Classificatory Prediction and Primitive Polynomial Construction of Linear Feedback Shift Registers using Decision Tree Approach

Shehroz S. Khan
America Online India Pvt. Limited,
RMZ Titanium, 3rd Floor,
135, Airport Road,
Bangalore-560017,
Karnataka, India
shehroz@aol.com

Abstract

Prediction of next bit of pseudo-random sequence is a critical issue in cryptography. If a pseudo-random sequence can be predicted then cipher / code is considered to be broken. In this paper we envisage data mining perspective to explore the relationships and regularities between the bits of pseudo-random sequence that paves the way for the prediction of subsequent next bits. We present the use of decision tree algorithm (C4.5) to predict the next bit of pseudo-random sequences generated by Linear Feedback Shift Registers (LFSR). We also successfully constructed the exact primitive polynomials that generate maximal length pseudo-random sequences of LFSRs using the association rules produced from our classificatory prediction approach. This technique is independent of the parameters and domain (cryptographic) used by the pseudo-random generator.

1 Introduction

Random number generation forms the core of information security in any cryptographic application. Unfortunately, there exists no technique to generate true random numbers, especially on computers that are typically designed to be deterministic. The best a computer can produce are pseudo-random numbers, which are generated from some random initial values. The pseudo-random numbers looks like random numbers and have good statistical properties [Knuth, 1997]. The period of pseudo-random sequence should be very high and should not repeat for a large length. A pseudo-random bit generator (PRBG) is a deterministic algorithm which given a truly-random binary sequence of length n, produces a binary sequence of length

l(n) that appears to be random, with *l( )* being a polynomial. The input to the PRBG is called the seed, and the output is called a pseudo-random bit sequence. Normally in cryptographic applications, random number is a number that cannot be predicted by an eavesdropper before it is generated. Cryptographically secure pseudo-random sequence must be unpredictable. It must be computationally infeasible to predict what the next random bit will be, given complete knowledge of the algorithm or hardware generating the sequence and all of the previous bits in the stream [Schneier, 1996]. Typically, if the pseudo-random numbers are to be in the range $[0\ldots n-1]$, an adversary cannot predict that number with probability slightly better than $1/n$. A more formal definition is given by Blum et al [Blum et al, 1984]. Let $g:\{0,1\}^n \to \{0,1\}^{l(n)}$ be an efficient (computable in polynomial time) function, with *l(n)* being a polynomial with *l(n)>n*. Let *X* and *I* be random variables uniformly distributed respectively on $\{0,1\}^n$ and on $\{1,\ldots,l(n)\}$. Then *g* is a next bit unpredictable PRBG, if for all adversaries *A* running in polynomial time the success probability (prediction probability) of *A* for *g* is

$$P\left[A\left(I, g(X)_{\{1,\ldots,I-1\}}\right) = g(X)_I\right] < \frac{1}{p(n)} \qquad \forall p \text{ , where } p \text{ is a polynomial.}$$

This paper deals with prediction of pseudo-random bits using data mining perspective. As mentioned in Friedman [Friedman et al, 1997], data mining is at best a vaguely defined field; its definition depends largely on the background and views of the definer. The view of Fayyad [Fayyad, 1997] is that any algorithm that enumerates patterns from data, or fits models to data, is data mining. Fayyad further viewed data mining to be a single step in a larger process of knowledge discovery in databases (KDD). KDD is considered to be a more encompassing process that includes data warehousing, target data selection, data cleaning, pre-processing, transformation and reduction, data mining, model selection, evaluation and interpretation, and finally consolidation and use of the extracted "knowledge". Data mining has applications in various fields like information retrieval [Can et al, 1987], finance [Kovalerchuck et al, 2000], bioinformatics [Eisen et al, 1998] etc.

The paper is organized is follows. Section 2 presents an introduction to data mining and C4.5 decision tree algorithm. We present the technique of *classificatory prediction* in section 3 along with the methodology we employed for the same. Classificatory prediction for Linear Feedback Shift Register is detailed in section 4. We conclude our presentation in section 5.

2 Data mining: Introduction to C4.5

As discussed above data mining aims at finding patterns from the data set. Our proposed approach belongs to inductive paradigm, because we try to generalize a concept (to predict next bit; 0 or 1) from set of examples and counter examples of the concept. The idea is to generate concept description that can predict the value of the class for all the previously seen instances.

Most of the data mining algorithms build decision trees, association rules as a result of their classification process. There are variety of methods, tools and software that work in this way, and they differ in the manner they construct these rules. For example, they might try to build

production rules in the form of nested "if …, then…" statements. There exist many inductive algorithms like CART [Breiman, 1984], ID3 [Quinlan,, 1986], C4.5 [Quinlan, 1993], SLIQ [Agrawal,1996]. We have chosen C4.5 algorithm as decision tree technique for classificatory prediction of pseudo-random sequences generated by LFSRs. C4.5 induces classification rules from training sets to form decision trees. The decision tree is defined as a tree in which each node is an attribute, each arc from this node is a possible value for that attribute, and each leaf is the expected value for the category of the pattern obtained following all the path from the root of the tree to that leaf. The general idea to construct a decision tree is to decide at each node which of the non-used attributes is most informative for the classification of all the patterns represented by the path from the root to that node. Applying this idea, recursively, for every node generates the decision tree. C4.5 uses the concept of gain ratio to make a tree of classificatory decisions with respect to a previously chosen target classification. The information gain can be described as the effective decrease in entropy resulting from making a choice as to which attribute to use and at what level. By considering which of the attributes is best for discriminating among cases at a particular node in the tree, we can build up a tree of decisions that allows us to navigate from the root of the tree to a leaf node by continually examining attributes.

C4.5 is an extension of the basic ID3 algorithm designed by Quinlan to address many issues not dealt with by ID3 like avoiding over fitting of the data, determining how deeply to grow a decision tree, reduced error pruning, post-pruning after induction of trees to increase accuracy, handling continuous attributes e.g. temperature, handling training data with missing attribute values etc. C4.5 is a widely used decision tree based classification algorithm with applications in various scientific areas. The C4.5 package with full source code is freely available on the Internet [Web site]. The decision trees generated by C4.5 algorithm can be easily analyzed and interpreted. The package allows the decision tree to be simplified, using pruning techniques, which reduces the size of the tree according to a user-defined level.

2.1 C4.5 package

We downloaded the C4.5 package Release 8 for Unix/Linux version [Website]. This package consists of four programs viz. ***C4.5, C4.5rules, Consult*** and ***Consultr***. We used the ***C4.5*** program to generate decision trees from a set of examples. We also used ***C4.5rules*** program to generate association rules from the decision trees generated by *C4.5* program.

3 Classificatory Prediction

As presented by Stinson [Stinson, 1995] the next bit predictor is an algorithm, when given all previous bits generated from a pseudo-random generator (PRBG), it can efficiently predict the next bit with higher than chance probability. Suppose we have a pseudo-random sequence of bits $p_1, p_2, \ldots p_n$ generated by a PRBG, then a next bit predictor should compute the p_{n+1}^{th} bit given the previous ones, with probability greater than $1/2$ without knowing the particular set of parameters used by the PRBG.

We concentrate our efforts to find patterns and regularities in the binary pseudo-random sequences using the data mining perspective. Here, the advantage lies in the fact that no a priori cryptographic domain knowledge is required to predict the next bits. We translate the technique of next bit prediction as a classification problem (*classificatory prediction*), to adjudge prediction accuracy with the bits to be predicted in hand. We explain the *classificatory prediction* problem and the methodology adopted as follows.

Methodology adopted: -

Suppose we have n bits generated from a PRBG i.e. $p_1, p_2, \ldots p_n$. Choose a suitable block size (b) as a training pattern (P_i) associated with a class label ($CL \rightarrow 0$ or 1) such that

$$P_1 = p_1, p_2, p_3 \ldots p_b \qquad CL \rightarrow p_{b+1}$$

$$P_2 = p_2, p_3, p_4 \ldots p_{b+1} \qquad CL \rightarrow p_{b+2}$$

$$\vdots$$

$$P_{n-b} = p_{n-b}, p_{n-b+1}, \ldots p_{n-1} \qquad CL \rightarrow p_n$$

The n-b patterns from P_1 to P_{n-b} serves as the pattern space for the classification model we have adopted. Out of these n-b patterns an appropriate number of patterns (α) are used for learning (training the C4.5 network) and the remaining patterns are used to predict (n-b-α) bits of the pseudo-random sequence. The learning process is dependent on a number of prefixed block sizes so as to accommodate maximum possible regularities, patterns and bit combinations of the pseudo-random sequence.

To use C4.5 package as a next bit predictor for classification problem, we supply α patterns as training data set, and n-b-α patterns as test data for the prediction of n-b-α bits. Intuitively, the size of the training data set should be sufficient to capture maximum regularities and extract generalized conclusions that yield high degree of prediction. The algorithmic steps followed are presented below: -

Algorithm: ***Classificatory_Prediction(p_i, n)***

Input: p_i – the binary pseudo-random sequence, n - the length of the pseudo-random sequence

Output: Classificatory prediction results on test data

1. Select a suitable block size (b)
2. Generate n-b blocks and associate every pattern with their class label, i.e. next bit (described in section 3)

$$P_1 = p_1, p_2, p_3 \ldots p_b \qquad CL \rightarrow p_{b+1}$$

$$P_2 = p_2, p_3, p_4 \ldots p_{b+1} \qquad CL \rightarrow p_{b+2}$$

$$\vdots$$

$$P_{n-b} = p_{n-b}, p_{n-b+1}, \ldots p_{n-1} \qquad CL \rightarrow p_n$$

3. Select α number of patterns for training
4. Select remaining n-b-α number of patterns as test data
5. Run the *C4.5* program to generate decision trees
6. Run the *C4.5rules* program to generate the association rules from the decision trees generated by *C4.5* program.
7. Choose the simplified simplified decision trees along with association rule set for further analysis

In the succeeding section we show the use of our proposed methodology for classificatory prediction of next bit of pseudo-random sequences generated by various degree of LFSRs. We also present an empirical formulation of the construction of primitive polynomials that defines maximal period LFSR.

4 Classificatory Prediction of Linear Feedback Shift Register

The simplest kind of feedback shift register is the Linear Feedback Shift Register (LFSR) [Schneier, 1996]. The feedback function is simply the XOR of certain bits (taps) in the register; the list of these bits is called a tap sequence. The degree of the polynomial is the length of the shift register (say l). The period of a LFSR is the length of the output sequence before it starts repeating. In order for an LFSR to have maximal period the polynomial formed from a tap sequence plus the constant 1 must be a primitive polynomial mod 2. A primitive polynomial of degree m is an irreducible polynomial that divides $x^{2^{m-1}}+1$, but not x^t+1 for any t that divides 2^m-1 [Golomb, 1982]. Cryptographers like to analyze the sequences generated from LFSRs to convince themselves that they are random enough to be secure. Barlekamp-Massey algorithm [Massey, 1969] can determine an LFSR of length l after examining $2l$ bits of the key stream. Once LFSR is determined (or the primitive polynomial is constructed), the cipher is considered to be broken.

Hernandez et al [Hernandez et al, 2000] reported a General next bit Predictor (GNBP) for predicting next bit for LFSR by converting the next bit predictor theoretical model into a classification problem using C4.5 as inductive algorithm. They showed the utility of their process by considering one particular primitive polynomial $\left(x^{15}+x^1+1\right)$ and predicting the subsequent next bits. The claims made in their research work were not comprehensive. The interpretation of their results cannot be generalized. They do not comment about the block size required to predict correctly. We analyze the same problem in a more comprehensive manner. We considered various primitive polynomials ranging from degree 10 to 17 to carry out our analysis. We looked into the problem of classificatory prediction in two-dimensional manner as

a) To check the minimum block size required to learn correctly from the pattern space
b) To check how many bits are needed to learn from the pattern space such that the prediction is maximally correct

We took various primitive polynomials (table 1) to start the process. As discussed in section 3, we generated *n-b* patterns for each of these polynomials respectively, where *n* is the period of the LFSR of degree *d* i.e. $n = 2^d - 1$ and *b* is the arbitrarily chosen block size. We carried out the experiment to check the minimum size of block required to learn correctly using C4.5 algorithm. At first, we took 99% of the pattern space for training and the remaining 1% for testing the classificatory prediction accuracy. It has been found experimentally that for a LFSR of degree *d*, the minimum block size required is *d*. Hernandez et al claimed that larger the block size, the better the prediction. They presented a value of block length equal to $10 * \log(n)$ to distinguish an unpredictable source from a predictable one. We agree to the first claim, but they did not justify this numerically. Experimentally we found a lower bound for the block size for maximum prediction. The results are summarized in table 1.

It can be seen from table 1 that if the block size for a primitive polynomial is less than its degree then the prediction accuracy is equivalent to chance probability. Hence from this point onwards it is established that *for a primitive polynomial of degree d, the minimum block size will be d for accurate classificatory prediction*. If we further increase the block size, there will be no significant change in the formation of decision trees and association rules as generated by C4.5 algorithm.

Degree of polynomial (*d*)	**Polynomial chosen**	**Block size (*b*)**	**Classificatory Prediction error (%age)**	**Block size (*b*)**	**Classificatory Prediction error (%age)**	**Block size (*b*)**	**Classificatory Prediction error (%age)**
10	$x^{10} + x^3 + 1$	9	36.4	10	0	11	0
11	$x^{11} + x^2 + 1$	10	52.4	11	0	12	0
12	$x^{12} + x^6 + x^4 + x + 1$	11	56.1	12	0	13	0
13	$x^{13} + x^4 + x^3 + x + 1$	12	50.0	13	0	14	0
14	$x^{14} + x^5 + x^3 + x + 1$	13	56.1	14	0	15	0
15	$x^{15} + x + 1$	14	50.0	15	0	16	0
16	$x^{16} + x^5 + x^3 + x^2 + 1$	15	51.5	16	0	17	0
17(a)	$x^{17} + x^3 + 1$	16	50.5	17	0	18	0
17(b)	$x^{17} + x^5 + 1$	16	54.4	17	0	18	0
17(c)	$x^{17} + x^6 + 1$	16	56.6	17	0	18	0

Table 1

The next important point to consider is to investigate that how many bits are required to learn from the pseudo-random output of LFSR to get correct prediction. To check these results we performed a comprehensive analysis for primitive polynomials ranging from degree 10 to 17. We fix the block size as equal to *d*. We generated 25 different pseudorandom sequences corresponding to every primitive polynomial by varying its initial settings. We take only the upper bounds results into account. Hernandez et al claimed that experimentally they found that training patterns needed to predict accurately is close to 1% of period of generator. Their analysis is based on a particular LFSR of degree 15; hence limitations exist in their claim.

We define an index *Bit Prediction Ratio (BPR)*, to estimate the ratio of minimum bits (*M*) required for correct classificatory prediction to the period of primitive polynomial $\left(2^d - 1\right)$ of the LFSR. Mathematically, $BPR = \frac{M}{\left(2^d - 1\right)} \times 100$

Experimentally, we found that *BPR* value for the under considered primitive polynomials varies from 0.13% to 14.23%. This variation in the values of ***BPR*** is due to the number of taps in LFSRs. *The higher the number of taps the more the number of bits required for correct classificatory prediction*. We can also infer that $x+b-1$ bits are required to predict the next $n-x-b+1$ bits correctly. Table 2 summarizes this result.

Degree of primitive polynomial (*d*)	**Training Patterns**	**Minimum bits needed, *M***	***BPR***
10	70	80	7.82
11	61	72	3.51
12	571	583	14.23
13	899	912	11.13
14	1473	1487	9.07
15	81	96	0.29
16	2558	2574	3.92
17(a)	159	176	0.13
17(b)	319	336	0.25
17(c)	319	336	0.25

Table 2

4.1 Primitive polynomial construction of LFSR

C4.5 makes classification decision on the basis of the decision trees and association rules generated from the learning data. We observe that the primitive polynomial can be determined from the pruned trees but not the way Hernandez et al showed. We observed experimentally that all the production rules lead to primitive polynomial generation in a different fashion. The empirical formulation of construction of primitive polynomial for any n degree LFSR can be summarized as under.

($a, b, c, d, \ldots$) - Significant bits / attributes from association rules

(n-a, n-b, n-c, n-$d, \ldots$) - Subtract from length of LFSR

(n-a+1, n-b+1, n-c+1, n-d+1, ...) - Add 1

Hence, $x^{n-a+1} + x^{n-b+1} + x^{n-c+1} + x^{n-d+1} + \ldots + 1$, is the required primitive polynomial.

The above rules are generated when block size was equal to d. It the block size exceeds d, the rules generated are equivalent, we only have to subtract the number of exceeding bits from the block size d to get the required primitive polynomial. Hence, *if significant bits are known then the exact primitive polynomial can be constructed*. The rule sets that we discovered for different polynomial are depicted in the form of significant attributes in table 3

As an example let us consider the primitive polynomial of degree 10 ($x^{10} + x^3 + 1$) as presented in table 1. The simplified association rules generated from decision trees after pruning are

If bit at position 1 is 0 and bit at position 8 is 1 then class label is 1

If bit at position 1 is 1 and bit at position 8 is 0 then class label is 1

If bit at position 1 is 1 and bit at position 8 is 1 then class label is 0

If bit at position 1 is 0 and bit at position 8 is 0 then class label is 0

C4.5rules program identifies bit 1 and 8 as significant attributes or bits. Here, we can infer that class label = bit 1 XOR bit8

Now consider bit 1 and 8 as the significant bits obtained from the association rules generated from decision trees. Since the rules are generated for a 10-degree LFSR, therefore subtract them from 10 and add 1 to them to get the desired primitive polynomial. The steps can be summarized as

(*1, 8*) - Significant bits from association rules

(*9, 2*) - Subtract from 10 (length of LFSR)

(*10, 3*) - Add 1

Hence, $x^{10} + x^3 + 1$ is the required polynomial.

We conclude from the above experiments and discussions that if we analyze of the results presented in table 1, 2 and 3 we can know a lot about the pseudo random sequence generated from

an LFSR. *We can estimate the minimum number of bits and determine the block size* (i.e. *length of LFSR*) *for correct classificatory prediction. We can also construct the primitive polynomial that generates the maximal period LFSR.*

Degree of primitive polynomial (*d*)	**Rules generated**
10	1, 8
11	1, 10
12	1, 7, 9, 12
13	1, 10, 11, 13
14	1, 10, 12, 14
15	1, 15
16	1, 12, 14, 15
17(a)	1, 15
17(b)	1, 13
17(c)	1, 12

Table 3

5. Conclusion

We propose an alternative approach to predict the next bit of pseudo-random binary sequences generated by LFSR based pseudo-random generators using C4.5 as decision tree algorithm. We show the utility of our approach for prediction of next bit of pseudo-random sequences generated by LFSRs by transforming the next bit prediction problem as classificatory prediction problem. We were successful in determining the primitive polynomial that defines the maximal period LFSR using the association rules generated from decision trees produced during classificatory prediction. Although, there exist techniques to predict and crypt analyze the above considered LFSR based PRBG's but they are domain specific. The advantage of our proposed approach is that it is domain independent and does not rely on the type of generator used and the parameters used by PRBG. We achieved high classificatory prediction accuracy for LFSRs, which proves that relationship and regularities exists between the pseudo-random bits. And therefore, LFSR based pseudo-random sequences should not hold priority in the wish list of a cryptographer. We would like to extend this work for other type of PRBG's and study the association rules thus produced. We hope to use this approach to check the classificatory prediction accuracy of other cryptographically secure pseudo-random number generators. We may achieve less prediction accuracy, in some cases, but sledging in this direction we will reduce the complexity domain of a cryptanalyst so that he has a reduced set of bits that can be subjected to a brute force attack in real time. Off course, this is not a simple task but if we can find some regularity

and patterns in cryptographically secure pseudo-random generators then it will definitely be conducive aid to the cryptanalyst.

References

[Agrawal, 1996] Agrawal, R., Arning, A., Bollinger, T., Mehta, M., Shafer, J., Srikant, R., The Quest Data mining System, *Proc. of 2nd International Conference on Knowledge Discovery in Databases and Data mining*, Portland, Oregon, August, 1996

[Blum et al , 1984] Blum, M., and Micali, S., How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Computing*. 13(4), 1984

[Breiman, 1984] Breiman, Friedman, J., Olshen, R., Stone, C., *Classification and Regression Trees*, 1984

[Can et al, 1987] Can, F., Ozkarahan, E., A Dynamical Cluster Maintenance System for Information Retrieval, *In Proceedings of 10th Annual International ACM SIGIR Conference*, 1987

[Eisen et al, 1998]Eisen,M., Spellman, P., Brown, P., Bostein, D., Cluster analysis and display of genome-wide expression patterns, in Proc. Nat. Acad. Sci. USA, vol. 95, 14863-14868, 1998

[Fayyad, 1997] Fayyad, U. M. "Editorial." Data Mining and Knowledge Discovery 1: 5-10, 1997

[Friedman, 1997] Friedman, J. H. Data Mining and Statistics: What's the Connection? *Proceedings of Computer Science and Statistics: the 29th Symposium on the Interface*, 1997

[Golomb, 1982] Golomb, S.W., Shift Register Sequences, San Francisco, Holden-Day, 1967, Reprinted by Aegean Park Press, 1982

[Hernandez et al, 2000] Hernandez, J.C., Sierra, J.M., Mex-Perera, C., Borrajo, D., Ribagorda, A., Isasi, P.,Using the general next bit predictor like an evaluation criteria, *In proceedings of NESSIE workshop*, Leuven, Belgium, 2000

[Key, 1976] Key, E.L., An Analysis of the Structure and Complexity of Nonlinear Binary Sequence generators, *IEEE Transactions on Information Theory*, v. IT-22, n. 6, 1976

[Knuth, 1997] Knuth, D.E.,, Semi numerical Algorithms, volume 2 of *The Art of Computer Programming*, Addison-Wesley, third edition , Reading, MA, USA, 1997

[Kovalerchuck et al, 2000]Kovalerchuck, B., Vityaev, E., *Datamining in Finance: Advances in Relational and Hybrid Methods*, Kluwer Publications, 2000

[Massey, 1969] Massey, J.L., Shift Register synthesis and BCH Decoding, *IEEE Transactions on Information Theory*, v. IT-15, n. 1, pages 122-127, 1969

[Quinlan, 1986] Quinlan, J, R., Induction of Decision Trees, Machine Learning Journal, 1986

[Quinlan, 1993] Quinlan, J, R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1993

[Schneier, 1996] Schneier, B., *Applied Cryptography*, second edition, *Protocols, Algorithms and Source Code in C*, John Wiley & Sons, Inc, 1996

[Stinson, 1995] Stinson, D, R., *Cryptography, Theory and Practice*, CRC Press, 1995

[Website] http://www2.cs.uregina.ca/~hamilton/courses/831/notes/ml/dtrees/c4.5/c4.5r8.tar.gz

Acknowledgements

The author is highly indebted to Amir Ahmad for resolving many algorithmic and implementation issues, critically reviewing this work and for his continuous support.