An Improved Linear Syndrome Algorithm in Cryptanalysis With Applications *

Kencheng Zeng[1], C.H. Yang[2], and T.R.N. Rao[2]

[1]Graduate School of USTC
Academia Sinica
P.O. Box 3908
Beijing
People's Republic of China

[2]The Center for Advanced Computer Studies
University of Southwestern Louisiana
P.O. Box 44330
Lafayette, LA 70504-4330

Abstract. *A new algorithm is developed for making attacks to certain comparatively simple LFSR based ciphersystems. Special attention is paid towards minimizing the solution distance and guaranteeing the success probability of the attacks. The algorithm is then applied to crack the random bit generators of Geffe (1973) and Beth-Piper (1984).*

I. Introduction

The linear syndrome (LS) method was discussed in [1] for the purpose of solving cryptanalytic problems reducible to the following mathematical setting: What is given is a certain segment of a binary sequence of the form $B = A + X$, where A is a linear recursive sequence with known feedback polynomial $f(x)$ and the sequence X is unknown but sparse in the sense that $Prob\left(x(t) = 1\right) = s_0 < \frac{1}{2}$, s_0 being called the *initial error rate* of the sequence A in the sequence B. What is required to

* This research is supported by Board of Regents of Louisiana Grant #86-USL(2)-127-03

A.J. Menezes and S.A. Vanstone (Eds.): Advances in Cryptology - CRYPTO '90, LNCS 537, pp. 34-47, 1991.

do is to recover from the captured segment of B the sequence A and hence the sequence X.

The method suggests to fix an integer $r \geqslant 3$, find out a set of r-nomial multiples

$$g(x) = 1 + x^{i_1} + \cdots + x^{i_{r-1}} \tag{1}$$

of the feedback polynomial $f(x)$, compute an odd number, say $2m + 1$, of *syndromes*

$$\sigma_{i,k}\left[g(x)\right] \triangleq \sum_{p=0}^{r-1} b(i + i_p - i_k), \tag{2}$$

and revise the signals $b(i)$ to new signals $b'(i)$ according to the rule of majority decision, namely, put $b'(i) = \overline{b(i)}$ if at least $m + 1$ syndromes are 1, otherwise $b'(i) = b(i)$.

It has been shown that the error rate s_1 of the sequence A in the sequence $B' = \{b'(i)\}$ is

$$s_1 = f_m(s_0) = p - (1 - 2p) \sum_{k=0}^{m-1} C_{2k+1}^{k} (pq)^{k+1}, \tag{3}$$

where

$$p \triangleq p(s_0) = \frac{1 - (1 - 2s_0)^{r-1}}{2}, \qquad q \triangleq 1 - p. \tag{4}$$

Further for any s_0, there exists an integer m_c, called the ***critical number***, such that

$$\lim_{k \to \infty} f_m^{(k)}(s_0) = \begin{cases} 0, & \text{if } m \geqslant m_c, \\ 1/2, & \text{if } m < m_c, \end{cases} \tag{5}$$

where $f_m^{(k)}$ denotes the k-fold self-composite of the function f_m.

Of course, the applicability of the convergence theorem (5), which suggests making iterated revision using a **fixed** number of $2m + 1$ syndromes, is based on the tacitly made assumption that during each iteration the error pattern is well-modelled as being independent so that the relation (3) between the old and new error rates s_i and s_{i+1} remains valid. However, the main disadvantage of the method is that it provides no way towards minimizing the *solution distance* and guaranteeing the *success probability*. The solution distance here refers to the length of the captured segment needed in solving the problem. The success probability is the probability of identifying the correct solution after a certain finite number of revisions.

In this paper, we give a new algorithm for solving the same problem, which will make up the above defects. We assume that $r = 3$ and the feedback polynomial $f(x)$ itself is a primitive trinomial. If $f(x)$ is not a trinomial, we can replace it by its trinomial multiple of the least possible degree, which can always be found by computing discrete logarithms [2].

We point out that some of these concepts can be found in [7] with some numerical experiments reported. However, their paper gave no consideration to such important issues of cryptanalysis as convergence of the method, solution distance, success probability, and possible applications. Beside the applications, the present paper is interesting in that the algorithm proposed here contains in itself a proof of its convergence much simpler than that given in [1].

II. The Improved LS Algorithm

(1) Supercritical and cleansing numbers

We start the new algorithm with the following.

Lemma. The function $w(s) \triangleq f_1(s) - s$ has in $\left(0, \frac{1}{2}\right)$ a single root $\alpha \approx 0.1294$. We have $w(s) < 0$ in $\left(0, \alpha\right)$, $w(s) > 0$ in $\left(\alpha, \frac{1}{2}\right)$, and $\lim_{k\to\infty} f_1^{(k)}(s) = 0$ if $0 \leqslant s < \alpha$.

Proof. Let $u = 1 - 2s$. We have

$$w(s) = p^2(3 - 2p) - s = \frac{(1-u^2)^2}{4}\left(3-(1-u^2)\right) - \frac{1-u}{2}$$
$$= -\frac{u(u-1)(u^4 + u^3 + \cdots + u - 2)}{4} = -\frac{u(u-1)h(u)}{4}.$$

But $h(u)$ increases strictly with u and $h(0) = -2$, $h(1) = 2$, so we see $h(u)$ has only one zero β in the open u-interval $(0, 1)$, which can be found by Newton's method of successive approximation to be $\beta \approx 0.7412$. From here we get the conclusion about the zero α and the sign of $w(s)$. Further, if $s \in \left(0, \alpha\right)$ and we define $s_0 \triangleq s$, $s_k \triangleq f_1(s_{k-1})$, then the sequence s_k, $k \geqslant 0$, decreases strictly to a certain limit $s^* \in \left[0, \alpha\right)$, and $w(s^*) = 0$. So we must have $s^* = 0$. This proves the lemma.

The main idea in the improved LS algorithm is to make the revisions with a **reducing** number of syndromes, with the length of the segment under processing being reduced correspondingly. The central role is played by the concept of *supercritical* and *cleansing* numbers.

Definition. By the supercritical number m_{sc} corresponding to the initial error rate s_0, we mean the least integer m such that the inequalities

$$s_k \triangleq f_{m-k+1}(s_{k-1}) < s_{k-1} \tag{6}$$

hold true for all $1 \leqslant k \leqslant m$. By the t-th cleansing number l_t corresponding to s_0

we mean the least integer l such that

$$f_1^{(l)}(s_{m_{sc}}) < 10^{-t}. \tag{7}$$

Theorem 1. For any $s_0 \in \left(0, \frac{1}{2}\right]$ and any $t \geqslant 0$, the supercritical number m_{sc} and the cleansing number l_t exist.

Proof. As pointed out in [1],

$$\lim_{m\to\infty} f_m(s_0) = p - (1-2p)\sum_{k=0}^{\infty} C_{2k+1}^{k}(pq)^{k+1} = 0.$$

Therefore, for m sufficiently large we have

$$s_1 = f_m(s_0) < \min(\alpha, s_0).$$

But we know from (3)

$$f_i(s) \leqslant f_1(s), \qquad \forall i \geqslant 1,$$

So we have

$$s_2 = f_{m-1}(s_1) < f_1(s_1) < s_1 < \alpha,$$
$$s_3 = f_{m-2}(s_2) < f_1(s_2) < s_2 < \alpha,$$

etc. This means the set of positive integers m, for which the condition (6) holds, is not empty, and the existence of the m_{sc} corresponding to s_0 follows from the well-ordering principle of the set of all non-negative integers. The existence of l_t is an easy consequence of the lemma given above, for we see from $f_1(s_{m_{sc}-1}) < s_{m_{sc}-1}$ that $s_{m_{sc}-1} < \alpha$.

Theorem 1 provides a natural justification to the following algorithm for finding m_{sc}:

Step 1: $1 \to m_{sc}$, $1 \to k$, $s_0 \to s$.

Step 2: If $k = 0$, stop. Otherwise compute $s' = f_k(s)$.

Step 3: If $s' < s$ then $k - 1 \to k$, $s' \to s$, and return to Step 2.

Step 4: $m_{sc} + 1 \to m_{sc}$, $m_{sc} \to k$, $s_0 \to s$, go to Step 2.

s_0	m_c	m_{sc}	l_4	$c(s_0, 4)$
0.03125	1	1	2	7
0.0625	1	1	3	9
0.09375	1	1	5	13
0.125	1	1	10	23
0.15625	2	2	8	23
0.1875	2	3	4	19
0.21875	3	4	2	23
0.25	3	5	1	37
0.28125	4	6	0	51
0.3125	6	7	1	85
0.34375	8	10	0	339
0.375	13	14	0	2387
0.40625	22	24	0	218451

The above is a table of critical, supercritical, and cleansing numbers, together with the numbers $c(s_0, t)$ to be defined later in Theorem 2, computed for values of s_0 at step length equal to $\frac{1}{32}$. It may be of some interest to note that $m_{sc} - m_c \leqslant 2$.

(2) The improved LS algorithm

Now we can state the new LS algorithm.

Theorem 2. There exists an algorithm which, when given as input the number t and a captured B-segment of length

$$N(s_0, t) = \left(1 + 2l_t + 2\sum_{m=1}^{m_{sc}} L(m)\right) n = c(s_0, t)\, n, \tag{8}$$

where $n = deg\, f(x)$ and $L(m) \triangleq 2^{\lfloor \frac{4m-1}{6} \rfloor}$, will give as output the state vector of the attacked LFSR at a certain moment i with success probability

$$P_{success} > (1 - 10^{-t})^n. \tag{9}$$

The computational complexity of the algorithm in terms of bit operations is

$$Q(s_0, t) = \left(6l_t^2 + 2m_{sc}(m_{sc} + 2)(2l_t + 1) + 4\sum_{j=1}^{m_{sc}} (2j + 1)D(j - 1)\right) n \tag{10}$$

$$= q(s_0, t)\, n,$$

where

$$D(j) \triangleq L(1) + L(2) + \cdots L(j), \qquad D(0) = 0. \tag{11}$$

Proof. The required algorithm is divided into two phases: the *reducing* phase, at which the initial error rate s_0 is reduced to $s_{m_{sc}} < \alpha$; and the *cleansing* phase, at which the remanent error rate $s_{m_{sc}}$ is rendered below 10^{-t}.

At the reducing phase we need $p = \lceil \frac{2m_{sc} + 1}{3} \rceil$ trinomial multiples of $f(x)$ to form the syndromes, which we choose to be

$$g_0(x) = f(x), \quad g_{i+1}(x) = g_i^2(x), \quad 0 \leqslant i \leqslant p - 2.$$

Observe that each trinomial $g(x) = 1 + x^{i_1} + x^{i_2}$ provides three syndrome formulas

$$\sigma_{i,k}(g) = \sum_{p=0}^{2} b(i + i_p - i_k), \quad k = 0, 1, 2$$

for checking the same ciphertext signal $b(i)$. We arrange the $3p$ syndrome formulas in the following two different ways:

(i) $\sigma_{i,0}(\mathcal{J}_0), \sigma_{i,1}(\mathcal{J}_0), \sigma_{i,2}(\mathcal{J}_0), \sigma_{i,0}(\mathcal{J}_1), \sigma_{i,1}(\mathcal{J}_1), \sigma_{i,2}(\mathcal{J}_1), \cdots, \sigma_{i,0}(\mathcal{J}_{p-1}), \sigma_{i,1}(\mathcal{J}_{p-1}), \sigma_{i,2}(\mathcal{J}_{p-1})$

(ii) $\sigma_{i,2}(\mathcal{J}_0), \sigma_{i,1}(\mathcal{J}_0), \sigma_{i,0}(\mathcal{J}_0), \sigma_{i,2}(\mathcal{J}_1), \sigma_{i,1}(\mathcal{J}_1), \sigma_{i,0}(\mathcal{J}_1), \cdots, \sigma_{i,2}(\mathcal{J}_{p-1}), \sigma_{i,1}(\mathcal{J}_{p-1}), \sigma_{i,0}(\mathcal{J}_{p-1})$

Reducing Phase

Step 1: $c(s_0, t)\ n \rightarrow N,\ m_{sc} \rightarrow m,\ nL(m) \rightarrow L.$

Step 2: For $L \leqslant i \leqslant N - L - 1$, compute the syndromes needed by the first $2m + 1$ formulas of (i) or (ii) according to $i \leqslant \dfrac{N}{2}$ or $i > \dfrac{N}{2}$,

do $\overline{b(i)} \rightarrow b(i)$, if at least $m+1$ syndromes are 1.

Step 3: $m - 1 \rightarrow m$. If $m = 0$, go to Step 5.

Step 4: $nL(m) + L \rightarrow L$, return to Step 2.

Cleansing Phase

Step 5: $l_t \rightarrow m, L + n \rightarrow L.$

Step 6: $m - 1 \rightarrow m$. If $m < 0$, stop.

Step 7: For $L \leqslant i \leqslant N - L - 1$, compute $\sigma_{i,0}(f),\ \sigma_{i,1}(f),\ \sigma_{i,2}(f)$,

do $\overline{b(i)} \rightarrow b(i)$, if at least two syndromes are 1. Return to Step 6.

Observe that at the $(m_{sc} - m + 1)$-th round of the reducing phase, the signals $b(i)$ with

$$L \leqslant i \leqslant N - L - 1 \tag{12}$$

are revised according to majority decision with $2m + 1$ syndromes computed from the wider range of signals $b(j)$ with

$$L - nL(m) \leqslant j \leqslant N - L + nL(m) - 1. \tag{13}$$

So we can conclude by reverse induction on m that after the $(m_{sc} - m + 1)$-th round of work

$$Prob\left\{b(i) \neq a(i)\right\} = s_m,$$

for all i satisfying (12), provided at the start of the round we have in the signals $b(j)$, with j satisfying (13), all the necessary data for computing the $2m + 1$ syndromes needed. Evidently, it suffices to check the point for the case

$$i = \lfloor \frac{N}{2} \rfloor, \quad 2m + 1 \equiv 1 \mod 3.$$

Using the easily checkable fact that $\sum_{k=1}^{m} L(k) \geqslant 2^{\lceil \frac{2m+1}{3} \rceil}$, it is easy to show that in this case the largest j, for which the signal $b(j)$ is used in computing the syndromes is

$$j_{\max} = \lceil \frac{N}{2} \rceil + n2^{\frac{2m+1}{3}} \leqslant N - L + nL(m) - 1.$$

This means the proviso is fulfilled at each round and the initial error rate s_0 definitely will be rendered below α after the reducing phase. That it will be further reduced to below 10^{-t} can be proved in a similar way. Thus we see that, after $m_{sc} + l_t$ rounds of iterated revision, the algorithm will output the n-bit vector

$$\left\{b(\frac{N-n}{2}),\ b(\frac{N-n}{2}+1),\ \cdots,\ b(\frac{N-n}{2}+n-1)\right\},$$

which coincides with the $\frac{N-n}{2}$-th state vector of the attacked LFSR with probability (9).

The formula (10) for the computational complexity can be derived by straightforward manipulations, and is omitted here. □

III. Cracking the Generators of Geffe and Beth-Piper

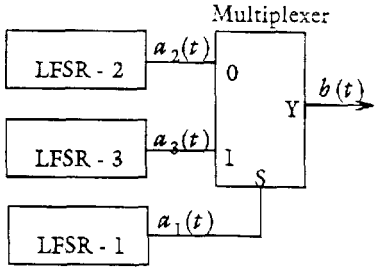

Figure 1. Generator **G**

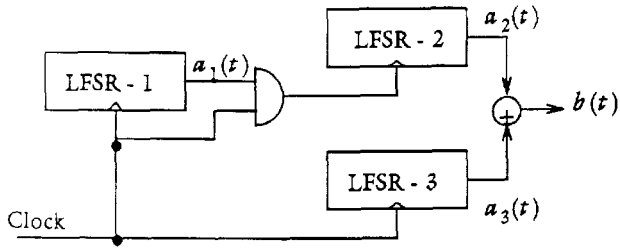

Figure 2. Generator **BP**

The Geffe [3] generator **G** and the Beth-Piper [4] generator **BP** (as described by Figures 1 and 2, respectively), are designed on the basis of quite different cryptographical ideas; but they can be cracked by one and the same method, at the expense of nearly the same amount of computation. It is assumed that all the feedback polynomials are primitive.

Theorem 3. If in the **G**-generator the feedback polynomials $f_i(x)$, $1 \leqslant i \leqslant 3$, of the LFSRs are *all known to the cryptanalyst* and $f_2(x)$ is a *trinomial* of degree n, then the system can be broken on a captured segment of length $N = 37\,n$, at the computational expense of $Q = 896\,n$ bit operations, disregarding the work not directly related to the LS algorithm.

Proof. The output signal of the **G**-generator at the moment t is

$$b(t) = a_1(t)a_3(t) + \overline{a_1(t)}a_2(t)$$
$$= a_2(t) + a_1(t)\Big[a_2(t) + a_3(t)\Big].$$

It follows from the balanced property of m-sequences that

$$s_0 = Prob\left\{b(t) \neq a_2(t)\right\} = \frac{1}{4}.$$

So we see LFSR-2 is the Achilles heel in the system. If, for example, $n \leqslant 100$ and we choose $t = 4$, then the sequence $A_2 = \{a_2(t)\}$ can be recovered by our algorithm with success probability exceeding 0.99, provided the length of the captured segment exceeds $37n$.

Now suppose we already have at hand the sequence A_2. Compare the signals $a_2(t)$ with $b(t)$, for $0 \leqslant t \leqslant 37n$, and mark those moments t_i, for which

$$a_2(t_i) + b(t_i) = 1.$$

It is easy to see that at these moments

$$a_1(t_i) = 1, \quad a_3(t_i) = b(t_i).$$

Divide each power x^{t_i} by $f_1(x)$ and $f_3(x)$ respectively to obtain the remainders

$$x^{t_i} = r_{i,0} + r_{i,1}x + \cdots + r_{i,n_1-1}x^{n_1-1} \quad mod\ f_1(x)$$

and

$$x^{t_i} = s_{i,0} + s_{i,1}x + \cdots + s_{i,n_3-1}x^{n_3-1} \quad mod\ f_3(x).$$

Then we shall have two linear systems, each containing the same number of approximately $9n$ equations

$$r_{i,0}a_1(0) + r_{i,1}a_1(1) + \cdots + r_{i,n_1-1}a_1(n_1-1) = 1,$$

and

$$s_{i,0}a_3(0) + s_{i,1}a_3(1) + \cdots + s_{i,n_3-1}a_3(n_3-1) = b(t_i).$$

If $n_1, n_2 << 9n$, then these systems will determine with probability nearly 1 (see [5]), the initial states of LFSR-1 and LFSR-3.

Theorem 4. If in the **BP**-generator, $f_1(x)$ and $f_3(x)$ are ***known*** trinomials of degrees not greater than n, while $f_2(x)$ is ***arbitrary and unknown***, then the system can be broken on a captured segment of length $N = 37\,n$, at the computational expense of $Q = 1792\,n$ bit operations, disregarding the work not directly related to the LS algorithm.

Proof. Denote the output signal of the clock-controlled LFSR-2 at the moment t by $a_2'(t)$, then we have

$$Prob\left[a_2'(t) = a_2'(t+1)\right] = Prob\left[a_1(t) = 0\right] + Prob\left[a_1(t) = 1\right] Prob\left[a_2(t) = a_2(t+1)\right]$$

$$= \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4}.$$

Thus if we write

$$a_2^*(t) = a_2'(t) + a_2'(t+1), \quad b^*(t) = b(t) + b(t+1) \quad a_3^*(t) = a_3(t) + a_3(t+1),$$

then we will encounter with the cracking problem $B^* = A_3^* + A_2^*$, where A_3^* is an m-sequence with the known trinomial feedback $f_3(x)$ and

$$s_0 = Prob\left[a_2^*(t) = 1\right] = \frac{1}{4}.$$

After cracking it with the improved LS algorithm, we will get at the sequence A_3^*. The signals $a_3(t)$ can be determined recursively by

$$a_3(0) = \epsilon, \qquad a_3(t) = a_3^*(t-1) + a_3(t-1)$$

where $\epsilon = 0$ or 1. The actual value of ϵ can be decided by the linear relations the signals of A_3 should satisfy, for here $f_3(x)$ is a trinomial.

Now suppose we have done this and have A_2' at hand. Define

$$a_1^*(t) \triangleq \begin{cases} 0, & \text{if } a_2'(t) = a_2'(t+1), \\ 1, & \text{if } a_2'(t) \neq a_2'(t+1). \end{cases}$$

Since, as can be easily seen,

$$Prob\left(a_1(t) = 1 \mid a_2'(t) = a_2'(t+1)\right) = \frac{1}{3},$$

we have

$$s_0 = Prob\left(a_1^*(t) \neq a_1(t)\right)$$

$$= Prob\left(a_2'(t) = a_2'(t+1)\right) Prob\left(a_1(t) = 1 \mid a_2'(t) = a_2'(t+1)\right) = \frac{1}{4}.$$

Once again, we meet with the problem

$$A_1^* = A_1 + X, \; s_0 = \frac{1}{4},$$

with the trinomial $f_1(x)$ assumed known. So we can recover the sequence A_1.

Finally, the sequence A_2 can be recovered by removing the repeating signals which appear when $a_1(t) = 1$. The unknown feedback polynomial $f_2(x)$ can then be found by the well-known Massey [6] synthesis algorithm, provided $deg\ f_2(x) \leqslant 18n$.

References

1. Kencheng Zeng and Minqiang Huang, *On the Linear Syndrome Method in Cryptanalysis*, CRYPTO 88, 1988.

2. Don Coppersmith, *Fast Evaluation of Logarithms in Fields of Characteristic Two*, IEEE Trans. Information Theory, IT-30, July 1984, pp. 587-594.

3. Philip R. Geffe, *How to Protect Data with Ciphers That Are Really Hard to Break*, Electronics, Jan. 4, 1973, pp. 99-101.

4. T. Beth and F.C. Piper, *The Stop-and-Go-Generator*, EUROCRYPT 84, 1984, pp. 88-92.

5. K.C. Zeng, C.H. Yang, and T.R.N. Rao, *On the Linear Consistency Test in Cryptanalysis*, to appear.

6. J. L. Massey, *Shift-Register Synthesis and BCH Decoding*, IEEE Trans. Information Theory, IT-15, Jan. 1969, pp. 122-127.

7. W. Meier and O. Staffelback, *Fast Correlation Attacks on Certain Stream Ciphers*, Journal of Cryptology, Vol. 1, pp. 159-176, 1989.