \tag{5.12}$$

and attempting to proceed with the steps listed in this section fails at Step 2 since equation (5.9) no longer provides new information as this value can be calculated from the previous partial state. The value required to proceed is $r_i^{(n-1-\gamma_{m-1})}$, $i = 0, \dots, n-\gamma_{m-1}$ and since it is the result of a nonlinear operation it can only be predicted with probability 2^{-1}. If we continue in this manner then the attack will need on average 2^n trials to find a correct initial state of the LFSR, no better than a brute force attack over all possible initial states.

In short, we can derive a new form of the attack to exploit the case that the NFF g is of the form (5.5) i.e. linear in its last variable, although in contrast, the use of a Galois LFSR in conjunction with a NFF g of the form (5.4) i.e. linear in its first variable does not seem to be vulnerable to the standard form of Golić's inversion attack.

5.5 Modular Golomb rulers in the shifts of the Galois LFSRs and filter generators

Since Golić's original design only took into account the Fibonacci model LFSR we shall now consider a new construction of a NFG that uses a Galois LFSR with a dense primitive polynomial f. We select positions $\gamma = \{i_0, \ldots, i_{k-1}\}$, a subset of $\{0, 1, \ldots, n-1\}$ as inputs to the filtering function in such a way that $(h_{i_0}, \ldots, h_{i_{k-1}})$ is an MGR. This NFG would be equivalent to tapping positions $j_0 = h_{i_0}, \ldots, j_{k-1} = h_{i_{k-1}}$ of a buffered section of length $l = \max(\gamma) - \min(\gamma)$ of the m-sequence. Using the Galois LFSR allows us to avoid this unnecessary buffering of terms, saving storage space. This construction would satisfy Golić's design criterion but it remains to be seen whether it would be susceptible to other forms of attack.

Following the results of Section 4.3, using the same dense primitive polynomial f we could choose $\gamma = D = \{i | c_i \neq 0\}$ and use Algorithm 4.1 to check whether Conjecture 4.9 is true in this case. If Algorithm 4.1 returns true then we have $|\gamma| = \mathrm{wt}(f) - 1$ which can be very close to n for some suitably chosen f. If Algorithm 4.1 returns false then we can choose $\gamma = \{i | c_i \neq 0, i \leq \frac{n}{2}\}$ and by Theorem 4.13 we know that H_D is guaranteed to be an MGR. Again, choosing a suitable f we can achieve $|\gamma|$ equal to, or lower and very close to $\lfloor \frac{n}{2} \rfloor + 1$.

Choosing an equivalent tapping sequence from a Fibonacci LFSR of length n such that they formed a Golomb ruler $|\gamma|$ would have an upper bound of $\sqrt{2n} + 1$ since $n \geq k(k-1)/2$, i.e. the length of the LFSR must be greater than the greatest difference between elements of γ. Therefore we would have a much smaller range of inputs available and if we needed a fixed number k of inputs we would need a larger Fibonacci LFSR, greater than $k(k-1)/2$, compared to around $2k$ for the Galois LFSR. This is illustrated by the following example:

Example 5.2. The first example in Table 4.1, after removing the elements in brackets, produces an MGR of order $k = 6$. A Fibonacci LFSR of same length $n = 7$ would allow us to produce a Golomb ruler of only $k = 4$ elements, which by Lemma 4.8 would also be an MGR modulo $2^n - 1$. For $k = 6$ elements we would need a Fibonacci LFSR of length $n = 17$ (see [14]).

The last example in Table 4.1 is a Galois LFSR of length $n = 23$ and after removing the elements in brackets, produces an MGR of order $k = 22$. A Fibonacci LFSR of same length $n = 23$ will allow us to produce a Golomb ruler (which by Lemma 4.8 would also be an MGR modulo $2^n - 1$) of order only $k = 6$. For order $k = 22$ we would need a Fibonacci LFSR of length $n = 356$ (see [14]).

Chapter 6

Observation on the Properties of *H*

This section outlines another small interesting property of the list of shifts H of the Galois LFSR that were noticed during experimentation. It is listed here as it may have important implications to further results involving the set, including a solution to Conjecture 4.9.

6.1 Reciprocal primitive polynomials

As may be expected we observed a relation between the shifts of systems with reciprocal polynomials.

Below is an explanation of the results that were observed.

Proposition 6.1. *Let* $f(x) = c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ *be a primitive polynomial then we know that its reciprocal* $g(x) = x^n f(x^{-1}) = c_0 x^n + c_1 x^{n-1} + \cdots + c_{n-1}x + c_n$ *is also primitive. The relative shifts corresponding to these two polynomials are reversed such that*

$$h_i = 2^n - \rho_{n-i-1}, \quad i = 0, \ldots, n-1 \tag{6.1}$$

or equivalently

$$h_i = 1 - \rho_{n-i-1} \bmod 2^n - 1, \quad i = 0, \ldots, n-1 \tag{6.2}$$

where ρ_i *are the shifts corresponding to* $g(x)$*.*

Proof. Let us first examine g, since α is a primitive root of f it stands that $\beta = \alpha^{-1}$

is a primitive root of g. Let us write

$$h_i = \log_\alpha \left(\sum_{k=i+1}^{n} c_k \alpha^{k-i} \right). \tag{6.3}$$

Now let us examine the corresponding shift of g

$$\begin{aligned} 2^n - \rho_{n-i-1} &= \log_\beta(\beta^{2^n}) - \log_\beta \left(\sum_{k=0}^{n-i-1} c_{n-k} \beta^{k-n+i+1} \right) && (6.4) \\ &= \log_\beta(\beta) - \log_\beta \left(\sum_{k=i+1}^{n} c_k \beta^{i-k+1} \right) && (6.5) \end{aligned}$$

we now substitute in α^{-1}

$$\begin{aligned} 2^n - \rho_{n-i-1} &= \log_{\alpha^{-1}}(\alpha^{-1}) - \log_{\alpha^{-1}} \left(\sum_{k=i+1}^{n} c_k \alpha^{k-i-1} \right) && (6.6) \\ &= \log_\alpha \left(\sum_{k=i+1}^{n} c_k \alpha^{k-i-1} \right) - \log_\alpha(\alpha^{-1}) && (6.7) \\ &= \log_\alpha \left(\sum_{k=i+1}^{n} c_k \alpha^{k-i} \right) && (6.8) \end{aligned}$$

□

Corollary 6.2. *Let $f(x)$ be a primitive polynomial and $g(x) = x^n f(x^{-1})$ be its reciprocal polynomial. The difference between shifts corresponding to these two polynomials are related such that*

$$h_i - h_{i-1} = \rho_{n-i} - \rho_{n-i-1}, \quad i = 1, \ldots, n-1 \tag{6.9}$$

where ρ_i are the shifts corresponding to $g(x)$.

Example 6.3. Let us now consider the polynomial pair f and $g = x^n f(x^{-1})$ whose coefficients are represented by 10110010111 and 11101001101 respectively. We have

$$f(x) = x^{10} + x^8 + x^7 + x^4 + x^2 + x + 1, \tag{6.10}$$

with

$$H_f = (1, 260, 220, (221), 72, (73), (74), 516, 1022, 0) \tag{6.11}$$

and

$$g(x) = x^{10} + x^9 + x^8 + x^6 + x^3 + x^2 + 1, \tag{6.12}$$

with

$$H_g = (1, (2), 508, 950, (951), (952), 803, (804), 764, 0). \tag{6.13}$$

Now, taking the values $H_f - H_g$ modulo $2^{10} - 1$ we get

$$H' = (0, 258, 735, 294, 144, 144, 294, 735, 258, 0). \tag{6.14}$$

Noting the symmetry in the list H' we may deduce the properties shown in Equations (6.1) and (6.9).

It may also be noted that all the information about the relative shifts of the sequences is contained within the pair of sets H_R and H'_R. Here the $_R$ notation describes a list division such that a set S with k elements becomes $S_R = (s_i \mid i < \lceil \frac{k}{2} \rceil)$.

This fragmentation of information may be useful in some analytic applications in which the information of full sets of data on the system is not readily available.

For example a system that leaks information about the content of only half of its memory cells may be run in reverse to obtain information about the content of half of the memory cells of its reciprocal system. Together these two data sets may provide enough exploitable information for a feasible attack to be launched on the original system.

Chapter 7

Conclusion

Upon examination of the Galois LFSR and its implementations there are some results which have significant implications to cryptography, especially in regard to the possible further applications of the properties of MGRs. These results reflect the ability of the Galois LFSR to effectively hold more information over time than its Fibonacci counterpart. We can see this presented in the shifts of the elementary sequences produced by each of the LFSRs memory cells. As was shown here, one implication of these results was in relation to Golić's inversion attack, and also as a property to be used to increase the rate of output of pseudorandom number generators (PRNGs). It is clear that there are more mathematical results yet to be discovered and in this regard, there is scope for much additional future research into the suitability of Galois LFSRs as replacements for Fibonacci LFSRs in cryptographic systems. For example being able to prove the existence of MGRs in the shifts of the LFSR may present a great improvement of the methods used to generate secure NFFs.

Proving this conjecture is of great future interest to the author and many approaches have been considered as starting points. One such approach is to take for example constructions of Golomb rulers as in [6] which have their foundations in finite field theory. If one such construction can be used to create a superset of the set of shifts of a Galois LFSR then one can infer that the child set is also a Golomb ruler.

Also of great interest for further study are the m-tuple bias results explored in Section 5.3. More targeted research in this area has the prospect of discovering a method to exploit the bias in order to mount an attack. As is very often the case, upon discovering a new attack work can then be directed more precisely in order to protect against it. Many of the results in this thesis would be very beneficial to research in this direction.

In a purely mathematical context it is also a step towards tying together many fields of study such as Linear Algebra, Combinatorics and Finite Geometry. The study of these fields leads to many crossovers in the computing community in the areas of Cryptography, Coding Theory and Electrical Engineering. Other advances in the study of computer science that yield faster and more efficient algorithms for calculating discrete logarithms may also provide further benefits in the fields of cryptography when combined with the Galois LFSR.

References

[1] Anderson, R. Searching for the optimum correlation attack. In *Fast Software Encryption* (Leuven, Belgium, 1995), B. Preneel, Ed., vol. 1008 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 137–143.

[2] Babu, R. *Discrete Mathematics.* Pearson Education India, 2012.

[3] Balakrishnan, V. *Introductory Discrete Mathematics.* Courier Corporation, 1991.

[4] Blackburn, S. R. Increasing the rate of output of m-sequences. *Information Processing Letters 51*, 2 (1994), 73–77.

[5] Ding, C., Xiao, G., and Shan, W. *The Stability Theory of Stream Ciphers.* Springer Science & Business Media, 1991.

[6] Drakakis, K. A review of the available construction methods for Golomb rulers. *Advances in Mathematics of Communications 3*, 3 (2009), 235–250.

[7] Everest, G. *Recurrence Sequences.* American Mathematical Society, 2003.

[8] Gardner, D., Sălăgean, A., and Phan, R. C.-W. Efficient generation of elementary sequences. In *Cryptography and Coding* (Oxford, UK, 2013), M. Stam, Ed., vol. 8308 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 16–27.

[9] Giuliani, K. J., and Gong, G. New LFSR-based cryptosystems and the trace discrete log problem (trace-DLP). In *Sequences and Their Applications - SETA 2004* (Seoul, Korea, 2004), T. Helleseth, D. Sarwate, H.-Y. Song, and K. Yang, Eds., vol. 3486 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 298–312.

[10] Golić, J. D. On decimation of linear recurring sequences. *The Fibonacci Quarterly 33*, 5 (1995), 407–411.

[11] Golić, J. D. On the security of nonlinear filter generators. In *Fast Software Encryption* (Cambridge, UK, 1996), D. Gollmann, Ed., vol. 1039 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 173–188.

[12] Golić, J. D., Clark, A., and Dawson, E. Generalized inversion attack on nonlinear filter generators. *IEEE Transactions on Computers 49*, 10 (2000), 1100–1109.

[13] Golomb, S. W. *Shift Register Sequences*, revised ed. Aegean Park Press, California, 1982.

[14] Graham, R. L., and Sloane, N. J. A. On additive bases and harmonious graphs. *SIAM Journal on Algebraic and Discrete Methods 1*, 4 (1980), 382–404.

[15] Grimaldi, R. P. Recurrence Relations. In *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 1999, pp. 178–189.

[16] Hell, M., Johansson, T., and Meier, W. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing 2*, 1 (2007), 86–93.

[17] Kagaris, D. Multiple-seed TPG structures. *IEEE Transactions on Computers 52*, 12 (2003), 1633–1639.

[18] Lempel, A., and Eastman, W. L. High speed generation of maximal length sequences. *IEEE Transactions on Computers C-20*, 2 (1971), 227–229.

[19] Lidl, R., and Niederreiter, H. *Introduction to Finite Fields and their Applications*, 2nd ed. Cambridge University Press, 1994.

[20] MacWilliams, F. J., and Sloane, N. J. A. *The Theory of Error-Correcting Codes*. North-Holland Publishing Company, 1978.

[21] Mantin, I., and Shamir, A. A Practical Attack on Broadcast RC4. In *Fast Software Encryption* (Yokohama, Japan, 2002), M. Matsui, Ed., vol. 2355 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 152–164.

[22] Massey, J. L. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory 15*, 1 (1969), 122–127.

[23] Miller, F. *Telegraphic Code to Insure Privacy and Secrecy in the Transmission of Telegrams*. C.M. Cornwell, 1882.

[24] Mullen, G. L., and Panario, D. *Handbook of Finite Fields.* CRC Press, 2013.

[25] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2011.

[26] Robshaw, M. J. B. Increasing the rate of output for m sequences. *Electronics Letters 27*, 19 (1991), 1710–1712.

[27] Rueppel, R., and Staffelbach, O. Products of linear recurring sequences with maximum complexity. *IEEE Transactions on Information Theory 33*, 1 (1987), 124–131.

[28] Shannon, C. E. Communication theory of secrecy systems. *Bell System Technical Journal 28*, 4 (1949), 656–715.

[29] Smyshlyaev, S. V. Perfectly balanced boolean functions and Golić conjecture. *Journal of Cryptology 25*, 3 (2012), 464–483.

[30] Sălăgean, A., Gardner, D., and Phan, R. C.-W. Index tables of finite fields and modular golomb rulers. In *Sequences and Their Applications – SETA 2012* (Waterloo, Canada, 2012), T. Helleseth and J. Jedwab, Eds., vol. 7280 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 136–147.

[31] Surböck, F., and Weinrichter, H. Interlacing properties of shift-register sequences with generator polynomials irreducible over $GF(p)$. *IEEE Transactions on Information Theory 24*, 3 (1978), 386–389.

[32] Teo, S.-G., Simpson, L. R., and Dawson, E. Bias in the nonlinear filter generator output sequence. *International Journal of Cryptology Research 2*, 1 (2010), 27–37.

[33] Udar, S., and Kagaris, D. LFSR reseeding with irreducible polynomials. In *On-Line Testing Symposium* (Heraklion, Greece, 2007), D. Gizopoulos, T. Mak, M. Nicolaidis, and A. Paschalis, Eds., IEEE, pp. 293–298.

[34] Vernam, G. S. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Transactions of the American Institute of Electrical Engineers XLV* (1926), 295–301.

[35] Zenner, E. Cryptanalysis of LFSR-based pseudorandom generators - a survey. Tech. rep., University of Mannheim, 2004.

Appendix A

Acronyms and Terminology

Acronyms

DLP	discrete logarithm problem
FSM	finite state machine
LFSR	linear-feedback shift register
MGR	modular Golomb ruler
NCF	nonlinear combining function
NCG	nonlinear combining generator
NFF	nonlinear filtering function
NFG	nonlinear filtering generator
OTP	one-time pad
PRNG	pseudorandom number generator
PRS	pseudorandom sequence
XOR	the bitwise exclusive or operation

Polynomial Classifications

Definition (Irreducible Polynomial)**.** [19, Definition 1.57] A polynomial $p \in F[x]$ is said to be *irreducible over* F if p has positive degree and $p = bc$ with $b, c \in F[x]$ implies that either b or c is a constant polynomial.

Definition (Minimal Polynomial)**.** [19, Definition 1.81] If $\theta \in F$ is algebraic over K, then the uniquely determined monic polynomial $g \in K[x]$ generating the ideal $J = \{f \in K[x] : f(\theta) = 0\}$ of $K[x]$ is called the *minimal polynomial* of θ over K.

Definition (Primitive Polynomial)**.** [19, Definition 3.15] A polynomial $f \in \mathbb{F}q[x]$ of degree $m \geq 1$ is called a *primitive* polynomial over $\mathbb{F}_q$ if it is the minimal polynomial over $\mathbb{F}_q$ of a primitive element of $\mathbb{F}_{q^m}$.

Terminology

Characteristic polynomial	(Definition 2.6, page 17)
Elementary sequence	(Definition 2.15, page 19)
Fibonacci linear-feedback shift register	(Definition 2.30, page 25)
Galois linear-feedback shift register	(Definition 2.32, page 27)
Golomb ruler	(Definition 4.1, page 42)
Linear recurrence sequence	(Definition 2.4, page 16)
log notation	(Definition 2.18, page 20)
m-sequence	(Definition 2.10, page 18)
Modular Golomb ruler	(Definition 4.3, page 42)
Polynomial order	(Definition 2.22, page 22)
Recurrence relation	(Definition 2.2, page 16)
Seq notation	(Definition 2.16, page 20)
Sequence decimation	(Definition 2.24, page 22)
Sequence interleaving	(Definition 2.26, page 23)
Sequence notation	(Definition 2.1, page 16)
Sequence phases	(Definition 2.21, page 21)
Shift notation	(Definition 2.11, page 18)
Trace notation	(Definition 2.13, page 19)

Appendix B

Online Encyclopedia of Integer Sequences: Sequences of Interest

Included here are sequences registered at the *Online Encyclopaedia of Integer Sequences* [25]. Each example includes its unique designation, a description, a formula where available and a sample of elements from the start of the sequence itself.

A001037 Number of irreducible polynomials of degree n over $\mathbb{F}_2$.

$$\left(\frac{1}{n}\sum_{d|n} 2^{\frac{n}{d}}\mu(d)\right)_{n=1}^{\infty} \tag{B.1}$$

$$2, 1, 2, 3, 6, 9, 18, 30, 56, 99, 186, 335, 630, 1161, 2182, \dots \tag{B.2}$$

A011260 Number of primitive polynomials of degree n over $\mathbb{F}_2$.

$$\left(\frac{\varphi(2^n - 1)}{n}\right)_{n=1}^{\infty} \tag{B.3}$$

$$1, 1, 2, 2, 6, 6, 18, 16, 48, 60, 176, 144, 630, 756, 1800, 2048, \dots \tag{B.4}$$

A027375 Number of binary sequences with minimal period n.

$$\left(\sum_{d|n} 2^{\frac{n}{d}}\mu(d)\right)_{n=1}^{\infty} \tag{B.5}$$

$$2, 2, 6, 12, 30, 54, 126, 240, 504, 990, 2046, 4020, 8190, \dots \tag{B.6}$$

Appendix C

Index Tables

Below are a collection of example index tables for fields of varying characteristic. The columns headed B_i contain a representation of the summation of powers of α in base i.

Table C.1: Index table of $\mathbb{F}_4$ with $f(x) = x^2 + x + 1$

Power	Sum	B$_2$	B$_4$
−	0	00	0
0	1	01	1
1	α	10	2
2	$\alpha + 1$	11	3

Table C.2: Index table of $\mathbb{F}_8$ with $f(x) = x^3 + x + 1$

Power	Sum	B$_2$	B$_8$
−	0	000	0
0	1	001	1
1	α	010	2
2	α^2	100	4
3	$\alpha + 1$	011	3
4	$\alpha^2 + \alpha$	110	6
5	$\alpha^2 + \alpha + 1$	111	7
6	$\alpha^2 + 1$	101	5

Table C.3: Index table of $\mathbb{F}_{16}$ with $f(x) = x^4 + x + 1$

Power	Sum	B_2	B_{16}
–	0	0000	0
0	1	0001	1
1	α	0010	2
2	α^2	0100	4
3	α^3	1000	8
4	$\alpha + 1$	0011	3
5	$\alpha^2 + \alpha$	0110	6
6	$\alpha^3 + \alpha^2$	1100	C
7	$\alpha^3 + \alpha + 1$	1011	B
8	$\alpha^2 + 1$	0101	5
9	$\alpha^3 + \alpha$	1010	A
10	$\alpha^2 + \alpha + 1$	0111	7
11	$\alpha^3 + \alpha^2 + \alpha$	1110	E
12	$\alpha^3 + \alpha^2 + \alpha + 1$	1111	F
13	$\alpha^3 + \alpha^2 + 1$	1101	D
14	$\alpha^3 + 1$	1001	9

Table C.4: Addition/multiplication table of $\mathbb{F}_{16}$ with $f(x) = x^4 + x + 1$

$_{\times}\diagdown^{+}$	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	**1**	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	**4**	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	6	**5**	7	6	5	4	B	A	9	8	F	E	D	C
4	4	8	C	**3**	1	2	3	C	D	E	F	8	9	A	B
5	5	A	F	7	**2**	3	2	D	C	F	E	9	8	B	A
6	6	C	A	B	D	**7**	1	E	F	C	D	A	B	8	9
7	7	E	9	F	8	1	**6**	F	E	D	C	B	A	9	8
8	8	3	B	6	E	5	D	**C**	1	2	3	4	5	6	7
9	9	1	8	2	B	3	A	8	**D**	3	2	5	4	7	6
A	A	7	D	E	4	9	3	F	5	**8**	1	6	7	4	5
B	B	5	E	A	1	F	4	7	C	2	**9**	7	6	5	4
C	C	B	7	5	9	E	1	A	6	1	D	**F**	1	2	3
D	D	9	4	1	C	8	5	2	F	B	6	3	**E**	3	2
E	E	F	1	D	3	2	C	9	7	6	8	4	A	**B**	1
F	F	D	2	9	6	4	B	1	E	C	3	8	7	5	**A**

Bold values displayed are the result of multiplication since any element added to itself equals 0.

Table C.5: Index table of $\mathbb{F}_{32}$ with $f(x) = x^5 + x^2 + 1$

Power	Sum	B_2	B_{10}
$-$	0	00000	0
0	1	00001	1
1	α	00010	2
2	α^2	00100	4
3	α^3	01000	8
4	α^4	10000	16
5	$\alpha^2 + 1$	00101	5
6	$\alpha^3 + \alpha$	01010	10
7	$\alpha^4 + \alpha^2$	10100	20
8	$\alpha^3 + \alpha^2 + 1$	01101	13
9	$\alpha^4 + \alpha^3 + \alpha$	11010	26
10	$\alpha^4 + 1$	10001	17
11	$\alpha^2 + \alpha + 1$	00111	7
12	$\alpha^3 + \alpha^2 + \alpha$	01110	14
13	$\alpha^4 + \alpha^3 + \alpha^2$	11100	28
14	$\alpha^4 + \alpha^3 + \alpha^2 + 1$	11101	29
15	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$	11111	31
16	$\alpha^4 + \alpha^3 + \alpha + 1$	11011	27
17	$\alpha^4 + \alpha + 1$	10011	19
18	$\alpha + 1$	00011	3
19	$\alpha^2 + \alpha$	00110	6
20	$\alpha^3 + \alpha^2$	01100	12
21	$\alpha^4 + \alpha^3$	11000	24
22	$\alpha^4 + \alpha^2 + 1$	10101	21
23	$\alpha^3 + \alpha^2 + \alpha + 1$	01111	15
24	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha$	11110	30
25	$\alpha^4 + \alpha^3 + 1$	11001	25
26	$\alpha^4 + \alpha^2 + \alpha + 1$	10111	23
27	$\alpha^3 + \alpha + 1$	01011	11
28	$\alpha^4 + \alpha^2 + \alpha$	10110	22
29	$\alpha^3 + 1$	01001	9
30	$\alpha^4 + \alpha$	10010	18

Appendix D

Sequences and Their Applications 2012

This paper was presented at Sequences and Their Applications (SETA) 2012 at the University of Waterloo, Canada.

Index Tables of Finite Fields and Modular Golomb Rulers

Ana Sălăgean, David Gardner, and Raphael Phan

Loughborough University, UK
{A.M.Salagean,D.Gardner2,R.Phan}@lboro.ac.uk

Abstract. For a Galois field GF(2^n) defined by a primitive element α with minimal polynomial f, the index table contains in row i the coordinates of α^i in the polynomial basis $\alpha^{n-1}, \alpha^{n-2}, \ldots, \alpha, 1$. Each column i in this table equals the m-sequence with characteristic polynomial f, shifted cyclically by some offset h_i.

In this paper we show that the set of the n shifts h_i contains large subsets which are modular Golomb rulers modulo $2^n - 1$ (i.e. all the differences are different). Let D be the set of integers j such that the coefficient of x^j in f is non-zero. We prove that the set H_D of shifts corresponding to columns $j \in D$ can be partitioned into two subsets (the columns in the left half of the table and the ones in the right half) each of which is a modular Golomb ruler. Based on this result and on computational data, we conjecture that in fact the whole set H_D is a modular Golomb ruler.

We give a polynomial time algorithm for deciding if given a subset of column positions, the corresponding shifts are a modular Golomb ruler. These results are applied to filter generators used in the design of stream ciphers. Golić recommends that in order to withstand his inversion attack, one of the design requirements should be that the inputs of the non-linear filtering function are taken from positions of a Fibonacci LFSR which form a Golomb ruler. We propose using a Galois LFSR instead and selecting positions such that the corresponding shifts form a modular Golomb ruler. This would allow for a larger number of inputs to be selected (roughly $n/2$ rather than $\sqrt{2n}$) while still satisfying Golić's requirement.

1 Preliminaries

First we recall the definitions of linear recurrent sequences and m-sequences.

Definition 1. *An infinite sequence* $\tilde{s} = s_0, s_1, \ldots$ *with elements in a field* K *is called a* linear recurrent sequence *if there exists a relation of the form* $s_{i+n} = c_{n-1}s_{i+n-1} + \cdots + c_1 s_{i+1} + c_0 s_i$ *for all* $i = 0, 1, \ldots,$ *where* $c_0, c_1, \ldots, c_{n-1} \in K$ *are constants. We associate to it a characteristic polynomial* $f(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$. *If* n *is minimal for the given sequence we call* n *the linear complexity of the sequence. A sequence which has a primitive polynomial as characteristic polynomial is called an* m-sequence.

T. Helleseth and J. Jedwab (Eds.): SETA 2012, LNCS 7280, pp. 136–147, 2012.

Recall that a binary m-sequence of linear complexity n has period $2^n - 1$.

We now introduce a notation for (cyclic) shifts of sequences:

Definition 2. *Given a sequence* $\tilde{s} = s_0, s_1, \ldots$, *we denote by* $\tilde{s} \ll k$ *the sequence obtained by shifting* $\tilde{s}$ *by* k *positions to the left, i.e. the sequence* $s_k, s_{k+1}, \ldots$.

If $\tilde{s}$ *is periodic with period* N *we denote by* $\tilde{s} \gg k$ *the sequence obtained by cyclicly shifting* $\tilde{s}$ *by* k *positions to the right, i.e. the sequence* $s_{N-k}, s_{N-k+1}, \ldots, s_{N-1}, s_0, s_1, \ldots$.

Obviously $(\tilde{s} \gg k) = (\tilde{s} \ll (N - k))$.

Next we recall a few facts about the construction of a finite field with 2^n elements, denoted $\mathrm{GF}(2^n)$.

Throughout the paper we fix $f = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0 \in \mathrm{GF}(2)[x]$ to be a primitive polynomial of degree n (hence $c_0 = 1$) and denote by α a root of f. We define $\mathrm{GF}(2^n)$ as $\mathrm{GF}(2)[x]/\langle f \rangle$, or equivalently as the algebraic extension field of $\mathrm{GF}(2)$ by α.

The elements of $\mathrm{GF}(2^n)$ can be represented in different ways; we are interested in the two most common representations: firstly we have the representation in the polynomial basis $\alpha^{n-1}, \alpha^{n-2}, \ldots, \alpha, 1$, whereby

$$\mathrm{GF}(2^n) = \{r_{n-1}\alpha^{n-1} + r_{n-2}\alpha^{n-2} + \cdots + r_1\alpha + r_0 | r_0, \ldots, r_{n-1} \in \mathrm{GF}(2)\}.$$

Secondly we have the representation as powers of the primitive root α, whereby

$$\mathrm{GF}(2^n) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^n-2}\}.$$

Since the first representation is convenient for addition and the second is convenient for multiplication (and multiplicative inverse), implementations often use lookup tables for conversion between the two representations, also called log/antilog tables or index tables. When n is large however, such tables can no longer be computed/stored due to their exponential size.

Converting from a power of α to the polynomial basis representation is relatively easy (polynomial time). However the reverse problem (given $r_{n-1}, \ldots, r_0$ find i such that $\alpha^i = r_{n-1}\alpha^{n-1} + r_{n-2}\alpha^{n-2} + \cdots + r_1\alpha + r_0$) is difficult and it is known as the Discrete Logarithm Problem (DLP) in $\mathrm{GF}(2^n)$.

We will study the index table that gives the representation of $1, \alpha, \alpha^2, \ldots, \alpha^{2^n-2}$ in the polynomial basis. That is, if we denote

$$\alpha^i = r_i^{(n-1)}\alpha^{n-1} + r_i^{(n-2)}\alpha^{n-2} + \cdots + r_i^{(1)}\alpha + r_i^{(0)},$$

the index table is the $2^n - 1$ by n matrix whose rows are indexed from 0 to $2^n - 2$ and the i-th row is the vector $(r_i^{(n-1)}, r_i^{(n-2)}, \ldots, r_i^{(1)}, r_i^{(0)})$. Note that the rows of this table are precisely all the n-bit vectors except the all-zero one. We will denote column j by $\tilde{r}^{(j)}$ and it will be convenient to view it as a periodic sequence of period $2^n - 1$.

It is known, and not difficult to prove, that each sequence $\tilde{r}^{(j)}$ (being the image under a projection homomorphism of the sequence $1, \alpha, \alpha^2, \ldots$) has characteristic polynomial f. Since f is primitive, $\tilde{r}^{(j)}$ is an m-sequence. For different values of j we obtain different cyclic shifts of this same m-sequence. We will choose $\tilde{r}^{(n-1)}$ as a reference point.

Definition 3. *For $j = 0, \ldots, n-1$ we denote by h_j the integer modulo $2^n - 1$ such that $\tilde{r}^{(j)} = (\tilde{r}^{(n-1)} \gg h_j)$. We denote by H the set $\{h_{n-1}, h_{n-2}, \ldots, h_1, h_0\}$.*

Determining H seems difficult for large fields where the index table cannot be computed in full. This problem was considered by Blackburn in [1]. In [1, Definition 3] he defines a set $\sum(f)$ that would correspond to $H \cup \{h_i - h_j | h_i, h_j \in H\}$, and searches for suitable values in this set in order to increase the rate of output of m-sequences by interleaving. In the next section we will prove certain properties of the elements of H without explicitly computing them.

It will be convenient to use the trace representation for m-sequences:

Theorem 1. *[5, Theorem 6.21] The elements of an m-sequence $\tilde{s} = s_0, s_1, \ldots$ over* GF(2) *can be expressed as $s_i = Tr(a\alpha^i) = \sum_{j=0}^{n-1} a^{2^j}(\alpha^{2^j})^i$, where α is a primitive root of the primitive characteristic polynomial of s and $a \in \mathrm{GF}(2^n)$, $a \neq 0$, is a constant, uniquely determined by the first n elements of the sequence.*

Since we will work with a fixed primitive polynomial f, it is only the constant a in the theorem above that determines which of the $2^n - 1$ shifts of the m-sequence we are dealing with. It is therefore convenient to introduce the following notation:

Definition 4. *We define $\mathrm{Seq}_\alpha(a)$ (also denoted $\mathrm{Seq}(a)$ if α is clear from the context) as the sequence $\tilde{s}$ whose i-th element is represented by*

$$s_i = Tr(a\alpha^i) = \sum_{j=0}^{n-1} a^{2^j}(\alpha^{2^j})^i. \tag{1}$$

Seq is linear, i.e. for any $a, b \in \mathrm{GF}(2^n)$ and $c \in \mathrm{GF}(2)$ we have:

$$\mathrm{Seq}(a) + \mathrm{Seq}(b) = \mathrm{Seq}(a+b) \tag{2}$$

$$c\,\mathrm{Seq}(a) = \mathrm{Seq}(ca) \tag{3}$$

The effect of shifting on sequences $\mathrm{Seq}(a)$ can be described as follows:

Lemma 1. *Let $a, a_1, a_2 \in GF(2^n)^*$ and h an integer. Then:*
(i) $(\mathrm{Seq}(a) \ll h) = \mathrm{Seq}(a\alpha^h)$ and $(\mathrm{Seq}(a) \gg h) = \mathrm{Seq}(a\alpha^{-h})$
(ii) $\mathrm{Seq}(a_2) = (\mathrm{Seq}(a_1) \gg h)$ where h is the discrete logarithm of $a_1 a_2^{-1}$.

Proof. (i) The i-th element of $(\mathrm{Seq}(a) \ll h)$ is the $(i+h)-th$ element of $\mathrm{Seq}(a)$,

$$s_{i+h} = \sum_{j=0}^{n-1} a^{2^j}(\alpha^{2^j})^{i+h} = \sum_{j=0}^{n-1} (a\alpha^h)^{2^j}(\alpha^{2^j})^i$$

which is indeed the i-th element of the sequence $\mathrm{Seq}(a\alpha^h)$ as in (1).

(ii) Write $\mathrm{Seq}(a_2) = \mathrm{Seq}(a_1 a_2 a_1^{-1}) = \mathrm{Seq}(a_1\alpha^{-h})$ and then use (i). □

The following notion appears in the literature in different equivalent forms and under different names: Golomb ruler, finite Sidon set, full positive difference set, etc.

Definition 5. *A* Golomb ruler *of order m is a set of integers $\{b_0, \dots, b_{m-1}\}$ with $b_0 < b_1 < \dots < b_{m-1}$, such that all the positive pairwise differences of elements are unique,i.e. $b_j - b_i \neq b_l - b_k$, for all $(i,j) \neq (k,l)$, $i < j$ and $k < l$.*

A modular Golomb ruler *modulo N is a set $\{b_0, \dots, b_{m-1}\}$ of numbers modulo N such that all the pairwise differences of elements are unique modulo N, i.e. $(b_j - b_i) \bmod N \neq (b_l - b_k) \bmod N$, for all $(i,j) \neq (k,l)$.*

There is no general construction for optimal (modular) Golomb rulers (i.e. minimum $b_{m-1} - b_0$ for given order m); tables for the currently known optimal values are available see [4], the Online Encyclopedia of Integer Sequences and the references therein. The following is immediate:

Lemma 2. *Let $B = \{b_0, \dots, b_{m-1}\}$ with $0 \le b_0 < b_1 < \dots < b_{m-1} < N$ be a Golomb ruler. If $b_{m-1} - b_0 < N/2$ then B is also a modular Golomb ruler modulo N.*

Proof. For all $i < j$, we consider a first set of differences as the differences $b_j - b_i$. These are all different because B is a Golomb ruler. Moreover, $b_j - b_i \le b_{m-1} - b_0 < N/2$. The second set of differences $b_i - b_j = N - (b_j - b_i) > N/2$ are all different among themselves, and also different from the first set of differences. □

2 Modular Golomb Rulers within the Set of Shifts of the Index Table of a Galois Ring

In this section we show that certain non-trivial subsets of H (where H is defined in Definition 3) are modular Golomb rulers. Moreover, we show that for suitable choices of the primitive polynomial f these subsets contain about half the elements of H.

For a start, all h_j are different. (If we assumed there exist $h_i = h_j$ then in each row of the index table entries i and j are identical. However, this is not possible as the table contains as rows all the possible binary vectors except the all-zero one.) As an easy consequence $h_j - h_i \neq h_k - h_i$ for all $j \neq k$.

Lemma 3. *$\tilde{r}^{(0)} = (\tilde{r}^{(n-1)} \gg 1)$ and $\tilde{r}^{(j)} = ((\tilde{r}^{(j-1)} + c_j \tilde{r}^{(n-1)}) \gg 1)$ for $1 \le j \le n-1$.*

Proof.

$$\begin{aligned}
\alpha^{i+1} &= r_{i+1}^{(n-1)}\alpha^{n-1} + r_{i+1}^{(n-2)}\alpha^{n-2} + \dots + r_{i+1}^{(1)}\alpha + r_{i+1}^{(0)} = \alpha\alpha^i \\
&= \alpha(r_i^{(n-1)}\alpha^{n-1} + r_i^{(n-2)}\alpha^{n-2} + \dots + r_i^{(1)}\alpha + r_i^{(0)}) \\
&= r_i^{(n-1)}\alpha^{n} + r_i^{(n-2)}\alpha^{n-1} + \dots + r_i^{(1)}\alpha^2 + r_i^{(0)}\alpha \\
&= r_i^{(n-1)}(c_{n-1}\alpha^{n-1} + \ldots + c_1\alpha + c_0) + r_i^{(n-2)}\alpha^{n-1} + \dots + r_i^{(1)}\alpha^2 + r_i^{(0)}\alpha \\
&= (r_i^{(n-1)}c_{n-1} + r_i^{(n-2)})\alpha^{n-1} + \dots + (r_i^{(n-1)}c_1 + r_i^{(0)})\alpha + (r_i^{(n-1)}c_0).
\end{aligned}$$

Since $\alpha^{n-1}, \alpha^{n-2}, \dots, \alpha, 1$ is a vector space basis, we have $r_{i+1}^{(j)} = r_i^{(n-1)}c_j + r_i^{(j-1)}$ and $r_{i+1}^{(0)} = r_i^{(n-1)}c_0$. □

Corollary 1. *(i)* $h_{n-1} = 0, h_0 = 1$ *and* $h_j = h_{j-1} + 1$ *for all* j *for which* $c_j = 0$.
(ii) If $f = x^n + x^j + 1$ *is a trinomial, then* $H = \{0, 2^n - 2, 2^n - 3, \ldots, 2^n - (n - j), j, j-1, \ldots, 3, 2, 1\}$

Hence for determining H it suffices to determine those h_j for which $c_j \neq 0$.

Let a be such that $\tilde{r}^{(n-1)} = \mathrm{Seq}(a)$. The value of a can be computed from the initial terms of $\tilde{r}^{(n-1)}$ but this will not be necessary for our purposes.

Theorem 2. *Let* $z_j = c_{j+1} + c_{j+2}\alpha + \cdots + c_{n-1}\alpha^{n-j-2} + \alpha^{n-j-1}$. *Then:*
(i) $z_0, z_1, \ldots, z_{n-1}$ *form a vector space basis for* $\mathrm{GF}(2^n)$.
(ii) $\tilde{r}^{(j)} = \mathrm{Seq}(az_j)$, *i.e.* $\alpha^{-h_j} = z_j$ *for all* $j = 0, \ldots n-1$.
(iii) $h_j - h_i$ *equals the discrete logarithm of* $z_i z_j^{-1}$.
(iv) If j *is such that* $c_j \neq 0$ *then* $h_j = h_{j-1} + 1 - h$ *where* h *equals the discrete logarithm of* $1 + z_{j-1}^{-1}$.

Proof. For (i), note that the z_j have different degrees. The proof of (ii) is by induction on j using Lemmas 1 and 3 as well as the linearity of Seq, i.e. equations (2) and (3). For (iii), write $\alpha^{h_j - h_i} = \alpha^{-h_i}\alpha^{h_j} = z_i z_j^{-1}$. Finally, (iv) is a particular case of (iii). □

Determining H is therefore equivalent to solving the particular instances of the DLP problem $\alpha^{-h_j} = z_j$, for $j = 0, 1, \ldots, n-1$ or alternatively solving particular instances of the State-based DLP as defined by Giuliani and Gong in [2, Definition 7]. Namely, given the n initial terms of $\tilde{r}^{(j)}$, determine the starting position h_j where the n terms 0,0,…,0,1 appear in $\tilde{r}^{(j)}$. It is shown in [2, Theorem 3] that the State-based DLP is equivalent to the DLP.

Theorem 3. *Let* $D \subseteq \{0, 1, \ldots, n-1\}$ *be a set of indices and* $H_D = \{h_i | i \in D\}$ *be the set of corresponding values of shifts. The set* H_D *is a modular Golomb ruler (modulo* $2^n - 1$*) if and only if for all distinct pairs* (i, j), (k, l) *of elements in* D *with* $i < j$, $k < l$, $j - i \leq l - k$ *we have*

$$z_i z_j^{-1} \neq z_k z_l^{-1} \tag{4}$$

$$z_i z_j^{-1} \neq z_k^{-1} z_l \tag{5}$$

Proof. Use Theorem 2(iii) and Definition 5. □

Based on the theorem above, Algorithm 1 decides whether H_D is a modular Golomb ruler for a given D.

Theorem 4. *Algorithm 1 has a time complexity of* $\mathcal{O}(n^4)$ *and needs* $\mathcal{O}(n^3)$ *extra memory space.*

Proof. Computing the polynomial basis representation of $z_i z_j^{-1}$ and of $z_i^{-1} z_j$ takes $\mathcal{O}(n^2)$ steps. The list L has at most $n(n-1)$ elements of n bits each, i.e. a total of $\mathcal{O}(n^3)$ bits. With an appropriate data structure, we can maintain the elements of L in lexicographic order and we do binary search to find out if an element is in the list or to insert a new element. We would then need $\log(n^2) = 2\log n$ list element comparisons, and each comparison takes n steps. Hence all operations inside the two nested **for** loops take $\mathcal{O}(n^2)$ steps. □

Algorithm 1. GolombRulerDecision(f, D)

```
Input: $f$ a primitive polynomial of degree $n$; $D \subseteq \{0, 1, \ldots, n-1\}$.
Output: True/False signifying whether $\{h_j | j \in D\}$ is a modular Golomb ruler.
begin
Initialise $L$ to the empty list
for $i = 0, 1, \ldots, n-1$ do
  for $j = i+1, \ldots, n-1$ do
    Compute the polynomial basis representation of $z_i z_j^{-1}$ and of $z_i^{-1} z_j$
    if ($z_i z_j^{-1}$ is in $L$) or ($z_i^{-1} z_j$ is in $L$) then
      return(False)
    else
      Insert $z_i z_j^{-1}$ and $z_i^{-1} z_j$ in $L$
    end if
  end for
end for
return(True)
end
```

For certain subsets of H we will be able to show that they are always modular Golomb rulers. Intuitively, runs of zero coefficients in f correspond to runs of consecutive integers in the corresponding shifts h_j by Corollary 1(i). In such regions of consecutive integers we can only choose very small subsets which are Golomb rulers. A much more promising source of Golomb ruler subsets comes from those h_j for which $c_j \neq 0$.

Next we will gather sufficient conditions for (4) and (5) to hold.

Lemma 4. *We use the notations of Theorem 3 and assume* c_i, c_j, c_k, c_l *are all non-zero.*
Each of the following conditions is sufficient for (4) to be satisfied:
(i) $j - i = l - k$
(ii) $i + l \leq n$
(iii) $j + k \geq n - 1$
Each of the following conditions is sufficient for (5) to be satisfied:
(iv) $j + l \leq n$
(v) $i + k \geq n - 1$.

Proof. We write $z_i = \alpha^{-(i+1)} v_i$ where $v_i = 1 + c_1\alpha + c_2\alpha^2 + \ldots + c_i\alpha^i$. We denote by next($i$) the smallest index $u > i$ such that $c_u \neq 0$. Note that next(i) $\leq j$.

The general idea of these proofs is that we assume for a contradiction that equality holds in (4) or in (5), respectively. We then simplify this equation to the point that only powers of α between $\alpha^0 = 1$ and α^{n-1} appear. Since this is a vector space basis of GF(2^n), an equality holds if and only if for all i the coefficients of the corresponding α^i are identical on the two sides of the equality. We then prove that this is not the case for our equality, obtaining thus a contradiction.

(i) Note that in this case we cannot have $i = k$, because $j - i = l - k$ would then imply $j = l$ and therefore $(i, j) = (k, l)$. Assuming equality in (4) and using

$j-i=l-k$, this simplifies to $v_kv_j=v_iv_l$. Writing $v_j=v_i+\alpha^{i+1}(c_{i+1}+c_{i+2}\alpha+\cdots+c_j\alpha^{j-i-1})$ and similarly for v_l the equality further simplifies to either

$$v_k(c_{i+1}+c_{i+2}\alpha+\cdots+c_j\alpha^{j-i-1})=\alpha^{k-i}v_i(c_{k+1}+c_{k+2}\alpha+\cdots+c_l\alpha^{l-k-1})$$

for the case $i<k$, or

$$\alpha^{i-k}v_k(c_{i+1}+c_{i+2}\alpha+\cdots+c_j\alpha^{j-i-1})=v_i(c_{k+1}+c_{k+2}\alpha+\cdots+c_l\alpha^{l-k-1})$$

for the case $i>k$. In the case of $i<k$, on the l.h.s. the smallest power of α is next(i) and the highest is $l-1$ and on the r.h.s. the smallest power is $k-i+\text{next}(k)$ and the highest is $l-1$. Since all powers of α are below n, all the corresponding coefficients of the powers of α must coincide on the l.h.s and r.h.s. This implies $\text{next}(i)=k-i+\text{next}(k)$, i.e. $i+\text{next}(i)=k+\text{next}(k)$. However, one can see that this is a contradiction because $i<k$, which due to the way we defined next implies $\text{next}(i)\le k<\text{next}(k)$. The case $i>k$ leads to a contradiction in a similar way.

(ii) We may assume $l-k>j-i$, as the case $l-k=j-i$ was covered by (i). Assuming equality in (4) we obtain $\alpha^{(l-k)-(j-i)}v_jv_k=v_lv_i$. On the l.h.s. the lowest power of α is $(l-k)-(j-i)>0$ and the highest is $l+i$. On the r.h.s. the lowest is 0 and the highest is $l+i$. The highest powers on both sides cancel out, leaving only powers of at most $l+i-1\le n-1$. Since all powers of α are below n, all the corresponding coefficients of the powers of α must coincide on the l.h.s and r.h.s. However this cannot be the case as the lowest powers with a non-zero coefficient are different on the two sides.

(iii) Again we may assume $l-k>j-i$, as the case $l-k=j-i$ was covered by (i). Note $i+l>j+k\ge n-1$. Assuming equality in (4) gives $z_iz_l=z_jz_k$. The powers of α range from some integer ≥ 0 to $2(n-1)-(i+l)<n-1$ on the l.h.s. and from some integer ≥ 0 to $2(n-1)-(j+k)\le n-1$ on the r.h.s.. That means the coefficients must be identical, hence $2(n-1)-(i+l)=2(n-1)-(j+k)$. But that implies $l-k=j-i$, which is not true.

(iv) Assuming equality in (5) gives $v_jv_l=\alpha^{l-k+j-i}v_iv_k$. Again, the powers of α range on the l.h.s. from 0 to $j+l$ and on the r.h.s from $l-k+j-i>0$ to $j+l$, with the highest ones canceling out and leaving powers of at most $j+l-1\le n-1$. The range needs to be the same on both sides. Contradiction.

(v) Note $j+l>i+k\ge n-1$. Assuming equality in (5) gives $z_jz_l=z_iz_k$. The powers of α range from some integer ≥ 0 to $2(n-1)-(j+l)<n-1$ on the l.h.s. and from some integer ≥ 0 to $2(n-1)-(i+k)\le n-1$ on the r.h.s.. Therefore the coefficients must be identical, which is not true as $2(n-1)-(j+l)<2(n-1)-(i+k)$. □

Theorem 5. *Let* $D=\{i|c_i\ne 0\}$ *and let* $D_1=\{i\in D, i\le \frac{n}{2}\}$ *and* $D_2=\{i\in D, i\ge \frac{n-1}{2}\}$. *Then* $H_{D_1}=\{h_i|i\in D_1\}$ *and* $H_{D_2}=\{h_i|i\in D_2\}$ *are modular Golomb rulers (modulo* 2^n-1*).*

Proof. For H_{D_1} all indices satisfy conditions (ii) and (iv) in Lemma 4. For H_{D_2} all indices satisfy conditions (iii) and (v) in Lemma 4. □

Conjecture 1. $H_D = \{h_i | c_i \neq 0\}$ is a modular Golomb ruler (modulo $2^n - 1$).

In view of Lemma 4, the missing cases for proving the conjecture are: showing that (4) holds when $j + k < n - 1$ and it also holds when $n < i + l$; showing that (5) holds when $i + k < n - 1$ and also when $n < j + l$. The experiments in the following section support this conjecture. Moreover, they allow us to state:

Proposition 1. *For all primitive polynomials f of degree 23 or less, $H_D = \{h_i | c_i \neq 0\}$ is a modular Golomb ruler (modulo $2^n - 1$).*

Finally, note that these results mean that H can have very large subsets which are modular Golomb rulers. One of the subsets in Theorem 5 will have at least $\lceil (\mathrm{wt}(f) - 1)/2 \rceil$ elements, where $\mathrm{wt}(f)$ is the Hamming weight of f (number of non-zero coefficients). If Conjecture 1 is true for a particular f (and this can be checked by Algorithm 1), the subset obtained is even larger, namely it has $\mathrm{wt}(f) - 1$ elements.

For many, but not all n, there exists a primitive polynomial of weight n for n odd or of weight $n-1$ for n even. It seems likely that for all n there are primitive polynomials of weight close to n, and therefore H contains in these cases a modular Golomb ruler subset consisting of almost the whole H (if Conjecture 1 is true). Moreover, it seems likely that for any n there are primitive polynomials f for which all or almost all coefficients in the lower half of f are non-zero, and therefore H contains in these cases a modular Golomb ruler subset consisting of half or almost half of the elements of H (by Theorem 5, so regardless whether Conjecture 1 is true).

3 Experiments

Brute force experimentation was performed on all Galois fields $\mathrm{GF}(2^n)$ with n from 2 to 23, examining all the different primitive polynomials for each n. In each case the full index table was produced, and the shifts $h_{n-1}, h_{n-2}, \ldots, h_1, h_0$ were computed by direct examination of the table. Some examples are described in Table 1, with the primitive polynomial f represented as $1c_{n-1}c_{n-2}\ldots c_1 1$. It was then verified (using Definition 5) that removing those h_j for which $c_j = 0$ (shown in brackets in Table 1) leaves indeed a subset which is a modular Golomb ruler. Thus it was verified that Conjecture 1 holds for all primitive polynomials up to degree $n = 23$. For $24 \leq n \leq 29$ we ran Algorithm 1 for all primitive polynomials f with $\mathrm{wt}(f) \geq n - 1$ and again Conjecture 1 was verified.

4 An Application to Galois LFSRs and Filter Generators

Linear recurrent sequences are often generated in practice by hardware devices called Linear Feedback Shift Registers (LFSR). There are two common types of LFSR, usually called the Fibonacci LFSR and the Galois LFSR. We recall these notions here. The registers of a Fibonacci LFSR of length n will be denoted by

Table 1. A selection of primitive polynomials f and the corresponding shifts H

$n = 7,\ f = 11111101,\ \mathrm{wt}(f) = 7,$ $H = \{0, 18, 119, 54, 39, (2), 1\}$
$n = 9,\ f = 1111000111,\ \mathrm{wt}(f) = 7,$ $H = \{0, 326, 461, (467), (466), (465), 464, 328, 1\}$
$n = 15,\ f = 1100000111100111,\ \mathrm{wt}(f) = 9,$ $H = \{0, (3971), (3970), (3969), (3968), (3967), 3966,$ $30091, 12457, 28329, (24624), (24623), 24622, 3973, 1\}$
$n = 21,\ f = 1010101011110110001101,\ \mathrm{wt}(f) = 13,$ $H = \{(0), 2097150, (1796558), 1796557, (1333708),$ $1333707, (1195372), 1195371, 1508706, 363026, 820032,$ $(1536625), 1536624, 543838, (134466), (134465), (134464),$ $134463, 1796561, (2), 1\}$
$n = 23,\ f = 111111110111111111111111,\ \mathrm{wt}(f) = 23,$ $H = \{0, 873419, 3430060, 2620257, 1534122, 7733539,$ $3311431, (6113933), 6113932, 7496295, 3308273, 7951902,$ $226119, 3941673, 4712702, 6113941, 3311438, 7733545,$ $1534127, 2620261, 3430063, 873421, 1\}$

$Q_0, Q_1, \ldots Q_{n-1}$. The content of register Q_j at time i will be denoted $q_i^{(j)}$ and the contents of all the registers at time i are called the state at time i. The initial state is the state at time 0. The sequence $\tilde{q}^{(j)}$ consists of the values of register Q_j in time, i.e. $q_0^{(j)}, q_1^{(j)}, \ldots$. Similarly for a Galois LFSR we denote the registers by $R_{n-1}, R_{n-2}, \ldots, R_0$ and the contents of the register R_j in time by $\tilde{r}^{(j)}$.

Definition 6. *A Fibonacci LFSR of length n (see Fig. 1) with characteristic polynomial $f(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ will update itself at each clock interval i according to the following*

$$q_{i+1}^{(j)} = \begin{cases} c_{n-1}q_i^{(n-1)} + \cdots + c_1 q_i^{(1)} + c_0 q_i^{(0)} & \text{if } j = n-1 \\ q_i^{(j+1)} & \text{otherwise.} \end{cases}$$

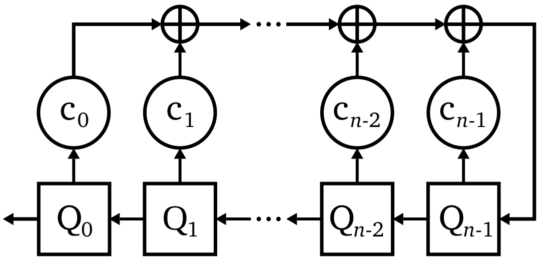


Fig. 1. A Fibonacci style LFSR

Definition 7. *A Galois LFSR of length n (see Fig. 2) with characteristic polynomial $f(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1x + c_0$ will update itself at each clock interval i according to the following*

$$r_{i+1}^{(j)} = \begin{cases} c_0 r_i^{(n-1)} & \textit{if } j = 0 \\ r_i^{(j-1)} + c_j r_i^{(n-1)} & \textit{otherwise.} \end{cases}$$

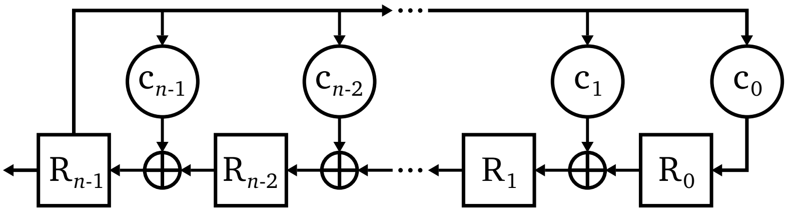


Fig. 2. A Galois style LFSR

The output of the Fibonacci LFSR is taken from register Q_0, i.e. equals $\tilde{q}^{(0)}$; the output of the Galois LFSR is taken from register R_{n-1}, i.e. equals $\tilde{r}^{(n-1)}$. It is known that a Fibonacci LFSR and a Galois LFSR with the same characteristic polynomial will produce the same output sequence provided the initial states are suitably chosen. We now fix the characteristic polynomial f to be the same primitive polynomial in both LFSRs, so both produce the same m-sequence.

In the Fibonacci LFSR each sequence representing the content of a register is equal to the neighbouring sequence shifted by one position. More precisely, $\tilde{q}^{(j)} = (\tilde{q}^{(j-1)} \ll 1)$. Taking the output sequence $\tilde{q}^{(0)}$ as reference, $\tilde{q}^{(j)} = (\tilde{q}^{(0)} \ll j) = (\tilde{q}^{(0)} \gg (2^n - 1 - j))$.

For a Galois LFSR with a primitive polynomial f which has a primitive root α, the state $(r_i^{(n-1)}, r_i^{(n-2)}, \ldots, r_i^{(0)})$ at time i can be interpreted as the coefficients of the element $r_i^{(n-1)}\alpha^{n-1} + r_i^{(n-2)}\alpha^{n-2} + \cdots + r_i^{(0)}$ of GF(2^n). Then the state at time i will be α^{i+k} where k is such that α^k corresponds to the initial state. We can see now that each sequence $\tilde{r}^{(j)}$ coincides with the sequence $\tilde{r}^{(j)}$ defined in Section 1, shifted by k positions to the left. We are only interested in the relative shifts of different $\tilde{r}^{(j)}$, hence the shifts by k will cancel out. Taking the output sequence $\tilde{r}^{(n-1)}$ as reference point, the other sequences $\tilde{r}^{(j)}$ can be obtained by shifting $\tilde{r}^{(n-1)}$ to the right by h_j positions, where h_j is as defined in Definition 3.

For designing stream ciphers, one of the classical constructions for the keystream generator is the filter generator (see Fig. 3). It consists of a binary LFSR (usually Fibonacci LFSR) generating an m-sequence of period $2^n - 1$ and a boolean function $g : \mathrm{GF}(2)^k \to \mathrm{GF}(2)$ with $k \leq n$, called a non-linear filtering function. The output of the generator is obtained by applying the function g to k selected registers of the LFSR, say $j_1, j_2, \ldots, j_k$. Hence the output at time i equals $g(q_i^{(j_1)}, q_i^{(j_2)}, \ldots, q_i^{(j_k)})$.

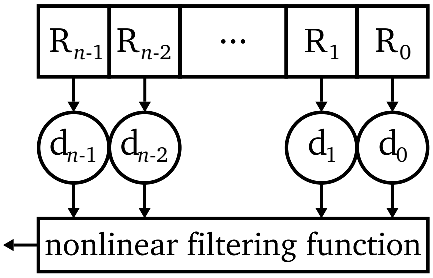


Fig. 3. An LFSR fed NFF

There is a large number of results concerning the recommended choice of the function g and the "tapped" positions $j_1, j_2, \ldots, j_k$ in order to avoid various cryptanalysis attacks. Here we are interested in the results of Golić [3]. In [3, Theorem 2], Golić gives a sufficient condition for a non-linear filtering function g to produce a purely random output provided its inputs come from a purely random sequence $\tilde{z}$ in such a way that the output at time i equals $g(z_{i-j_1}, z_{i-j_2}, \ldots, z_{i-j_k})$ for fixed tapping positions $j_1, \ldots, j_k$. In our notation, the input of g at time i consists of the i-th elements of the sequences $(\tilde{z} \gg j_1), (\tilde{z} \gg j_2), \ldots, (\tilde{z} \gg j_k)$. The sufficient condition is that g is linear in the first or last variable. Golić conjectured the condition was also necessary, see [6] for further results on this conjecture. Based on this result, Golić introduces an inversion attack for generators which use a Fibonacci LFSR and tapping position $j_1, \ldots, j_k$ together with a non-linear filtering function which is linear in the first or last variable. He recommends that for withstanding this attack one design criterion is that the tapped positions of the Fibonacci LFSR should form a full positive difference set (Golomb ruler). Note that since the tapped positions are in the range 0 to $n-1$, they also form a modular Golomb ruler modulo $2^n - 1$, by Lemma 2, as $n - 1 < (2^n - 1)/2$ for all $n \geq 2$.

Golić's results still apply if we use an m-sequence of complexity n and we buffer t terms for some $t \geq n$; we can then tap any positions $j_1, j_2, \ldots, j_k$ provided $\max_{u=1,\ldots,k}(j_u) - \min_{u=1,\ldots,k}(j_u) \leq t$. We suspect that in that case Golić's design criterion would need to be enhanced, requiring that the tapped positions be a modular Golomb ruler modulo $2^n - 1$ (the period of the m-sequence) rather than simply a Golomb ruler. The two are no longer equivalent if the range $\max_{u=1,\ldots,k}(j_u) - \min_{u=1,\ldots,k}(j_u)$ exceeds $2^{n-1} - 1$. Buffering $t > n$ terms would allow a larger number of positions to be tapped while still satisfying Golić's design criterion. This would come at the cost of extra storage.

We propose constructing a filter generator that uses a Galois LFSR with a dense primitive polynomial f. We then select positions $D = \{i_1, i_2, \ldots, i_k\} \subseteq \{0, 1, \ldots, n-1\}$ as inputs to the filtering function in such a way that $\{h_{i_1}, h_{i_2}, \ldots, h_{i_k}\}$ is a modular Golomb ruler. The filter generator thus constructed would be equivalent to tapping positions $j_1 = h_{i_1}, j_2 = h_{i_2}, \ldots, j_k = h_{i_k}$ of a buffered section of length $t = \min_{l=1,\ldots,k}(\max_{u=1,\ldots,k}((j_l - j_u) \bmod (2^n - 1))$ of the m-sequence, with the advantage that we do not need to actually buffer such a long section, we are only using the n memory registers of the Galois LFSR. This construction would satisfy Golić's design criterion. It remains to be seen whether it would be susceptible to other forms of attack.

According to the discussion at the end of Section 2, we can choose $D = \{i | c_i \neq 0\}$ and check whether Conjecture 1 is true in this case by running Algorithm 1. If the answer is positive, we have $k = \mathrm{wt}(f) - 1$, which can be very close to n for suitably chosen f. If Algorithm 1 returns a negative result, we can still chose $D = \{i | c_i \neq 0, i \leq n/2\}$ and H_D is guaranteed to be a modular Golomb ruler by Theorem 5. For suitably chosen f we can then have k equal, or lower but very close to $\lfloor n/2 \rfloor + 1$. If we had to choose inputs from a Fibonacci LFSR of length n so that they are a Golomb ruler, the well known bound $n \geq k(k-1)/2$ would mean $k < \sqrt{2n} + 1$, hence a much smaller number of inputs are available. Equivalently, if we required some fixed number k of inputs, we would need a much larger length n for the Fibonacci LFSR, namely more than $k(k-1)/2$ compared to approximately $2k$ for the Galois LFSR. The following example illustrates this:

Example 1. The first example in Table 1, after removing the elements in brackets, produces a modular Golomb ruler of order $k = 6$. A Fibonacci LFSR of same length $n = 7$ would allow us to produce a Golomb ruler (which by Lemma 2 would also be a modular Golomb ruler modulo $2^n - 1$) of only $k = 4$ elements. For $k = 6$ elements we would need a Fibonacci LFSR of length $n = 17$ (see [4]).

The last example in Table 1 is a Galois LFSR of length $n = 23$ and after removing the elements in brackets, produces a modular Golomb ruler of order $k = 22$. A Fibonacci LFSR of same length $n = 23$ will allow us to produce a Golomb ruler (which by Lemma 2 would also be a modular Golomb ruler modulo $2^n - 1$) of order only $k = 6$. For order $k = 22$ we would need a Fibonacci LFSR of length $n = 356$ (see [4]).

Acknowledgements. We would like to thank Simon Blackburn for a useful discussion regarding his paper [1].

References

1. Blackburn, S.R.: Increasing the Rate of Output of m-Sequences. Information Processing Letters 51, 73–77 (1994)
2. Giuliani, K., Gong, G.: New LFSR-Based Cryptosystems and the Trace Discrete Log Problem (trace-DLP). In: Helleseth, T., Sarwate, D., Song, H.-Y., Yang, K. (eds.) SETA 2004. LNCS, vol. 3486, pp. 298–312. Springer, Heidelberg (2005)
3. Golić, J.D.: On the Security of Nonlinear Filter Generators. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 173–188. Springer, Heidelberg (1996)
4. Graham, R.L., Sloane, N.J.A.: On Additive Bases and Harmonious Graphs. Siam Journal on Algebraic and Discrete Methods 1, 382–404 (1980)
5. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications. Cambridge University Press (1994)
6. Smyshlyaev, S.V.: Perfectly Balanced Boolean Functions and Golić Conjecture. Journal of Cryptology, 1–20 (2011)

Appendix E

IMA International Conference on Cryptography and Coding 2013

This paper was presented at the IMA International Conference on Cryptography and Coding (IMACC) 2013 in Oxford, UK.

Efficient Generation of Elementary Sequences

David Gardner[1], Ana Sălăgean[1], and Raphael C.-W. Phan[2]

[1] Department of Computer Science, Loughborough University, UK
`{D.Gardner2,A.M.Salagean}@lboro.ac.uk`
[2] Faculty of Engineering, Multimedia University, Malaysia
`raphael@mmu.edu.my`

Abstract. Given an irreducible non-primitive polynomial g of degree n over $\mathbb{F}_2[x]$ we aim to compute in parallel all the elementary sequences with minimal polynomial g (i.e. one sequence from each class of equivalence under cyclic shifts). Moreover, they need to each be in a suitable phase such that interleaving them will produce an m-sequence with linear complexity $\deg(g)$; this m-sequence is therefore produced at the rate of $q = (2^n - 1)/\mathrm{ord}(g)$ bits per clock cycle. A naive method would use q LFSRs so our aim is to use considerably fewer. We explore two approaches: running a small number of Galois LFSRs with suitable seeds and using certain registers, possibly with a small amount of buffering; alternatively using only one (Galois or Fibonacci) LFSR and computing certain linear combinations of its registers. We ran experiments for all irreducible polynomials of degree n up to 14 and for each n we found that efficient methods exist for at least one m-sequence. A combination of the two approaches above is also described.

Keywords: Fibonacci LFSR, Galois LFSR, m-sequences, interleaving, elementary sequences.

1 Introduction

The linear feedback shift register (LFSR) has become a standard way to generate linearly recurrent sequences and in particular m-sequences, i.e. sequences which have the maximum period given the length of the LFSR. m-sequences are commonly used as good approximations of random binary sequences, so called pseudo-random or pseudo-noise sequences. Sequences with such properties are invaluable in symmetric key cryptographic systems implemented both in hardware and software. Hence rapid generation of m-sequences without excessive memory requirements is desirable. There are many other areas of applications for m-sequences.

Some authors have explored various ways to artificially increase the rate of output of such cryptographic systems; Robshaw[6] described a method of interleaving a number of phases of the desired m-sequence, all these phases being generated synchronously by separate LFSRs (see also [3]). Blackburn[1] then extended this idea by using a smaller number of Galois LFSR in order to produce the required synchronous sequences exploiting the fact that a Galois LFSR with

M. Stam (Ed.): IMACC 2013, LNCS 8308, pp. 16–27, 2013.

a primitive feedback polynomial can produce a greater spread of phases of a single m-sequence than an equivalent Fibonacci LFSR. The main difficulty in [1] lies in identifying a primitive feedback polynomial that will produce sequences in the required phases.

Surböch and Weinrichter[8] explored interleaving "elementary sequences" instead of m-sequences. In the present paper we are further developing this approach. An elementary sequence is one whose minimal polynomial g is irreducible but not necessarily primitive. For a fixed non-primitive g, there are several such sequences, not all of which are cyclic shifts of one another. More precisely, with equivalence given by cyclic shifts, there are exactly $q = (2^n - 1)/\mathrm{ord}(g)$ classes, each having $\mathrm{ord}(g)$ elements.

Given an m-sequence s of length $2^n - 1$ it is known that for any proper factor q of $2^n - 1$, the (improper) decimations of s by q are elementary sequences, all having the same irreducible minimal polynomial g.

Taking the reverse approach, we can interleave q elementary sequences in order to obtain an m-sequence. Not every arbitrary collection of q elementary sequences with the same irreducible minimal polynomial will produce an m-sequence by interleaving. The sequences need to satisfy further conditions: to be inequivalent under cyclic shifts, and to be in a certain phase, see Theorem 4. Our aim is to obtain in an efficient way exactly such a collection of sequences.

Our general approach is to use a certain number (less than q) of Galois or Fibonacci LFSRs, and then extract from different registers the sequences we need, by possibly using additional buffering or XORs. We look then at two particular cases of this approach. In the first case, we use a small number of Galois LFSRs and possibly some small amount of buffering, but no additional XORs. We exploit the fact that, unlike a Fibonacci LFSR, each register of a Galois LFSR can produce elementary sequences from different equivalence classes. In the second particular case we consider, we only run a single (Galois or Fibonacci) LFSR and obtain all the required sequences by doing further XOR operations between different registers. We exploit the fact that the n sequences obtained from the n registers form a basis in the vector space of all elementary sequences with fixed minimal polynomial g. We ran experiments for all irreducible polynomials of degree n up to 14, where $2^n - 1$ is not prime.

While our construction of elementary sequences was targeted at fast generation of m-sequences, there are other possible applications of this construction. In coding theory, elementary sequences are the codewords of minimal cyclic codes (also known as irreducible cyclic codes), hence our construction will produce the non-equivalent codewords of such a code. In cryptography, one could use the elementary sequences as inputs to a non-linear function in order to construct a filtering generator for a stream cipher.

2 Preliminaries

Throughout the paper all the fields are finite fields. We define linear recurring sequences as usual and note elementary sequences and m-sequences as special cases:

Definition 1. *An infinite sequence $s = s_0, s_1, \ldots$ with elements in a field K is called a* homogeneous linear recurring sequence *if there exists a homogeneous linear recurrence relation of the form $s_{i+n} = c_{n-1}s_{i+n-1} + \cdots + c_1 s_{i+1} + c_0 s_i$ for all $i = 0, 1, \ldots$, where $c_0, c_1, \ldots, c_{n-1} \in K$ are constants. We associate to it a* characteristic polynomial *$f(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$. If n is minimal for the given sequence we call n the linear complexity of s and f the minimal polynomial of s.*

An elementary sequence *is a sequence that has an irreducible minimal polynomial. If the minimal polynomial is moreover primitive, the the sequence is called* m-sequence*.*

Recall that an m-sequence with linear complexity n has period $|K|^n - 1$, so binary m-sequences have period $2^n - 1$. More generally, a sequence with irreducible minimal polynomial f has period equal to $\mathrm{ord}(f)$, where $\mathrm{ord}(f)$ denotes the order of f, i.e. the minimum integer $k > 0$ such that $f | x^k - 1$. We have that $\mathrm{ord}(f)$ is a factor of $|K|^{\deg(f)} - 1$ with $\mathrm{ord}(f) = |K|^{\deg(f)} - 1$ achieved iff f is primitive.

Decimation and its inverse, interleaving, will play an important role in our constructions:

Definition 2. *Given a sequence $s = s_0, s_1, \ldots$, its q-*decimation *starting at position j is the sequence u such that $u_i = s_{j+iq}$. If j is not specified, it is by default $j = 0$.*

Note that if s has period N then any q-decimation of s has (not necessarily minimal) period $N/\gcd(N, q)$.

Definition 3. *The* interleaving *of q sequences $u^{(0)}, u^{(1)}, \ldots, u^{(q-1)}$ is the sequence s defined as $s_{j+iq} = u_i^{(j)}$.*

Note that if all the sequences $u^{(0)}, u^{(1)}, \ldots, u^{(q-1)}$ have (not necessarily minimal) period d, then their interleaving will have (not necessarily minimal) period dq.

For the rest of this paper we will restrict to binary sequences. Some of these constructions could be applicable to other fields, but this will be a subject for further research. We will denote by $\mathbb{F}_{2^n}$ the finite field with 2^n elements.

Any linear recurrent sequence can be represented using the trace transformation. We are interested in elementary sequences:

Theorem 1. *[4, Theorem 6.24] Let $s_0, s_1, \ldots$ be a linear recurring sequence in $\mathbb{F}_2$ whose characteristic polynomial g is irreducible over $\mathbb{F}_2$ and has degree n. Let β be a root of g in the extension field $\mathbb{F}_{2^n}$. Then there exists a uniquely determined $a \in \mathbb{F}_{2^n}$ such that $s_i = \mathrm{Tr}(a\beta^i) = \sum_{k=0}^{n-1} a^{2^k}(\beta^{2^k})^i$, $i = 0, 1, \ldots$.*

We introduce the following short hand notation to refer to elementary sequences in terms of their trace representation:

Definition 4. *We define $\mathrm{Seq}_\beta(a)$ as the sequence s whose i-th element is represented by $s_i = \mathrm{Tr}(a\beta^i)$.*

We note that Seq is linear: for any $a_1, a_2 \in \mathbb{F}_{2^n}$ and $c_1, c_2 \in \mathbb{F}_2$ we have

$$c_1 \mathrm{Seq}_\beta(a_1) + c_2 \mathrm{Seq}_\beta(a_2) = \mathrm{Seq}_\beta(c_1 a_1 + c_2 a_2)$$

For elements β, x in a finite field we will denote by $\log_\beta(x)$ the discrete logarithm of x in the base β, i.e. the smallest positive integer i with the property $\beta^i = x$, if such an integer exists. When β is not a primitive element of the fields we must remember that $\log_\beta$ will not always be defined.

Given $s = s_0, s_1, \ldots$ we denote by $(s \ll k)$ the sequence obtained by shifting s by k positions to the left, i.e. the sequence $s_k, s_{k+1}, \ldots$. If s is periodic with period N we denote by $s \gg k$ the sequence obtained by cyclicly shifting s by k positions to the right, i.e. $s_{N-k}, s_{N-k+1}, \ldots, s_{N-1}, s_0, s_1$. Note that $(s \gg k) = (s \ll (N-k))$.

If two sequences of period N are such that one can be obtained from the other by a (cyclic) shift, then we say the two sequences are equivalent (under cyclic shifts). The different cyclic shifts of a sequence are sometimes called "phases", especially in engineering contexts.

Shifting relates to the Seq() notation as follows:

Lemma 1. *Let $b, b_1, b_2 \in \mathbb{F}^*_{2^n}$, and $h \in \mathbb{Z}$.*

$$(\mathrm{Seq}_\beta(b) \gg h) = \mathrm{Seq}_\beta(b\beta^{-h})$$
$$(\mathrm{Seq}_\beta(b) \ll h) = \mathrm{Seq}_\beta(b\beta^{h})$$
$$(\mathrm{Seq}_\beta(b_1) \gg h) = \mathrm{Seq}_\beta(b_2) \Leftrightarrow h = \log_\beta(b_1 b_2^{-1}).$$

Proof. The proof is similar to [7, Lemma 1], as it does not depend on whether the minimal polynomial of β is primitive or not.

3 Fibonacci and Galois LFSRs

We recall the definition of both the Fibonacci and Galois Linear Feedback Shift Registers (LFSRs) and establish the notations that we will use later. As there are some variations in the literature, we will define everything explicitly.

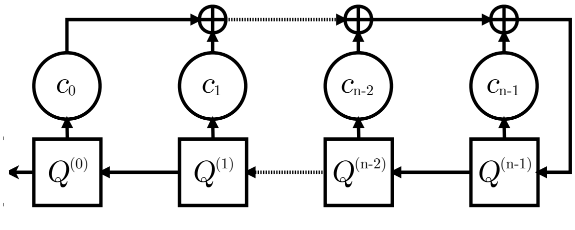


Fig. 1. Fibonacci LFSR

A Fibonacci LFSR is depicted in Fig. 1. We will denote the n registers $Q^{(0)}, Q^{(1)}, \ldots, Q^{(n-1)}$. At time $t = 0, 1, \ldots$ the content of register $Q^{(j)}$ will be denoted $q_t^{(j)}$, so the contents of the register $Q^{(j)}$ over time forms the sequence $q^{(j)} = q_0^{(j)}, q_1^{(j)}, \ldots$. The contents of all the registers at time t, i.e. the n-tuple $(q_t^{(0)}, \ldots, q_t^{(n-1)})$, is known as the state of the LFSR at time t. The initial state of the LFSR is the *state* at time 0. The content of the registers will be updated at each clock interval i according to the following

$$q_{i+1}^{(j)} = \begin{cases} c_{n-1}q_i^{(n-1)} + \cdots + c_1 q_i^{(1)} + c_0 q_i^{(0)} & \text{if } j = n-1 \\ q_i^{(j+1)} & \text{otherwise.} \end{cases} \tag{1}$$

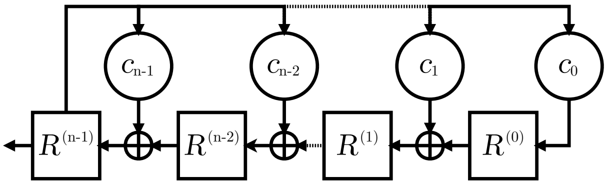


Fig. 2. Galois LFSR

The Galois LFSR, shown in Figure 2, will produce from each register $R^{(j)}$, $j = 0, 1, \ldots, n-1$ a sequence denoted $r^{(j)} = r_0^{(j)}, r_1^{(j)}, \ldots$. These values are updated at each clock interval i according to the following

$$r_{i+1}^{(j)} = \begin{cases} c_0 r_i^{(n-1)} & \text{if } j = 0 \\ r_i^{(j-1)} + c_j r_i^{(n-1)} & \text{otherwise.} \end{cases} \tag{2}$$

There are two related polynomials associated to the (Fibonacci or Galois) LFSR, namely the *feedback polynomial* $c_0 x^n + c_1 x^{n-1} + \cdots + c_{n-1}x + 1$ and the *characteristic polynomial* $f(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$. The feedback polynomial is the reciprocal of the characteristic polynomial, so if one of these polynomials is irreducible/primitive, so is the other.

Both the Fibonacci and the Galois LFSR produce as their output $q^{(0)}$, respectively $r^{(n-1)}$ a sequence with characteristic polynomial f. The initial states are in a one-to-one correspondence with the sequences with characteristic polynomial f.

Given the initial state of a Fibonacci LFSR, a Galois LFSR that produces exactly the same output sequence will need to have initial state given by

$$r_0^{(j)} = q_0^{(n-1-j)} + c_{n-1}q_0^{(n-1-j-1)} + \ldots + c_{j+1}q_0^{(0)}$$

for $j = 0, 1, \ldots, n-1$.

Conversely, given the initial state of a Galois LFSR, a Fibonacci LFSR that produces exactly the same output sequence will need to have initial state

$$q_0^{(n-1-j)} = r_0^{(j)} + c_{n-1}q_0^{(n-1-j-1)} + \ldots + c_{j+1}q_0^{(0)}$$

for $j = n-1, n-2, \ldots, 0$.

4 Efficient Generation of Elementary Sequences for Interleaving

We first need to recall the number of (inequivalent) elementary sequences and their connection to m-sequences. The following two theorems are known results, but we make them more precise, as needed for our purposes.

Theorem 2. *Given an irreducible polynomial g of degree n and order d, there are exactly $2^n - 1$ distinct non-zero elementary sequences with minimal polynomial g. There are $q = (2^n - 1)/d$ classes of equivalence (under cyclic shifts), each having d elements.*

If β is a root of g and α is a primitive element of $\mathbb{F}_{2^n}$ such that $\beta = \alpha^q$ then each of the classes above is of the form $\{\mathrm{Seq}_\beta(\alpha^{i+jq}) | j = 0, 1, \ldots, d-1\}$ for $i = 0, 1, \ldots, q-1$.

Proof. The sequences can be written as $\mathrm{Seq}_\beta(a)$ with each of the $2^n - 1$ elements $a \in \mathbb{F}^*_{2^n}$ uniquely identifying a sequence. By Lemma 1, two sequences $\mathrm{Seq}_\beta(b_1)$ and $\mathrm{Seq}_\beta(b_2)$ are equivalent under cyclic shifts iff there is an integer h such that $b_1 = b_2\beta^h$.

Theorem 3. *(cf. [8] and [5, Theorem 11, Ch. 8, §4]) Let $s = s_0, s_1, \ldots$ be an m-sequence with composite period $2^n - 1 = dq$. Let α be a root of the minimal polynomial of s and let $a \in \mathbb{F}_{2^n}$ be such that $s = \mathrm{Seq}_\alpha(a)$. Then the q-decimation of s (staring at positions $0, 1, \ldots q-1$) will result in the q sequences $s^{(0)}, \ldots, s^{(q-1)}$ such that $s^{(j)} = \mathrm{Seq}_\beta(a\alpha^j)$ for $j = 0, 1, \ldots q-1$ and $\beta = \alpha^q$.*

Proof. The i-th element of $s^{(j)}$ is $s_i^{(j)} = s_{j+iq} = \mathrm{Tr}(a\alpha^{j+iq}) = \mathrm{Tr}(a\alpha^j(\alpha^q)^i) = \mathrm{Tr}(a\alpha^j(\beta)^i)$, which is exactly the i-th element of the sequence $\mathrm{Seq}_\beta(a\alpha^j)$.

The result above and its proof immediately lead to the following generalisation:

Theorem 4. *Let s be an m-sequence of linear complexity n and minimal polynomial f. Assume $2^n - 1 = dq$ is a non-trivial factorisation. Let g be an irreducible polynomial of order d, degree n and let β be a root of g. Let $u^{(0)}, \ldots, u^{(q-1)}$ be elementary sequences with minimal polynomial g.*

We have the following equivalence: s can be obtained by interleaving $u^{(0)}, \ldots, u^{(q-1)}$ if and only if there is a primitive root α of f such that $\beta = \alpha^q$, and there is an $a \in \mathbb{F}_{2^n}$ such that $s = \mathrm{Seq}_\alpha(a)$ and $u^{(j)} = \mathrm{Seq}_\beta(a\alpha^j)$ for $j = 0, 1, \ldots, q-1$.

Note that if we start from an irreducible non-primitive polynomial g of degree n and order d and construct the finite field $\mathbb{F}_{2^n}$ as the algebraic extension by β, a root of g, then there are several primitive elements $\alpha \in \mathbb{F}_{2^n}$ which are solutions for $\beta = \alpha^q$ (with $q = (2^n - 1)/d$). Moreover these α need not all have the same minimal polynomial.

Our goal is, given g, to produce one elementary sequence from each of the q equivalence classes; moreover these sequences should be in the correct phase relative to each other (as described by Theorem 4) such that they may be interleaved to generate an m-sequence.

A first, straightforward approach would be to generate these sequences using q (Galois or Fibonacci) LFSRs with suitably chosen initial states. Let us examine in more detail how to compute the initial states. Assume we are given n initial terms of the target m-sequence s.

The initial states of the q Fibonacci LFSRs can be obtained by computing a further $(q-1)n$ terms of s (to give us a total of qn terms) and q-decimating the qn terms to obtain the q initial states.

The initial states of the q Galois LFSRs can be obtained by computing first the initial states of the Fibonacci LFSRs, and then transforming them into the equivalent initial states of Galois LFSRs as described at the end of Section 3. An alternative efficient way of computing the Galois initial states is described in [9] and [2].

However, we are interested in obtaining the desired sequences from less than q LFSRs. To this end, we will examine closer what sequences we can obtain from each of the registers of an LFSR, rather than just from the output register.

Theorem 5. *Let $g = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ be an irreducible non-primitive polynomial with root β.*

Consider a Fibonacci and a Galois LFSR, both with characteristic polynomial g and initial states chosen such that they both produce the same output $\mathrm{Seq}_\beta(a)$ for some given $a \in \mathbb{F}_{2^n}$.

The register $Q^{(j)}$ of the Fibonacci LFSR will produce the sequence $q^{(j)} = \mathrm{Seq}_\beta(a\beta^j) = (\mathrm{Seq}_\beta(a) \ll j)$, for $j = 0, 1, \ldots, n-1$.

The register $R^{(j)}$ of the Galois LFSR will produce the sequence $r^{(j)} = \mathrm{Seq}_\beta(av_j)$ where $v_j = c_{j+1} + c_{j+2}\beta + \cdots + c_{n-1}\beta^{n-j-2} + \beta^{n-j-1}$, for $j = 0, 1, \ldots, n-1$. Moreover, if α is a primitive element of $\mathbb{F}_{2^n}$ such that $\beta = \alpha^q$, where $q = (2^n-1)/\mathrm{ord}(g)$, then $r^{(j)} = \mathrm{Seq}_\beta(a\alpha^{h_j}) = (\mathrm{Seq}_\beta(a\alpha^{k_j}) \ll l_j)$ where $h_j = \log_\alpha v_j = k_j + l_j q$ and $0 \le k_j < q$.

Proof. For the Fibonacci LFSR the result is immediate and well known. For the Galois LFSR, (2) can be rewritten as $r^{(0)} = (r^{(n-1)} \gg 1)$ and $r^{(j)} = ((r^{(j-1)} + c_j r^{(n-1)}) \gg 1)$ for $1 \le j \le n-1$. By induction, and using Lemma 1 we obtain the required expression, similar to [7, Theorem 2].

This result tells us that for Fibonacci LFSRs all registers contain the same sequence, shifted by $1, 2, \ldots, n-1$ positions, so we cannot hope to obtain sequences from different equivalence classes. The situation is more interesting for Galois LFSRs. Here, depending on the characteristic polynomial, we may obtain sequences from several classes. It all depends on the values of $h_j = \log_\alpha v_j$. If h_j is not a multiple of q (or equivalently $v_j \notin \langle\beta\rangle$) then the sequence produced in the register $R^{(j)}$ is inequivalent to (i.e. not a cyclic shift of) the output of the LFSR. Moreover if h_j and h_k are not congruent modulo q then the sequences in registers j and k are inequivalent. So we can hope to obtain several inequivalent elementary sequences from the same Galois LFSR, and thus obtain from less than q LFSRs all the q elementary sequences needed for interleaving. Of course we also need to examine the relative shifts of the sequences, given by the integer quotients l_j in the Theorem above. Experimental data using this approach is described in Section 4.1.

An alternative approach to obtaining more than one elementary sequence from one LFSR is the following. It is well known that the set of sequences with given minimal polynomial g forms a vector space over $\mathbb{F}_2$. Therefore, if we have a basis we can obtain any other sequence as a linear combination.

Theorem 6. *Let* $a_0, a_1, \ldots, a_{n-1} \in \mathbb{F}_{2^n}$. *We have the following equivalence:* $a_0, a_1, \ldots, a_{n-1}$ *is a basis of* $\mathbb{F}_{2^n}$ *(viewed as an* n*-dimensional vector space over* $\mathbb{F}_2$*) if and only if* $\mathrm{Seq}_\beta(a_0), \mathrm{Seq}_\beta(a_1), \ldots, \mathrm{Seq}_\beta(a_{n-1})$ *is a basis of the set of sequences with minimal polynomial* g *(viewed as an* n*-dimensional vector space over* $\mathbb{F}_2$*).*

Proof. Let $a \in \mathbb{F}_{2^n}$. $a_0, a_1, \ldots, a_{n-1}$ is a basis of $\mathbb{F}_{2^n}$ iff there are unique $c_0, c_1, \ldots, c_{n-1} \in \mathbb{F}_2$ such that $a = \sum_{i=0}^{n-1} c_i a_i$ iff there are unique $c_0, c_1, \ldots, c_{n-1} \in \mathbb{F}_2$ such that $\mathrm{Seq}_\beta(a) = \mathrm{Seq}_\beta(\sum_{i=0}^{n-1} c_i a_i) = \sum_{i=0}^{n-1} c_i \mathrm{Seq}_\beta(a_i)$ iff $\mathrm{Seq}_\beta(a_0), \mathrm{Seq}_\beta(a_1), \ldots, \mathrm{Seq}_\beta(a_{n-1})$ is a basis.

Corollary 1. *With the notations of Theorem 5, each of the sets of sequences* $\{q^{(j)} | j = 0, 1, \ldots, n-1\}$ *and* $\{r^{(j)} | j = 0, 1, \ldots, n-1\}$ *form a basis for the set of sequences with minimal polynomial* g *(viewed as a* n*-dimensional vector space over* $\mathbb{F}_2$*).*

Hence it turns out that the sequences from the n LFSR registers do form a basis (regardless whether Fibonacci or Galois LFSR) so any other sequence can be obtained as a linear combination of the registers. Examples of this approach are discussed in Section 4.2.

More generally we can combine the two approaches above. Namely, for speeding up the computation of the desired set of sequences $A = \{\mathrm{Seq}_\beta(a\alpha^j) | j = 0, \ldots, q-1\}$ we will use the following general method: we employ several (but less that q) Galois or Fibonacci LFSRs. We consider the set B of sequences generated in each register of each of the LFSRs. This set B will contain some of the sequences in A. Any remaining sequences in A can be obtained from the sequences in B by either buffering (if the sequence exists in B but is in the wrong phase) or by computing a linear combination of sequences in B. Experimental results using this combined method are a subject of further research.

4.1 Using Several Galois LFSRs

For given irreducible characteristic polynomials g we examine the Galois LFSR, computing the equivalence class and the relative shift for each register, see Theorem 5. We then identify suitable registers that produce sequences from different classes, preferably in the same phase or with a small difference of phase. We then determine the number of Galois LFSRs we need to produce all the elementary sequences we need for interleaving. It suffices to run the experiments for only one initial state, as the classes and the shifts are computed relative to the output sequence.

We ran experiments for all irreducible polynomials g of degree n up to 14, where $2^n - 1$ is not prime. For each g we computed all the possible values of α and their minimal polynomial f (which will also be the minimal polynomial of the interleaved sequence). Table 1 shows a few examples of constructions yielding an elementary sequence in the correct phase from each equivalence class. The "Classes" and "Shifts" n-tuples display the values $(k_{n-1}, \ldots, k_0)$ and $(l_{n-1}, \ldots, l_0)$, with the notations from Theorem 5. The list "Interleave" specifies which registers we need to interleave, with a triple of the form $(R^{(j)}, a\alpha^i, l_j)$ signifying that we have to use a buffer of l_j terms for register $R^{(j)}$ of a Galois LFSR initialised such that the LFSR output is $\mathrm{Seq}_\beta(a\alpha^i)$. Finally, the LFSRs value indicates the total number of Galois LFSRs we need.

For all lengths in our experiment we were able to determine at least one efficient construction, in the sense that the number of LFSRs needed for the construction is less than q, the total number of sequences required. For each n there is at least one g which will produce 2 of the required sequences, for most n at least 3 sequences can be acquired from one LFSR. If we allow q to increase, such that the elementary sequences become shorter, we generally see that we can obtain elementary sequences from more equivalence classes from a single LFSR, thus lowering the total number of LFSRs needed for the full construction.

4.2 Using One LFSR and Linear Combinations of Its Registers

We wish to produce the required elementary sequences using only one LFSR, either Fibonacci or Galois, but allow ourselves to linearly combine the output of the registers to produce our sequences. This is possible according to Corollary 1. Let a be such that the output of our LFSR is $\mathrm{Seq}_\beta(a)$.

To obtain a particular desired sequence $\mathrm{Seq}_\beta(a\alpha^j)$ for $j = 1, \ldots, q-1$ as a linear combination of the registers it suffices to represent α^j in the basis $1, \beta, \ldots, \beta^{n-1}$ for a Fibonacci LFSR, or in basis $v_0, v_1, \ldots, v_{n-1}$ for a Galois one. Alternatively we can consider the first n terms of the sequence $\mathrm{Seq}_\beta(a\alpha^j)$, viewed as an element in the vector space $\mathbb{F}_2^n$ and write them in the basis consisting of the first n elements of each of the sequences corresponding to the registers of the LFSR. Any of these approaches will amount to solving an $n \times n$ system of linear equations over $\mathbb{F}_2$, which is extremely fast. In our experiments we set up the LFSRs so as to generate the m-sequence in its "impulse response" form, i.e. starting with initial terms $00\ldots01$.

Again we ran experiments for all irreducible polynomials g of degree n up to 14, where $2^n - 1$ is not prime. We computed, for both Fibonacci and Galois LFSRs, the linear combinations of registers needed to produce the required sequences needed for interleaving. Tables 3 and 2 show a few examples of possible constructions using Fibonacci and Galois LFSRs respectively. The tables display which registers need to be XORed in order to produce the sequences required.

Table 1. Generating m-Sequences by Interleaving Registers of Several Galois LFSRs

$n=4 \quad g=11111$	$d=5 \quad q=3 \quad \alpha=1110$	$f=10011$
Classes: $(0,1,1,0)$	Shifts: $(4,4,1,0)$	LFSRs: 2
Interleave: $(R^{(0)},a,0),(R^{(1)},a,1),(R^{(0)},a\alpha,0)$		
$n=6 \quad g=1010111$	$d=21 \quad q=3 \quad \alpha=101$	$f=1100001$
Classes: $(0,2,1,1,0,0)$	Shifts: $(20,8,1,0,1,0)$	LFSRs: 2
Interleave: $(R^{(0)},a,0),(R^{(2)},a,0),(R^{(2)},a\alpha,0)$		
$n=8 \quad g=111010111$	$d=17 \quad q=15 \ \alpha=10011100$	$f=$
Classes $=(0,3,13,13,13,13,3,0)$	Shifts $=(16,0,13,12,0,16,2,0)$	LFSRs: 6
Interleave: $(R^{(0)},a,0),(R^{(0)},a\alpha,0),(R^{(0)},a\alpha^2,0),(R^{(7)},a,0),(R^{(7)},a\alpha,0),(R^{(7)},a\alpha^2,0),$ $(R^{(3)},a\alpha^8,0),(R^{(3)},a\alpha^9,0),(R^{(0)},a\alpha^8,0),(R^{(0)},a\alpha^9,0),(R^{(0)},a\alpha^{10},0),(R^{(7)},a\alpha^8,0),$ $(R^{(7)},a\alpha^9,0),(R^{(3)},a,0),(R^{(3)},a\alpha,0)$		
$n=9 \quad g=1001100101$	$d=73 \quad q=7 \quad \alpha=111011111$	$f=1001101111$
Classes: $(0,0,5,5,5,4,0,0,0)$	Shifts: $(72,71,0,72,71,63,2,1,0)$	LFSRs: 4
Interleave: $(R^{(0)},a,0),(R^{(0)},a\alpha,0),(R^{(6)},a\alpha^4,0),(R^{(6)},a\alpha^5,0),(R^{(0)},a\alpha^4,0),(R^{(6)},a,0),(R^{(6)},a\alpha,0)$		
$n=10 \ g=10000001111$	$d=341 \ q=3 \quad \alpha=1010000110$	$f=10010000001$
Classes: $(0,1,2,0,0,0,0,0,0,0)$	Shifts: $(340,1,101,6,5,4,3,2,1,0)$	LFSRs: 2
Interleave: $(R^{(0)},a,0),(R^{(8)},a,1),(R^{(0)},a\alpha^2,0)$		

Table 2. Generating m-Sequences by XORing and Interleaving the Registers of one Galois LFSR

$n=4 \quad g=11111$	$d=5 \quad q=3 \ \alpha=1010$	$f=10011$
Classes: $(0,2,2,0)$	Shifts: $(4,2,4,0)$	XORs: 2
Interleave: $R^{(1)}\oplus R^{(2)}\oplus R^{(3)},R^{(2)},R^{(3)}$		
$n=6 \quad g=1010111$	$d=21 \quad q=3 \ \alpha=101$	$f=1000011$
Classes: $(0,2,1,1,0,0)$	Shifts: $(20,8,1,0,1,0)$	XORs: 2
Interleave: $R^{(5)},R^{(3)},R^{(1)}\oplus R^{(3)}\oplus R^{(5)}$		
$n=8 \quad g=101111011$	$d=85 \quad q=3 \ \alpha=100101$	$f=101011111$
Classes: $(0,0,0,2,2,0,0,0)$	Shifts: $(84,14,13,45,49,32,1,0)$	XORs: 4
Interleave: $R^{(3)}\oplus R^{(4)}\oplus R^{(6)},R^{(2)}\oplus R^{(5)}\oplus R^{(7)},R^{(4)}$		
$n=8 \quad g=110100011$	$d=85 \quad q=3 \ \alpha=10010000$	$f=110001101$
Classes: $(0,1,1,1,1,1,1,0)$	Shifts: $(84,67,66,65,64,70,69,0)$	XORs: 4
Interleave: $R^{(3)}\oplus R^{(5)}\oplus R^{(6)},(R^{(2)}\oplus R^{(3)}),(R^{(2)}\oplus R^{(3)})\oplus R^{(6)}$		
$n=10 \ g=10010011001$	$d=341 \ q=3 \ \alpha=1110011010$	$f=10000100111$
Classes: $(0,0,0,1,1,1,1,0,0,0)$	Shifts: $(340,339,338,85,91,90,89,2,1,0)$	XORs: 5
Interleave: $R^{(3)}\oplus R^{(6)}\oplus R^{(8)},R^{(8)}\oplus R^{(9)},R^{(4)}\oplus R^{(5)}\oplus R^{(9)}$		
$n=10 \ g=10010011001$	$d=341 \ q=3 \ \alpha=1111100010$	$f=10001100101$
Classes: $(0,0,0,2,2,2,2,0,0,0)$	Shifts: $(340,339,338,312,318,317,316,2,1,0)$	XORs: 5
Interleave: $R^{(3)}\oplus(R^{(7)}\oplus R^{(8)}),R^{(4)}\oplus(R^{(7)}\oplus R^{(8)}),R^{(4)}\oplus R^{(7)}\oplus R^{(9)}$		
$n=10 \ g=10000001111$	$d=341 \ q=3 \ \alpha=1010000110$	$f=10000001001$
Classes: $(0,1,2,0,0,0,0,0,0,0)$	Shifts: $(340,1,101,6,5,4,3,2,1,0)$	XORs: 2
Interleave: $R^{(3)},R^{(5)}\oplus R^{(6)},R^{(7)}\oplus R^{(9)}$		

Interleaving these sequences in order yields the m-sequence with minimal polynomial f as shown in the table. Some optimisations are denoted by brackets, for example in Table 2 the fourth example shows that the computation $(R^{(2)}\oplus R^{(3)})$ can then be reused for computing $(R^{(2)}\oplus R^{(3)})\oplus R^{(6)}$ with only one additional $\oplus$. A full optimisation is outside the scope of this paper. For all lengths in our

Table 3. Generating m-Sequences by XORing and Interleaving the Registers of one Fibonacci LFSR

$n = 4$	$g = 11111$	$d = 5$	$q = 3$	$\alpha = 1010$	$f = 10011$	XORs: 2
Interleave: $Q^{(2)} \oplus Q^{(0)}, Q^{(1)} \oplus Q^{(0)}, Q^{(0)}$						
$n = 6$	$g = 1010111$	$d = 21$	$q = 3$	$\alpha = 101$	$f = 1000011$	XORs: 2
Interleave: $Q^{(0)}, Q^{(2)} \oplus Q^{(0)}, Q^{(4)} \oplus Q^{(0)}$						
$n = 6$	$g = 1010111$	$d = 21$	$q = 3$	$\alpha = 111000$	$f = 1101101$	XORs: 4
Interleave: $Q^{(3)} \oplus Q^{(0)}, Q^{(3)} \oplus Q^{(1)}, Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}$						
$n = 8$	$g = 101110111$	$d = 85$	$q = 3$	$\alpha = 1100011$	$f = 100011101$	XORs: 4
Interleave: $Q^{(3)} \oplus Q^{(1)} \oplus Q^{(0)}, Q^{(5)} \oplus Q^{(2)}, Q^{(4)} \oplus Q^{(0)}$						
$n = 8$	$g = 110001011$	$d = 85$	$q = 3$	$\alpha = 11101011$	$f = 101101001$	XORs: 4
Interleave: $Q^{(4)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(3)}, Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}$						
$n = 8$	$g = 111011101$	$d = 85$	$q = 3$	$\alpha = 1111110$	$f = 101110001$	XORs: 4
Interleave: $Q^{(4)} \oplus Q^{(3)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(1)}, Q^{(3)} \oplus Q^{(0)}$						
$n = 8$	$g = 100111111$	$d = 85$	$q = 3$	$\alpha = 1111100$	$f = 110101001$	XORs: 5
Interleave: $Q^{(4)} \oplus Q^{(3)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(2)}, (Q^{(5)} \oplus Q^{(2)}) \oplus Q^{(4)} \oplus Q^{(0)}$						
$n = 10$	$g = 10000001111$	$d = 341$	$q = 3$	$\alpha = 11000110$	$f = 11100011101$	XORs: 5
Interleave: $Q^{(6)} \oplus Q^{(5)} \oplus Q^{(1)} \oplus Q^{(0)}, (Q^{(6)} \oplus Q^{(5)}) \oplus Q^{(3)} \oplus Q^{(2)}, Q^{(0)}$						
$n = 10$	$g = 10000001111$	$d = 341$	$q = 3$	$\alpha = 1010000110$	$f = 10000001001$	XORs: 2
Interleave: $Q^{(6)}, Q^{(4)} \oplus Q^{(3)}, Q^{(2)} \oplus Q^{(0)}$						

experiment we were able to determine at least one efficient construction, in the sense that for obtaining each of the required sequences the maximum number of XORs was $\lfloor (n+3)/2 \rfloor$, but for most n this value was as low as $\lfloor (n+1)/2 \rfloor$.

5 Conclusion

We developed a general framework for efficiently producing several elementary sequences (i.e. linearly recurrent sequences with an irreducible characteristic polynomial) such that they are inequivalent under cyclic shifts and if needed are, moreover, in suitable phases so that interleaving them produces an m-sequence. Experimental data identified irreducible polynomials particularly suited for this purpose. Future work will aim to further improve the efficiency of the method by analysing in more depth the gate complexity of the different constructions, combining the different approaches, and improving the efficiency of determining the initial states.

References

1. Blackburn, S.R.: Increasing the Rate of Output of m-Sequences. Information Processing Letters 51(2), 73–77 (1994)
2. Kagaris, D.: Multiple-Seed TPG Structures. IEEE Transactions on Computers 52(12), 1633–1639 (2003)

3. Lempel, A., Eastman, W.L.: High speed generation of maximum length sequences. IEEE Transactions on Computers 20(2), 227–229 (1971)
4. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications. Cambridge University Press (1994)
5. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland (1978)
6. Robshaw, M.J.B.: Increasing the Rate of Output for m Sequences. Electronics Letters 27(19), 1710–1712 (1991)
7. Sălăgean, A., Gardner, D., Phan, R.: Index Tables of Finite Fields and Modular Golomb Rulers. In: Helleseth, T., Jedwab, J. (eds.) SETA 2012. LNCS, vol. 7280, pp. 136–147. Springer, Heidelberg (2012)
8. Surböck, F., Weinrichter, H.: Interlacing Properties of Shift-Register Sequences with Generator Polynomials Irreducible Over GF(p). IEEE Transactions on Information Theory 24(3), 386–389 (1978)
9. Udar, S., Kagaris, D.: LFSR Reseeding with Irreducible Polynomials. In: On-Line Testing Symposium, pp. 293–298 (2007)

Appendix F

Galois LFSR Shift Results

The following results display the shifts of the stages of Galois LFSRs with primitive feedback polynomial f with degrees ranging from 2 to 11. The polynomial f here is displayed in the form $d_n 2^n + d_{n-1} 2^{n-1} + \cdots + d_1 2 + d_0$ such that $f(x) = d_n x^n + d_{n-1} x^{n-1} + \cdots + d_1 x + d_0$ and the shifts are relative to the sequence in memory cell $R^{(n-1)}$. The results of polynomials with degrees 12 to 23 are not reproduced here due to the lack of space, in this range a further 634, 358 polynomials were tested.

Table F.1: Shifts of Galois LFSRs of length 3

Polynomial	Shifts H
`1011`	$(1, 2, 0)$
`1101`	$(1, 6, 0)$

Table F.2: Shifts of Galois LFSRs of length 4

Polynomial	Shifts H
`10011`	$(1, 2, 3, 0)$
`11001`	$(1, 13, 14, 0)$

Table F.3: Shifts of Galois LFSRs of length 5

Polynomial	Shifts H
`100101`	$(1, 2, 3, 30, 0)$
`101001`	$(1, 2, 29, 30, 0)$
`101111`	$(1, 2, 25, 11, 0)$
`110111`	$(1, 20, 21, 18, 0)$
`111011`	$(1, 14, 11, 12, 0)$
`111101`	$(1, 21, 7, 30, 0)$

Table F.4: Shifts of Galois LFSRs of length 6

Polynomial	Shifts H
`1000011`	$(1, 2, 3, 4, 5, 0)$
`1011011`	$(1, 2, 50, 54, 55, 0)$
`1100001`	$(1, 59, 60, 61, 62, 0)$
`1100111`	$(1, 26, 27, 28, 24, 0)$
`1101101`	$(1, 9, 10, 14, 62, 0)$
`1110011`	$(1, 40, 36, 37, 38, 0)$

Table F.5: Shifts of Galois LFSRs of length 7

Polynomial	Shifts H
`10000011`	$(1, 2, 3, 4, 5, 6)$
`10001001`	$(1, 2, 3, 4, 125, 126)$
`10001111`	$(1, 2, 3, 4, 95, 86)$
`10010001`	$(1, 2, 3, 124, 125, 126)$
`10011101`	$(1, 2, 3, 39, 107, 126)$
`10100111`	$(1, 2, 102, 103, 104, 113)$
`10101011`	$(1, 2, 43, 44, 19, 20)$
`10111001`	$(1, 2, 21, 89, 125, 126)$
`10111111`	$(1, 2, 39, 54, 119, 18)$
`11000001`	$(1, 122, 123, 124, 125, 126)$
`11001011`	$(1, 40, 41, 42, 37, 38)$
`11010011`	$(1, 90, 91, 86, 87, 88)$
`11010101`	$(1, 108, 109, 84, 85, 126)$
`11100101`	$(1, 15, 24, 25, 26, 126)$
`11101111`	$(1, 56, 34, 35, 31, 54)$
`11110001`	$(1, 42, 33, 124, 125, 126)$
`11110111`	$(1, 74, 97, 93, 94, 72)$
`11111101`	$(1, 110, 9, 74, 89, 126)$

Table F.6: Shifts of Galois LFSRs of length 8

Polynomial	Shifts H
`100011101`	$(1, 2, 3, 4, 101, 48, 254, 0)$
`100101011`	$(1, 2, 3, 68, 69, 241, 242, 0)$
`100101101`	$(1, 2, 3, 39, 40, 223, 254, 0)$
`101001101`	$(1, 2, 47, 48, 49, 44, 254, 0)$
`101011111`	$(1, 2, 245, 246, 108, 179, 121, 0)$
`101100011`	$(1, 2, 140, 193, 194, 195, 196, 0)$
`101100101`	$(1, 2, 212, 207, 208, 209, 254, 0)$
`101101001`	$(1, 2, 33, 216, 217, 253, 254, 0)$
`101110001`	$(1, 2, 208, 155, 252, 253, 254, 0)$
`110000111`	$(1, 100, 101, 102, 103, 104, 98, 0)$
`110001101`	$(1, 60, 61, 62, 63, 116, 254, 0)$
`110101001`	$(1, 14, 15, 187, 188, 253, 254, 0)$
`111000011`	$(1, 158, 152, 153, 154, 155, 156, 0)$
`111001111`	$(1, 142, 163, 164, 165, 160, 140, 0)$
`111100111`	$(1, 116, 96, 91, 92, 93, 114, 0)$
`111110101`	$(1, 135, 77, 148, 10, 11, 254, 0)$

Table F.7: Shifts of Galois LFSRs of length 9

Polynomial	Shifts H
`1000010001`	$(1, 2, 3, 4, 5, 508, 509, 510, 0)$
`1000011011`	$(1, 2, 3, 4, 5, 488, 195, 196, 0)$
`1000100001`	$(1, 2, 3, 4, 507, 508, 509, 510, 0)$
`1000101101`	$(1, 2, 3, 4, 79, 80, 37, 510, 0)$
`1000110011`	$(1, 2, 3, 4, 417, 101, 102, 103, 0)$
`1001011001`	$(1, 2, 3, 343, 344, 339, 509, 510, 0)$
`1001011111`	$(1, 2, 3, 237, 238, 156, 181, 52, 0)$
`1001101001`	$(1, 2, 3, 173, 168, 169, 509, 510, 0)$
`1001101111`	$(1, 2, 3, 263, 275, 276, 167, 92, 0)$
`1001110111`	$(1, 2, 3, 214, 455, 278, 279, 442, 0)$
`1001111101`	$(1, 2, 3, 137, 164, 403, 466, 510, 0)$
`1010000111`	$(1, 2, 389, 390, 391, 392, 393, 193, 0)$
`1010010101`	$(1, 2, 453, 454, 455, 449, 450, 510, 0)$
`1010100011`	$(1, 2, 73, 74, 32, 33, 34, 35, 0)$
`1010100101`	$(1, 2, 62, 63, 57, 58, 59, 510, 0)$
`1010101111`	$(1, 2, 396, 397, 333, 334, 164, 452, 0)$
`1010110111`	$(1, 2, 412, 413, 349, 171, 172, 460, 0)$
`1010111101`	$(1, 2, 53, 54, 160, 119, 50, 510, 0)$
`1011001111`	$(1, 2, 372, 296, 297, 298, 177, 440, 0)$
`1011010001`	$(1, 2, 475, 432, 433, 508, 509, 510, 0)$
`1011011011`	$(1, 2, 492, 162, 163, 158, 499, 500, 0)$
`1011110101`	$(1, 2, 462, 393, 352, 458, 459, 510, 0)$
`1011111001`	$(1, 2, 46, 109, 348, 375, 509, 510, 0)$

Table F.8: Shifts of Galois LFSRs of length 9 continued

Polynomial	Shifts H
`1100010011`	$(1, 115, 116, 117, 118, 111, 112, 113, 0)$
`1100010101`	$(1, 477, 478, 479, 480, 438, 439, 510, 0)$
`1100011111`	$(1, 429, 430, 431, 432, 259, 271, 427, 0)$
`1100100011`	$(1, 399, 400, 401, 394, 395, 396, 397, 0)$
`1100110001`	$(1, 409, 410, 411, 95, 508, 509, 510, 0)$
`1100111011`	$(1, 250, 251, 252, 177, 224, 247, 248, 0)$
`1101001111`	$(1, 136, 137, 400, 401, 402, 175, 134, 0)$
`1101011011`	$(1, 127, 128, 288, 289, 106, 124, 125, 0)$
`1101100001`	$(1, 316, 317, 24, 507, 508, 509, 510, 0)$
`1101101011`	$(1, 387, 388, 406, 223, 224, 384, 385, 0)$
`1101101101`	$(1, 12, 13, 354, 349, 350, 20, 510, 0)$
`1101110011`	$(1, 264, 265, 288, 335, 260, 261, 262, 0)$
`1101111111`	$(1, 405, 406, 420, 53, 187, 264, 403, 0)$
`1110000101`	$(1, 319, 119, 120, 121, 122, 123, 510, 0)$
`1110001111`	$(1, 328, 464, 465, 466, 467, 461, 326, 0)$
`1110110101`	$(1, 52, 340, 341, 163, 99, 100, 510, 0)$
`1110111001`	$(1, 70, 233, 234, 57, 298, 509, 510, 0)$
`1111000111`	$(1, 186, 51, 45, 46, 47, 48, 184, 0)$
`1111001011`	$(1, 378, 337, 110, 111, 112, 375, 376, 0)$
`1111001101`	$(1, 72, 335, 214, 215, 216, 140, 510, 0)$
`1111010101`	$(1, 60, 348, 178, 179, 115, 116, 510, 0)$
`1111011001`	$(1, 420, 345, 236, 237, 249, 509, 510, 0)$
`1111100011`	$(1, 85, 241, 253, 80, 81, 82, 83, 0)$
`1111101001`	$(1, 460, 331, 356, 274, 275, 509, 510, 0)$
`1111111011`	$(1, 109, 248, 325, 459, 92, 106, 107, 0)$

Table F.9: Shifts of Galois LFSRs of length 10

Polynomial	Shifts H
`10000001001`	$(1, 2, 3, 4, 5, 6, 7, 1021, 1022, 0)$
`10000011011`	$(1, 2, 3, 4, 5, 6, 58, 491, 492, 0)$
`10000100111`	$(1, 2, 3, 4, 5, 363, 364, 365, 84, 0)$
`10000101101`	$(1, 2, 3, 4, 5, 200, 201, 360, 1022, 0)$
`10001100101`	$(1, 2, 3, 4, 703, 347, 348, 349, 1022, 0)$
`10001101111`	$(1, 2, 3, 4, 255, 698, 699, 168, 574, 0)$
`10010000001`	$(1, 2, 3, 1017, 1018, 1019, 1020, 1021, 1022, 0)$
`10010001011`	$(1, 2, 3, 890, 891, 892, 893, 965, 966, 0)$
`10011000101`	$(1, 2, 3, 919, 163, 164, 165, 166, 1022, 0)$
`10011010111`	$(1, 2, 3, 15, 998, 999, 89, 90, 944, 0)$
`10011100111`	$(1, 2, 3, 907, 422, 428, 429, 430, 473, 0)$
`10011110011`	$(1, 2, 3, 204, 623, 618, 941, 942, 943, 0)$
`10011111111`	$(1, 2, 3, 545, 497, 709, 732, 979, 585, 0)$
`10100001101`	$(1, 2, 820, 821, 822, 823, 824, 817, 1022, 0)$
`10100011001`	$(1, 2, 858, 859, 860, 861, 105, 1021, 1022, 0)$
`10100100011`	$(1, 2, 441, 442, 443, 216, 217, 218, 219, 0)$
`10100110001`	$(1, 2, 675, 676, 677, 321, 1020, 1021, 1022, 0)$
`10100111101`	$(1, 2, 261, 262, 263, 48, 327, 258, 1022, 0)$
`10101000011`	$(1, 2, 490, 491, 751, 752, 753, 754, 755, 0)$
`10101010111`	$(1, 2, 472, 473, 789, 790, 391, 392, 746, 0)$
`10101101011`	$(1, 2, 581, 582, 393, 399, 400, 288, 289, 0)$
`10110000101`	$(1, 2, 207, 200, 201, 202, 203, 204, 1022, 0)$
`10110001111`	$(1, 2, 506, 240, 241, 242, 243, 616, 763, 0)$
`10110010111`	$(1, 2, 508, 950, 951, 952, 803, 804, 764, 0)$
`10110100001`	$(1, 2, 664, 823, 824, 1019, 1020, 1021, 1022, 0)$
`10111000111`	$(1, 2, 962, 112, 360, 361, 362, 363, 991, 0)$
`10111100101`	$(1, 2, 766, 697, 976, 761, 762, 763, 1022, 0)$
`10111110111`	$(1, 2, 549, 178, 259, 254, 108, 109, 273, 0)$
`10111111011`	$(1, 2, 674, 86, 36, 870, 82, 846, 847, 0)$

Table F.10: Shifts of Galois LFSRs of length 10 continued

Polynomial	Shifts H
11000010011	$(1, 356, 357, 358, 359, 360, 352, 353, 354, 0)$
11000010101	$(1, 269, 270, 271, 272, 273, 533, 534, 1022, 0)$
11000100101	$(1, 805, 806, 807, 808, 581, 582, 583, 1022, 0)$
11000110111	$(1, 993, 994, 995, 996, 746, 395, 396, 991, 0)$
11001000011	$(1, 670, 671, 672, 664, 665, 666, 667, 668, 0)$
11001001111	$(1, 713, 714, 715, 85, 86, 87, 635, 711, 0)$
11001011011	$(1, 899, 900, 901, 937, 938, 240, 896, 897, 0)$
11001111001	$(1, 81, 82, 83, 406, 401, 820, 1021, 1022, 0)$
11001111111	$(1, 423, 424, 425, 410, 65, 240, 506, 421, 0)$
11010001001	$(1, 58, 59, 131, 132, 133, 134, 1021, 1022, 0)$
11010110101	$(1, 735, 736, 624, 625, 631, 442, 443, 1022, 0)$
11011000001	$(1, 532, 533, 966, 1018, 1019, 1020, 1021, 1022, 0)$
11011010011	$(1, 127, 128, 784, 86, 87, 123, 124, 125, 0)$
11011011111	$(1, 624, 625, 897, 76, 77, 843, 873, 622, 0)$
11011111101	$(1, 177, 178, 942, 154, 988, 938, 350, 1022, 0)$
11100010111	$(1, 700, 551, 552, 553, 554, 547, 548, 698, 0)$
11100011101	$(1, 33, 661, 662, 663, 664, 912, 62, 1022, 0)$
11100100001	$(1, 940, 659, 660, 661, 1019, 1020, 1021, 1022, 0)$
11100111001	$(1, 551, 594, 595, 596, 602, 117, 1021, 1022, 0)$
11101000111	$(1, 326, 476, 477, 470, 471, 472, 473, 324, 0)$
11101001101	$(1, 260, 220, 221, 72, 73, 74, 516, 1022, 0)$
11101010101	$(1, 278, 632, 633, 234, 235, 551, 552, 1022, 0)$
11101011001	$(1, 80, 934, 935, 25, 26, 1009, 1021, 1022, 0)$
11101100011	$(1, 33, 628, 629, 278, 28, 29, 30, 31, 0)$
11101111101	$(1, 751, 915, 916, 770, 765, 846, 475, 1022, 0)$
11110001101	$(1, 261, 408, 781, 782, 783, 784, 518, 1022, 0)$
11110010011	$(1, 313, 389, 937, 938, 939, 309, 310, 311, 0)$
11110110001	$(1, 450, 856, 325, 326, 769, 1020, 1021, 1022, 0)$
11111011011	$(1, 402, 151, 181, 947, 948, 127, 399, 400, 0)$
11111110011	$(1, 603, 518, 784, 959, 614, 599, 600, 601, 0)$
11111111001	$(1, 439, 45, 292, 315, 527, 479, 1021, 1022, 0)$

Table F.11: Shifts of Galois LFSRs of length 11

Polynomial	Shifts H
100000000101	$(1, 2, 3, 4, 5, 6, 7, 8, 9, 2049, 0)$
100000010111	$(1, 2, 3, 4, 5, 6, 7, 450, 451, 3250, 0)$
100000101011	$(1, 2, 3, 4, 5, 6, 137, 138, 439, 3655, 0)$
100000101101	$(1, 2, 3, 4, 5, 6, 655, 656, 1729, 2049, 0)$
100001000111	$(1, 2, 3, 4, 5, 97, 98, 99, 100, 3878, 0)$
100001100011	$(1, 2, 3, 4, 5, 1199, 856, 857, 858, 3236, 0)$
100001100101	$(1, 2, 3, 4, 5, 265, 1196, 1197, 1198, 2049, 0)$
100001110001	$(1, 2, 3, 4, 5, 1380, 1445, 2044, 2045, 2049, 0)$
100001111011	$(1, 2, 3, 4, 5, 1325, 1355, 625, 1107, 2987, 0)$
100010001101	$(1, 2, 3, 4, 599, 600, 601, 602, 297, 2049, 0)$
100010010101	$(1, 2, 3, 4, 1279, 1280, 1281, 636, 637, 2049, 0)$
100010011111	$(1, 2, 3, 4, 1952, 1953, 1954, 1972, 281, 2073, 0)$
100010101001	$(1, 2, 3, 4, 1513, 1514, 1715, 1716, 2045, 2049, 0)$
100010110001	$(1, 2, 3, 4, 778, 779, 773, 2044, 2045, 2049, 0)$
100011001111	$(1, 2, 3, 4, 1430, 555, 556, 557, 1709, 3227, 0)$
100011010001	$(1, 2, 3, 4, 1275, 1269, 1270, 2044, 2045, 2049, 0)$
100011100001	$(1, 2, 3, 4, 603, 668, 2043, 2044, 2045, 2049, 0)$
100011100111	$(1, 2, 3, 4, 1475, 1663, 1868, 1869, 1870, 2704, 0)$
100011101011	$(1, 2, 3, 4, 675, 1260, 1274, 1275, 1190, 2904, 0)$
100011110101	$(1, 2, 3, 4, 1790, 1310, 668, 1915, 1916, 2049, 0)$
100100001101	$(1, 2, 3, 633, 634, 635, 636, 637, 864, 2049, 0)$
100100010011	$(1, 2, 3, 1750, 1751, 1752, 1753, 1797, 1798, 2296, 0)$
100100100101	$(1, 2, 3, 1365, 1366, 1367, 997, 998, 999, 2049, 0)$
100100101001	$(1, 2, 3, 1882, 1883, 1884, 1877, 1878, 2045, 2049, 0)$
100100111011	$(1, 2, 3, 1407, 1408, 1409, 1119, 870, 1762, 2332, 0)$
100100111101	$(1, 2, 3, 419, 420, 421, 166, 88, 1452, 2049, 0)$
100101000101	$(1, 2, 3, 1397, 1398, 1308, 1309, 1310, 1311, 2049, 0)$
100101001001	$(1, 2, 3, 170, 171, 164, 165, 166, 2045, 2049, 0)$
100101010001	$(1, 2, 3, 332, 333, 534, 535, 2044, 2045, 2049, 0)$
100101011011	$(1, 2, 3, 1730, 1731, 1350, 1351, 248, 1671, 2423, 0)$
100101110011	$(1, 2, 3, 988, 989, 1391, 1409, 1095, 1096, 2998, 0)$
100101110101	$(1, 2, 3, 1298, 1299, 1505, 286, 254, 255, 2049, 0)$
100101111111	$(1, 2, 3, 357, 358, 1303, 226, 256, 950, 2645, 0)$
100110000011	$(1, 2, 3, 1558, 836, 837, 838, 839, 840, 3254, 0)$
100110001111	$(1, 2, 3, 1944, 614, 615, 616, 617, 1052, 3207, 0)$
100110101011	$(1, 2, 3, 522, 857, 858, 16, 17, 1943, 2151, 0)$
100110101101	$(1, 2, 3, 928, 1588, 1589, 1248, 1249, 665, 2049, 0)$
100110111001	$(1, 2, 3, 395, 801, 802, 287, 391, 2045, 2049, 0)$
100111000111	$(1, 2, 3, 1491, 494, 1120, 1121, 1122, 1123, 3731, 0)$
100111011001	$(1, 2, 3, 1657, 1761, 1246, 1247, 1653, 2045, 2049, 0)$
100111100101	$(1, 2, 3, 728, 743, 595, 1956, 1957, 1958, 2049, 0)$
100111110111	$(1, 2, 3, 1237, 1058, 1627, 339, 608, 609, 3471, 0)$
101000000001	$(1, 2, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2049, 0)$
101000000111	$(1, 2, 107, 108, 109, 110, 111, 112, 113, 4043, 0)$

Table F.12: Shifts of Galois LFSRs of length 11 continued

Polynomial	Shifts H
`101000010011`	$(1, 2, 503, 504, 505, 506, 507, 248, 249, 3845, 0)$
`101000010101`	$(1, 2, 130, 131, 132, 133, 134, 126, 127, 2049, 0)$
`101000101001`	$(1, 2, 737, 738, 739, 740, 650, 651, 2045, 2049, 0)$
`101001001001`	$(1, 2, 1049, 1050, 1051, 681, 682, 683, 2045, 2049, 0)$
`101001100001`	$(1, 2, 850, 851, 852, 1783, 2043, 2044, 2045, 2049, 0)$
`101001101101`	$(1, 2, 225, 226, 227, 147, 994, 995, 222, 2049, 0)$
`101001111001`	$(1, 2, 90, 91, 92, 1453, 1305, 1320, 2045, 2049, 0)$
`101001111111`	$(1, 2, 381, 382, 383, 1508, 991, 567, 1683, 3906, 0)$
`101010000101`	$(1, 2, 1921, 1922, 1914, 1915, 1916, 1917, 1918, 2049, 0)$
`101010010001`	$(1, 2, 1411, 1412, 767, 768, 769, 2044, 2045, 2049, 0)$
`101010011101`	$(1, 2, 182, 183, 2031, 2032, 2033, 27, 179, 2049, 0)$
`101010100111`	$(1, 2, 1598, 1599, 1405, 1406, 698, 699, 700, 2274, 0)$
`101010101011`	$(1, 2, 1995, 1996, 27, 28, 1889, 1890, 995, 3099, 0)$
`101010110011`	$(1, 2, 1649, 1650, 20, 21, 851, 821, 822, 3272, 0)$
`101010110101`	$(1, 2, 98, 99, 297, 298, 292, 94, 95, 2049, 0)$
`101011010101`	$(1, 2, 1953, 1954, 1756, 1750, 1751, 1949, 1950, 2049, 0)$
`101011011111`	$(1, 2, 1147, 1148, 1415, 1493, 1494, 1716, 705, 3523, 0)$
`101011101001`	$(1, 2, 1793, 1794, 1762, 543, 749, 750, 2045, 2049, 0)$
`101011101111`	$(1, 2, 1631, 1632, 1701, 767, 394, 395, 848, 3281, 0)$
`101011110001`	$(1, 2, 132, 133, 1380, 738, 258, 2044, 2045, 2049, 0)$
`101011111011`	$(1, 2, 1008, 1009, 150, 896, 54, 712, 1525, 2569, 0)$
`101100000011`	$(1, 2, 1823, 904, 905, 906, 907, 908, 909, 3185, 0)$
`101100001001`	$(1, 2, 1184, 1411, 1412, 1413, 1414, 1415, 2045, 2049, 0)$
`101100010001`	$(1, 2, 1751, 1446, 1447, 1448, 1449, 2044, 2045, 2049, 0)$
`101100110011`	$(1, 2, 1239, 1041, 1042, 1043, 96, 616, 617, 3477, 0)$
`101100111111`	$(1, 2, 1346, 1296, 1297, 1298, 854, 991, 825, 2400, 0)$
`101101000001`	$(1, 2, 319, 1392, 1393, 2042, 2043, 2044, 2045, 2049, 0)$
`101101001011`	$(1, 2, 671, 1421, 1422, 1415, 1416, 1417, 333, 3761, 0)$
`101101011001`	$(1, 2, 1383, 799, 800, 459, 460, 1120, 2045, 2049, 0)$
`101101011111`	$(1, 2, 215, 1611, 1612, 1554, 1555, 318, 1230, 3989, 0)$
`101101100101`	$(1, 2, 1826, 1053, 1054, 1901, 1821, 1822, 1823, 2049, 0)$
`101101101111`	$(1, 2, 1934, 1316, 1317, 1290, 1872, 1873, 1458, 2106, 0)$
`101101111101`	$(1, 2, 1938, 528, 529, 1976, 468, 1323, 1935, 2049, 0)$
`101110000111`	$(1, 2, 1329, 90, 982, 983, 984, 985, 986, 3432, 0)$
`101110001011`	$(1, 2, 1795, 1020, 1013, 1014, 1015, 1016, 895, 3199, 0)$
`101110010011`	$(1, 2, 219, 1474, 289, 290, 291, 106, 107, 3987, 0)$
`101110010101`	$(1, 2, 1869, 2021, 15, 16, 17, 1865, 1866, 2049, 0)$
`101110101111`	$(1, 2, 1111, 243, 502, 503, 1661, 1662, 1239, 3541, 0)$
`101110110111`	$(1, 2, 752, 714, 1694, 1695, 1689, 169, 170, 2697, 0)$
`101110111101`	$(1, 2, 947, 1932, 207, 208, 353, 1777, 944, 2049, 0)$
`101111001001`	$(1, 2, 596, 1960, 1882, 1627, 1628, 1629, 2045, 2049, 0)$
`101111011011`	$(1, 2, 1226, 1349, 1740, 1805, 1806, 1345, 1634, 2460, 0)$
`101111011101`	$(1, 2, 1104, 271, 1695, 1840, 1841, 116, 1101, 2049, 0)$
`101111100111`	$(1, 2, 1819, 1207, 1774, 972, 650, 651, 652, 3187, 0)$

Table F.13: Shifts of Galois LFSRs of length 11 continued

Polynomial	Shifts H
101111101101	$(1, 2, 113, 725, 1580, 72, 1519, 1520, 110, 2049, 0)$
110000001011	$(1, 310, 311, 312, 313, 314, 315, 316, 307, 3787, 0)$
110000001101	$(1, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 225, 2049, 0)$
110000011001	$(1, 1207, 1208, 1209, 1210, 1211, 1212, 490, 2045, 2049, 0)$
110000011111	$(1, 1208, 1209, 1210, 1211, 1212, 1213, 1571, 1957, 2889, 0)$
110001010111	$(1, 1505, 1506, 1507, 1508, 1331, 1332, 52, 53, 2592, 0)$
110001100001	$(1, 1189, 1190, 1191, 1192, 849, 2043, 2044, 2045, 2049, 0)$
110001101011	$(1, 828, 829, 830, 831, 1563, 1279, 1280, 825, 3269, 0)$
110001110011	$(1, 1477, 1478, 1479, 1480, 1234, 1742, 1473, 1474, 2620, 0)$
110010000101	$(1, 1798, 1799, 1800, 1541, 1542, 1543, 1544, 1545, 2049, 0)$
110010001001	$(1, 249, 250, 251, 295, 296, 297, 298, 2045, 2049, 0)$
110010010111	$(1, 802, 803, 804, 2022, 2023, 2024, 252, 253, 3295, 0)$
110010011011	$(1, 1173, 1174, 1175, 549, 550, 551, 1843, 1170, 2924, 0)$
110010011101	$(1, 1940, 1941, 1942, 1757, 1758, 1759, 574, 1829, 2049, 0)$
110010110011	$(1, 238, 239, 240, 1179, 1180, 766, 234, 235, 3859, 0)$
110010111111	$(1, 1530, 1531, 1532, 1897, 1898, 905, 490, 185, 2567, 0)$
110011000111	$(1, 1211, 1212, 1213, 1258, 1957, 1958, 1959, 1960, 2886, 0)$
110011001101	$(1, 1430, 1431, 1432, 1952, 1005, 1006, 1007, 809, 2049, 0)$
110011010011	$(1, 1812, 1813, 1814, 1282, 868, 869, 1808, 1809, 2285, 0)$
110011010101	$(1, 1225, 1226, 1227, 1197, 2027, 2028, 398, 399, 2049, 0)$
110011100011	$(1, 573, 574, 575, 306, 814, 568, 569, 570, 3524, 0)$
110011101001	$(1, 951, 952, 953, 639, 657, 1059, 1060, 2045, 2049, 0)$
110011110111	$(1, 549, 550, 551, 269, 694, 639, 127, 128, 3548, 0)$
110100000011	$(1, 1740, 1741, 1732, 1733, 1734, 1735, 1736, 1737, 2357, 0)$
110100001111	$(1, 374, 375, 1112, 1113, 1114, 1115, 1116, 1775, 3723, 0)$
110100011101	$(1, 1152, 1153, 1032, 1033, 1034, 1035, 1028, 253, 2049, 0)$
110100100111	$(1, 1354, 1355, 1951, 1952, 1953, 1265, 1266, 1267, 2743, 0)$
110100101101	$(1, 1714, 1715, 631, 632, 633, 626, 627, 1377, 2049, 0)$
110101000001	$(1, 1608, 1609, 1910, 1911, 2042, 2043, 2044, 2045, 2049, 0)$
110101000111	$(1, 1497, 1498, 408, 409, 1462, 1463, 1464, 1465, 2600, 0)$
110101010101	$(1, 1052, 1053, 158, 159, 2020, 2021, 52, 53, 2049, 0)$
110101011001	$(1, 104, 105, 2031, 2032, 1190, 1191, 1526, 2045, 2049, 0)$
110101100011	$(1, 1222, 1223, 768, 769, 485, 1217, 1218, 1219, 2875, 0)$
110101101111	$(1, 628, 629, 471, 472, 1379, 1877, 1878, 1988, 3469, 0)$
110101110001	$(1, 857, 858, 773, 774, 788, 1373, 2044, 2045, 2049, 0)$
110110010011	$(1, 877, 878, 205, 1497, 1498, 1499, 873, 874, 3220, 0)$
110110011111	$(1, 330, 331, 745, 1425, 1426, 1427, 984, 764, 3767, 0)$
110110101001	$(1, 376, 377, 1800, 697, 698, 317, 318, 2045, 2049, 0)$
110110111011	$(1, 768, 769, 182, 1937, 1938, 1932, 1212, 765, 3329, 0)$
110110111101	$(1, 413, 414, 703, 242, 243, 308, 699, 822, 2049, 0)$
110111001001	$(1, 285, 286, 1178, 929, 639, 640, 641, 2045, 2049, 0)$
110111010111	$(1, 478, 479, 300, 1119, 771, 772, 161, 162, 3619, 0)$
110111011011	$(1, 1282, 1283, 836, 116, 110, 111, 1866, 1279, 2815, 0)$
110111100001	$(1, 940, 941, 1423, 693, 723, 2043, 2044, 2045, 2049, 0)$

Table F.14: Shifts of Galois LFSRs of length 11 continued

Polynomial	Shifts H
`110111100111`	$(1, 1156, 1157, 1568, 1133, 51, 865, 866, 867, 2941, 0)$
`110111110101`	$(1, 522, 523, 1336, 1994, 1152, 1898, 1039, 1040, 2049, 0)$
`111000000101`	$(1, 1996, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 2049, 0)$
`111000011101`	$(1, 1385, 1062, 1063, 1064, 1065, 1066, 1958, 719, 2049, 0)$
`111000100001`	$(1, 1831, 1948, 1949, 1950, 1951, 2043, 2044, 2045, 2049, 0)$
`111000100111`	$(1, 719, 1262, 1263, 1264, 1265, 1257, 1258, 1259, 3378, 0)$
`111000101011`	$(1, 553, 583, 584, 585, 586, 1639, 1640, 550, 3544, 0)$
`111000110011`	$(1, 839, 88, 89, 90, 91, 790, 835, 836, 3258, 0)$
`111000111001`	$(1, 1684, 925, 926, 927, 928, 1554, 557, 2045, 2049, 0)$
`111001000111`	$(1, 1331, 789, 790, 791, 783, 784, 785, 786, 2766, 0)$
`111001001011`	$(1, 696, 781, 782, 783, 95, 96, 97, 693, 3401, 0)$
`111001010101`	$(1, 227, 1348, 1349, 1350, 642, 643, 449, 450, 2049, 0)$
`111001011111`	$(1, 899, 844, 845, 846, 1717, 1718, 644, 841, 3198, 0)$
`111001110001`	$(1, 657, 178, 179, 180, 385, 573, 2044, 2045, 2049, 0)$
`111001111011`	$(1, 894, 1181, 1182, 1183, 1997, 915, 480, 891, 3203, 0)$
`111001111101`	$(1, 1140, 1396, 1397, 1398, 1076, 274, 841, 229, 2049, 0)$
`111010000001`	$(1, 1203, 1597, 1598, 2041, 2042, 2043, 2044, 2045, 2049, 0)$
`111010010011`	$(1, 1248, 1795, 1796, 24, 25, 26, 1244, 1245, 2849, 0)$
`111010011111`	$(1, 70, 1924, 1925, 1823, 1824, 1825, 204, 1921, 4027, 0)$
`111010100011`	$(1, 545, 1995, 1996, 716, 717, 540, 541, 542, 3552, 0)$
`111010111011`	$(1, 1572, 1886, 1887, 1276, 1277, 929, 1748, 1569, 2525, 0)$
`111011001111`	$(1, 1466, 1781, 1782, 1628, 296, 297, 298, 1778, 2631, 0)$
`111011011101`	$(1, 650, 1878, 1879, 359, 353, 354, 1334, 1296, 2049, 0)$
`111011110011`	$(1, 1501, 1920, 1921, 1409, 1354, 1779, 1497, 1498, 2596, 0)$
`111011111001`	$(1, 1424, 1439, 1440, 1709, 421, 990, 811, 2045, 2049, 0)$
`111100001011`	$(1, 1676, 273, 932, 933, 934, 935, 936, 1673, 2421, 0)$
`111100011001`	$(1, 1160, 996, 1431, 1432, 1433, 1434, 104, 2045, 2049, 0)$
`111100110001`	$(1, 1180, 339, 1491, 1492, 1493, 618, 2044, 2045, 2049, 0)$
`111100110111`	$(1, 584, 270, 1750, 1751, 1752, 420, 266, 267, 3513, 0)$
`111101011101`	$(1, 1494, 809, 386, 387, 1545, 1546, 1805, 937, 2049, 0)$
`111101101011`	$(1, 1422, 60, 170, 171, 669, 1576, 1577, 1419, 2675, 0)$
`111101101101`	$(1, 59, 590, 175, 176, 758, 731, 732, 114, 2049, 0)$
`111101110101`	$(1, 1234, 1200, 1653, 1654, 1281, 347, 416, 417, 2049, 0)$
`111110000011`	$(1, 842, 91, 477, 835, 836, 837, 838, 839, 3255, 0)$
`111110010001`	$(1, 26, 1767, 76, 94, 95, 96, 2044, 2045, 2049, 0)$
`111110010111`	$(1, 1980, 127, 1844, 223, 224, 225, 123, 124, 2117, 0)$
`111110011011`	$(1, 1720, 1284, 1064, 621, 622, 623, 1303, 1717, 2377, 0)$
`111110100111`	$(1, 1151, 1207, 1404, 330, 331, 1202, 1203, 1204, 2946, 0)$
`111110101101`	$(1, 1942, 818, 1730, 493, 494, 436, 437, 1833, 2049, 0)$
`111110110101`	$(1, 1476, 1343, 332, 554, 555, 633, 900, 901, 2049, 0)$
`111111001101`	$(1, 353, 1223, 1057, 1194, 750, 751, 752, 702, 2049, 0)$
`111111010011`	$(1, 520, 1863, 1558, 1143, 150, 151, 516, 517, 3577, 0)$
`111111100101`	$(1, 1859, 365, 1481, 1057, 540, 1665, 1666, 1667, 2049, 0)$
`111111101001`	$(1, 598, 1098, 1792, 1822, 745, 1690, 1691, 2045, 2049, 0)$

Appendix G

Grain's Nonlinear Filtering Function Keystream Bias

The following results describe the run counts of the output of the 5-bit boolean function used in Grain [16],

$$\begin{aligned} h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 \\ + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4. \end{aligned} \tag{G.1}$$

Primitive polynomials of degree 5 to 9 are used to clock the LFSR and are represented here in the form $d_n2^n + d_{n-1}2^{n-1} + \cdots + d_12 + d_0$ such that $f(x) = d_nx^n + d_{n-1}x^{n-1} + \cdots + d_1x + d_0$. The tuples contain the number of length m bit runs that are contained in one period of keystream, $m = 1, \ldots, n$. The input sequences used are generated by Galois LFSRs and have run frequencies of the form $(2^{n-2}, \ldots, 2, 1, 1)$.

Table G.1: Keystream bias from Galois LFSRs of length 5

Polynomial	Count of m-tuples
`100101`	$(18, 4, 2, 0, 1)$
`101001`	$(15, 4, 2, 2, 0)$
`101111`	$(11, 6, 1, 2, 0)$
`110111`	$(7, 4, 3, 1, 2)$
`111011`	$(14, 9, 3, 0, 0)$
`111101`	$(13, 8, 2, 0, 0)$

Table G.2: Keystream bias from Galois LFSRs of length 6

Polynomial	Count of m-tuples
`1000011`	$(23, 14, 2, 1, 2, 0)$
`1011011`	$(23, 7, 7, 2, 0, 0)$
`1100001`	$(23, 11, 7, 0, 1, 0)$
`1100111`	$(19, 15, 5, 1, 0, 0)$
`1101101`	$(22, 12, 1, 2, 2, 0)$
`1110011`	$(23, 14, 5, 1, 1, 0)$

Table G.3: Keystream bias from Galois LFSRs of length 7

Polynomial	Count of m-tuples
`10000011`	$(42, 19, 13, 6, 0, 0, 0)$
`10001001`	$(44, 22, 10, 1, 2, 1, 0)$
`10001111`	$(40, 22, 10, 3, 2, 0, 0)$
`10010001`	$(42, 24, 7, 5, 0, 1, 0)$
`10011101`	$(42, 23, 10, 0, 2, 0, 1)$
`10100111`	$(40, 20, 10, 4, 1, 0, 0)$
`10101011`	$(42, 22, 7, 4, 2, 0, 0)$
`10111001`	$(43, 21, 10, 2, 1, 1, 0)$
`10111111`	$(43, 21, 9, 7, 0, 0, 0)$
`11000001`	$(41, 22, 10, 1, 0, 1, 1)$
`11001011`	$(41, 22, 9, 3, 2, 0, 0)$
`11010011`	$(45, 23, 9, 3, 2, 0, 0)$
`11010101`	$(44, 19, 12, 0, 3, 0, 0)$
`11100101`	$(42, 21, 10, 3, 2, 0, 0)$
`11101111`	$(39, 22, 8, 4, 0, 1, 1)$
`11110001`	$(42, 23, 7, 4, 1, 1, 0)$
`11110111`	$(40, 22, 6, 6, 1, 0, 0)$
`11111101`	$(41, 22, 10, 1, 3, 0, 0)$

Table G.4: Keystream bias from Galois LFSRs of length 8

Polynomial	Count of m-tuples
`100011101`	$(79, 40, 18, 7, 1, 2, 0, 0)$
`100101011`	$(81, 40, 16, 11, 2, 0, 0, 0)$
`100101101`	$(81, 41, 19, 7, 1, 2, 0, 0)$
`101001101`	$(82, 41, 18, 6, 3, 1, 0, 0)$
`101011111`	$(78, 42, 19, 4, 4, 1, 0, 0)$
`101100011`	$(80, 42, 18, 5, 6, 0, 0, 0)$
`101100101`	$(81, 40, 20, 5, 4, 1, 0, 0)$
`101101001`	$(80, 41, 18, 7, 2, 0, 1, 0)$
`101110001`	$(78, 41, 21, 6, 1, 1, 1, 0)$
`110000111`	$(79, 42, 19, 5, 5, 0, 0, 0)$
`110001101`	$(79, 42, 19, 6, 3, 1, 0, 0)$
`110101001`	$(80, 40, 19, 7, 2, 0, 1, 0)$
`111000011`	$(81, 41, 18, 7, 2, 0, 1, 0)$
`111001111`	$(78, 40, 18, 7, 4, 0, 0, 0)$
`111100111`	$(80, 42, 20, 2, 5, 1, 0, 0)$
`111110101`	$(80, 42, 17, 8, 2, 1, 0, 0)$

Table G.5: Keystream bias from Galois LFSRs of length 9

Polynomial	Count of m-tuples
`1000010001`	$(154, 82, 36, 14, 5, 1, 0, 1, 0)$
`1000011011`	$(156, 80, 37, 12, 6, 1, 1, 0, 0)$
`1000100001`	$(157, 80, 37, 14, 4, 0, 1, 1, 0)$
`1000101101`	$(156, 81, 38, 11, 8, 0, 1, 0, 0)$
`1000110011`	$(159, 80, 37, 11, 7, 2, 0, 0, 0)$
`1001011001`	$(158, 82, 36, 15, 5, 1, 0, 1, 0)$
`1001011111`	$(156, 82, 36, 12, 7, 2, 0, 0, 0)$
`1001101001`	$(158, 82, 37, 14, 4, 0, 1, 1, 0)$
`1001101111`	$(153, 83, 37, 12, 5, 3, 0, 0, 0)$
`1001110111`	$(153, 81, 37, 12, 5, 3, 0, 0, 0)$
`1001111101`	$(158, 82, 38, 13, 5, 0, 2, 0, 0)$
`1010000111`	$(154, 81, 37, 12, 5, 3, 0, 0, 0)$
`1010010101`	$(158, 81, 38, 11, 8, 0, 1, 0, 0)$
`1010100011`	$(159, 82, 36, 12, 6, 1, 1, 0, 0)$
`1010100101`	$(156, 81, 37, 12, 6, 1, 1, 0, 0)$
`1010101111`	$(153, 81, 36, 12, 7, 2, 0, 0, 0)$
`1010110111`	$(155, 81, 38, 13, 4, 2, 1, 0, 0)$
`1010111101`	$(155, 81, 36, 12, 6, 1, 1, 0, 0)$
`1011001111`	$(155, 80, 36, 13, 5, 3, 0, 0, 0)$
`1011010001`	$(154, 82, 37, 14, 4, 0, 1, 1, 0)$
`1011011011`	$(155, 80, 37, 12, 7, 2, 0, 0, 0)$
`1011110101`	$(155, 81, 36, 12, 6, 1, 1, 0, 0)$
`1011111001`	$(158, 81, 36, 13, 5, 1, 0, 1, 0)$

Table G.6: Keystream bias from Galois LFSRs of length 9 continued

Polynomial	Count of m-tuples
1100010011	$(159, 82, 38, 11, 7, 2, 0, 0, 0)$
1100010101	$(158, 80, 36, 12, 8, 0, 1, 0, 0)$
1100011111	$(155, 83, 36, 12, 5, 4, 0, 0, 0)$
1100100011	$(156, 81, 36, 12, 7, 1, 1, 0, 0)$
1100110001	$(155, 81, 37, 14, 5, 1, 0, 1, 0)$
1100111011	$(157, 81, 36, 11, 10, 1, 0, 0, 0)$
1101001111	$(153, 82, 36, 12, 5, 3, 0, 0, 0)$
1101011011	$(157, 81, 36, 11, 7, 2, 0, 0, 0)$
1101100001	$(157, 80, 36, 13, 5, 1, 0, 1, 0)$
1101101011	$(160, 83, 38, 12, 6, 1, 1, 0, 0)$
1101101101	$(154, 81, 36, 13, 6, 1, 1, 0, 0)$
1101110011	$(158, 81, 36, 12, 9, 1, 0, 0, 0)$
1101111111	$(156, 83, 37, 13, 4, 2, 1, 0, 0)$
1110000101	$(158, 82, 37, 12, 6, 1, 1, 0, 0)$
1110001111	$(154, 81, 36, 13, 4, 2, 1, 0, 0)$
1110110101	$(155, 81, 37, 13, 5, 0, 2, 0, 0)$
1110111001	$(157, 81, 39, 12, 7, 0, 0, 1, 0)$
1111000111	$(157, 83, 36, 14, 4, 2, 1, 0, 0)$
1111001011	$(156, 80, 38, 10, 9, 1, 0, 0, 0)$
1111001101	$(157, 82, 36, 12, 6, 1, 1, 0, 0)$
1111010101	$(159, 81, 36, 12, 6, 1, 1, 0, 0)$
1111011001	$(157, 81, 36, 13, 6, 1, 0, 1, 0)$
1111100011	$(158, 81, 37, 12, 7, 2, 0, 0, 0)$
1111101001	$(157, 81, 37, 14, 4, 0, 1, 1, 0)$
1111111011	$(155, 80, 37, 11, 7, 2, 0, 0, 0)$

Table G.7: Keystream bias from Galois LFSRs of length 10

Polynomial	Count of m-tuples
`10000001001`	$(311, 161, 73, 24, 12, 3, 0, 0, 1, 0)$
`10000011011`	$(306, 160, 73, 25, 11, 5, 1, 0, 0, 0)$
`10000100111`	$(308, 162, 72, 24, 12, 3, 2, 0, 0, 0)$
`10000101101`	$(307, 162, 72, 24, 12, 2, 1, 1, 0, 0)$
`10001100101`	$(309, 162, 72, 24, 12, 2, 1, 1, 0, 0)$
`10001101111`	$(305, 161, 72, 25, 11, 3, 2, 0, 0, 0)$
`10010000001`	$(310, 163, 73, 24, 13, 1, 1, 0, 1, 0)$
`10010001011`	$(308, 162, 73, 24, 10, 5, 1, 0, 0, 0)$
`10011000101`	$(310, 162, 72, 24, 11, 4, 0, 2, 0, 0)$
`10011010111`	$(306, 163, 72, 24, 11, 3, 2, 0, 0, 0)$
`10011100111`	$(310, 161, 72, 24, 12, 1, 3, 0, 0, 0)$
`10011110011`	$(310, 163, 72, 24, 11, 3, 2, 0, 0, 0)$
`10011111111`	$(308, 160, 73, 25, 11, 3, 2, 0, 0, 0)$
`10100001101`	$(308, 161, 73, 24, 11, 4, 0, 1, 0, 0)$
`10100011001`	$(309, 161, 74, 25, 12, 3, 0, 0, 1, 0)$
`10100100011`	$(310, 160, 73, 24, 10, 5, 1, 0, 0, 0)$
`10100110001`	$(308, 162, 74, 24, 12, 3, 0, 0, 1, 0)$
`10100111101`	$(306, 162, 73, 24, 12, 2, 1, 1, 0, 0)$
`10101000011`	$(312, 161, 72, 24, 10, 5, 1, 0, 0, 0)$
`10101010111`	$(305, 162, 73, 24, 11, 3, 2, 0, 0, 0)$
`10101101011`	$(310, 160, 72, 24, 10, 5, 1, 0, 0, 0)$
`10110000101`	$(311, 162, 72, 24, 12, 2, 1, 1, 0, 0)$
`10110001111`	$(307, 162, 73, 24, 11, 3, 2, 0, 0, 0)$
`10110010111`	$(307, 163, 72, 24, 12, 1, 3, 0, 0, 0)$
`10110100001`	$(309, 161, 73, 24, 12, 3, 0, 0, 1, 0)$
`10111000111`	$(308, 161, 73, 25, 11, 3, 2, 0, 0, 0)$
`10111100101`	$(310, 163, 72, 24, 12, 2, 1, 1, 0, 0)$
`10111110111`	$(305, 161, 73, 24, 12, 1, 3, 0, 0, 0)$
`10111111011`	$(307, 161, 73, 24, 11, 5, 1, 0, 0, 0)$

Table G.8: Keystream bias from Galois LFSRs of length 10 continued

Polynomial	Count of m-tuples
`11000010011`	$(309, 162, 72, 25, 11, 3, 2, 0, 0, 0)$
`11000010101`	$(309, 161, 73, 24, 12, 2, 1, 1, 0, 0)$
`11000100101`	$(311, 161, 74, 24, 11, 4, 0, 1, 0, 0)$
`11000110111`	$(306, 162, 72, 24, 11, 3, 2, 0, 0, 0)$
`11001000011`	$(308, 162, 74, 24, 11, 5, 1, 0, 0, 0)$
`11001001111`	$(305, 160, 73, 24, 12, 1, 3, 0, 0, 0)$
`11001011011`	$(311, 163, 72, 24, 11, 3, 2, 0, 0, 0)$
`11001111001`	$(309, 163, 72, 24, 12, 3, 0, 0, 1, 0)$
`11001111111`	$(310, 160, 73, 24, 11, 3, 2, 0, 0, 0)$
`11010001001`	$(310, 161, 73, 24, 12, 3, 0, 0, 1, 0)$
`11010110101`	$(308, 163, 73, 24, 11, 4, 0, 1, 0, 0)$
`11011000001`	$(308, 161, 73, 24, 12, 3, 0, 0, 1, 0)$
`11011010011`	$(313, 162, 72, 24, 10, 5, 1, 0, 0, 0)$
`11011011111`	$(306, 162, 72, 25, 12, 1, 3, 0, 0, 0)$
`11011111101`	$(307, 161, 72, 25, 12, 4, 0, 1, 0, 0)$
`11100010111`	$(306, 161, 72, 24, 12, 1, 3, 0, 0, 0)$
`11100011101`	$(307, 161, 72, 24, 11, 4, 0, 1, 0, 0)$
`11100100001`	$(312, 161, 73, 24, 12, 3, 0, 0, 1, 0)$
`11100111001`	$(309, 162, 73, 24, 14, 1, 1, 0, 1, 0)$
`11101000111`	$(307, 161, 72, 25, 11, 3, 2, 0, 0, 0)$
`11101001101`	$(313, 162, 72, 24, 12, 2, 1, 1, 0, 0)$
`11101010101`	$(310, 161, 72, 24, 11, 4, 0, 1, 0, 0)$
`11101011001`	$(311, 164, 72, 25, 14, 1, 1, 0, 1, 0)$
`11101100011`	$(308, 163, 72, 24, 11, 3, 2, 0, 0, 0)$
`11101111101`	$(307, 161, 72, 24, 11, 4, 0, 1, 0, 0)$
`11110001101`	$(307, 162, 72, 25, 11, 4, 0, 1, 0, 0)$
`11110010011`	$(311, 161, 73, 25, 11, 3, 2, 0, 0, 0)$
`11110110001`	$(307, 161, 72, 24, 12, 3, 0, 0, 1, 0)$
`11111011011`	$(309, 161, 75, 24, 11, 3, 2, 0, 0, 0)$
`11111110011`	$(311, 164, 72, 25, 12, 3, 2, 0, 0, 0)$
`11111111001`	$(308, 162, 73, 24, 12, 3, 0, 0, 1, 0)$