Cryptanalysis of the Nonlinear FeedForward Generator

S.S. Bedi and N. Rajesh Pillai

Scientific Analysis Group, DRDO, Delhi 110054

Abstract. The nonlinear feedforward generator is one of the commonly used building blocks of stream ciphers. This paper describes a novel known-plaintext attack for cryptanalyzing nonlinear feedforward generator. The plaintext requirement of the attack is only twice the length of the shift register. The implementation of this attack could identify the initial settings of the system for a 128 stage register and randomly chosen nonlinear feedforward function of 10 variables in few minutes on a P-II 300 MHz machine.

1 Introduction

The nonlinear feedforward generator is one of the commonly used building blocks of stream ciphers. This paper describes a new technique for cryptanalyzing feedforward generator. Given just 2n bits of the plain-text, where n is the length of shift register, the attack determines the initial setting of the shift register. The basic idea is to form a system of Boolean equations describing relationship between the keysequence (obtained by xoring crypt and the known plain text), and the initial settings. We then use the techniques developed by Zakrevskij & Vasilkova [8] for solving a system of nonlinear equations.

The proposed attack is very efficient and in most of the cases we got results within few minutes. We believe that this attack can be extended to attack other building blocks of stream ciphers also.

The Nonlinear feedforward generator is made up of a linear feedback shift register (LFSR) and a nonlinear Boolean function. The shift register is allowed to run (with its feedback polynomial deciding the bit to be shifted in). The nonlinear feedforward function (NLFF) takes some of the bits of the LFSR as input and calculates the output bit. The location in LFSR from where the input bit for NLFF is picked is called a tappoint or tap for short. Figure.1 shows a block diagram of a Nonlinear feedforward generator.

It is known (corollary 5.6 of [4]) that the lower bound for Linear complexity (LC) of NLFF Generator can be expressed in terms of L, the length of shift register, k, the number of taps, under suitable conditions as $LC \geq {}^{n}C_{k}$.

For L = 50, k = 8, we have $LC \geq {}^{50}C_{8} = 536878650 \approx 5.4 * 10^{8}$

In this paper we will be dealing with *feedforward functions with equidistant taps.*

C. Pandu Rangan, C. Ding (Eds.): INDOCRYPT 2001, LNCS 2247, pp. 188–194, 2001.

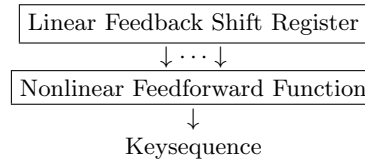

Fig. 1. Nonlinear Feedforward Generator

2 Attacks on Nonlinear FeedForward Generator

We summarize the few existing known results in this area. Correlation attacks are the most common class of attacks applied on stream ciphers. See [5] for overview of these type of attacks. Correlation attacks work when there is correlation between output sequence and input sequence to the nonlinear function. Anderson [1] describes how to pick up subsequences with optimum correlation properties for mounting the correlation attack on feedforward generators for known plaintexts. Correlation attacks work for ciphertext-only attack also. They need ciphertext length dependent on the value of correlation. Typical ciphertext length required would vary from 4000 bits (very optimistic case) to order of 10^6.

Among the known-plaintext attacks, the generalized inversion attack [2], [3] is very effective in cryptanalyzing nonlinear filter generators. It has a complexity which is exponential in the distance(d) between first and last tap points of the feedforward function. The generalized inversion attack will be practical if the feedforward function has its tap points bunched together or if it can be converted to this form (say by uniform decimation) This attack is based on theory of branching processes.

Our attack uses a totally different approach to this problem. We use the description of the system to setup nonlinear equations capturing relations between initial settings (unknown) and the bits of the output sequence (known part). Then we solve the system for the unknown initial variables.

3 Solving Nonlinear Boolean Equations

Solving an arbitrary set of Boolean equations (also known as the satisfiability problem) is an NP-Complete problem. Currently only exponential time algorithms are available for the general case. But if the system of equations is of a special kind, solutions can be obtained efficiently. We have used the technique of local reduction developed by Zakrevskij and Vasilkova [8] in our work. Local Reduction technique can solve large systems of nonlinear boolean equations efficiently when number of variables in each equation is small, (typically less than 10) and there are equations having large overlaps in their sets of variables.

3.1 Local Reduction Technique of Solving System of Nonlinear Equations

Given a system of nonlinear Boolean equations of the form $F_i = 1$, the local reduction technique takes two equations at a time and reduces the number of points in the solution space to be tried out.

Representation of the equations: For each equation, system will store the variables in the equation and the onset. (Onset of an equation is the set of points at which the equation is satisfied.) For e.g. $xy \oplus yz = 1$ will be represented as the set of (binary) numbers $\{110, 011\}$ over the variables (x, y, z). This means that the equation is satisfied when $x = 1, y = 1, z = 0$ and when $x = 0, y = 1, z = 1$ and at no other points.

Given two equations, whose variable sets have a nonempty intersection, say

$$Eq1 : varlist_1 = (a, b, c, d, e), onset_1 = \{01101, 11010, 10011\}$$
$$Eq2 : varlist_2 = (c, d, e, f, g, h), onset_2 = \{101110, 001101, 010010\}$$

List of elements common to $varlist_1$ and $varlist_2 = (c, d, e) \neq \phi$

From Eq1 we can infer that the 3-tuple (c,d,e) is 101 or 010 or 011 whenever Eq1 is true. The set $S_1 = \{101, 010, 011\}$ is called the projection of the onset of Eq1 on (c, d, e). From Eq2 we can infer that the 3-tuple (c, d, e) is 101 or 001 or 010 whenever Eq2 is true. We shall call $S_2 = \{101, 001, 010\}$ as projection of onset of Eq2 on (c, d, e). A correct solution to the system satisfies both the equations simultaneously. So all correct solutions to the system should have the 3-tupe (c,d,e) = 101 or 010 (a value from intersection of S_1 and S_2). Using this information we can delete all those elements from the $onset_1$ and $onset_2$ whose projection on (c, d, e) is not in the set $S_1 \cap S_2$ Using this reduction we can get an equivalent system of equations (i.e. solution set of the new system of equations is the same as the solution set for the old system). Applying this reduction on our system we get

$$NewEq1 : varlist_1 = (a, b, c, d, e), onset_1 = \{01101, 11010\}$$
$$NewEq2 : varlist_2 = (c, d, e, f, g, h), onset_2 = \{101110, 010010\}$$

We execute this operation sequentially on pairs where it can be applied. The procedure terminates when either some onset becomes empty (case when set of equations is inconsistent) or some reduced set of functions is obtained where the given operation cannot be applied on any pair. This process of reducing the size of onsets by considering a pair of equations at a time is called local reduction. Once a reduced set of functions is obtained, we can combine the onsets of each equation to get onset for the whole system. (Combining is done by doing a 'join' operation over the onsets.) For example, in the above case the solution set for the system of equations will be

$$(a, b, c, d, e, f, g, h) = \{01101110, 11010010\}$$

This complexity of local reduction is proportional to number of points in the onset or roughly exponential in number of variables per equation, which is why number of variables in each equation is to be kept small.

3.2 Removal of Common Factors

We also used the technique of removal of common factors. The basic idea behind this is very simple. Since the nonlinear equations are of the type $F_i = 1$, if some sub-term occurs as a common factor in all the terms of F_i, it can be factored out. In our case suppose we have

$$Eq_i : varlist_i = (a, b, c, d, e), onset_i = \{10100, 10101, 10110\}$$

We see that the triple (a,b,c) = (101) in all the points of the onset, so we can infer that a=1, b=0 and c=1 in the solution for the whole system of equations. We can substitute the values inferred to get reduced system of equations. In other words we are factoring out $ab'c$ and substituting the value in other equations. This operation of factoring out is linear in size of onset or roughly exponential in number of variables in the equation.

4 Description of the Algorithm

4.1 Making Equations

We try to make equations expressing relation between initial contents of the shift register and the bits of key sequence. Using system description such equations can be easily made. The initial contents will be the variables and the key sequence bits will be the constants. To ensure that the system of equations satisfy the criteria for Local reduction to be applicable, we had to make number of variables per equation less than 10.

This was achieved by introducing new variables for the bits generated by the LFSR. A set of linear equations based on the feedback polynomial giving relation between these new variables and the initial variables is added to the system of equations.

Given

1. Feedback polynomial of degree n,
2. Feedforward function of m variables and its tap points. Without loss of generality we assume that the tap points of the nonlinear function are in the last m consecutive stages of the shift register. (as taps are equidistant)
3. 2n consecutive bits of output sequence,

Our system of equations will be as follows:

- Variables involved : $x_1, ...x_{2n+m}$ corresponding to the 2n+m bits generated by LFSR while producing 2n bits of output sequence.
- 2n nonlinear equations, each equation over m variables, describing output bit in terms of m bits of the LFSR. These equations are formed using the feedforward function. We represent these equations in the form of onsets.

– n+m linear equations expressing each of the bits x_{n+1} to x_{2n+m} in terms of the previous n bits. These equations are formed using feedback function of LFSR. We represent these equations in the Algebraic Normal Form.

Representing the linear equations also as onsets would bring uniformity of representation to the system but would constrain the feedback polynomial to have a low density (number of taps less than 10).

4.2 Solving the System of Equations

In our case we had a system of nonlinear equations (2n equations), and a system of linear equations (n+m equations) over the same set of variables (2n+m variables). Each nonlinear equation had 10 or fewer variables. There was no constraint over the set of linear equations. We applied local reduction to the set of nonlinear equations. In case some onsets get reduced to singleton sets, we get the values of all the variables in that equation. For e.g.

$$varlist_i = (a, b, c), onset_i = \{011\}$$

Means that correct solution to the system has a=0, b=1 and c=1.

We substitute the values in both the nonlinear and linear system of equations. In case some linear equation gets reduced to an equation over a single variable, we can infer the value of the variable and substitute it back into the system of equations. We repeat this process local reduction followed by substitution again over the reduced system of equations. When no further reductions are possible, (and system is not yet solved) the reduced system of equations is saved and we perform tree search over the reduced onset space of the system. For this we first pick the equation with least number of points in the onset and then substitute values corresponding to each point in the onset. The new system obtained by the substitution is passed through the same process of local reduction followed by substitution. In case the substitution was correct, we get closer to our solution otherwise the system of equations leads to a contradiction. In that case we undo the substitution and try the next possible substitution. At every stage before applying local reduction we apply factoring out of common terms. The outline of this algorithm in pseudo code is given in Figure 2.

5 Results

The results of applying our algorithm to systems of nonlinear feedforward generators of different parameters are given in Table 1. The results have been obtained on P-II 300MHz system with 64MB RAM.

6 Conclusions

A novel method of cryptanalysis of nonlinear feedforward generator has been described. This method is a known-plaintext attack and works even when just

```
Input:  Feedback polynomial f of degree n,
        Feedforward function g of m variables and
        First 2n bits of the key sequence.
Output: The initial content of the Shift register

Making equations:
The first 2n+m bits of the sequence generated by LFSR are
considered as variables. We form two sets of equations.
* n+m linear equations are formed based on f, describing the
  relationship between the 2n+m variables.
* 2n nonlinear equations describing the given key sequence in
  terms of the variables
The n+m linear equations are represented in ANF
The nonlinear equations are taken in the form Fi = 1
Internal representation of the nonlinear equation is in the
form of sets. We store onsets, i.e. set of points at which
Fi evaluates to 1.

Pseudocode of the algorithm for solving the system of equations: -

numvarfound =0    /* number of variables whose values are found */
WHILE numvarfound < n
  Apply removal of common factors, local reduction on set of
     nonlinear eqns.
  WHILE some variables are found DO
    Substitute the values found in all equations
    Apply removal of common factors, local reduction on set of
        nonlinear eqns
  END WHILE
  IF  numvarfound > n  BREAK /* tree search over reduced eqns */
  Search for equations with least number of points in the onset.
  Try Substituting values as given by each point one by one.
    Check for consistency.
    IF not consistent, undo substitution.
    ELSE try solving the reduced system of equations
END WHILE

Make linear equations expressing the variables found in terms
of first n variables.
Solve system of the linear equations to get values of
the first n variables.
OUTPUT value of first n variables
```

Fig. 2. Algorithm for Cryptanalysis of Nonlinear feedforward generator

$2n$ bits of the keysequence is available, where n is the length of the shift register. The implementation took few minutes to find initial settings of the system with 128-degree polynomial and 10 variable feedforward function.

Table 1. Experimental Results

Number of taps for NLFF	Degree of Polynomial	Time taken
8	64	5.3s
8	64	6.5s
8	71	10.9s
8	128	29.5s
10	128	64.9s

The basic idea of the attack is to express the generator using a set of nonlinear boolean equations of a certain form so that they can be solved efficiently. We expressed the output of the feedforward generator using a system of equations with limited number of variables per equation and lot of common variables between equations. Zakreviskij's method was then used to solve the resulting system of Boolean equations.

We believe that this method can be extended to attack other building blocks of cryptosystems also.

Acknowledgements

The authors would like to thank Prof. A D Zakreviskij for kindly giving a copy of his paper and Prof. C E Veni Madhavan for useful discussions.

References

1. Ross Anderson: Searching for the Optimum Correlation Attack. Proc. Fast Software Encryption -Leuven '94, B. Preneel, ed., 1995
2. Jovan Dj. Golic: On the Security of Nonlinear Filter Generators FAst Software Encryption – Cambridge '96, D. Gollmann, ed., 1996.
3. Jovan Dj. Golic, Andrew Clark, Ed Dawson: Generalized Inversion Attack on Nonlinear Filter Generators IEEE Transactions on Computers, Vol. 49, No. 10, October 2000.
4. Rainer A. Rueppel: Analysis and Design of Stream Ciphers. Springer Verlag Communication and Control Engineering Series 1986.
5. Gustavus J. Simmons (Ed.): Contemporary Cryptology- The Science of Information Integrity. IEEE Press 1992.
6. Arkadij .D. Zakrevskij: Solving system of logical equations by the method of local reduction. Doklady NAN B, 1999, v. 43, No. 5, pp. 5-8. (in Russian).
7. Arkadij .D. Zakrevskij, Irina Vasilkova: Cryptanalysis of the Hagelin Machine by the method of spreading of constants Proc. of Third International Conference of Computer Aided Design of Discrete Devices (CAD DD 99) Minsk, November 10-12 (1999), Vol. 1, pp. 140-147.
8. Arkadij .D. Zakrevskij, Irina Vasilkova: Reducing Large Systems of Boolean Equations. Fourth International Workshop on Boolean Problems, 21-22 Sep. (2000).