Cyclic Linear Codes:

2016. október 21., péntek 14:25

Encoding $u(x) \rightarrow$ [LFF SR] $\rightarrow c(x)$, $g(x)$

1) $\forall s \in C \rightarrow s' = Ss \in C$
2) $\forall s, s' \in C \rightarrow (\lambda s + \rho s') \in C$

$\Rightarrow \exists g(x):$
- $\deg(g(x)) = n-k$
- $g_{n-k} = 1$
- $\forall c(x): c(x) = u(x)g(x)$
- $g(x) \mid x^n - 1$

Proof: $\exists a(x) \in C$

$\deg(a(x)) = m < \deg(c(x))$

$g(x) := a_m^{-1} a(x) = a_m^{-1}(a_0 + a_1 x + \dots + a_m x^m) = a_0 a_m^{-1} + a_1 a_m^{-1} x + \dots + x^m \in C$

$g(x)$ unique: $\nexists g'(x): (g(x) - g'(x)) \in C$ and $\deg(g(x) - g'(x)) < m$

Cyclic: $g(x);\ x \cdot g(x);\ x^2 \cdot g(x);\ \dots\ x^{n-m-1} g(x) \in C$

Linearity: $u_0 g(x) + u_1 x g(x) + \dots + u_{n-m-1} x^{n-m-1} g(x) = (u_0 + u_1 x + \dots + u_{n-m-1} x^{n-m-1}) g(x) = u(x) g(x)$

$\nexists c(x): c(x) = u(x)g(x) + r(x) \rightarrow \frac{c(x)}{g(x)}$

$u(x)$: divisor, $r(x)$: remainder, $\deg(r(x)) < \deg(g(x)) = m$

$c(x) - u(x)g(x) = r(x)$

$c'(x) = r(x)$

↳ cannot be a codeword

↳ if it is a codeword → contradiction

Lowest codeword degree: m ← because

$m = n-k \quad n-m-1 = k-1$

$c(x) = u(x) g(x)$

$g(x)$ divide without remainder

$u(x) \rightarrow$ [LFF SR] $\rightarrow c(x)$, $g(x)$

RS codes are cyclic: → $c(x)\big|_{x = \lambda^i,\ i = 1 \dots n-k} = 0$ → $c(x) = c_0 + c_1 x + \dots + c_{n-1} x^{n-1}$

$c(\lambda) = 0$
$c(\lambda^2) = 0$
$\vdots$
$c(\lambda^{n-k}) = 0$

$\lambda, \lambda^2, \dots \lambda^{n-k}$ are roots of $c(x)$

MDS: optimal performance + coding is implemented on SR: realtime

$c(x) = \prod_{i=1}^{n-k} (x - \lambda^i) \cdot u(x)$; $\deg(u(x)) = k-1$

$c(x) = g(x) u(x)$; $g(x) = \prod_{i=1}^{n-k} (x - \lambda^i)$

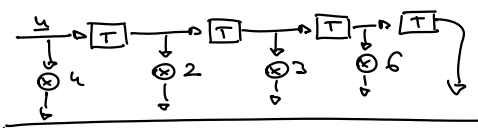
$t = 2$; q prime; $n-k = 2 \cdot 2 = 4$

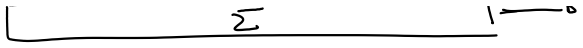
$3 \in GF(7)$ $\quad C(6,2)$

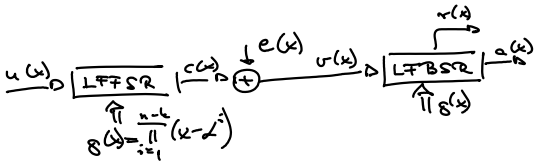
q	n	k
1	–	–
2	–	–
3	–	–
4	–	–
5	–	–
7	6	2

$g(x) = \prod_{i=1}^{n-k} (x - 3^i) = \prod_{i=1}^{4} (x - 3^i) = (x-3)(x-2)(x-6)(x-4) = (x+4)(x+5)(x+1)(x+3) =$

$= (x^2 + 2x + 6)(x^2 + 4x + 3) = x^4 + 2x^3 + 6x^2 + 4x^3 + x^2 + 3x + 3x^2 + 6x + 4 = x^4 + 6x^3 + 3x^2 + 3x + 4$







Known: $v(x) = u(x)g(x) + e(x)$

observable — known

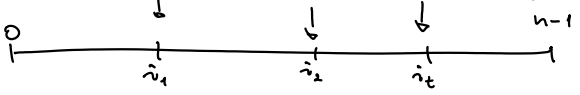
$v(x) = a(x)g(x) + r(x)$ — computable

given — computable — known

division with remainder $\deg(r(x)) < n-k$

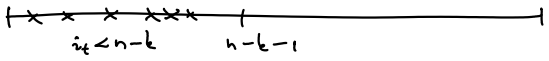
$a(x) = u(x)$; $r(x) = e(x)$ ← it work only in case of low error configuration

We assume t numbers of errors: $\underline{e} = (0 \dots 0\, e_{i_1} 0 \dots 0\, e_{i_2} 0 \dots 0\, e_{i_t} 0 \dots 0)$



$e(x) = 0 + 0 \cdot x + e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \dots + e_{i_t} x^{i_t}$

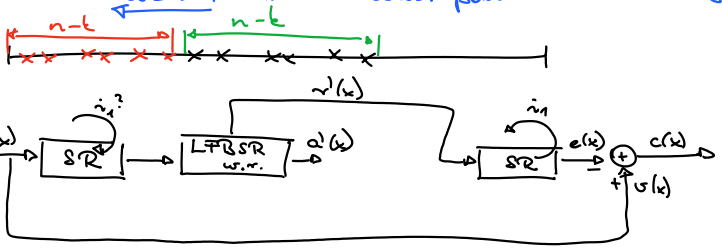
Low error configuration:



$\deg(e(x)) < n-k$

Assumption: $i_t - i_1 < n-k$ "bursty nature"

we shift errors to lower positions with shift registers



$x^{-i} v(x) = a'(x) g(x) + r'(x)$

$r'(x) = x^{-i_1} e(x)$

$e(x) = x^{i_1} r'(x)$

How to find i_1?

$v(x) = u(x)g(x) + e(x) = \big(u(x) + b(x)\big) g(x) + r(x) \Rightarrow v(x) \bmod g(x) = e(x) \bmod g(x)$

$e(x) = b(x)g(x) + r(x)$

$x^{-\delta} v(x) = a'(x) g(x) + r'(x)$

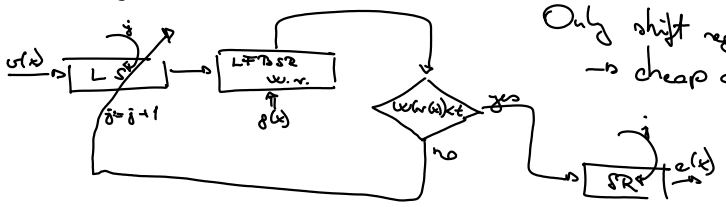
$x^{-\delta} e(x) = b'(x) g(x) + r'(x)$

$x^{-\delta} e(x) - r'(x) = b(x)g(x) \in C$

$w(x^{-\delta} e(x)) < t$; $w(b(x)g(x)) \geq 2t+1$

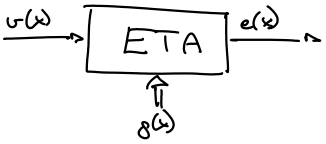
$w(r'(x)) \geq t$

$x^{-\delta} e(x) = r'(x) \rightarrow e(x) = x^{i} r'(x)$



Only shift registers are needed!
→ cheap and fast

Error Trapping Algorithm



$g(x)$ $\qquad$ $g(x)$

$$c(x) = u(x)\,g(x) \quad \text{so} \quad u(x) = \frac{c(x)}{g(x)}$$

$$g(x) = \prod_{i=1}^{n-1} (x - x_i)$$