Lesson 2021-03-11

March 21, 2021

1 Introduction

1.1 Problems in LFSR or Lehmer PRNG

LFSR or Lehmer PRNG are not cryptographic PRNGS. The problem is the linear property, because if you intercept the s_i you can solve an easy system.

$$\begin{cases} s_1 = as_0 + bmod(m) \\ s_2 = as_1 + bmod(m) \\ s_3 = as_2 + bmod(m) \end{cases} \tag{1}$$

1.2 KPA ATTACK

The attack is the (kpa) known plaintext attack. We assume that the enemy knows in somehow:

$$\begin{cases} ...x_{m+j}...x_j... \\ ...y_{m+j}...y_j... \end{cases} \Leftrightarrow \begin{cases} ...s_{m+j}...s_j... \\ ...y_{m+j}...y_j... \end{cases} \tag{2}$$

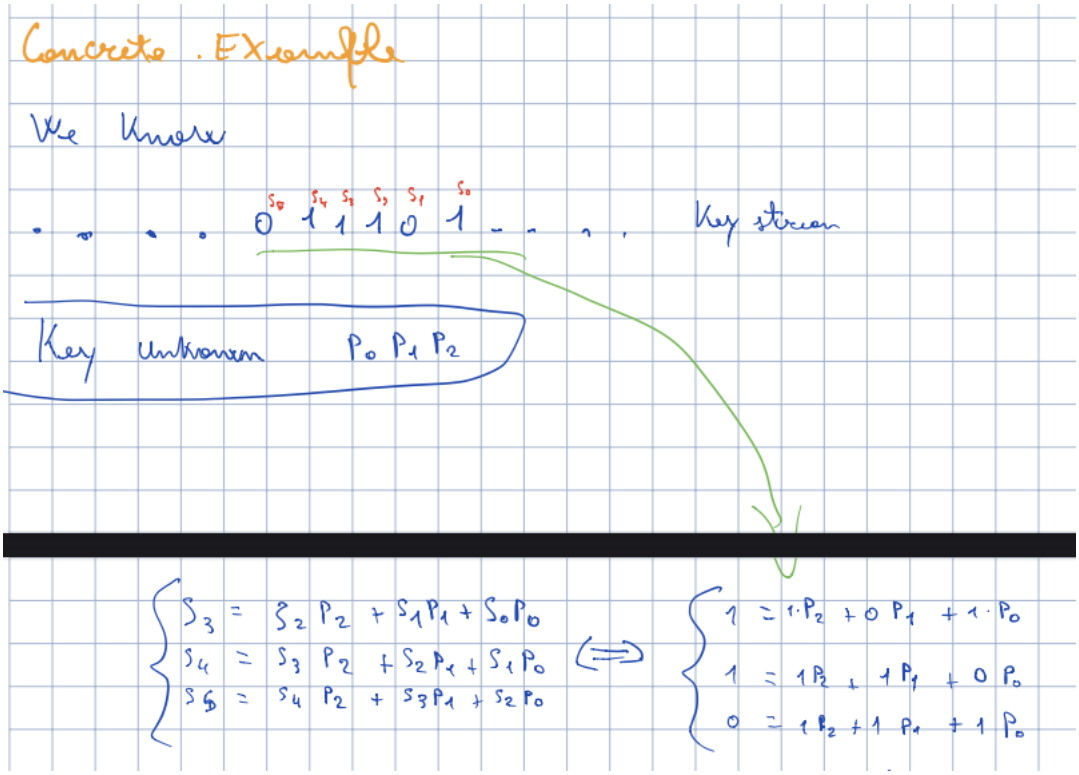
So in a LFSR with m bits and a key $[p_{m-1}....p_0]$ we have :

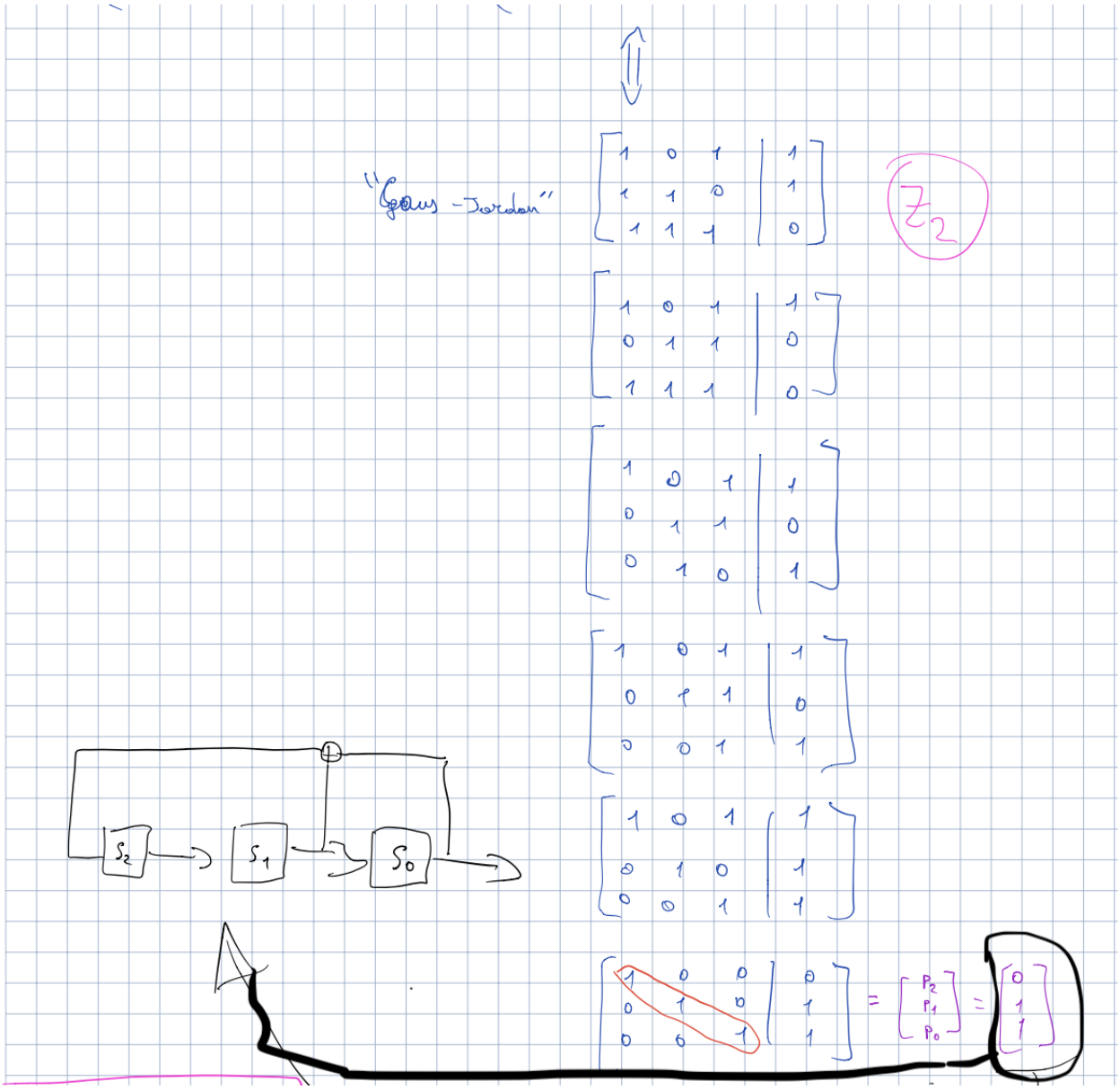
$$s_m = s_{m-1}p_{m-1} + s_{m-2}p_{m-2} + ... + s_1p_1 + s_0p_0 \tag{3}$$

and an enemy needs only 2m bits of key stream to solve the system.

1.2.1 Example

In case of LFSR of m bits in general if you intercept a segment of 2m bits then you obtain m equations with m unkown $p_m....p_0$ and ypu can solve the system with $O(m^2)$


Concrete Example
We know
s_0 s_4 s_3 s_2 s_1 s_0
0 1 1 1 0 1 - - - - Key stream
Key unknown P_0 P_1 P_2
$s_3 = s_2 P_2 + s_1 P_1 + s_0 P_0$
$s_4 = s_3 P_2 + s_2 P_1 + s_1 P_0$
$s_5 = s_4 P_2 + s_3 P_1 + s_2 P_0$
$\Leftrightarrow$
$1 = 1 \cdot P_2 + 0 P_1 + 1 \cdot P_0$
$1 = 1 P_2 + 1 P_1 + 0 P_0$
$0 = 1 P_2 + 1 P_1 + 1 P_0$


"Gauss - Jordan"
$\mathbb{Z}_2$
S_2
S_1
S_0
P_2
P_1
P_0

1.3 Stream Ciphers: Permutation

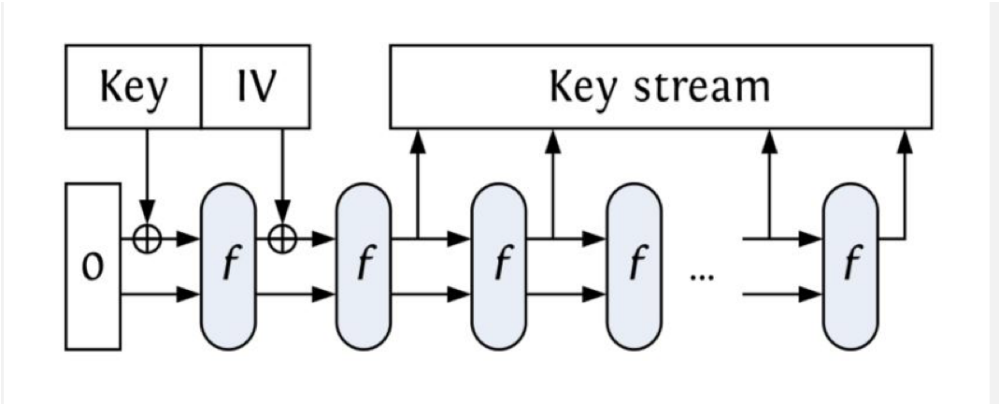
ORACLES. For convenience, a random oracle R is a map from $\{0,1\}^*$ to $\{0,1\}^\infty$ chosen by selecting each bit of $R(x)$ uniformly and independently, for every x. Of course no actual protocol uses an infinitely long output, this just saves us from having to say how long "sufficiently long" is. We denote by 2^∞ the set of all random oracles.

A random oracle is random but preserves the responses. So it has memory:

$$R : \{0,1\}^* \rightarrow \{0,1\}^\infty$$

$R(x) = [b_1.....]$
$R(y) = [c_1.....]$
.
.
$R(x) = [b_1.....]$

1.4 Keccak-Sponge key stream



A sponge function is built from three components:

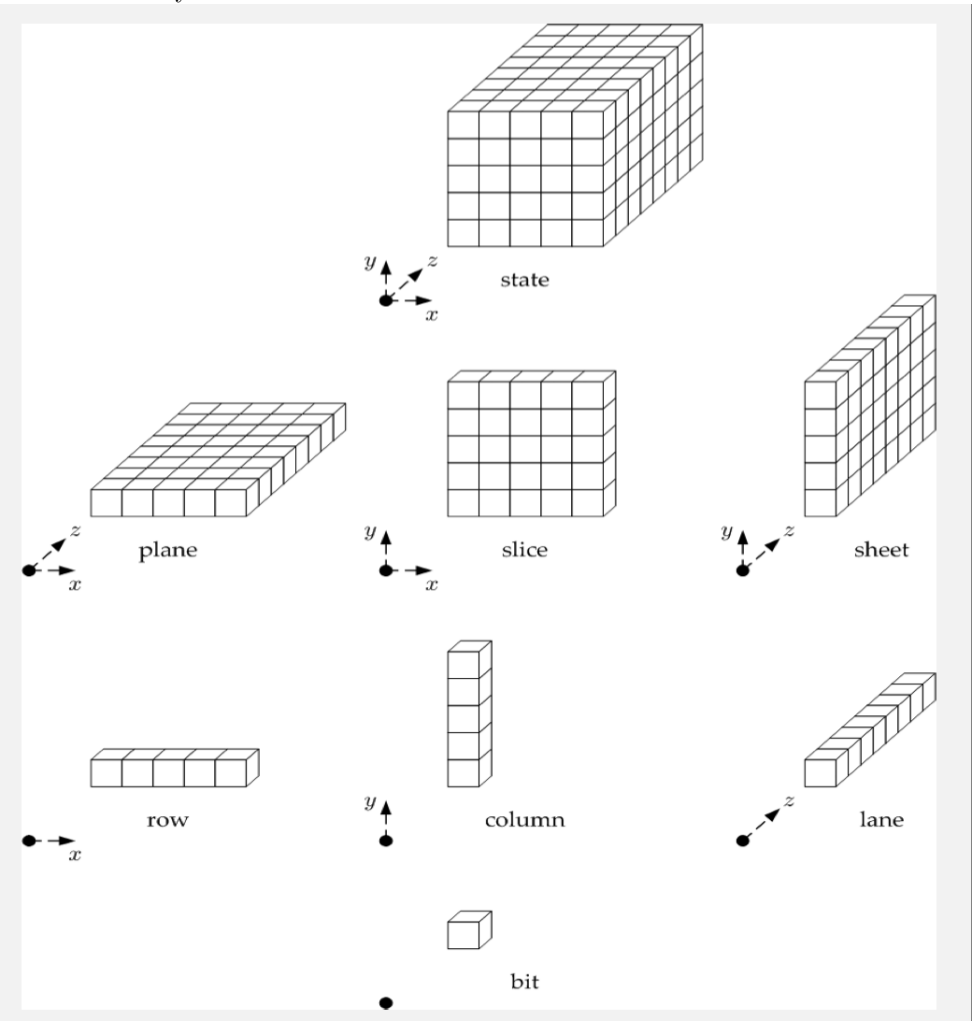
- State, containing bits,
- a function that transforms the state memory (often it is a pseudorandom permutation of the $2^{|bits|}$ state values)
- a padding function Pad.

The State memory is divided into two sections: Bitrate(r) and the remaining part the Capacity(c).

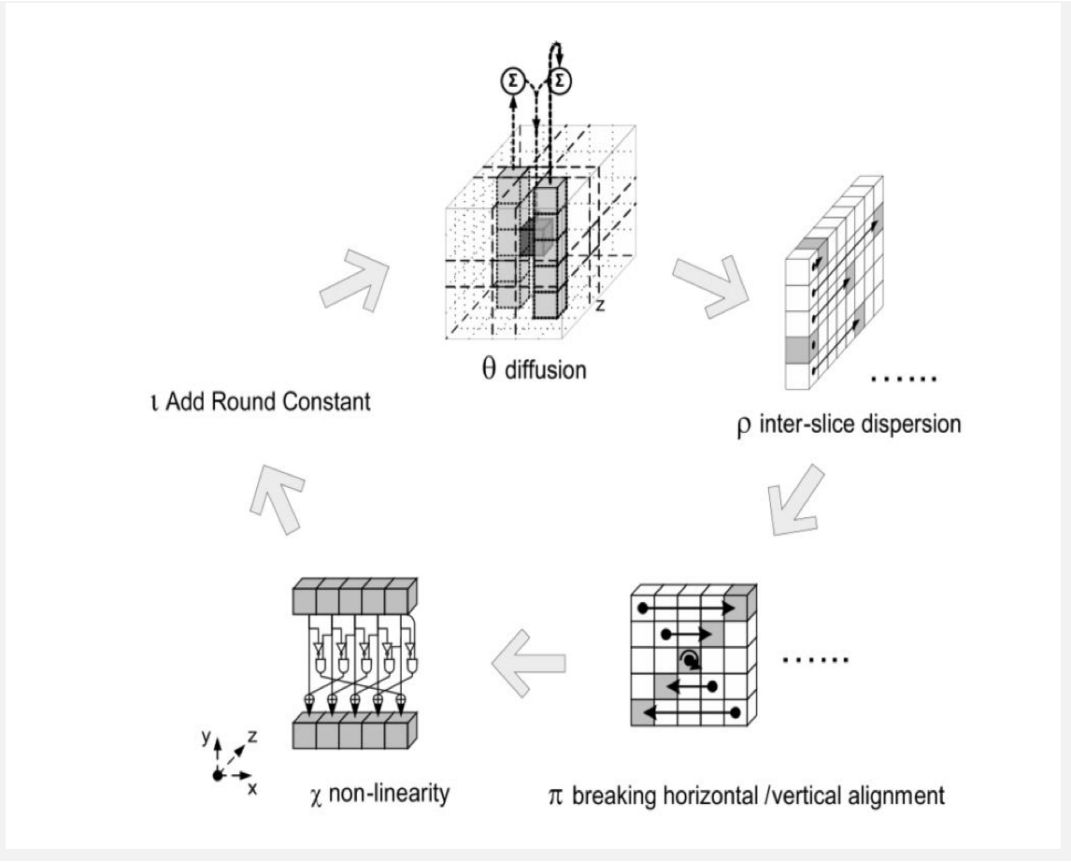
There are two phases one of absorption in which the system uses the key and the iv to modify the current state. then there is the squeeze phase in which the system generates the key stream.

1.5 The Keccak-f permutations

There are 7 Keccak-f permutations of $\{0,1\}^b$, indicated Keccak-f[b] , where b $= 25 \times 2^l$ and l ranges from 0 to 6. The elements of $\{0,1\}^b$ are regarded as a 3-dimensional array



To compute Keccak-f[b](x) there are 12 + 2l rounds, e.g. Keccak-f[25] has 12 rounds, Keccak- f[1600] has 24 rounds. A round consists of 5 invertible steps mappings:


θ diffusion
ρ inter-slice dispersion
π breaking horizontal /vertical alignment
χ non-linearity
ι Add Round Constant