Using classifiers to predict linear feedback shift registers

Hernández J.C., Isasi P.[1], Sierra J.M., Mex-Perera, C.[2], Ramos B.

Computer Science Department
Security Group
Carlos III University
28911 Leganés
Madrid, Spain

{jcesar, sierra, benja1}@inf.uc3m.es

(1)

Computer Science Department
Artificial Intelligence Group
Carlos III University
28911 Leganés
Madrid, Spain

isasi@ia.uc3m.es

(2)

Department of Electronic & Electrical Engineering
Bradford University
Bradford, UK

J.C.Mex-Perera@brad.ac.uk

Abstract

In [1] some new ideas that justify the use of artificial intelligence techniques in cryptanalysis are presented. The main objective of that paper was to show that the theoretical next bit prediction problem can be transformed into a classification problem, and this classification problem could be solved with the aid of some AI algorithms. In particular, they showed how a well-known classifier called c4.5 could predict the next bit generated by a linear feedback shift register (LFSR, a widely used model of pseudorandom number generator) very efficiently and, most importantly, without any previous knowledge over the model used.

In this paper we look for other classifiers, apart from c4.5, that could be useful in the prediction of LFSRs. We conclude that the selection of c4.5 in [1] was adequate, because it shows the best accuracy of all the classifiers tested. However, we have found other classifiers that produce interesting results, and we suggest that these algorithms must be taken into account in the future, when trying to predict more complex LFSR-based models. Finally, we show some other properties that make the c4.5 algorithm the best choice for this particular cryptanalytic problem.

0-7803-6636-0/01/$10.00 ©2001 IEEE

Introduction

A linear feedback shift register (LFSR) is a particular model of pseudorandom number generator (PRNG), that is, an algorithm that is able of generating a sequence of pseudorandom binary digits. It consists of a series of cells that are filled with the value of an binary initialisation vector. At every step, the contents of the cells are shifted right one position and the XOR of the values of a subset of the cells is placed in the leftmost cell, producing the rightmost bit. A schematic representation of this procedure can be seen in the image below:

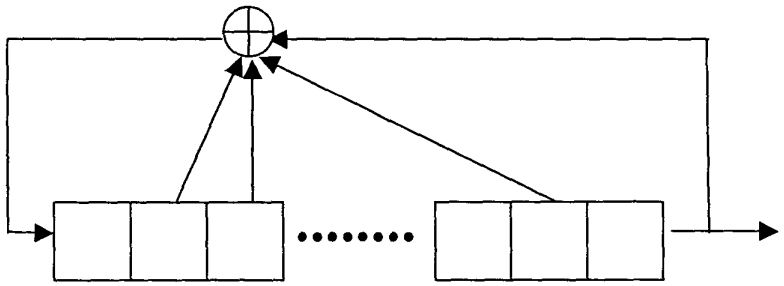

Figure 1: General scheme of a Linear Feedback Shift Register, where the contents of some of the cells are xored to fill the leftmost cell

LFSRs have the interesting property of being easily analysed and represented using algebraic techniques. For example, the subset of cells that generate, by XORing their contents, the new leftmost bit, sometimes called the feedback bit, can be seen as a polynomial over GF(2). So if the feedback bit is obtained by XORing the contents of cell 0, cell 7 and cell 11 we can, alternatively, say that the feedback bit is produced by the polynomial $x^{11}+x^7+1$ over GF(2), etc... so we refer to it as the feedback polynomial. This feedback polynomial P(X), with the number of cells L and its initial value IV, characterize any LFSR.

To get a maximal period LFSR (i.e. an LFSR with period 2^L-1) we need three conditions:

1) The degree of the feedback polynomial equals the number of cells (deg(P(X))=L
2) The feedback polynomial is primitive
3) The initial value at least one cell is not null (not null IV)

Compared with other models of PRNGs, LFSRs are particularly quick and simple, and specially well suited for hardware implementations. Apart from that, the binary sequences that maximal-period LFSRs generate have extremely good random properties. In particular they satisfy all Golomb's randomness postulates:

Let s be a sequence of period N, then the Golomb's postulates require that:

R1) In the cycle s^N of s the number of 1's differs from the number of 0's by, at most, 1.
R2) In the cycle s^N at least half the runs have length 1, at least one-fourth have length 2, at least one-eighth have length 3, etc. Moreover, for each of these lengths there are almost equally many gaps (groups of consecutive 0's) and blocks (groups of consecutive 1's).

R3) The autocorrelation function $C(t)$ is two-valued. That is, for some integer K,

$$N\,C(t) = \sum_{i=0}^{N-1}(2s_i - 1)(2s_{i+t} - 1) = \begin{cases} N \text{ if } t = 0 \\ K \text{ if } 1 \le t \le N-1 \end{cases}$$

Every binary sequence that satisfies Golomb's randomness postulates is called a pseudorandom noise sequence or pn-sequence. So maximal period LFSRs generate pn-sequences.

In spite of these excellent random properties, LFSRs have one strongly undesirable property: they are easily predictable. Using the Berlekamp-Massey [2] algorithm (a powerful mathematical framework as said in [5]) any LFSR with a feedback polynomial of degree L can be predicted with the knowledge of only 2L consecutive output bits.

Despite these bad prediction properties, LFSRs are the basic building block of most stream ciphers proposed in the cryptographic literature. Nowadays state-of-the-art stream ciphers usually combine the output of multiple LFSRs using a highly non-linear function. However, the study of new alternatives for predicting LFSRs remain interesting because they can provide new ways of attacking real-life strong stream ciphers that are highly based in combining LFSRs.

In [1] the authors try to show that some techniques inherited from the field of artificial intelligence can be very useful in cryptanalysis. By approximating the next bit prediction theoretical problem with a classification problem, they justify the use of AI classifiers in cryptanalysis. In particular, they prove that a well-known classifier algorithm called c4.5 is able to predict, without any previous knowledge, the output of two different models of pseudorandom number generators, namely a linear congruential generator (LCG) and an LFSR, after *learning* and extracting rules from their output. Furthermore, they showed how the c4.5 classifier could predict the LFSR with 100% accuracy and how it was capable of finding relationships between the output bits that implied they were generated by a feedback polynomial. So c4.5 was able not only to predict the LFSR but also to discover the pseudorandom number generator model used.

In this work we look for other classifiers, apart from c4.5, that could be useful in predicting LFSRs. In [1] c4.5 was selected and shown to be useful, but with some lack of solid arguments in its favour. In this article we want to compare c4.5 against other classifiers to see if our previous election was correct and, if not, to find better candidates.

For this, we have tested around twelve different classifiers, some of them with changing parameters, at predicting two different LFSRs. This two LFSRs have been chosen to be quite different in degree and weight, for testing how this differences could affect the prediction. The two LFSRs used had degrees 15 and 17, and primitive feedback polynomials $x^{15}+x+1$ and $x^{17}+x^{12}+x^{10}+x^9+x^6+x^4+1$.

The general working procedure was to use these LFSRs output to generate a different number of classification instances (from 250 to 32.000) following the method presented in [1]. Then, we used the 66% of them to train the classifiers, and the rest to test their prediction accuracy. It is important to note than the test set is formed with previously unseen instances, so the classifiers are fronted with completely new instances to test the goodness of the models they have created over the training data.

Results

We have used the Waikato Environment for Knowledge Analysis (WEKA) tool [4] as a common interface for all the classifiers tested. WEKA is free and open, so it would be easy for other researchers to verify or even extend our results. Another advantage of WEKA is that it has been developed in Java, so it is also platform independent. WEKA has also some drawbacks: the tool is far from efficient and have serious difficulties in working with large sets of instances. However, it is nowadays one of the best tools at hand.

Let us see the results obtained by the different classifiers over the output generated by the LFSR with feedback polynomial $x^{15}+x+1$:

Number of instances	ZeroR	Decision Stump	Decision Table X 1 -S 5	Hyperpipes	IB-1	IBk -K 5 -W 0
250	49,3671%	56,9620%	54,4304%	50,6329%	62,0253%	65,8228%
500	53,6585%	58,5366%	59,1463%	46,3415%	71,9512%	75,6098%
1000	47,9042%	52,6946%	100,0000%	52,0958%	70,9581%	77,5449%
2000	49,5549%	46,5875%	100,0000%	50,4451%	78,6350%	87,6855%
4000	50,0000%	48,4490%	100,0000%	50,0000%	83,1610%	95,7164%
8000	50,5895%	50,5895%	47,7892%	49,4105%	87,8408%	98,7472%
16000	50,0736%	48,5278%	47,7181%	49,9264%	89,8785%	100,0000%

Table 1A. Percentage of correctly classified instances by different classifiers over the output of the LFSR with feedback polynomial $x^{15}+x+1$

Number of instances	ID3	J48	Logistic	Prism	Kernel Density	Naive Bayes
250	72,1519%	92,4051%	50,6329%	89,8734%	62,0253%	56,9620%
500	100,0000%	100,0000%	66,4634%	100,0000%	73,7805%	63,4146%
1000	91.3174%	100,0000%	48,8024%	100,0000%	75,7485%	49,7006%
2000	72.1068%	100,0000%	53,2641%	100,0000%	82,6409%	52,2255%
4000	87.7400%	100,0000%	47,2674%	100,0000%	88,7740%	47,3412%
8000	90.5674%	100,0000%	49,9263%	99,2631%	93,5520%	49,5210%
16000	90.5962%	100,0000%	48,3070%	99,7240%	98,6750%	48,1229%

Table 1B. Percentage of correctly classified instances by different classifiers over the output of the LFSR with feedback polynomial $x^{15}+x+1$

As expected, the prediction problem (at least stated in this way) is too difficult for most classifiers. Most classifiers do not achieve results significantly better, not worse, than those that could be expected at random. For these classifiers, increasing the number of instances to learn from does not produce better results. With some of them, a huge increase in the number of instances will be needed, but with others even this will not be enough. Anyway, it is clear that we must reject this classifying algorithms because, in contrast, with exactly the same problem and information, others achieve a perfect 100% accuracy.

We have used the same name notation of WEKA to help other researches follow our results. At first sight, we can conclude that the best classifier of all is c4.5 (WEKA's J48) because it has the better accuracy for every different number of instances. This confirms that the election of c4.5 as a reference classifier was correct, and now it is totally justified. However, our work has also revealed the existence of other classifiers that, although not as good as c4.5, offer very interesting results.

It is important to note that in real life cryptanalysis, any prediction result that is consistently and significantly better than random is enough to reveal a weaknesses and to discard a particular pseudorandom number generator, so perfect accuracy is never the real objective but an additional advantage. In both tables above we see how, when a given classifier performs significantly better than just by chance, increasing the number of classification instances to learn generally increases the accuracy result, to end always with a result very close to a 90 percent, or better.

Now, let us see the results obtained by the different classifiers over the output generated by the LFSR with feedback polynomial $x^{17}+x^{12}+x^{10}+x^9+x^6+x^4+1$:

Number of instances	ZeroR	OneR	DecisionStump	DecisionTable X 1 -S 5	Hyperpipes
250	46,753200%	51,948100%	44,155800%	48,051900%	53,246800%
500	48,765400%	46,913600%	46,913600%	55,555600%	48,765400%
1000	46,988000%	52,108400%	52,108400%	92,771100%	46,988000%
2000	50,446400%	47,470200%	47,470200%	55,357100%	50,446400%
4000	49,704100%	47,707100%	47,707100%	52,071000%	50,295900%
8000	49,262500%	51,474900%	51,474900%	49,778800%	49,262500%
16000	49,742300%	50,607500%	50,607500%	50,699600%	49,742300%
32000	49,429700%	49,466500%	49,420500%	48,749100%	49,429700%

Table 2A. Percentage of correctly classified instances by different classifiers over the output of the LFSR with feedback polynomial $x^{17}+x^{12}+x^{10}+x^9+x^6+x^4+1$

Number of instances	IB-1	IBk -K 1 -W 0	ID3	J48.J48	J48.PART
250	54.545500%	55.844200%	45.454500%	42.857100%	51.948100%
500	57.407400%	60.493800%	56.790100%	50.617300%	51.851900%
1000	51.506000%	56.325300%	56.927700%	55.722900%	57.530100%
2000	50.892900%	53.273800%	49.702400%	49.553600%	51.636900%
4000	52.218900%	53.032500%	49.408300%	48.520700%	69.970400%
8000	53.539800%	51.991200%	50.811200%	53.539800%	85.472000%
16000	53.994800%	54.363000%	58.983800%	60.898400%	96.833600%
32000	53.449200%	54.865700%	59.722200%	89.597100%	97.691300%

Table 2B. Percentage of correctly classified instances by different classifiers over the output of the LFSR with feedback polynomial $x^{17}+x^{12}+x^{10}+x^9+x^6+x^4+1$

Number of instances	Logistic	Prism	KernelDensity	Naive BayesSimple	Voted Perceptron
250	45.454500%	48.051900%	55.844200%	42.857100%	48.051900%
500	47.530900%	46.296300%	60.493800%	48.765400%	46.296300%
1000	48.795200%	46.686700%	55.722900%	48.192800%	53.012000%
2000	45.684500%	45.238100%	52.083300%	45.982100%	49.256000%
4000	49.778100%	46.671600%	53.402400%	50.739600%	51.701200%
8000	48.893800%	44.063400%	52.507400%	49.041300%	48.820100%
16000	49.208400%	76.343900%	55.081000%	49.392500%	49.981600%
32000	49.061800%	90.406500%	55.831500%	49.209000%	49.411300%

Table 2C. Percentage of correctly classified instances by different classifiers over the output of the LFSR with feedback polynomial $x^{17}+x^{12}+x^{10}+x^9+x^6+x^4+1$

The above tables confirm the conclusions we extracted from the LFSR of the last example. They clearly show that c4.5-based classifiers (WEKA implements two variants, J48.J48 and

J48.PART) achieve better than the rest. We have included more classifiers in our test than in the example above, but none of the new classifiers performs significantly better than chance.

Only the classifiers IB-1, IB1, ID3, the two J48 variants, Prism and Kernel Density produce results that are consistently and significantly better that those that could be expected at random. Even in these cases, the results are significantly poorer than those of the last example. This is due more to the increase in the weight (7 instead of 3, more than the double) of the primitive polynomial than to the slight increase (17 for 15) in its degree. The best results in the tables above are worse than those of tables 1A and 1B, even though we have used twice the number (32000 instead of 16000) of learning examples. This shows that the problem of predicting this LFSR is clearly harder.

To which extend the increase in the weight of the polynomial must be followed with an increase of the number of training examples to get exactly the same prediction accuracy is a very interesting point, but completely out of the scope of the present article. It is sufficient, for our purpose, to know that for this kind of prediction methodology based in AI classifiers, an increase in the weight of the primitive polynomial makes the problem much harder than the same increase in the degree of the feedback polynomial. Anyway, most of the feedback polynomials used in LFSRs nowadays are sparse.

Additional c4.5 advantages

We have completely justified the use of the c4.5 classifier in [1], showing that it produces the best results when compared against other algorithms. Only the Prism classifier achieves comparable results, but has some drawbacks that make it less recommendable than the c4.5 algorithm. Between then, perhaps the most important is that it does not guarantee that all test instances will be classified, something completely unacceptable in our particular application. When working with so many instances, efficiency starts playing an important role, and again, the c4.5 algorithm is slightly more efficient in time and memory than Prism.

Another advantage of the c4.5 algorithm is its tree pruning phase, in which it prunes long subtrees that do not increase the prediction accuracy significantly. As a result of this phase, simpler and usually better models arise, as seen in the next Figure:

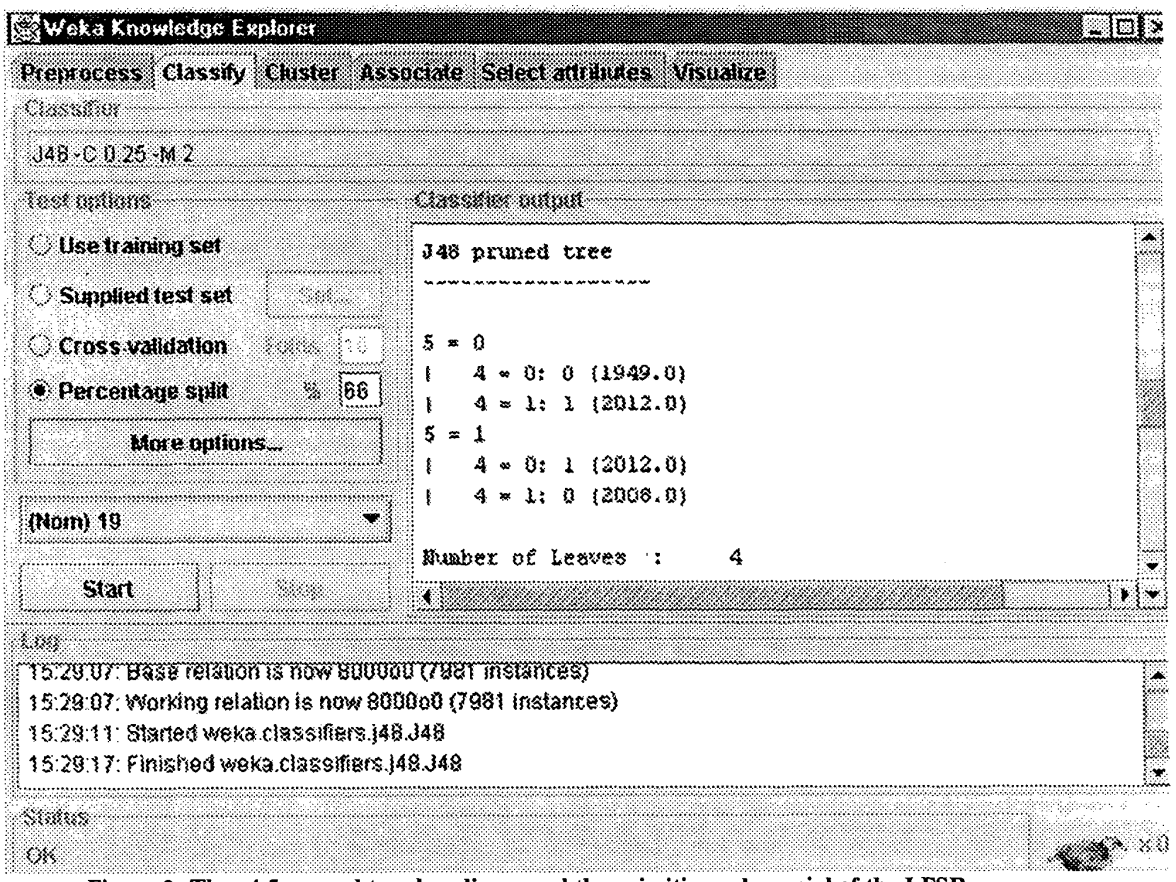

Figure 2: The c4.5 pruned tree has discovered the primitive polynomial of the LFSR

This is one of the greatest advantages of c4.5: when the training data is enough, it also discovers the relation between the output bits induced by the feedback polynomial. In the example above, as shown in Figure 2, we can see the pruned tree:

```
5 = 0
|  4 = 0: 0 (1949.0)
|  4 = 1: 1 (2012.0)
5 = 1
|  4 = 0: 1 (2012.0)
|  4 = 1: 0 (2008.0)
```

that can be interpreted as

```
if the bit in the position 5 is 0
        then
                if the bit in the position 4 is 0 then class is 0
                else class is 1
        else
                if the bit in the position 4 is 0 then class is 1
                else class is 0
```

This rules are equivalent to

class = bit4 XOR bit 5

In the training and test data, that had a framelength value of 20 (from bit0 to bit19), the class was represented by bit 19. So we have

bit 19 = bit 4 XOR bit 5

or, alternatively,

$x^{19} = x^4 \oplus x^5$

And, adding x^{19} to both terms of the equation we get

$x^{19} \oplus x^{19} = x^{19} \oplus x^5 \oplus x^4$

that is equivalent to

$x^{19} \oplus x^5 \oplus x^4 = 0$

and dividing over x^4 we get

$x^{15} \oplus x \oplus 1 = 0$

Which is exactly the feedback polynomial of the LFSR.

So c4.5, apart from being able to predict with perfect accuracy the output of this pseudorandom number generator, did also discovered the model used by finding its feedback polynomial.

In the case of the second LFSR, due to its more than double weight, c4.5 needed more instances to learn but achieved a 97.69% accuracy with 32000 instances. Increasing the number of instances would also slightly increase the accuracy to reach a 100%, and would make c4.5 discover the feedback polynomial.

Conclusions

It is clear that this work cannot pretend to be definitive or exhaustive in the matter. There are many other classifiers that had not been tested, and many additional possibilities (combining multiple classifiers, bagging, boosting, etc.) that could also achieve good results. However, we feel we have provided enough evidence, after testing more than 15 different classifiers, to prove that the c4.5 algorithm is probably the best option for the cryptanalytic problem of predicting an LFSR.

As a result of this search for other good classifiers, we have also found a bunch of other algorithms that, in this particular problem, perform consistently and significantly better that chance. This is exactly what a cryptanalyst could dream of: the exact accuracy obtained, being a 60% or a 95% is not so important as far as a significant statistical deviation from random behaviour proves the existence of a weaknesses.

LFSRs, the pseudorandom number generator model used in this work, are very simple generators. Furthermore, there are much more efficient algorithms for predicting them like the Berlekamp-Massey algorithm.

We feel, anyway, that our new prediction approach is interesting at least for two reasons: Firstly, the particular model of generator used is supposed by the Berlekamp-Massey algorithm but not by our classifier-based model. This implies that the Berlekamp-Massey algorithm could not predict any other non-LFSR based model. On the other hand, as our classifier-based model does not suppose (and does not take advantage of it) any underlying generator model, it could be used to predict different types of non-LFSR based generators.

In this way, we are approaching the ideal concept of a general next bit predictor that, starting with zero knowledge can *learn* from different instances of a pseudorandom number generator to finally be able to predict it. Finally, this basic approach, when extended and improved, will offer a new way for trying to break modern state-of-the-art stream ciphers that are based in the combination of various LFSRs.

References

[1] Using the general next bit predictor as an evaluation criteria
First New European Schemes for Signatures, Integrity and Encryption (NESSIE) Workshop
KU Leuven, Belgium. October 2000
J.C.Hernández, J.M.Sierra, C.Mex-Perera, D.Borrajo, A.Ribagorda, P.Isasi

[2] Shift-register synthesis and BCH decoding
J.L. Massey
IEEE Transactions of Information Theory, vol. 15, pp. 122-127. 1969

[3] Handbook of applied Cryptography
Alfred J. Menezes, Paul C. Van Oorscht, Scott A. Vanstone
CRC Press, 1997

[4] Data mining: Practical Machine Learning Tools and Techniques with Java Implementations
Ian H. Witten, Eibe Frank
Morgan Kaufmann Publishers, October 1999

[5] Cryptology Frequently Asked Questions
Item 2.1.5.1: What is a Linear Feedback Shift Register?
RSA Laboratories 2001

Bibliography

Shift Register Sequences
S.W. Golomb
Holden-Day, San Francisco, 1967

Analysis and design of stream ciphers

R.A. Rueppel
Springer-Verlag, Berlin 1986

Stream Ciphers
M.J.B. Robshaw
Technical Report TR-701
RSA Laboratories, 1995

Linear recurrence relations over finite fields
E.S. Selmer
Department of Mathematics
University of Bergen, Norway. 1996