Cryptography
Lecture 3

Pseudorandom generators
LFSRs

Remember

- One Time Pad is ideal
- With OTP you need the same transmission capacity via an already secure channel for the key as you can then secure via this key. Thus it is an extremely unpractical cipher.
- It can be used when a secure channel is available for a short time, and data to transmit exist later.

Emulating OTP

- If you cannot use a true random sequence of sufficient length, try to use a pseudorandom sequence created from a shorter key
- The generator is deterministic and finite, so the sequence is periodic
- Every single character should be as “unpredictable” as possible

Pseudorandom sequences

- A well-known generator is $x_i = ax_{i-1} \bmod p$
- With a=3 for example, $x_0=1$ and p = 7 we get

  1, 3, 2, 6, 4, 5, 1, 3, 2, 6, 4…..

  and with a=2 we get

  1, 2, 4, 1, 2, 4
- If p is prime we can get the maximum period p-1
- Values for a that give this period are called generators
- Two known values will reveal all the rest….

Sequence properties

- Part of the sequence is sure to be known to the cryptanalyst
- Every single bit in the unknown part should be as hard to guess as possible for all that do not know the key
- Since the sequence is periodic, the period must then be as long as possible, since the known part repeats after the period length

Binary sequences

- Since this is all about IT systems, we normally have and want binary values
- Studying binary sequences does not limit achievable results
- If nothing else is said in what follows, values are binary and addition is taken as addition modulo 2
- 1+1 mod 2= 0 Thus -1 mod 2 = +1

Linear Feedback Shift Registers

- LFSRs are known to produce (binary) sequences with good pseudorandom properties
- LFSRs can be thoroughly analysed and predicted with the algebra of extended Galois fields, GF(p^n)
- So let us look at LFSRs!

Terminology

- In many subjects, there is a strict accepted standard of what letters to use for what concepts, like v for velocity, x for unknowns, i for indices etc.
- In the subject of LFSR analysis, there is no such standard usage.
- So get used to from the start that we can use G here for what is called C there etc.

Extended Galois fields

- We use tuples of n elements. (s_0, s_1,…s_{n-1})
- These are *not* vectors
- How then can we do calculations?
- Regard s_i as the coefficient for α in a polynomial $s_0 \alpha^{n-1} + s_1 \alpha^{n-2} +,\ldots + s_{n-1} \alpha^0$
- Define α as the root of another polynomial $\Pi(\alpha) = \alpha^n + c_1 \alpha^{n-1} + \ldots + c_n = 0$
- Do everything modulo $\Pi(\alpha)$!

How to create a sequence

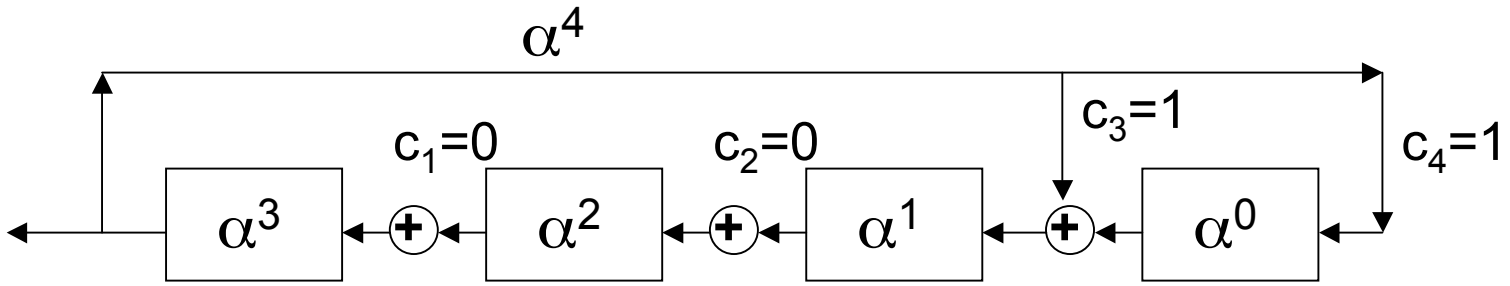
- Start with a tuple of length n, $\mathbf{x}_0$, and a good $\Pi(\alpha)$ ("good" is defined later)
- Look at the tuple as coefficients in a polynomial in α
- Let $\mathbf{x}_i = \alpha\mathbf{x}_{i-1}$ modulo $\Pi(\alpha)$
- I. e. regard α as the root of $\Pi(\alpha)$, so that $\Pi(\alpha) = 0$ and you can express α^n in lower powers of α

Example, p=2, n=4, $\Pi(\alpha)=\alpha^4+\alpha+1$

4-tuple	Polynomial	4-tuple	Polynomial
0001	1	0101	α^2+1
0010	α	1010	$\alpha^3+\alpha$
0100	α^2	0111	$\alpha^2+\alpha+1$
1000	α^3	1110	$\alpha^3+\alpha^2+\alpha$
0011	$\alpha+1$	1111	$\alpha^3+\alpha^2+\alpha+1$
0110	$\alpha^2+\alpha$	1101	$\alpha^3+\alpha^2+1$
1100	$\alpha^3+\alpha^2$	1001	α^3+1
1011	$\alpha^3+\alpha+1$	0001	1

LFSRs

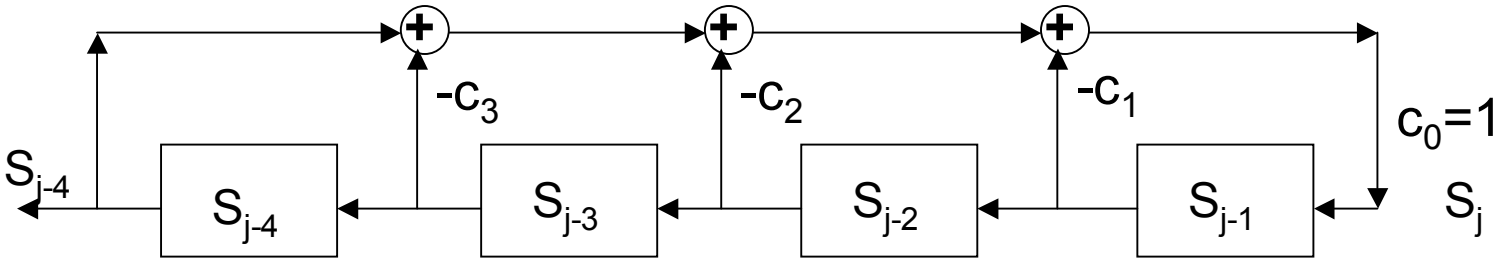
- For those familiar with LFSRs, the operations on the bit level on the previous slide are well known: Clocking an LFSR!



Linear Feedback Shift register, Galois model

Fibonacci versus Galois models

- There is an alternative way of creating LFSRs, the Fibonacci model
- As will be shown later they are equivalent



Linear Feedback Shift register, Fibonacci model

The shift register equation

From the figure we see that the output, s_i is

- $s_j = (-c_n s_{j-n}) + (-c_{n-1} s_{j-n+1}) + \ldots + (-c_1 s_{j-1})$ for any $j \geq n$
- Use the z transform!(If the signal value of the sequence S at time i is s_i, the z transform is $s_0 + s_1 z^{-1} + s_2 z^{-2} + \ldots\ldots$)
- Write the feedback as a polynomial in z^{-1}
- Call that polynomial $C(z) = 1 + c_1 z^{-1+} \ldots + c_n z^{-n}$

Theorem to remember

- **All sequences generated by LFSRs can be written as $S(z^{-1})=P(z^{-1})/C(z^{-1})$**

(Since it is of no importance what we call the unknown, we usually use x instead of z^{-1})

- **All sequences that can be written as $S(z^{-1})=P(z^{-1})/C(z^{-1})$ can be generated by a LFSR**
- Degree of P < degree of C

Proof of the theorem

- The proof is given in the Course Notes

Fibonacci versus Galois

- From the shift register equation and the proof of the theorem, it is easy to see that if you reverse the order of the feedback coefficients in one model, you get the feedback coefficients in the other
- The coefficients in the polynomial P are simply the values stored in the LFSR when you start generating a sequence in the Galois model

Period lengths

- If you pass every possible state of the shift register before you get back to a previous state, you obviously have the longest possible period
- Since we cannot use the all 0 state, the number of states, and thus output symbols in a period, is p^n-1 (2^n-1 for binary sequences)
- For all lengths there are feedback polynomials giving this period length (field generators)
- They are called ***primitive*** or maximum length polynomials

Primitive, prime and non-prime

- The period for a sequence generated by
  - A primitive polynomial is p^n-1
  - A prime but not primitive polynomial is a factor of p^n-1
  - A polynomial, which is the product of several polynomials of lower degree (a non-prime polynomial), varies with the starting state

More on period lengths

- A prime, but not primitive, polynomial is a power of a primitive polynomial, and that power divides the period. Thus its period picks out evenly spaced states in the longer period
- A composite polynomial $G(z)=A(z)B(z)$ can get a starting state $P(z)=A(z)C(z)$. Then, since $S(z)=P(z)/G(z)=C(z)/B(z)$, the period is of course at most that of $B(z)$, and B has lower degree than G
- A composite polynomial can thus have periods that are the least common multiplier of the periods for all combinations of its prime factors

Finding the period length

- The period of a polynomial is defined as the lowest N for which the polynomial G(z) divides $1-z^{-N}$
- This is also the period of the sequence generated by the polynom ial (see Course Notes for proof)
- Thus there is a Q(z) with degree less than N such that $G(z)Q(z)=1-z^{-N}$ and $1/G(z)=Q(z)+z^{-N}/G(z)$
- So if you simply divide 1 by G(z), the first time the remainder contains just one term, z^{-x}, that value of x is the period!

Choosing a good polynomial

- Find a primitive polynomial of suitable length from table!
- If you cannot choose, test for factors and also for any short basic period through division!

But do not use simple LFSRs!

- The key should be at least the starting state, thus n bits for a length of n in the LFSR
- From the shift register equation, we can see that with knowledge of 2n consecutive bits, we get n equations and can solve for unknown previous states and polynomial coefficients!
- To prevent this we must send at least n new key bits for every 2n encrypted bits!