2.3 Cracking an LFSR Stream XOR Encryption

Let us break down the abstract setting of Section 2.2 to an explicit procedure for cracking an XOR cipher that uses an LFSR sequence as keystream. (This section doesn't depend on 2.1 or 2.2 but contains a direct approach.)

Consider a key bitstream $u_0, u_1, \ldots$ generated by an LFSR by the formula $u_n = s_1 u_{n-1} + \cdots + s_l u_{n-l}$. Assume a plaintext a is XOR encrypted using this key stream, resulting in the ciphertext c, where $c_i = a_i + u_i$ for $i = 0, 1, \ldots$ What are the prospects of an attacker who knows a chunk of the plaintext?

Well, assume she knows the first $l+1$ bits $a_0, \ldots, a_l$ of the plaintext. She immediately derives the corresponding bits $u_0, \ldots, u_l$ of the key stream, in particular the initial state of the LFSR. For the yet unknown coefficients s_i she knows a linear relation:

$$s_1 u_{l-1} + \cdots + s_l u_0 = u_l.$$

Each additional known plaintext bit yields one more relation, and having l relations, from $2l$ bits of known plaintext, the easy linear algebra over the field $\mathbb{F}_2$ finds a unique solution (in non-degenerate cases).

So assume we know the first $2l$ bits $u_0, \ldots, u_{2l-1}$ from an LFSR of length l. The state vector

$$u_{(i)} = (u_i, \ldots, u_{i+l-1}) \quad \text{for } i = 0, 1, \ldots$$

is the register content for step i (in reversed order compared with Figure 1.7). Thus the analysis focusses on the states, not directly on the output. The recursion in matrix form (for $n \geq l$) is

$$\begin{pmatrix} u_{n-l+1} \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \\ s_l & s_{l-1} & \ldots & s_1 \end{pmatrix} \begin{pmatrix} u_{n-l} \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix}$$

or more parsimoniously (the indices being substituted by $m = n - l + 1$)

$$u_{(m)} = S \cdot u_{(m-1)} \quad \text{for } m \geq 1$$

where S is the companion matrix. As a further step we collect l consecutive state vectors $u_{(i)}, \ldots, u_{(i+l-1)}$ in a state matrix

$$U_{(i)} = \begin{pmatrix} u_i & u_{i+1} & \ldots & u_{i+l-1} \\ u_{i+1} & u_{i+2} & \ldots & u_{i+l} \\ \vdots & \vdots & \ddots & \vdots \\ u_{i+l-1} & u_{i+l} & \ldots & u_{2l-2} \end{pmatrix}$$

and set $U = U_{(0)}$, $V = U_{(1)}$. This yields the formula

$$V = S \cdot U$$

that expresses the unknown coefficients $s_1, \ldots, s_l$ by the known plaintext bits $u_0, \ldots, u_{2l-1}$. Most notably it allows us to write down the solution immediately—provided that the matrix U is invertible:

$$S = V \cdot U^{-1}.$$

The matrix S explicitly displays the coefficients $s_1, \ldots, s_l$. We'll discuss the invertibility later on.

Example

Assume we are given a ciphertext:

```
10011100 10100100 01010110 10100110 01011101 10101110
01100101 10000000 00111011 10000010 11011001 11010111
00110010 11111110 01010011 10000010 10101100 00010010
11000110 01010101 00001011 11010011 01111011 10110000
10011111 00100100 00001111 01010011 11111101
```

We suspect that the cipher is XOR with a key stream from an LFSR of length $l = 16$. The context suggest that the text is in German and begins with the word "Treffpunkt" (meeting point). To solve the cryptogram we need 32 bits of plaintext, that is the first four letters only, presupposed that the theory applies. This gives 32 bits of the key stream:

```
01010100 01110010 01100101 01100110 = T r e f
10011100 10100100 01010110 10100110   cipher bits
-------- -------- -------- --------
11001000 11010110 00110011 11000000   key bits
```

Sage sample 2.1 determines the coefficient matrix. Its last row tells us that all $s_i = 0$ except $s_{16} = s_5 = s_3 = s_2 = 1$.

Now we know the LFSR and the initial state, and can reconstruct the complete key stream—yes, it is the same as in Section 1.10—and write down the plaintext (that by the way begins a bit differently from our guess).

We have shown that the cofficients are uniquely determined assuming the state matrix $U = U_{(0)}$ is invertible. As a consequence in this case the LFSR is completely known, and all output bits are predictable. We have yet to discuss the case where the matrix U is singular.

If one of the first l state vectors (= rows of the matrix U) is zero, then all following state vectors are zero too, and prediction is trivial.

Thus we may assume that none of these vectors are zero, but that they are linearly dependent (reinventing the Noetherian principle for this special

Sage Example 2.1 Determining a coefficient matrix

```
sage: l = 16
sage: kbits =
      [1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,0,0,0,1,1,0,0,1,1,1,1,0,0,0,0,0,0]
sage: ulist = []
sage: for i in range(0,l):
        state = kbits[i:(l+i)]
        ulist.append(state)
sage: U = matrix(GF(2),ulist)
sage: det(U)
1
sage: W = U.inverse()
sage: vlist = []
sage: for i in range(1,l+1):
        state = kbits[i:(l+i)]
        vlist.append(state)
sage: V = matrix(GF(2),vlist)
sage: S = V*W
sage: S
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0]
```

scenario). Then there is a smallest index $k \geq 1$ such that $u_{(k)}$ is contained in the subspace spanned by $u_{(0)}, \ldots, u_{(k-1)}$, and we find coefficients $t_1, \ldots, t_k \in \mathbb{F}_2$ such that

$$u_{(k)} = t_1 u_{(k-1)} + \cdots + t_k u_{(0)}.$$

Then also $u_{(k+1)} = S \cdot u_{(k)} = t_1 S \cdot u_{(k-1)} + \cdots + t_k S \cdot u_{(0)} = t_1 u_{(k)} + \cdots + t_k u_{(1)}$,

and by induction we get

$$u_{(n)} = t_1 u_{(n-1)} + \cdots + t_k u_{(n-k)} \quad \text{for all } n \geq k.$$

This formula predicts all the following bits.

Discussion

- For a singular state matrix this consideration yields a shorter LFSR (of length $k < l$) that generates exactly the same sequence. Then our method doesn't determine the coefficients of the original register but nevertheless correctly predicts the sequence.
- If the bits the attacker knows aren't just the first ones but $2l$ contiguous ones at a later position, then the theorem yields only the prediction of the following bits. In the main case of an invertible state matrix U the LFSR is completely known and may be run backwards to get the previous bits. For a singular state matrix we achieve the same effect using the shorter LFSR constructed above.
- The situation where $2l$ bits of the key stream are known but at non-contiguous positions is slightly more involved. We get linear relations that contain additional (unknown) intermediate bits. If m is the number of these then we get $l + m$ linear equations for $l + m$ unknown bits.
- What if the length l of the LFSR is unknown? Exhaustively trying all values $l = 1, 2, 3, \ldots$ is nasty but feasible. A better approach is provided by the Berlekamp-Massey algorithm, see Section 3.3 that is efficient also without knowledge of l.