4.3 General attacks on LFSR based stream ciphers

Recalling our initial discussion on possible attack scenarios, we now assume that $\mathbf{z} = z_1, z_2, \ldots, z_N$ is a known keystream sequence from a generator and our task is to find a distinguishing attack or a key recovery attack that can be performed with complexity lower than exhaustive key search.

4.3.1 Linear complexity and Berlekamp-Massey algorithm

As the generator has only finite memory, the keystream sequence $\mathbf{z} = z_1, z_2, \ldots$ will be periodic (after possibly removing some of the first symbols). Any such sequence can be generated by an LFSR. One possible approach would then be to replace the entire generator with an (in general very long) LFSR. This leads us to the following definition.

Definition 4.3. *The* linear complexity *of a sequence* $\mathbf{s}$ *(finite or semi-infinite), denoted* $L(\mathbf{s})$*, is the length of the shortest LFSR that can produce the sequence.*

To find the shortest LFSR producing a certain sequence we use the *Berlekamp-Massey algorithm.* The algorithm is given as follows.

Berlekamp-Massey algorithm

```
 IN: A sequence $\mathbf{s} = (s_0, s_1, \ldots, s_{N-1})$ of length $N$.
OUT: The shortest LFSR $< C(D), L >$ generating $\mathbf{s}$.
  1. Initilization $C(D) = 1$, $L = 0$, $C^*(D) = 1$, $d^* = 1$, $m = -1$, $n = 0$.
  2. While $(n < N)$ do the following:
         2.1 Compute the discrepancy
                 $d = s_n - \sum_{i=1}^{L} -c_i s_{n-i}.$
         2.2 If $d \neq 0$ do the following:
           $T(D) = C(D)$, $C(D) = C(D) - d \cdot (d^*)^{-1} \cdot C^*(D) D^{n-m}$.
           If $L \leq n/2$ then $L = n + 1 - L$, $C^*(D) = T(D)$, $d^* = d$, $m = n$.
         2.3 $n = n + 1$.
  3. Return $< C(D), L >$
```

Example 4.12. *Assume that we have observed the finite sequence* $\mathbf{s} = (0, 0, 1, 1, 0, 0, 0, 1)$. *Find the shortest LFSR generating this sequence.*

Solution: *Use the Berlekamp-Massey algorithm. Going through the steps we receive the following intermediate values.*

s_n	d	$T(D)$	$C(D)$	L	$C^*(D)$	d^*	m	n
-	-	-	1	0	1	1	-1	0
0	0	-	1	0	1	1	-1	1
0	0	-	1	0	1	1	-1	2
1	1	*1*	$1+D^3$	3	1	1	2	3
1	1	*1*	$1+D+D^3$	3	1	1	2	4
0	1	*1*	$1+D+D^2+D^3$	3	1	1	2	5
0	0	*1*	$1+D+D^2+D^3$	3	1	1	2	6
0	1	$1+D+D^2+D^3$	$1+D+D^2+D^3+D^4$	4	$1+D+D^2+D^3+D^4$	1	6	7
1	0	$1+D+D^2+D^3$	$1+D+D^2+D^3+D^4$	4	$1+D+D^2+D^3+D^4$	1	6	8

A shortest LFSR has connection polynomial $C(D) = 1 + D + D^2 + D^3 + D^4$ and length 4.

The running time of the Berlekamp-Massey algorithm for determining the linear complexity of a length n sequence is $O(n^2)$ operations. The algorithm actually delivers one connection polynomial and the length L of the LFSR. If (and only if) $L \leq N/2$ there is a unique connection polynomial. Otherwise, there are several possible connection polynomials all with the same length.

Assume that we want to find the shortest LFSR generating the sequence $\mathbf{s}$, where now $\mathbf{s}$ is a semi-infinite periodic sequence with period T. Then the Berlekamp-Massey algorithm can provide the solution if the input is the length $2T$ sequence $(s_0, s_1, \ldots, s_{T-1}, s_0, s_1, \ldots, s_{T-1})$. In fact, you only need to process the first $T + k$ symbols of the sequence, where k is the first positive integer such that $(s_0, s_1, \ldots, s_{T-1}, s_0, s_1, \ldots, s_{k-1})$ has linear complexity $\leq k$.

The above procedure is very useful if we want to establish the linear complexity of some sequence but we do not know the period of the sequence, only some upper bound. Then we can run the Berlekamp-Massey algorithm until we know that enough consecutive symbols have been processed and we will receive both the connection polynomial and the linear complexity.

In connection with linear complexity, we introduce a few operations on sequences. Let $\mathbf{s^1} = s_0^1, s_1^1, s_2^1, \ldots$ and $\mathbf{s^2} = s_0^2, s_1^2, s_2^2, \ldots$ be two sequences. Let $f(x_1, x_2)$ be a function in two variables, $x_1, x_2 \in \mathbb{F}_q$. By

$$\mathbf{s} = f(\mathbf{s^1}, \mathbf{s^2})$$

we mean the sequence $\mathbf{s} = f(s_0^1, s_0^2), f(s_1^1, s_1^2), f(s_2^1, s_2^2), \ldots$.

As every Boolean function can be written in algebraic normal form we pay special attention to the two cases $f(x_1, x_2) = x_1 + x_2$ and $f(x_1, x_2) = x_1 x_2$.

Theorem 4.10. *Let $\mathbf{s^1}$ and $\mathbf{s^2}$ be two sequences with linear complexity $L(\mathbf{s^1})$ and $L(\mathbf{s^2})$ respectively. Then*

- *If $f(x_1, x_2) = x_1 + x_2$ then $L(f(\mathbf{s^1}, \mathbf{s^2})) \leq L(\mathbf{s^1}) + L(\mathbf{s^2})$.*
- *If $f(x_1, x_2) = x_1 x_2$ then $L(f(\mathbf{s^1}, \mathbf{s^2})) \leq L(\mathbf{s^1}) L(\mathbf{s^2})$.*

Example 4.13. *Assume that we have a generator generating a keystream sequence of the form*

$$\mathbf{z} = f(\mathbf{s^1}, \mathbf{s^2}, \mathbf{s^3}, \mathbf{s^4}),$$

where $\mathbf{s^i}$, $i = 1, \ldots, 4$ *are LFSR sequences with period* $2^{L_i} - 1$ *(m-sequences). Let*

$$f(x_1, x_2, x_3, x_4) = x_1 + x_2x_3 + x_3x_4 + x_2x_4.$$

Find a bound on the linear complexity of the keystream sequence $L(\mathbf{z})$*.*

Solution: *Using Theorem 4.10 we get*

$$L(\mathbf{z}) = L(f(\mathbf{s^1}, \mathbf{s^2}, \mathbf{s^3}, \mathbf{s^4})) \leq L(\mathbf{s^1})+L(\mathbf{s^2})L(\mathbf{s^3})+L(\mathbf{s^3})L(\mathbf{s^4})+L(\mathbf{s^2})L(\mathbf{s^4}) \leq L_1+L_2L_3+L_3L_4+L_2L_4.$$

4.3.2 Correlation Attacks

see description of Project 3

4.3.3 Linear distinguishing attacks

One of the possible tools to analyze a cipher is a linear distinguishing attack approach. This approach has been used to perform attacks on many ciphers. The approach can be regarded as a generalization of linear cryptanalysis, invented by Matsui in 1993 for analysis of block ciphers. The basic idea is to introduce linear approximations of all nonlinear operations in a specific "path" of the cipher. The path should be such that it connects some know values, which in this case must be key stream symbols. If the linear approximation is true, this leads to a linear relationship among the known key stream symbols. If it is not true, we can think of the error introduced by the linear approximation as truly random noise. In summary, some linear combination of key stream symbols corresponding to the linear relationship discussed above can be viewed as a sample from a very noisy (but not uniform) distribution. By collecting many such samples, we can eventually distinguish the distribution they are drawn from, from the uniform distribution. This results in a successful distinguishing attack, in the sence that it shows that the samples are not from a truly random generator. In the last step, the standard approach is to apply hypothesis testing between two known distributions.

Nonlinear parts of a cipher are substituted by some linear functions, and the introduced error are compensated for by introducing noise variables. If $S(x)$ is a nonlinear function and x is taken from a field this corresponds to the notation

$$S(x) = R(x) + N_x,$$

where $R(x)$ is a linear function and N_x denotes a new unknown random variable with a (usually) biased distribution. We simply write $\mathbf{R}$ to denote such a specific linear approximation. For increased efficiency of the attack, the introduced noise variables should have a significant bias. This is completely determined by the choice of the linear approximation of the corresponding nonlinear operation.

After approximation, all operations in the "linearized" part of the cipher are linear. In this case we find some linear expression L_1 including only symbols from the output stream of the "linearized" cipher (the linear cipher without noise variables), which is always equal to a constant (usually equal to 0).

If we then apply the expression L_1 to the real cipher, where now the noise variables are included, then obviously we get the second linear expression L_2 consisting of the noise variables only, that sum to some linear function of output symbols L_1. If we know the distribution of each noise variable, then it does not take much effort to get the distribution of the linear combination of all of them, L_2. In this way, we collect samples from some nonuniform distribution. A collection of n samples is then collected. If the output is a truly random sequence, then the collected samples are drawn from the uniform distribution, since any nonzero linear combination of uniformly distributed random variables produces samples from the uniform distribution as well. Finally, the decision rule from the hypothesis testing algorithm gives the answer.

Example 4.14. *Assume that a generator is constructed as follows. A long LFSR with connection polynomial* $C(D) = 1 + D^{34} + D^{69}$ *is generating a sequence* $\mathbf{s} = (s_0, s_1, s_2, \ldots)$ *in* $\mathbb{F}_2$. *The output of the generator is generated as*

$$z_i = f(s_i, s_{i+1}, s_{i+2}, s_{i+3}), i = 0, 1, \ldots,$$

where f *is the Boolean function* $f(x_1, x_2, x_3, x_4) = x_1 + x_2 + x_3 + x_1x_2x_3x_4$. *The key is the initial state of the LFSR. This is called a* filter generator. *Try to describe a linear distinguishing attack on the generator.*

Solution: *First we note that the LFSR sequence obeys the recursion*

$$s_i + s_{i+35} + s_{i+69} = 0, i = 0, 1, \ldots.$$

Next, a linear distinguishing attack replaces nonlinear cipher components with linear ones. In this case the nonlinear part is the f *function. A suitable choice is replace* f *with the linear function* $g(x_1, x_2, x_3, x_4) = x_1 + x_2 + x_3$. *As is easily checked, these two functions evaluate to the same value 15 times out of 16. After replacement, we have*

$$z_i = g(s_i, s_{i+1}, s_{i+2}, s_{i+3}) = s_i + s_{i+1} + s_{i+2}, i = 0, 1, \ldots.$$

Now we try to find a set of dependent linear equations. As $s_i + s_{i+35} + s_{i+69} = 0$ *we also have*

$$s_i + s_{i+1} + s_{i+2} + s_{i+35} + s_{i+36} + s_{i+37} + s_{i+69} + s_{i+70} + s_{i+71} = 0, i = 0, 1, \ldots.$$

But $z_i = s_i + s_{i+1} + s_{i+2}$, *so we must have* $z_i + z_{i+35} + z_{i+69} = 0$ *(assuming that* g *always gives the result of* f*).*

So in our attack we create a sequence of sample values $Q_i = z_i + z_{i+35} + z_{i+69}, i = 0, 1, \ldots.$ *Calculating* $P(Q_i)$ *gives* $P(Q_i = 0) = 0.835$.

The number of zeros in the $\mathbf{Q}$ *sequence has a binomial distribution with probability* 0.835. *If the* $\mathbf{z}$ *sequence would have been generated from a truly random source, then the number of zeros in the* $\mathbf{Q}$ *sequence has a binomial distribution with probability* 0.5. *One applies a hypothesis test to distinguish between these two cases.*

4.4 More Exercises

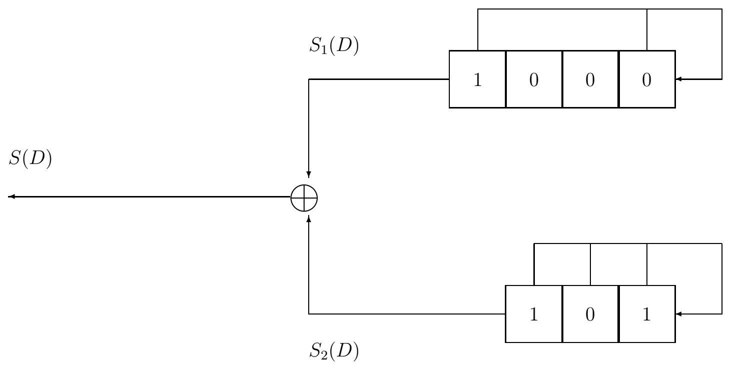
Exercise 4.13.

a) *Find the shortest LFSR in GF(2) that generates the sequence* [101100001].

b) *Find the shortest LFSR in GF(2) that generates the sequence* $[101100001]^\infty$.

Exercise 4.14. *Find the shortest linear feedback shift register over the field GF(2^4) that generates the following sequence of 4-tuples:* A,6,0,B,0,1,4,1,9,B,E. *Hexadecimal notation has been used. The 4-tuple (a_3, a_2, a_1, a_0) corresponds to $a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$. The field GF($2^4$) is generated by the irreducible polynomial $\pi(x) = x^4 + x^3 + x^2 + x + 1$ and $\pi(\alpha) = 0$.*

Exercise 4.15. *A cipher manufacturer has a large collection of short shift registers, $L \leq 4$. As a part in one of his ciphers we find the following construction, generating a sequence $S(D)$.*



a) *Find the expressions $S_i(D) = \frac{P_i(D)}{C_i(D)}$ for $S_1(D)$ and $S_2(D)$.*

b) *If the price for the construction is proportional to the sum of the D-elements, did the designer find the best construction?*

Exercise 4.16 *Find the shortest LFSR that generates the sequence*

$$s = [10000010101000]^\infty + [1010011]^\infty$$

in the field GF(2).