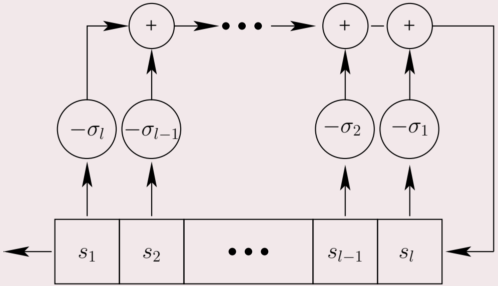
ulm university universität uulm

Fast Recursive Linearized-Feedback Shift-Register Synthesis

Yahia Hassan and Vladimir Sidorenko

Ulm University, TAIT

Linear feedback shift-register R



$\sigma_i \in \mathbb{F}_Q$,; R generates $s_1, s_2, \ldots, s_N \in \mathbb{F}_Q$; Find $\min l$

Linear means:

A register R generates $s, \tilde{s} \Rightarrow R$ generates $\alpha s + \beta \tilde{s}\ \forall \alpha, \beta \in \mathbb{F}_Q$

Connection polynomial:

$$\sigma(x) \triangleq 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_l x^l$$

Problem 0. (Solve the key equation for Reed–Solomon codes)

Let $s = s_1, s_2, \ldots, s_N$ be a sequence over a field $\mathbb{F}_Q$. Find the smallest integer $l \geq 0$ for which the system

$$s_n = -\sum_{i=1}^{l} \sigma_i s_{n-i} \quad n = l+1, \ldots, N$$

has a solution $\sigma_1, \ldots, \sigma_l \in \mathbb{F}_Q$. Find also one such solution.

- l is the *linear complexity* of s
- Gaussian elumination has complexity: $\mathcal{O}(N^3)$ oper. in $\mathbb{F}_Q$
- The Berlekamp–Massey algorithm solves Problem 0 with complexity $\mathcal{O}(N^2)$
- Blahut, Afanassiev: complexity $\mathcal{O}(N \log^2 N)$

The Berlekamp–Massey Algorithm

input: N, $s = s_1, \ldots, s_N$

$l \leftarrow 0$, $\sigma(x) \leftarrow 1$, $\sigma'(x) \leftarrow 1$, $n' \leftarrow 0$, $d' \leftarrow 1$

for *each* n *from* 1 *to* N **do**

  $d \leftarrow \sum_{j=0}^{l} \sigma_j s_{n-j}$

  if $d \neq 0$ **then**

    if $2l \geq n$ **then**

      $\sigma(x) \leftarrow \sigma(x) - \frac{d}{d'} x^{n-n'} \sigma'(x)$

    else

      $\tilde{\sigma}(x) \leftarrow \sigma(x)$

      $\sigma(x) \leftarrow \sigma(x) - \frac{d}{d'} x^{n-n'} \sigma'(x)$, $l \leftarrow n - l$

      $\sigma'(x) \leftarrow \tilde{\sigma}(x)$, $n' \leftarrow n$, $d' \leftarrow d$

output: l, $\sigma(x)$

Notations

Extention field

$$Q = q^m, \qquad \mathbb{F}_Q = \mathbb{F}_{q^m}$$

Frobenius power

For any integer i: $\quad x^{[i]} \triangleq x^{q^i}$

Problem 1. (Solve the key equation for Gabidulin codes)

Let $s = s_1, s_2, \ldots, s_N$ be a sequence over a field $\mathbb{F}_Q$. Find the smallest integer $l \geq 0$ for which the system

$$s_n = -\sum_{i=1}^{l} \sigma_i s_{n-i}^{[i]} \quad n = l+1, \ldots, N \tag{1}$$

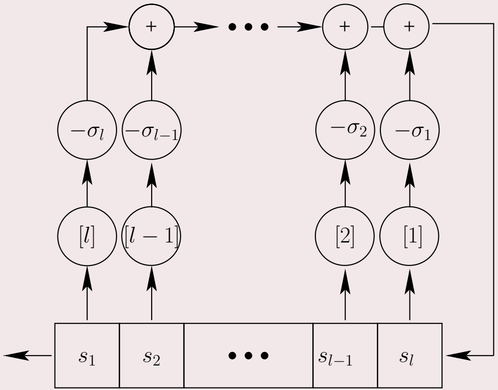
has a solution $\sigma_1, \ldots, \sigma_l \in \mathbb{F}_Q$. Find also one such solution.

Complexity: $\mathcal{O}(N^3)$

We would like:

- to solve the problem with low complexity

Linearized feedback shift-register



q-linearized means:

A register R generates $s, \tilde{s} \Rightarrow R$ generates $\alpha s + \beta \tilde{s}$ $\forall \alpha, \beta \in \mathbb{F}_q$

A Q-linearized shift-register is a *linear* shift-register, since $x^Q = x$.

Shortest linearized feedback shift-register synthesis

Problem 1a. (Equivalent to Problem 1)

Given a sequence s over $\mathbb{F}_Q$, find a shortest q-linearized shift-register that generates s.

Definition (Linearized complexity)

The minimal length l of q-linearized shift-register generating a sequence s over $\mathbb{F}_Q$ is called *q-linearized complexity of the sequence s.*

Remark

The Q-linearized complexity of a sequence s over $\mathbb{F}_Q$ is simply the linear complexity of s.

Motivation

- Problem 1 is equivalent to the problem of solving the key equation when decoding Gabidulin codes up to half the code distance in rank metric
- Gabidulin codes have already found many applications in channel coding and cryptography
- Recently Kötter and Kschischang have shown that network coding can be based on Gabidulin codes

Known Results and our contribution

Known Results

- In 1991, Paramonov and Tretjakov (PT) suggested and proved an efficient algorithm similar to the Berlekamp-Massey (BM) algorithm to solve Problem 1
- In 2004, Richter and Plass (RP) independently obtained a similar algorithm and announced it without complete proof
- Complexity of the PTRP algorithm is $\mathcal{O}(N^2)$
- In 2010, the Wachter at al. algorithm with complexity $\mathcal{O}(N^{1.68}\log N)$, based on Euclidean alg, Aho-Hopcroft

Our contribution

- Fast PTRP algorithm having complexity $\mathcal{O}(N^{1.68}\log N)$, based on Berlekamp-Massey, Blahut, Afanassiev

Linearized polynomials

Definition

A q-linearized polynomial (or q-polynomial) over $\mathbb{F}_{q^m} = \mathbb{F}_Q$ is a polynomial of the form

$$f(x) = \sum_{i=0}^{t} f_i x^{[i]},$$

where $f_i \in \mathbb{F}_Q$. If $f_t \neq 0$ we call t the q-*degree* of $f(x)$ and denote it by $\deg_q f(x) = t$.

Symbolic product

$$f(x) \otimes g(x) = f(g(x)).$$

Connection polynomial

$$\sigma(x) = \sum_{i=0}^{l} \sigma_i x^{[i]},$$

Algorithm PTRP. Paramonov-Tretjakov-Richter-Plass

input: $N,\ s = s_1, \ldots, s_N$

$l \leftarrow 0,\ \sigma(x) \leftarrow x,\quad \sigma'(x) \leftarrow x,\ n' \leftarrow 0,\ d' \leftarrow 1,\ h \leftarrow 0$

for *each* n *from* 1 *to* N **do**

  $d \leftarrow \sum_{j=0}^{l} \sigma_j s_{n-j}^{[j]}$

  if $d \neq 0$ **then**

    if $2l \geq n$ **then**

      $\sigma(x) \leftarrow \sigma(x) - d\left(\frac{x}{d'}\right)^{[n-n']} \otimes \sigma'(x)$

    else

      $h \leftarrow h+1,\ \sigma^{(h)}(x) \leftarrow \sigma(x),\ l_h \leftarrow l$

      $\sigma(x) \leftarrow \sigma(x) - d\left(\frac{x}{d'}\right)^{[n-n']} \otimes \sigma'(x),\quad l \leftarrow n - l$

      $\sigma'(x) \leftarrow \sigma^{(h)}(x),\ n' \leftarrow n,\ d' \leftarrow d$

output: $l,\ \sigma(x),$

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}

$\deg_q(\Lambda(x))$

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	$\deg_q(\Lambda(x))$
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	0

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	$\deg_q(\Lambda(x))$
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	0
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	$\deg_q(\Lambda(x))$
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	0
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	$\deg_q(\Lambda(x))$
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	0
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	2

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	$\deg_q(\Lambda(x))$
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	0
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	2
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	2

PTRP Algorithm - Complexity grows

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	$\deg_q(\Lambda(x))$
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	0
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	1
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	2
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	2
							$\vdots$									
S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	8

Idea of acceleration

Motivation and Main Idea

- Exploit the short length of $\sigma(x)$ at the early iterations.
- Divide the sequence into batches.
- Update the sequence after processing every batch.

S_0	S_1	....	$S_{\frac{d-1}{2}-2}$	$S_{\frac{d-1}{2}-1}$	$S_{\frac{d-1}{2}}$	$S_{\frac{d-1}{2}+1}$	....	S_{d-3}	S_{d-2}

S_0	S_1	....	$S_{\frac{d-1}{2}-2}$	$S_{\frac{d-1}{2}-1}$

$S'_{\frac{d-1}{2}}$	$S'_{\frac{d-1}{2}+1}$	....	S'_{d-3}	S'_{d-2}

Theorem (PTRP algorithm)

Problem 1 can be solved as follows. Set $\sigma^{(-1)}(x) = B^{(-1)}(x) = x$ *and* $\ell_{-1} = 0$. *For iterations* $i = 0, 1, \ldots, n-1$ *compute*

$$\triangle_i = \sum_{j=0}^{\ell_{i-1}} \sigma_j^{(i-1)} S_{i-j}^{[j]}, \qquad (2)$$

$$\delta_i = \begin{cases} 1, & \text{if } \triangle_i \neq 0 \text{ and } 2\ell_{(i-1)} \leq i \\ 0, & \text{otherwise} \end{cases},$$

$$\ell_i = \delta_i(i + 1 - \ell_{i-1}) + (1 - \delta_i)\ell_{i-1},$$

$$\begin{bmatrix} \sigma^{(i)}(x) \\ B^{(i)}(x) \end{bmatrix} = M^{(i)}(x) \otimes \begin{bmatrix} \sigma^{(i-1)}(x) \\ B^{(i-1)}(x) \end{bmatrix}, \qquad (3)$$

where $M^{(i)}(x) \triangleq \begin{bmatrix} x & -\triangle_i x^{[1]} \\ \triangle_i^{-1}\delta_i x & (1-\delta_i)x^{[1]} \end{bmatrix}.$

Definitions

for $i = 0, 1, \ldots, n-1$

$$\begin{bmatrix} \sigma^{(i)}(x) \\ B^{(i)}(x) \end{bmatrix} = \left[\prod_{k=0}^{i} M^{(k)}(x)\right] \otimes \begin{bmatrix} x \\ x \end{bmatrix} \quad (4)$$

$$\Upsilon^{(i)}(x) \triangleq \prod_{k=1}^{i} M^{(k)}(x) \quad (5)$$

$$S(x) = S_0 x^{[0]} + S_1 x^{[1]} + \cdots + S_{n-1} x^{[n-1]}$$

$$V^{(i)}(x) \triangleq \Upsilon^{(i)}(x) \otimes \begin{bmatrix} S(x) \\ S(x) \end{bmatrix} \quad (6)$$

Algorithm 1: Recursive PTRP Algorithm

Input: $S(x) = S_0 + S_1 x^{[1]} + \cdots + S_{n-1} x^{[n-1]}$; n
Output: $\sigma(x)$; ℓ

begin

$V(x) = \begin{bmatrix} S(x) \\ S(x) \end{bmatrix}$; $\Upsilon(x) \leftarrow \begin{bmatrix} x & 0 \\ 0 & x \end{bmatrix}$; $\Upsilon'(x) \leftarrow \begin{bmatrix} x & 0 \\ 0 & x \end{bmatrix}$;

$\ell \leftarrow 0$; $i \leftarrow 0$; $\tau \leftarrow n$; ($\Upsilon'(x)$, ℓ, i, and τ are global);

RecPTRP$(\Upsilon(x), V(x))$**;**

$\sigma(x) = \Upsilon'_{11}(x) + \Upsilon'_{12}(x)$**;**

end

Algorithm 2: Splitting subroutine **RecPTRP**$(\Upsilon(x), V(x))$

```
begin
    Stack.Push(Υ(x), V(x));
    if τ ≠ 1 then
        τ ← τ/2;
        RecPTRP(Υ(x), V(x));
        V(x) ← Υ'(x) ⊗ V(x);     Υ(x) ← Υ'(x);
        RecPTRP(Υ(x), V(x));
        Υ'(x) ← Υ'(x) ⊗ Υ(x);    τ ← τ * 2;
    else
        △_i ← V_{1,i};
        if △_i = 0 and 2ℓ > i then
            δ_i ← 0;
        else
            δ_i ← 1;    ℓ ← i + 1 − ℓ;
        Υ'(x) ← [ x, −△_i x^[1] ; △_i^{-1} δ_i x, (1 − δ_i) x^[1] ];   i ← i + 1;
    Stack.Pop(Υ(x), V(x));
end
```

$$\Upsilon'(x) \leftarrow \begin{bmatrix} x & -\triangle_i x^{[1]} \\ \triangle_i^{-1}\delta_i x & (1-\delta_i)x^{[1]} \end{bmatrix}; \quad i \leftarrow i+1;$$

Complexity

- Denote $\mathcal{M}(n)$ the complexity of the symbolic product of two q-polynomials of degree n. Wachter at al. suggested an algorithm with complexity

$$\mathcal{M}(n) = \mathcal{O}(n^{1.69})$$

- The suggested recursive algorithm has complexity

$$\kappa = \mathcal{O}(\mathcal{M}(n) \log n)$$

operations in the field $\mathbb{F}_{q^m}$.

- Using Wacher's algorithm we have complexity

$$\mathcal{O}(n^{1.69} \log(n))$$

Conclusions

- A fast algorithm to find a shortest linearized shift-register generating a given sequence s is suggested.
- The algorithm has asimptotical complexity $\kappa = \mathcal{O}(\mathcal{M}(n)\log n)$
- With Wachter's fast multiplication our algorithm has complexity $\mathcal{O}(n^{1.69}\log(n))$

Acknowledgment

The authors are very thankful to Antonia Wachter

Accompanying matrices

$$B = \begin{pmatrix} s_1 & s_2^{[-1]} & \dots & s_{N-1}^{[-N+2]} & s_N^{[-N+1]} \\ s_2 & s_3^{[-1]} & \dots & s_N^{[-N+2]} & * \\ \vdots & \vdots & & \vdots & \vdots \\ s_N & * & \dots & * & * \end{pmatrix}.$$

Lemma

If the first c columns of B are linearly independent, then the length l of every shift-register generating these sequences satisfies $l \geq c$.

Lemma

If PTRP algorithm reaches a shift-register length l then the first l columns of the accompanying B-matrix are linearly independent.