Stream Ciphers

A stream cipher enciphers individual characters, usually bits, of a plaintext message one at a time, with a cipher that varies with time. Block ciphers are *memoryless*, in the sense that the same function is used to encipher successive blocks. Stream ciphers, in contrast, must have memory. As such they are *state functions* because the current state S_i of the function is recorded in a memory buffer. We saw how various modes of operation (CFB, OFB) turn a memoryless block cipher into a state function by feedback buffer. A *keystream* is a sequence of characters generated from the key and the current state, as input to the stream cipher.

In a **synchronous stream ciphers** the keystream is generated independently of the plaintext message (or ciphertext). Given a key K, if the initial state is designated S_0, then, for each cycle $i = 0, 1, 2, \ldots$, the following equations describe the generation of the keystream, ciphertext, and next state:

$$\begin{aligned} \textit{Key stream function:} \quad & k_i = g_K(S_i) \\ \textit{Output function:} \quad & c_i = h(k_i, m_i) \\ \textit{Next state function:} \quad & S_{i+1} = f_K(S_i) \end{aligned}$$

An additive binary stream cipher is defined to be a synchronous stream cipher in which $h = \texttt{XOR}$.

Exercise. Identify each of these functions for the OFB mode of operation.

A **self-synchronizing** or **asynchronous** stream cipher is a stream cipher in which the keystream is a function of the key and a fixed number of previous ciphertext characters. Given a key K and initial state $S_0 = (c_{-t}, \ldots, c_{-1})$, the keystream and ciphertext are generated for each cycle $i = 0, 1, 2, \ldots$ as for a synchronous stream cipher:

$$\begin{aligned} \textit{Key stream function:} \quad & k_i = g_K(S_i) \\ \textit{Output function:} \quad & c_i = h(k_i, m_i) \end{aligned}$$

with the next state set to $S_{i+1} = (c_{i+1-t}, \ldots, c_i)$.

Exercise. Identify the components of a self-synchronizing stream cipher defined by a block cipher in CBC and 1-bit CFB modes.

Properties of Stream Ciphers

Synchronous stream ciphers.

- Synchronization: Sender and receiver are required to be synchronized in terms of both state and key.
- Error propagation: None — a bit error in the ciphertext affects precisely one bit in the deciphered plaintext, provided that synchronization is maintained.

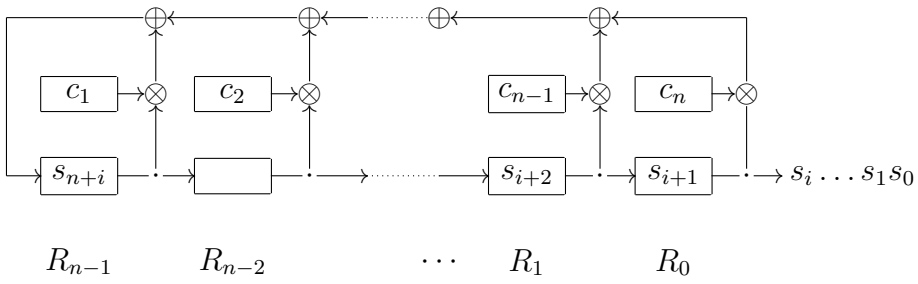
- Attacks and features: Property (1) means that an active adversary can use insertion, deletion, or replay of ciphertext; property (2) implies that the affects of these changes effect direct changes on the deciphered plaintext, which might be exploited.

Asynchronous stream ciphers.

- Synchronization: An insertion, deletion, or change in ciphertext characters results in loss of only a fixed number of deciphered plaintext characters, after which the deciphering self-synchronizes.
- Error propagation: A ciphertext error in transmission affects at most t characters of the deciphered plaintext.
- Attacks and features: The error propagation makes active modification more easily detected, while self-synchronization makes insertion, deletion, or reply of ciphertext blocks more difficult to detect. Since each plaintext character influences subsequent ciphertext, an asynchronous stream cipher is better at masking plaintext structure or redundancies.

Linear Feedback Shift Registers

A linear feedback shift register implements a keystream function, and which can be simply described by a schematic diagram of the following form:



Before discussing the mathematical definition of linear feedback shift registers (LFSR's), we address the question "Why?". A LFSR is essentially an elementary algorithm for generating a keystream, which has the following desirable properties:

1. Easy to implement in hardware.

2. Produce sequences of long period.

3. Produce sequences with good statistical properties.

4. Can be readily analyzed using algebraic techniques.

A LFSR is defined by n *stages*, labelled $R_{n-1}, \ldots, R_1, R_0$, each storing one bit, and having one input and output, and a timer which mark clock cycles $i = 0, 1, 2, \ldots$. At the i-th clock cycle:

1. The contents of stage 0 is output;

2. The contents of R_i moves to R_{i-1}, for $1 \leq i \leq n-1$; and

3. Stage R_{n-1} is the bit sum of a prescribed subset of stages $0, 1, \ldots, n-2$.

We may denote the contents of stage R_j at time i by s_{i+j}, and the algorithm for updating the contents of stage R_{n-1} gives a recurrence relation

$$s_{n+i} = \sum_{j=0}^{n-1} c_{n-j} s_{i+j},$$

where c_j, $1 \leq j \leq n$ are fixed bit constants specifying the stages which contribute to the bit sum. By setting $c_0 = 1$ we can express the relation as $\sum_{j=0}^{n} c_{n-j} s_{i+j} = 0$.

We identify the constants c_k with coefficients of a polynomial

$$g(x) = \sum_{k=0}^{n} c_k x^k,$$

which we call the *connection polynomial* of the LFSR. Moreover, if can take the LFSR output bits s_j as the coefficients of a power series

$$s(x) = \sum_{j=0}^{\infty} s_j x^j,$$

then the recurrence relation expresses the fact that $s(x)g(x)$ is a polynomial $f(x)$ of degree less than n. In other words, the power series takes the form $s(x) = f(x)/g(x)$.

Exercise. Verify that the equality $f(x) = s(x)g(x)$, for $f(x)$ a polynomial of degree less than n, gives rise to the the stated recurrence for the coefficients of $s(x)$.

The LFSR is said to be *nonsingular* if $c_n \neq 0$. It should be clear that the condition $c_n = \cdots c_n - k = 0$ describes a LFSR in which the feedback reduces to at most $n-k$ terms, hence after the initial k bits are output, reduces to a sequence which can be modelled by a LFSR of length $n-k$. For this reason we hereafter assume that the LFSR is nonsingular.

We note that since the next state of the shift register (i.e. the contents of the collection of stages) depends only on the current contents, and there are 2^n possible states, it is clear that the output sequence is eventually periodic. Since the all zero initial state maps to itself, it is clear that the maximal period for any LFSR of length n is $2^n - 1$. The connection polynomial is said to be *primitive* if the period of the LFSR output sequence, beginning at any nonzero state, is $2^n - 1$.

We note that the output sequence has period N if and only if $(X^N + 1)s(x)$ is a polynomial of degree at most $N - 1$. On the other hand, since $s(x) = f(x)/g(x)$, if $f(x)$ and $g(x)$ have no common factor, then it follows by the unique factorization of polynomials that $g(x)$ divides $X^N + 1$. In particular, if $g(x)$ is *irreducible*, since $\deg(f(x)) < \deg(g(x))$,

it follows that $f(x)$ and $g(x)$ have no common factors. In summary, an irreducible connection polynomial of a LFSR must divide $x^N + 1$ where N is the period of any nonzero output sequence.

The theorem below shows that in fact every polynomial $g(x)$ in $\mathbb{F}_2[x]$ with nonzero constant term must divides $X^N + 1$ for some N. The special feature of irreducible connection polynomials, and especially primitive polynomials, is that we will be able to compute the value of N and, for primitive polynomials, that it is takes the the maximal possible value.

Lemma 2 *If $g(x)$ is not divisible by x, then there exists a polynomial $u(x)$ such that $x\,u(x) \bmod g(x) = 1$.*

Proof. Since the constant term of $g(x)$ is 1, there is a polynomial $u(x)$ such that $x\,u(x) = g(x) + 1$, from which the lemma follows. □

Theorem 3 *Every polynomial $g(x)$ in $\mathbb{F}_2[x]$ coprime to x divides $x^N + 1$ for some N.*

Proof. Consider the sequence of remainders $\bmod g(x)$:

$$1 \bmod g(x),\ x \bmod g(x),\ x^2 \bmod g(x),\ x^3 \bmod g(x), \ldots$$

Since every remainder is a unique polynomial of degree at most $n-1$, there are at most 2^n distinct elements in this sequence. It follows that there is some N such that $x^i \bmod g(x)$ equals $x^{N+i} \bmod g(x)$ for all sufficiently large i. Since $g(x)$ is not divisible by x, it follows from the previous lemma that we can cancel the powers of x^i to obtain $x^N \bmod g(x) = 1$. We conclude that $x^N + 1$ is divisible by $g(x)$. □

Periods of LFSR's

We begin with some theorems regarding LFSR's and their connection polynomials. First, we make or recall some standard definitions. We define a polynomial $g(x)$ to be *irreducible* if the only factorization $g(x) = h(x)k(x)$ is with $h(x)$ or $k(x)$ equal to the constant polynomial. We define the order of x modulo $g(x)$ to be the smallest power x^N of x such that $x^N \bmod g(x)$ equals 1. A polynomial $g(x)$ is of degree n is said to be *primitive* if the order of x modulo $g(x)$ is $2^n - 1$. The next theorem shows that the definition of primitive given in the previous lecture agrees with the current one.

Theorem 4 *The period of a sequence generated by a LFSR is independent of the nonzero initial state if the connection polynomial is irreducible, and the period takes the maximal value $2^n - 1$ if and only if the connection polynomial is primitive.*

Since there are exactly $2^n - 1$ possible nonzero states, it is clear that a LFSR that produces an output sequence with this period in fact cycles through all such states, so the

period is independent of the initial state. As a consequence of the theorem, a primitive polynomial is irreducible. We now prove the theorem.

Proof. We first note that all possible $2^n - 1$ output sequences are given by the rational expressions $s(x) = f(x)/g(x)$, where $f(x)$ runs through the all nonzero polynomials of degree less than the connection polynomial $g(x)$. If the $g(x)$ is irreducible, then this expression is minimal — $f(x)$ and $g(x)$ have no common factors, so there is no cancellation.

Next we note that the minimal period N of any power series $s(x)$ is the degree of the smallest $x^N + 1$ for which $(x^N + 1)s(x)$ is a polynomial. If also $s(x) = f(x)/g(x)$ is in minimal form, then for any such $x^N + 1$, the denominator $g(x)$ divides $x^N + 1$. If the connection polynomial $g(x)$ is irreducible, then the denominator equals $g(x)$ independently of $s(x)$ and the initial state which defines it, and so also is the period constant.

Finally the last statement follows by noting that $g(x)$ divides $x^N + 1$ if and only if $x^N \bmod g(x) = 1$. □

For cryptographic purposes, it is desirable to have sequences which have very long period. The advantage of LFSR's in this respect is that the period grows exponentially in the length of the shift register. For small value of n, 2, 3, or 4, the value of the maximal possible period, $N = 2^n - 1$, is still trivially small. But as the table below shows, with modest values of n we are able to efficiently generate sequences with enormous period.

n	$2^n - 1$	n	$2^n - 1$	n	$2^n - 1$
1	1	11	2047	21	2097151
2	3	12	4095	22	4194303
3	7	13	8191	23	8388607
4	15	14	16383	24	16777215
5	31	15	32767	25	33554431
6	63	16	65535	26	67108863
7	127	17	131071	27	134217727
8	255	18	262143	28	268435455
9	511	19	524287	29	536870911
10	1023	20	1048575	30	1073741823

As particular examples, the primitive trinomials such as $x^{23}+x^5+1$, $x^{29}+x^2+1$, $x^{31}+x^3+1$, and $x^{41}+x^3+1$ define very efficiently computable recurrence relations for maximal length LFSR's.

Linear Complexity

In addition to practical applications for generating pseudo-random sequences, LFSR's are a useful theoretical tool for the characterization of other binary sequences. The *linear complexity* of a binary sequence is a measure of the structure of the sequence – a low linear complexity implies a cryptographically weak sequence.

An infinite sequence $s = s_0, s_1, \ldots$ is said to be generated by a LFSR if it is the output sequence of the shift register for some initial state. The linear complexity $L(s)$

for an infinite sequence s is defined to be 0 if s is the all zero sequence, equal to the minimum length of a LFSR which generates it if s is periodic, and equal to ∞ otherwise. The linear complexity $L(s)$ of a finite sequence $s = s_0, s_1, \ldots, s_{n-1}$ is defined to be the minimum length of a shift register which generates some sequence with initial segment s. The *linear complexity profile* of an infinite sequence s is the sequence $L_1(s), L_2(s), \ldots$, where $L_i(s)$ is the linear complexity of first i terms of s.