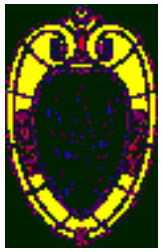

CS/ECE/Math 435
Introduction to Cryptography

Professor Chris DeMarco
Department of Electrical & Computer Engineering
University of Wisconsin-Madison

Spring Semester 2021

Return to Berlekamp-Massey (B-M) Algorithm

Context: An unauthorized decryptor (an "attacker") is assumed able to "snoop" sequential bits of a pseudo-random sequence. Because we'll treat the x bits as sequential in time, label as

---

$$x[0],\ x[1],\ x[2] \ldots\ x[n]$$

↑ first observed x at position 0

↑ present time position "n"

[Bach notes use terminology of a "position" in the x sequence; I'll say "time position"]

Goal: At any time point n, attacker wishes to have a LFSR of order $r \leq n$ such that the LFSR produces the sequence $x[0], x[1] \ldots x[n]$.

Note that as a new $x[n]$ is observed, the old LFSR of order $r \leq n-1$ may still "work", or the attacker may need to update the LFSR coefficients and possibly increase the order.

The perspective of an LFSR is that of "looking forward in time", to compute a new x.

The perspective of the B-M algorithm is that of "looking backward in time" to check whether or not previous x's have a "discrepancy with" the newly observed $x[n]$ (i.e. if the previously estimated LFSR "works" to produce $x[n]$ $\Leftrightarrow$ "no discrepancy"; if the LFSR fails to produce the new $x[n]$, "discrepancy".

We'll want a different organization of the LFSR update equation to make it more convenient to "look backward" in time.

Recall

LFSR update rule: (Bach notes, e.g. p. 20-1)

$$c_0 \cdot x_0 + \ldots + c_{n-2} x_{n-2} + c_{n-1} \cdot x_{n-1} = x_n$$

VERSUS

Alternate form of recurrance relation (Bach notes p. 22-1):

$$x_n + b_1 \cdot x_{n-1} + b_2 \cdot x_{n-2} + \ldots + b_k \cdot x_{n-k} = 0$$

where $c_{n-1} = b_1$, $c_{n-2} = b_2$ ...

CAUTION ON NOTATION IN BACH

Bach notes do not use new coefficient notation of b's, as done here. Instead, Bach notes again use c's for coefficients in the recurrance relations. c's on p. 22-1 of Bach are not the same as c's on p. 20-1!

In lecture, we'll use b's to emphasize different form of recurrance relation (but b's directly define c's provided $k \geq n$.)

Role of Connection Polynomial, and its relation to recurrance relation.

In slight abuse of notation, let Z denote time delay operator, i.e. in bit sequence $x[0] \ldots x[n]$

$$Z\,x[n] = x[n-1]$$

$$Z^2\,x[n] = x[n-2]$$

$$\vdots$$

$$Z^k\,x[n] = x[n-k]$$

Then, associate connection polynomial with recurrance relation

$$x[n] + b_1 x[n-1] + b_2 x[n-2] + \ldots b_k x[n-k]$$

$$\Leftrightarrow 1 \cdot x[n] + b_1 z \cdot x[n] + b_2 z^2 \cdot x[n-2] + \ldots + b_k z^k x[n]$$

$$\Leftrightarrow \underbrace{\left(1 + b_1 z + b_2 z^2 + \ldots + b_k z^k\right)}_{\text{this is connection polynomial, } f \text{ on p. 22-1 of Bach}} \cdot x[n]$$

(again, don't be confused by re-use of notation; f was previously the characteristic poly).

Knowing that we can always recover the c's of an LFSR from the b's of a recurrance relation or connection polynomial, the B-K algorithm focuses on connection polynomial.

Definition : Given an observed sequence $x[0], \ldots x[n]$, a connection polynomial $f(z)$ of length k is said to be 'correct' at time position n

i) $f(z)\, x[n] = 0$

OR

(ii) (degenerate case) $k > n$

Observation

i) is equivalent to having an LFSR to produces the X sequence,

ii) is degenerate, because if the number of observed x's is smaller than the order of the LFSR, we could always "back fill" x's prior to $x[0]$ to make the LFSR work.

A connection polynomial is said to be correct before time position n if it is correct for time positions $0, 1, 2, \cdots n-1$.

Splicing Theorem

Suppose $f(z)$ is correct before time position n, but incorrect at n. And for some $m \leq n$, suppose $g(z)$ is correct before time position m, but incorrect at m.

Construct $h(z) = f(z) + z^{n-m} g(z)$

Then $h(z)$ is correct for all time positions $\leq n$.

Illustration: Suppose $n = 5$, $m = 3$, and f has length $F = 4$, g has length $G = 2$,

$$f(z) = 1 + b_1 z + b_2 z^2 + b_3 z^3 + b_4 z^4$$

$$g(z) = 1 + a_1 z + a_2 z^2$$

$f(z) \times [n]$ possibilities $\begin{cases} 1 & \text{at } n = 5 \\ 0 & \text{at } n = 4 \\ \text{or} & \\ n \leq 4 & \end{cases}$

$g(z) \times [m]$ possibilities $\begin{cases} 1 & \text{at } m = 3 \\ 0 & \text{at } m = 2 \\ \text{or} & \\ m < 2 & \end{cases}$

Now, suppose we consider time position 5; from given conditions

$$f(z) \times [5] \neq 0 \Rightarrow f(z) \times [5] = 1$$

(while $f(z) \times [4] = 0$)

$$g(z) \times [3] = 1$$

(while $g(z) \times [2] = 0$)

But we construct $h(z) \times [5]$

$$= \left[f(z) + z^{5-3} \cdot g(z) \right] \times [5]$$

$$= \underbrace{f(z) \times [5]}_{= 1} + \underbrace{g(z) \cdot z^2 \cdot x[5]}_{g(z) \times [3]}$$

$$= 1 + 1 = 0 \; !!$$

So indeed, $h(z)$ is correct at time position 5.

And $h(z) \times [4]$

$= \left(f(z) + g(z)\, z^2\right) \times [4]$

$= \underbrace{f(z) \times [4]}_{0} + \underbrace{g(z) \times [2]}_{0} = 0$

So $h(z)$ is also correct at time position 4. (and indeed, will be correct for all earlier time positions).

KEY IDEA: Splicing provides a method to combine two connection polynomials that are correct for "shorter histories" to obtain a new connection polynomial that is correct over a longer history.

Algorithm to follow keeps a running tally of two connection polynomials, $f(z)$, $g(z)$. At each iteration, a new observation of an X value is fetched, and $f(z)$ is updated so it is correct up to that new observation.

B-M algorithm

Initialize

No x's yet observed $\Rightarrow n = -1$, $m = -1$

(roughly speaking, both our connection polynomials are in a "-1" state, before we observe $x[0]$).

Set polynomials: $f = 1$ (no b_k's yet identified)

length $F = 0$

$g = 1$ (no a_k's yet identified)

length $G = 0$

To indicate that no polynomial h is yet constructed, set length $H = -1$.

LOOP

REPEAT (for each new observation of an X bit):

$n \leftarrow n+1$; Fetch new observation $X[n]$

TEST: Is $f(z)$ correct at time position n?

IF YES, no change to any other quantity (in particular, $f(z)$ and $g(z)$ unchanged, and coefficients of $f(z)$ remain our correct estimate of the LFSR that produces $X[0], \ldots, X[n]$)

IF NO ($f(z)$ is not correct at n)

UPDATES:

$$h(z) = f(z) + z^{(n-m)} \cdot g(z)$$

length H = max (H, $G + n - m$)

if ($H > F$) { indicates that length of $f(z)$ must increase }

$g(z) = f(z)$ { update g with old f polynomial }

length G = length F

$m = n$

end if

$f(z) = h(z)$ } { update f with new h polynomial }

$F = H$ }

RETURN TO START OF REPEAT LOOP

B-M algorithm provides a very efficient plaintext attack on LFSR-based stream ciphers

(Roughly: LFSR-based stream ciphers very secure against ciphertext-only attacks, but offer poor security against plaintext attacks).

Strength against ciphertext only attacks maintained by any pseudo-random sequence. Weakness against plaintext attacks is associated with the linear equations of the LFSR.

FIX: Keep basic structure of a shift register approach, but introduce nonlinearity into update relation. Desirable characteristic of nonlinearity: identification of its functional inverse should be difficult.