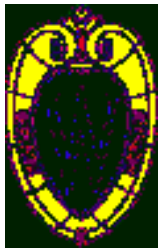

CS/ECE/Math 435
Introduction to Cryptography

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

Spring Semester 2021

Plaintext attacks on LFSR-based Stream Ciphers.

A key question will be what length of corresponding plaintext/ciphertext is required; denote this length "m." The attacker is assumed to know

Plaintext bits	$w_0\ w_1\ w_2 \cdots w_{m-1}$
Ciphertext bits	$y_0\ y_1\ y_2 \cdots y_{m-1}$

THEREFORE, ATTACKER HAS $\Rightarrow$ $x_0\ x_1\ x_2 \cdots x_{m-1}$ $\leftarrow$ bits of the LFSR generated pseudo-random sequence

Note that starting index/subscript of "0" on prior page is arbitrary; starting index of plaintext/ciphertext known to attacker need not coincide with "true" starting index of x_0 in LFSR.

Key idea is simple - just re-arrange LFSR update equations, treating x's as the <u>known</u> quantities, c's as unknowns to be solved:

Structure of " x's known, c's unknown " equations:

$$x_0 \cdot c_0 + x_1 \cdot c_1 + \cdots x_{n-1} \cdot c_{n-1} = x_n$$

$$x_1 \cdot c_1 + x_2 \cdot c_2 + \cdots x_n \cdot c_{n-1} = x_{n+1}$$

$$\vdots$$

$$x_{n-1} \cdot c_1 + x_n \cdot c_2 + \cdots x_{2n-1} \cdot c_{n-1} = x_{n+n}$$

$$\Rightarrow \underbrace{\begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ x_1 & x_2 & & x_n \\ \vdots & & & \\ x_{n-1} & x_n & \cdots & x_{2n-1} \end{bmatrix}}_{\text{denote this matrix } [X]} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} x_n \\ x_{n+1} \\ \vdots \\ x_{2n-1} \end{bmatrix}$$

So we first obtain a square matrix of known x's when we have observed x_0 through $2n-1$ $\Rightarrow$ attacker must have length of known plaintext/ciphertext of at least

$m = 2n.$

But... LFSR produces pseudo-random bits as x_i's. Statistics of bits produced by LFSR do a good job "mimicking" truly random bit (for example, as produced by sequence of fair coin tosses, heads = 1, tails = 0).

Q: How do we know $[X]$ is invertible in $\mathbb{Z}_2$? (i.e. $\det(X) = 1$).

Ans: Given pseudo-random nature of x_i's we ("usually") don't know that X is invertible, but ... properties of $\mathbb{Z}_2$ help us.

We ask closely related questions:

Q1: Given $X \in \mathbb{Z}_2^{k \times k}$, with uniformly distributed random elements, what is Probability $[\det(X) = 1]$?

Q2: (From previous lecture 23): What fraction of matrices in $\mathbb{Z}_2^{k \times k}$ are invertible? (we'll answer this Q2 in limit as k is "large", $k \to \infty$)

To answer Q2, return to Jordan's formula for number of invertible matrices in $\mathbb{Z}_N$.

Here, using form of Jordan's formula from Back notes p. 8-2, $N = 2$, and here k plays the role of n in formula; we'll let $k \to \infty$. And $p/2 = \{2\}$.

$$\underbrace{2^{k^2}}_{\text{this is total number of } k \times k \text{ matrices in } \mathbb{Z}_2} \times \underbrace{\left[\left(1-\frac{1}{2}\right)\cdot\left(1-\frac{1}{2^2}\right)\left(1-\frac{1}{2^3}\right)\cdots\right]}_{\text{this is fraction of all matrices that are invertible - this is what we want!}}$$

$$\prod_{i \geq 1}^{\pi} \left(1 - 1/2^i\right) = 0.288788\ldots$$

So, about 29% of (large) matrices on $\mathbb{Z}_2$ are invertible. So roughly speaking, we may expect that on average, attack needs to "snoop" $\frac{1}{0.29} \times 2n$ bits of corresponding plaintext/ciphertext, in order to obtain n linearly independent rows in matrix

$$\begin{cases} x_0 \; x_1 \; \cdots \; x_{n-1} \\ \vdots \\ x_{n-1} \; \cdots \; x_{2n-1} \\ x_n \; \cdots \; x_{2n} \\ x_{n+11} \; \cdots \; x_{2n+1} \end{cases}$$

keep observing more rows, until you obtain n linearly independent rows

While we will not prove it here, LFSR's constructed with c's selected from a primitive polynomial have property that $n \times n$ X is always invertible (i.e., no need to observe more than $2n$ bits to obtain invertible X to solve for c's).

Alternate Formulation of Solution for c's from observed x's:

Berlekamp-Massey Algorithm

Idea: Observe previous X^{-1} method "waited" for sufficient number of observed x's to allow solution for correct c's "all in one shot."

What if instead we view c's as vector to be iteratively updated, from (probably incorrect) initial guess $\underline{c}^{(0)}$,

to a "better" $\underline{c}^{(1)}$, ultimately converging to a correct $\underline{c}^{(k)}$.

Obvious questions:

- How to quantify a "better" $\underline{c}$?
- How to update $\underline{c}$'s so that they get better at each update?
- Must this update law converge to correct $\underline{c}^{(k)}$ for a "acceptable" number of iterations, k?

Recall our previous definition of the characteristic polynomial for a LFSR:

$$f(X) = c_0 + c_1 \cdot X^1 + c_2 X^2 + \ldots c_{n-1} X^{n-1} + X^n$$

↑ X a scalar $\in \mathbb{Z}_2$

Here we define a related, possibly lower degree polynomial, the connection polynomial for an LFSR. Back notes use notation "f" again; I'll use "$g_k(z)$" to denote connection polynomial of length k; here $k \leq n$.

$$g_k(z) = 1 + c_1 \cdot z^1 + c_2 z^2 + \dots c_{k-1} z^{k-1} + c_k z^k$$

↑ again, z is polynomial's scalar argument, $z \in \mathbb{Z}_2$

Caution: We say this connection polynomial has length $k \leq n$, not degree k, because it's perfectly possible that $c_k = 0$ (so the z^k term may disappear)

We don't yet have an update law to get better C's. But as a hint of how we may use these connection polynomials, consider a situation in which the attacker observes x^s from index $n-k$ to index n, i.e. attack observes:

$$\underbrace{x_{n-k} \quad x_{n-k+1} \quad x_{n-k+2} \cdots x_n}$$

Given these $(k+1)$ x values observed,

we will define g_k as "correct at position n" if

- $x_n + c_1 x_{n-1} + c_2 \cdot x_{n-2} + \cdots + c_k \cdot x_{n-k} = 0$

  OR

- (degenerate case) $n < k$

Define g_k as "correct before position n" if it is correct at positions $0, 1, 2, \ldots, n-1$.

We'll need one more construct before we can define our algorithm for iterative refinement of $\underline{C}^S$: the construct of "splicing" for connection polynomials.