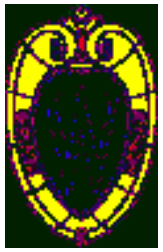


CS/ECE/Math 435
Introduction to Cryptography

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

Spring Semester 2021

Vulnerability of LFSR-based stream ciphers to plaintext attack is associated with the linearity of the LFSR update equation.

Suggests a "fix" in the form of introducing nonlinearity that is "harder to invert" in plaintext attack.

A Nonlinear Shift Register (NSR) device.

Syllabus / Coverage Note: Bach course notes devote considerable time to NSR's in sections 23 & 24.

Coverage here will focus only on multiplexer approach of section 24 (you'll be responsible only for multiplexers on upcoming HW, and on final exam)

Idea: A more complex (harder to invert) NSR algorithm may incorporate one or more LFSR's within the algorithm.

For simplicity of design, one NSR uses LFSR's for the portion of the algorithm with "memory" (i.e. dependence of $x[k]$ on past $x[k-1]$, $x[k-2]$, ...).

Nonlinearity added is memoriless; i.e., it involves only value(s) at "present" time position k

Most common NSR algorithm of this type is termed a multiplexer. It can be constructed with a single higher order LFSR, but more easily visualized as multiple, lower order LFSR's.

Simplest multiplexer algorithm 1 utilizes $(1 + 2^1) = 3$ LFSR's

LFSR "A", label its sequence of bits

$$R = \{ r[0], r[1], r[2] \ldots \}$$

LFSR "0", with sequence

$$S = \{ s[0], s[1], s[2] \ldots \}$$

LFSR "1", with sequence

$$T = \{ t[0], t[1], t[2] \ldots \}$$

Notation (standard in Boolean): for given stream bit, e.g. r_0, $\overline{r_0}$ = "NOT r_0" denotes logical complement

Then, memoriless output nonlinearity produces NSR sequence $X = \{x[0], x[1] \ldots$ as:

$$x[k] = \overline{r[k]} \cdot s[k] + r[k] \cdot t[k]$$

(the multiply operations introduce nonlinearity)

i.e. at time position k, if

$r[k] = 0$, choose NSR output

$$x[k] = s[k]$$

OR

if $r[k] = 1$, choose NSR output

$$x[k] = t[k]$$

LFSR stream "A" is said to be the "address stream," it chooses the address among the "data streams," stream 0 or stream 1.

We labeled this simplest case as NSR 1.

Q: What would constitute generalization to NSR 2?

Ans: 2 address streams, and 2^2 data streams.

Address LFSR's yield P (as most significant bit of address)
R (as least significant bit of address)

Data streams: LFSR 00 yields S
LFSR 01 " T
LFSR 10 " U
LFSR 11 " V

At time position k, two bit address $(p[k]\ r[k])$ selects ("multiplexes")

$$x[k] = \begin{cases} s[k] & \text{for } 00 \\ t[k] & \text{for } 01 \\ u[k] & \text{for } 10 \\ v[k] & \text{for } 11 \end{cases}$$

This multiplex NSR algorithm generalizes easily to m address LFSR's, 2^m data LFSR's.

Note that key information sent from encryption entity to authorized decryption entities will be:

If each LFSR has order n:

$$\text{Key space size} = (m + 2^m) \cdot \underbrace{2n}_{\substack{n \text{ coefficients} \\ + n \text{ initial conditions} \\ \text{for each LFSR}}}$$

(plus a very small amount of data to "label" our address & data streams)

Multiplexer NSR will be specific NSR algorithm treated here.

But key ideas of NSR play important role in the modern Block ciphers on $\mathbb{Z}_2$ to follow.

Modern (post 1970's) block ciphers:

"DES" = Data Encryption Standard

"AES" = Advanced Encryption Standard

Block ciphers on $\mathbb{Z}_2$, with block size m, and length ℓ key (e.g., for Hill, matrix operator, $M \in \mathbb{Z}_2^{m \times m}$.)

However, alternative, more general viewpoint is often helpful:

Block cipher on $\mathbb{Z}_2$ as a Boolean function taking:

$m + \ell$ inputs;

m ↑ plaintext block

ℓ ↑ key

e.g. DES specifics

$m = 64$

$\ell = 56$

and producing

m outputs

↑

ciphertext block.

We take a qualitative insight from our experience with LFSR's versus NSR's: we want to choose our encryption function to be nonlinear ($\Rightarrow$ no simple Hill ciphers or periodic ciphers).

Added qualitative lesson from LFSR/NSR: algorithmic components of one encryption scheme used together to build more secure overall algorithm.

"Modes of Operation" for Block Ciphers

Simplest case, only examined thus far in 435, is known as the "codebook mode"

Plaintext

$x_0, x_1, \ldots x_{m-1} \; ; \; x_m \; x_{m+1} \; \ldots \; x_{2m-1}$

Block cipher
Boolean function

$B_k(\underline{x})$

k denotes the key

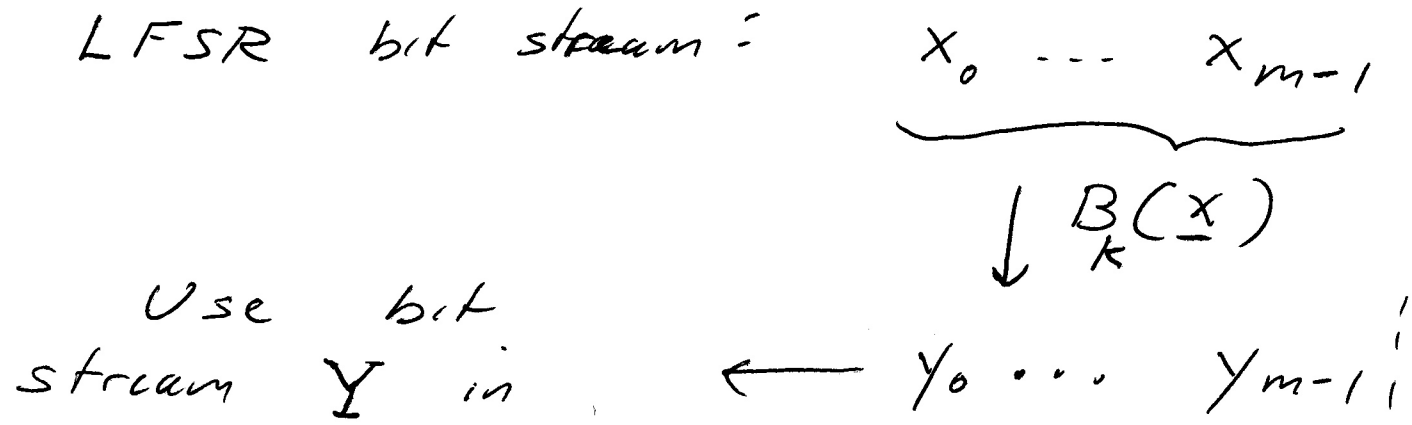
$B_k(\underline{x})$

$y_0, y_1, \ldots y_{m-1} \; ; \; y_m, y_{m+1} \ldots \; y_{2m-1}$

Ciphertext

"Counter - Mode"

This can be thought of as an improvement on our memoriless nonlinearity NSR: here we use the (yet to be specified) nonlinear $B_k(\underline{x})$ as a nonlinearity with memory:



Output Feedback Mode

Use B_k to directly construct a NSR, producing pseudo-random sequence $s_0, s_1, s_2 \ldots$ used in a stream cipher:

s_0:

plaintext $X = x_0, x_1, x_2 \ldots$

ciphertext $Y = y_0, \ldots$

$$y_i = x_i + s_i \quad \text{(stream cipher)}$$

where

$$s_i = f\left(B_k\begin{pmatrix} s_{i-1} \\ s_{i-2} \\ \vdots \\ s_{i-m} \end{pmatrix}\right)$$

where f "selects" the last component of the vector output of $B_k \begin{pmatrix} s_{i-1} \\ \vdots \\ s_{i-m} \end{pmatrix}$

Block Chaining Mode

Notation: Underscore as vector block

Plaintext

$$\underline{X}_1 = \begin{matrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{matrix}$$

$$\underline{X}_2 = \begin{matrix} x_m \\ x_{m+1} \\ \vdots \\ x_{2m-1} \end{matrix}$$

Ciphertext

$$\underline{Y}_0 = \begin{matrix} y_0 \\ \vdots \\ y_{m-1} \end{matrix}$$

$$\underline{Y}_1 = \begin{matrix} y_m \\ \vdots \\ y_{2m-1} \end{matrix}$$

Note: Introduce an arbitrary "priming block"

For $i = 1, 2, 3, \ldots$

$$\underline{Y}_i = B_k(\underline{Y}_{i-1} + \underline{X}_i)$$

Cipher Feedback Mode

For $i = 1, 2, 3, \ldots$

$$\underline{Y}_i = \underline{X}_i + B_k(\underline{Y}_{i-1})$$