

Title: A model for the self-synchronizing n-valued LFSR based descrambler.

Author: Peter Lablans, Ternarylogic LLC.

Date: May 5, 2005

The advantages of n-valued LFSR based scramblers and descramblers.

LFSR based scramblers and descramblers have attractive advantages over other means of n-valued coding and decoding:

1. easy to implement and operate in hardware or software
2. streaming decoder, no additional requirements for word or symbol synchronization
3. self-synchronizing, symbol errors will not propagate
4. high security, inherently superior correlation properties
5. will break up most symbol patterns and can create a pseudo-randomized signal
6. faster and more efficient in sequence detection than correlation methods
7. no coding overhead

Introduction:

The descrambler as the counterpart of an n-valued LFSR based scrambler is a long standing problem that has been, up till now, unresolved. While it is possible to create an n-value LFSR based scrambler, there was no model for the matching descrambler, which rendered the n-valued LFSR scrambler useless for coding purposes.

While interest in randomizing non-binary digital signals has been limited, it now appears to be gaining interest rapidly. Especially as multi-level digital signals are being considered for bandwidth and capacity constrained, applications. Applying n-level digital signals in digital document watermarking is another emerging application. Solving the issue of the n-valued descrambler has become a timely issue.

A frustrating aspect is that the problem seems so simple. One assumes that there should be a simple and elegant solution, comparable to the binary LFSR based descrambler. Such a solution now exists. This paper describes the underlying approach to that solution.

The LFSR based scrambler.

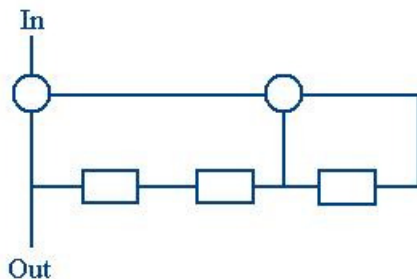
A scrambler is a coder with as its purpose to change the appearance of a sequence of symbols. An LFSR based scrambler is a streaming cipher, working on a continuous and, if desired, on a real-time basis. There is no need for block or word synchronization as scramblers are symbol oriented.

The scrambled sequence should obviously not look like the original sequence. One reason for applying a scrambler is to break up series of undesirable patterns. An all 0s sequence is often such an undesirable pattern. All 0s sequences have no transitions and may upset circuitry for clock extraction, detection and synchronization purposes.

The most common form of scrambling is the combining by way of a logic function of an “unknown” to be scrambled sequence with a “known” or at least “predictable” sequence. In binary technology the combining function is generally the Exclusive Or function. As one symbol is replaced by another one these codes are called substitution codes.

While a LFSR based scrambler is a substitution coder it differs from standard substitution coders by not using a substitution code book or tableau.

The following figure shows a diagram of a Linear Feedback Shift Register scrambler.



The square blocks represent individual elements of a shift register. Each element can retain one symbol or symbol value. Not shown (but assumed to be present) is a clock signal under which the shift register operates.

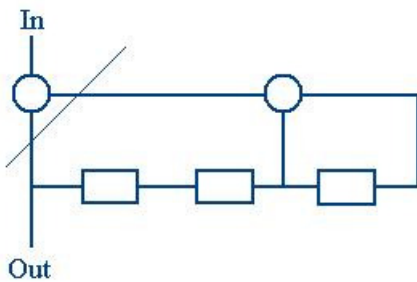
When the clock signal is for instance high, each element assumes the symbol value presented at its left side input. One can say that on a high clock signal each symbol is shifted one position to the right.

The circles in the diagram represent a logic function which operates on two inputs and generates a signal on an output. All signals, in this configuration, are moving in a counter clock-wise direction.

The to be scrambled sequence is provided by input In and the scrambled sequence is provided on output Out.

The diagram should be familiar to those skilled in the art as a binary scrambler wherein all logic functions are Exclusive Or functions.

Further analysis shows what actually happens in the scrambler. The following figure shows the scrambler separated into two parts.



The part of the scrambler to the right of the slanted line may be called the Scrambling Unit. The part to the left of the slanted line may be called the Combining Function.

What happens is the following: the initial content of the shift register causes the Scrambling Unit to generate a symbol which is provided to one input of the Combining Function. A to be scrambled symbol is provided on the input In of the Combining Function. And the Combining Function generates a (scrambled) symbol on Out.

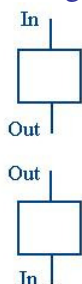
The substitution coding that takes place in this scrambler is the combining of a symbol with a delayed and altered version of a previous one.

There is another way of looking at the coding process. The Scrambling Unit may be considered to be a sequence generator that produces a Scrambling Sequence. The Combining Function may be considered to be combining an incoming data sequence with the locally generated Scrambling Sequence. This assumption is valid because the incoming symbol is not yet reflected in the Scrambling Sequence.

While imprecise as a model (though elsewhere we have put this in verifiable expressions), this approach will explain the working model for the descrambler.

The LFSR based descrambler.

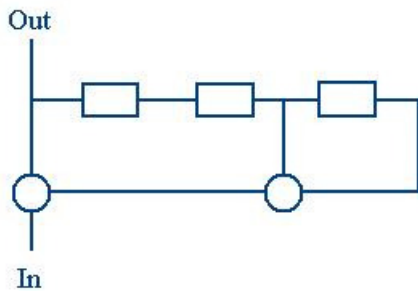
One would like to recover the original sequence from the scrambled sequence at the receiving end. A diagram of the process of scrambling and descrambling is shown in the following figure:





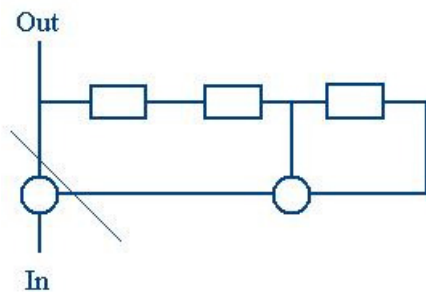
A signal Out is created from a signal In. It is reasonable to expect that a signal In can be created from a signal Out by executing the scrambling process in reverse. One could call this process “unscrambling”. The diagram also suggests that the terms scrambler and descrambler are relative to their usage, and that the scrambler can be used as the descrambler and the descrambler as the scrambler.

The following figure shows a diagram of the LFSR based descrambler that will descramble the signal that was generated by the previously shown scrambler. This model complies with the known binary descrambler.



In the binary LFSR based descrambler the applied functions are also XOR functions. XOR functions are identical to modulo-2 additions. This has led to one suggested model for the n-valued scrambler and descrambler. In this (flawed) n-valued scrambler and descrambler model all functions are modulo-n adders. It is easy to check that in such a configuration the descrambler does not descramble for n greater than 2.

In further analysis, like before with the scrambler, the descrambler can be separated into two parts as shown in the following figure.



The part to the right of the line can be called the Descrambling Unit and the part to the left of the slanted line is the Combining Function.

This all appears to comply with the scrambling and unscrambling concepts. However a closer look at the Descrambling Unit, and especially the direction of the signals, shows that the Descrambling Unit in the descrambler is identical to the Scrambling Unit in the scrambler. The signals in the descrambler now move clock-wise, which at first sight is the reverse of the scrambler. In fact the flow of the signal is exactly the same as in the Scrambling Unit of the scrambler.

The conclusion has to be that a signal is descrambled by further scrambling it. It is thus not correct to speak about the descrambler as an “unscrambler”, because the descrambling process is not the reverse of the scrambling process.

This may provide the explanation why it has been apparently so difficult to find solutions for the non-binary LFSR based descrambler: it has been treated as an “unscrambler”, wherein all the processes of the scrambler have to be reversed.

The key to the solution is to consider the function to the left of the slanted line in the descrambler diagram to be a Separating Function. At this stage we should be reminded of the fact that the Scrambling Unit is considered to be a local sequence generator.

The Separating Function in the descrambler separates a locally generated sequence from a sequence. This sequence was originally created in the scrambler by combining a locally generated sequence with a data sequence by a Combining Function. The conclusion has to be that: by separating a local sequence from a sequence that was created by combining such local sequence to a data sequence one will be left with the original data sequence.

Thus the problem has been solved. Specific functions have been identified elsewhere and the solution is part of a series of pending Ternarylogic LLC patents.

Related literature.

There is an abundance of literature on binary and non-binary, shift register based solutions for coding and sequence generation. It has its theoretical basis in Finite Field theory. The [home page](#) of Professor Guang Gong of the University of Waterloo contains lecture notes on Finite Field Theory and sequence design. The book: Sequence Design for Communications Applications, by Fan and Darnell provides a good connection between Finite Field theory, LFSR circuits and sequence design.

A [webpage](#) with a symbolic explanation of the binary LFSR based descrambler is: The Mathematics of LFSR Digital Signal Spreading by Richard DeVaul of MIT Media Lab. With appropriate changes in notation for non-binary operations, and by putting conditions on the logic operations, the expressions can describe all n-valued LFSR based descramblers.

Inventions.

The LFSR based descramblers are subjects of a series of patents and patent applications.

The original binary LFSR based scrambler and descrambler were invented by Fracassi and Tammaru of Bell Laboratories. They applied for patent with the US Patent and Trademark Office in 1965 and were granted Patent **4,304,962** in 1981.

At least one known flawed model of the n-valued LFSR based [descrambler](http://www.mathworks.com/access/helpdesk_r13/help/toolbox/commblks/ref/descrambler.html) comprised of modulo-n adders is known to exist and is part of the Matlab software package. It can be found at URL: http://www.mathworks.com/access/helpdesk_r13/help/toolbox/commblks/ref/descrambler.html

An n-valued shift-register-based coder, comprising adders and a decoder or “unscrambler”, comprising subtraction functions have been invented by Jason Mix, Michael Leddige and Howard Heck of Intel. They filed for a Patent on this invention with the USPTO in 2003 and the Patent Application is published in the USPTO Pregrant Patent Application database as number **20030063677**.

The solution for the n-valued LFSR based scramblers and descramblers has been invented by Peter Lablans, the author of this article. A first generic, n-valued descrambler solution has been submitted to USPTO for Patent in 2003. The Patent Application is published in the USPTO Pregrant Patent Application database as number **20050053240**. A second non-provisional patent application, comprising all generic descrambling solutions, has been submitted to USPTO in 2004 and will be published later this year.

United States Patents and Published Pregrant Patent Applications can be viewed at www.uspto.gov .

About Ternarylogic LLC: Ternarylogic LLC is a New Jersey based R&D company. Its mission is to create novel MVL technology. The company owns a portfolio of inventions related to scramblers/descramblers, sequence generators and sequence detectors, sequence correlators, gates and inverters based circuitry, non-binary multipliers, latches and other non-volatile memory elements, optical disk applications and MC-DSSS technology.

© 2005, Ternarylogic LLC. All rights reserved