# Skew cyclic codes: Hamming distance and decoding algorithms[1]

**J. Gómez-Torrecillas**[†], F. J. Lobillo[†], G. Navarro[‡]

[†]**Department of Algebra and CITIC, University of Granada**

[‡]Department of Computer Sciences and AI, and CITIC, University of Granada

NCRA V, Lens, June 14, 2017

[1]Research supported by grants MTM2013-41992-P and MTM2016-78364-P from Spanish government and FEDER.

Based on:

- GLN, **Generating idempotents in ideal codes**, ACM Communications in Computer Algebra, Vol 48, No. 3, Issue 189, September 2014, ISSAC poster abstracts, pp. 113-115.
- GLN, **A new perspective of cyclicity in convolutional codes**, IEEE Transactions on Information Theory 62 (5) (2016), 2702–2706.
- GLN, **A Sugiyama-like decoding algorithm for convolutional codes**, 2016. arXiv:1607.07187.
- GLN, **Ideal codes over separable ring extensions**, IEEE Transactions on Information Theory 63 (5) (2017), 2796–2813.

# Index

# Convolutional codes

$\mathbb{F}$ a finite field, $k \leq n$ positive integers. A *rate $k/n$ convolutional transducer $G$* transforms

$$\text{information sequences } \mathbf{u} = \dots \mathbf{u}_{-1}\mathbf{u_0}\mathbf{u_1}\dots \quad (\mathbf{u}_i \in \mathbb{F}^k)$$

into

$$\text{code sequences } \mathbf{v} = \dots \mathbf{v_{-1}}\mathbf{v_0}\mathbf{v_1}\dots \quad (\mathbf{v}_i \in \mathbb{F}^n).$$

# Convolutional codes

$\mathbb{F}$ a finite field, $k \leq n$ positive integers. A *rate $k/n$ convolutional transducer* $G$ transforms

$$\text{information sequences } \mathbf{u} = \dots \mathbf{u}_{-1}\mathbf{u_0}\mathbf{u_1} \dots \quad (\mathbf{u}_i \in \mathbb{F}^k)$$

into

$$\text{code sequences } \mathbf{v} = \dots \mathbf{v_{-1}}\mathbf{v_0}\mathbf{v_1} \dots \quad (\mathbf{v}_i \in \mathbb{F}^n).$$

**Requirements:** Write $\mathbf{u} = \sum_{i=-i_0}^{\infty} \mathbf{u}_i t^i \in \mathbb{F}^k((t)) \cong \mathbb{F}((t))^k$, $\mathbf{v} = \sum_{j=-j_0}^{\infty} \mathbf{v}_i t^j \in \mathbb{F}^n((t)) \cong \mathbb{F}((t))^n$. Then

$$\mathbf{v} = \mathbf{u}G,$$

where $G$ is an $k \times n$ full rank matrix with entries in $\mathbb{F}(t)$.

# Convolutional codes

$\mathbb{F}$ a finite field, $k \leq n$ positive integers. A *rate $k/n$ convolutional transducer* $G$ transforms

$$\text{information sequences } \mathbf{u} = \dots \mathbf{u}_{-1}\mathbf{u_0}\mathbf{u_1} \dots \quad (\mathbf{u}_i \in \mathbb{F}^k)$$

into

$$\text{code sequences } \mathbf{v} = \dots \mathbf{v_{-1}}\mathbf{v_0}\mathbf{v_1} \dots \quad (\mathbf{v}_i \in \mathbb{F}^n).$$

**Requirements:** Write $\mathbf{u} = \sum_{i=-i_0}^{\infty} \mathbf{u}_i t^i \in \mathbb{F}^k((t)) \cong \mathbb{F}((t))^k$, $\mathbf{v} = \sum_{j=-j_0}^{\infty} \mathbf{v}_i t^j \in \mathbb{F}^n((t)) \cong \mathbb{F}((t))^n$. Then

$$\mathbf{v} = \mathbf{u}G,$$

where $G$ is an $k \times n$ full rank matrix with entries in $\mathbb{F}(t)$.

## Definition

A *rate $k/n$ convolutional code $\mathcal{D}$* over $\mathbb{F}$ is the image of a rate $k/n$ convolutional transducer $G$, that is

$$\mathcal{D} = \{\mathbf{u}G : \mathbf{u} \in \mathbb{F}^k((t))\} \subseteq \mathbb{F}^n((t)).$$

# Convolutional codes

$\mathbb{F}$ a finite field, $k \leq n$ positive integers. A *rate $k/n$ convolutional transducer* $G$ transforms

$$\text{information sequences } \mathbf{u} = \dots \mathbf{u}_{-1}\mathbf{u_0}\mathbf{u_1} \dots \quad (\mathbf{u}_i \in \mathbb{F}^k)$$

into

$$\text{code sequences } \mathbf{v} = \dots \mathbf{v_{-1}}\mathbf{v_0}\mathbf{v_1} \dots \quad (\mathbf{v}_i \in \mathbb{F}^n).$$

**Requirements:** Write $\mathbf{u} = \sum_{i=-i_0}^{\infty} \mathbf{u}_i t^i \in \mathbb{F}^k((t)) \cong \mathbb{F}((t))^k$, $\mathbf{v} = \sum_{j=-j_0}^{\infty} \mathbf{v}_i t^j \in \mathbb{F}^n((t)) \cong \mathbb{F}((t))^n$. Then

$$\mathbf{v} = \mathbf{u}G,$$

where $G$ is an $k \times n$ full rank matrix with entries in $\mathbb{F}(t)$.

## Definition

A *rate $k/n$ convolutional code* $\mathcal{D}$ over $\mathbb{F}$ is the image of a rate $k/n$ convolutional transducer $G$, that is

$$\mathcal{D} = \{\mathbf{u}G : \mathbf{u} \in \mathbb{F}^k((t))\} \subseteq \mathbb{F}^n((t)).$$

## Definition (Vector space version)

A rate $k/n$ convolutional code over $\mathbb{F}$ is a $k$–dimensional vector subspace of $\mathbb{F}(t)^n$.

# Cyclic convolutional codes (module version)

**Lemma**

*The map $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ is a bijection between the set of rate $k/n$ convolutional codes and the set of submodules of rank $k$ of $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ that are direct summands.*

# Cyclic convolutional codes (module version)

## Lemma

*The map* $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ *is a bijection between the set of rate* $k/n$ *convolutional codes and the set of submodules of rank* $k$ *of* $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ *that are direct summands.*

## Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

# Cyclic convolutional codes (module version)

## Lemma

*The map* $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ *is a bijection between the set of rate* $k/n$ *convolutional codes and the set of submodules of rank* $k$ *of* $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ *that are direct summands.*

## Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

Thus, a cyclic structure could be introduced by taking $\mathbb{F}^n \cong A := \mathbb{F}[x]/\langle x^n - 1\rangle$, and suitable ideals of $A[t]$ as cyclic convolutional codes. But...

# Cyclic convolutional codes (module version)

### Lemma

*The map* $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ *is a bijection between the set of rate* $k/n$ *convolutional codes and the set of submodules of rank* $k$ *of* $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ *that are direct summands.*

### Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

Thus, a cyclic structure could be introduced by taking $\mathbb{F}^n \cong A := \mathbb{F}[x]/\langle x^n - 1 \rangle$, and suitable ideals of $A[t]$ as cyclic convolutional codes. But...

**Problem:** Piret (1976) observed that no non-block codes are obtained in this way.

# Cyclic convolutional codes (module version)

## Lemma

*The map $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ is a bijection between the set of rate $k/n$ convolutional codes and the set of submodules of rank $k$ of $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ that are direct summands.*

## Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

Thus, a cyclic structure could be introduced by taking $\mathbb{F}^n \cong A := \mathbb{F}[x]/\langle x^n - 1 \rangle$, and suitable ideals of $A[t]$ as cyclic convolutional codes. But...

**Problem:** Piret (1976) observed that no non-block codes are obtained in this way.

**Solution:** Piret's (1976) idea is skewing the multiplication of $A[t]$.

# Cyclic convolutional codes (module version)

## Lemma

*The map $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ is a bijection between the set of rate $k/n$ convolutional codes and the set of submodules of rank $k$ of $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ that are direct summands.*

## Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

Thus, a cyclic structure could be introduced by taking $\mathbb{F}^n \cong A := \mathbb{F}[x]/\langle x^n - 1\rangle$, and suitable ideals of $A[t]$ as cyclic convolutional codes. But...

**Problem:** Piret (1976) observed that no non-block codes are obtained in this way.

**Solution:** Piret's (1976) idea is skewing the multiplication of $A[t]$. Technically, this means to define cyclic convolutional codes as suitable left ideals of a skew polynomial ring $A[t;\sigma]$.

# Cyclic convolutional codes (module version)

## Lemma

*The map $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ is a bijection between the set of rate $k/n$ convolutional codes and the set of submodules of rank $k$ of $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ that are direct summands.*

## Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

Thus, a cyclic structure could be introduced by taking $\mathbb{F}^n \cong A := \mathbb{F}[x]/\langle x^n - 1\rangle$, and suitable ideals of $A[t]$ as cyclic convolutional codes. But...

**Problem:** Piret (1976) observed that no non-block codes are obtained in this way.

**Solution:** Piret's (1976) idea is skewing the multiplication of $A[t]$. Technically, this means to define cyclic convolutional codes as suitable left ideals of a skew polynomial ring $A[t;\sigma]$.
This idea has been further developed, (e.g. considering more general finite $\mathbb{F}$–algebras $A$) by Roos (1979), Gluesing-Luerssen/Schmale (2004), López-Permouth/Szabo (2013), GLN (2014, 2017), among others.

# Cyclic convolutional codes (module version)

## Lemma

*The map $\mathcal{D} \mapsto \mathcal{D} \cap \mathbb{F}^n[t]$ is a bijection between the set of rate $k/n$ convolutional codes and the set of submodules of rank $k$ of $\mathbb{F}^n[t] \cong \mathbb{F}[t]^n$ that are direct summands.*

## Definition (Module version)

A rate $k/n$ convolutional code is a rank $k$ direct summand of $\mathbb{F}^n[t]$.

Thus, a cyclic structure could be introduced by taking $\mathbb{F}^n \cong A := \mathbb{F}[x]/\langle x^n - 1\rangle$, and suitable ideals of $A[t]$ as cyclic convolutional codes. But...

**Problem:** Piret (1976) observed that no non-block codes are obtained in this way.

**Solution:** Piret's (1976) idea is skewing the multiplication of $A[t]$. Technically, this means to define cyclic convolutional codes as suitable left ideals of a skew polynomial ring $A[t; \sigma]$.
This idea has been further developed, (e.g. considering more general finite $\mathbb{F}$–algebras $A$) by Roos (1979), Gluesing-Luerssen/Schmale (2004), López-Permouth/Szabo (2013), GLN (2014, 2017), among others.

**Main difficulty/opportunity:** Dealing with idempotents in $A[t; \sigma]$.

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1 \rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

**Proposal:** (GLN (2016)). A cyclic convolutional could be a left ideal of $\mathcal{R} = \mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle$, where $\sigma$ is an $\mathbb{F}$–automorphism of order $n$ of $\mathbb{F}(t)$.

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

**Proposal:** (GLN (2016)). A cyclic convolutional could be a left ideal of $\mathcal{R} = \mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle$, where $\sigma$ is an $\mathbb{F}$–automorphism of order $n$ of $\mathbb{F}(t)$.

**Features:**

- $\mathbb{F}(t)[x;\sigma]$ is a left and right Euclidean domain.

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

**Proposal:** (GLN (2016)). A cyclic convolutional could be a left ideal of $\mathcal{R} = \mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle$, where $\sigma$ is an $\mathbb{F}$–automorphism of order $n$ of $\mathbb{F}(t)$.

**Features:**

- $\mathbb{F}(t)[x;\sigma]$ is a left and right Euclidean domain.
- $\mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle \cong M_n(\mathbb{F}(t)^\sigma)$.

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

**Proposal:** (GLN (2016)). A cyclic convolutional could be a left ideal of $\mathcal{R} = \mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle$, where $\sigma$ is an $\mathbb{F}$–automorphism of order $n$ of $\mathbb{F}(t)$.

**Features:**

∘ $\mathbb{F}(t)[x;\sigma]$ is a left and right Euclidean domain.

∘ $\mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle \cong M_n(\mathbb{F}(t)^\sigma)$.

∘ When the generator of the code is carefully chosen, we get an MDS code with respect to the Hamming distance.

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

**Proposal:** (GLN (2016)). A cyclic convolutional could be a left ideal of $\mathcal{R} = \mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle$, where $\sigma$ is an $\mathbb{F}$–automorphism of order $n$ of $\mathbb{F}(t)$.

**Features:**

∘ $\mathbb{F}(t)[x;\sigma]$ is a left and right Euclidean domain.

∘ $\mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle \cong M_n(\mathbb{F}(t)^\sigma)$.

∘ When the generator of the code is carefully chosen, we get an MDS code with respect to the Hamming distance.

∘ Efficient decoding schemes (like Sugiyama, or Peterson-Gorenstein-Zierler) are adapted to this non-commutative setting.

# Cyclic convolutional codes (vector space version).

Would we define a cyclic convolutional code as an ideal of $\mathbb{F}(t)[x]/\langle x^n - 1\rangle \cong \mathbb{F}(t)^n$?

If so, then we get nothing but cyclic block codes (the factors of $x^n - 1$ have coefficients in $\mathbb{F}$).

**Proposal:** (GLN (2016)). A cyclic convolutional could be a left ideal of $\mathcal{R} = \mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle$, where $\sigma$ is an $\mathbb{F}$–automorphism of order $n$ of $\mathbb{F}(t)$.

**Features:**

◦ $\mathbb{F}(t)[x;\sigma]$ is a left and right Euclidean domain.

◦ $\mathbb{F}(t)[x;\sigma]/\langle x^n - 1\rangle \cong M_n(\mathbb{F}(t)^\sigma)$.

◦ When the generator of the code is carefully chosen, we get an MDS code with respect to the Hamming distance.

◦ Efficient decoding schemes (like Sugiyama, or Peterson-Gorenstein-Zierler) are adapted to this non-commutative setting.

◦ The theory (including the algorithms) work for any field $L$.

# Skew cyclic codes

So, let $L$ be a field, and $\sigma : L \to L$ a field automorphism of finite order $n$. Set $K = L^\sigma$. Consider the ring $\mathcal{R} = L[x;\sigma]/\langle x^n - 1\rangle \cong M_n(K)$.
**Note:** The multiplication rule in $L[x;\sigma]$ is $xa = \sigma(a)x$ for all $a \in L$.

# Skew cyclic codes

So, let $L$ be a field, and $\sigma : L \to L$ a field automorphism of finite order $n$. Set $K = L^\sigma$. Consider the ring $\mathcal{R} = L[x;\sigma]/\langle x^n - 1\rangle \cong M_n(K)$.
**Note:** The multiplication rule in $L[x;\sigma]$ is $xa = \sigma(a)x$ for all $a \in L$.

We have the isomorphism of $L$–vector spaces $\mathfrak{p} : L^n \to \mathcal{R}$ sending $(c_0, c_1, \ldots, c_{n-1})$ onto $c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$.
**Note:** The classical identification of equivalence classes and its representatives is (and will be) made.

# Skew cyclic codes

So, let $L$ be a field, and $\sigma : L \to L$ a field automorphism of finite order $n$. Set $K = L^\sigma$. Consider the ring $\mathcal{R} = L[x;\sigma]/\langle x^n - 1\rangle \cong M_n(K)$.
**Note:** The multiplication rule in $L[x;\sigma]$ is $xa = \sigma(a)x$ for all $a \in L$.

We have the isomorphism of $L$–vector spaces $\mathfrak{p} : L^n \to \mathcal{R}$ sending $(c_0, c_1, \ldots, c_{n-1})$ onto $c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$.
**Note:** The classical identification of equivalence classes and its representatives is (and will be) made.

## Definition

A $k$–dimensional $L$–linear code $\mathcal{C} \subseteq L^n$ of dimension $n$ is said to be a *skew cyclic* code if $\mathfrak{p}(\mathcal{C})$ is a left ideal of $\mathcal{R}$.

**Note:** We will identify $\mathcal{C}$ with $\mathfrak{p}(\mathcal{C})$. Since every left ideal of $\mathcal{R}$ is principal, we will often speak of the "skew code generated by a polynomial", aggravating the abuse of language.

# Skew cyclic codes

So, let $L$ be a field, and $\sigma : L \to L$ a field automorphism of finite order $n$. Set $K = L^\sigma$. Consider the ring $\mathcal{R} = L[x;\sigma]/\langle x^n - 1\rangle \cong M_n(K)$.
**Note:** The multiplication rule in $L[x;\sigma]$ is $xa = \sigma(a)x$ for all $a \in L$.

We have the isomorphism of $L$–vector spaces $\mathfrak{p} : L^n \to \mathcal{R}$ sending $(c_0, c_1, \ldots, c_{n-1})$ onto $c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$.
**Note:** The classical identification of equivalence classes and its representatives is (and will be) made.

## Definition

A $k$–dimensional $L$–linear code $\mathcal{C} \subseteq L^n$ of dimension $n$ is said to be a *skew cyclic* code if $\mathfrak{p}(\mathcal{C})$ is a left ideal of $\mathcal{R}$.

**Note:** We will identify $\mathcal{C}$ with $\mathfrak{p}(\mathcal{C})$. Since every left ideal of $\mathcal{R}$ is principal, we will often speak of the "skew code generated by a polynomial", aggravating the abuse of language.

**Note:** When $L = \mathbb{F}$ is a finite field, these codes lie in the realm of the theory developed by Boucher/Chaussade/Geiselmann/Loidreau/Ulmer (2007, 2009), among others, where the name is taken.

# Skew RS codes

**Idea**: Choosing carefully generator polynomials of skew codes, we get very concrete results.

# Skew RS codes

**Idea**: Choosing carefully generator polynomials of skew codes, we get very concrete results.

## Construction method of Skew RS codes

- Choose a normal basis $\{\alpha, \sigma(\alpha), \ldots, \sigma^{n-1}(\alpha)\}$ of the field extension $L/K$.

# Skew RS codes

**Idea**: Choosing carefully generator polynomials of skew codes, we get very concrete results.

## Construction method of Skew RS codes

- Choose a normal basis $\{\alpha, \sigma(\alpha), \ldots, \sigma^{n-1}(\alpha)\}$ of the field extension $L/K$.
- Set $\beta = \alpha^{-1}\sigma(\alpha)$

# Skew RS codes

**Idea**: Choosing carefully generator polynomials of skew codes, we get very concrete results.

## Construction method of Skew RS codes

- Choose a normal basis $\{\alpha, \sigma(\alpha), \ldots, \sigma^{n-1}(\alpha)\}$ of the field extension $L/K$.
- Set $\beta = \alpha^{-1}\sigma(\alpha)$
- $x^n - 1$ decomposes as the least common left multiple

$$x^n - 1 = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-1}(\beta)\right]_\ell$$

# Skew RS codes

**Idea**: Choosing carefully generator polynomials of skew codes, we get very concrete results.

## Construction method of Skew RS codes

- Choose a normal basis $\{\alpha, \sigma(\alpha), \ldots, \sigma^{n-1}(\alpha)\}$ of the field extension $L/K$.
- Set $\beta = \alpha^{-1}\sigma(\alpha)$
- $x^n - 1$ decomposes as the least common left multiple

$$x^n - 1 = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-1}(\beta)\right]_\ell$$

- For $1 \leq k < n$, the left ideal $\mathcal{C}$ of $\mathcal{R}$ generated by

$$g = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-k-1}(\beta)\right]_\ell$$

is an SCC of length $n$ and dimension $k$. We call them Reed-Solomon skew codes.

# Skew RS codes

**Idea**: Choosing carefully generator polynomials of skew codes, we get very concrete results.

## Construction method of Skew RS codes

- Choose a normal basis $\{\alpha, \sigma(\alpha), \dots, \sigma^{n-1}(\alpha)\}$ of the field extension $L/K$.
- Set $\beta = \alpha^{-1}\sigma(\alpha)$
- $x^n - 1$ decomposes as the least common left multiple

$$x^n - 1 = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \dots, x - \sigma^{n-1}(\beta)\right]_\ell$$

- For $1 \leq k < n$, the left ideal $\mathcal{C}$ of $\mathcal{R}$ generated by

$$g = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \dots, x - \sigma^{n-k-1}(\beta)\right]_\ell$$

is an SCC of length $n$ and dimension $k$. We call them Reed-Solomon skew codes.

## Theorem

*The Hamming minimum distance of $\mathcal{C}$ is $\delta = n - k + 1$. Thus, it is an MDS code.*

# Parity check matrix

A parity check matrix of the skew RS code is given by

$$H = \begin{pmatrix} N_0(\beta) & N_0(\sigma(\beta)) & \dots & N_0(\sigma^{n-k-1}(\beta)) \\ N_1(\beta) & N_1(\sigma(\beta)) & \dots & N_1(\sigma^{n-k-1}(\beta)) \\ N_2(\beta) & N_2(\sigma(\beta)) & \dots & N_2(\sigma^{n-k-1}(\beta)) \\ \vdots & \vdots & \ddots & \vdots \\ N_{n-1}(\beta) & N_{n-1}(\sigma(\beta)) & \dots & N_{n-1}(\sigma^{n-k-1}(\beta)) \end{pmatrix}.$$

Here, for $\gamma \in L$,

$$N_j(\gamma) = \gamma\sigma(\gamma)\dots\sigma^{j-1}(\gamma)$$

is the remainder of the *left division* of $x^j$ by $x - \gamma$, that is

$$x^j = q(x)(x-\gamma) + N_j(\gamma).$$

# Transmission and syndromes

Our skew RS code $\mathcal{C}$ is generated by

$$g = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-k-1}(\beta)\right]_\ell$$

# Transmission and syndromes

Our skew RS code $\mathcal{C}$ is generated by

$$g = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-k-1}(\beta)\right]_\ell$$

Set $\tau = \lfloor \frac{\delta-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$ which is the maximum number of errors that the code can correct.

# Transmission and syndromes

Our skew RS code $\mathcal{C}$ is generated by

$$g = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-k-1}(\beta)\right]_\ell$$

Set $\tau = \lfloor \frac{\delta-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$ which is the maximum number of errors that the code can correct.

$$m \longrightarrow c = mg \longrightarrow y = c + e$$

$$\sum_{i=0}^{k-1} m_i x^i \qquad \sum_{i=0}^{n-1} c_i x^i \qquad \sum_{i=0}^{n-1} c_i x^i + \sum_{j=1}^{\nu} e_j x^{k_j} \qquad \nu \leq \tau$$

$$\boxed{\text{message}} \longrightarrow \boxed{\text{encoded}} \longrightarrow \boxed{\text{received}}$$

# Transmission and syndromes

Our skew RS code $\mathcal{C}$ is generated by

$$g = \left[x - \beta, x - \sigma(\beta), x - \sigma^2(\beta), \ldots, x - \sigma^{n-k-1}(\beta)\right]_\ell$$

Set $\tau = \lfloor \frac{\delta-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$ which is the maximum number of errors that the code can correct.

$$m \longrightarrow c = mg \longrightarrow y = c + e$$

$$\sum_{i=0}^{k-1} m_i x^i \qquad \sum_{i=0}^{n-1} c_i x^i \qquad \sum_{i=0}^{n-1} c_i x^i + \sum_{j=1}^{\nu} e_j x^{k_j} \qquad \nu \leq \tau$$

$$\boxed{\text{message}} \longrightarrow \boxed{\text{encoded}} \longrightarrow \boxed{\text{received}}$$

From the received polynomial $y = \sum_{j=0}^{n-1} y_j x^j$ we can compute the *syndromes*

$$S_i = \sum_{j=0}^{n-1} y_j N_j(\sigma^i(\beta)), \qquad i = 0, \ldots, n-1$$

# Error locator and error evaluator

We thus know the *syndrome polynomial*, defined as

$$S = \sum_{i=0}^{2\tau-1} \sigma^i(\alpha) S_i x^i.$$

# Error locator and error evaluator

We thus know the *syndrome polynomial*, defined as

$$S = \sum_{i=0}^{2\tau-1} \sigma^i(\alpha) S_i x^i.$$

Define the *locator polynomial*

$$\lambda = [1 - \sigma^{k_1}(\beta)x, 1 - \sigma^{k_2}(\beta)x, \ldots, 1 - \sigma^{k_\nu}(\beta)x]_r$$

# Error locator and error evaluator

We thus know the *syndrome polynomial*, defined as

$$S = \sum_{i=0}^{2\tau-1} \sigma^i(\alpha) S_i x^i.$$

Define the *locator polynomial*

$$\lambda = [1 - \sigma^{k_1}(\beta)x, 1 - \sigma^{k_2}(\beta)x, \ldots, 1 - \sigma^{k_\nu}(\beta)x]_r$$

For $1 \leq j \leq \nu$ we thus know that there is $p_j$ of degree $\nu - 1$ such that $\lambda = (1 - \sigma^{k_j}(\beta)x)p_j$.

# Error locator and error evaluator

We thus know the *syndrome polynomial*, defined as

$$S = \sum_{i=0}^{2\tau-1} \sigma^i(\alpha) S_i x^i.$$

Define the *locator polynomial*

$$\lambda = [1 - \sigma^{k_1}(\beta)x, 1 - \sigma^{k_2}(\beta)x, \dots, 1 - \sigma^{k_\nu}(\beta)x]_r$$

For $1 \leq j \leq \nu$ we thus know that there is $p_j$ of degree $\nu - 1$ such that $\lambda = (1 - \sigma^{k_j}(\beta)x)p_j$. Define the *evaluator polynomial*

$$\omega = \sum_{j=1}^{\nu} e_j \sigma^{k_j}(\alpha) p_j \tag{1}$$

# Error locator and error evaluator

We thus know the *syndrome polynomial*, defined as

$$S = \sum_{i=0}^{2\tau-1} \sigma^i(\alpha) S_i x^i.$$

Define the *locator polynomial*

$$\lambda = [1 - \sigma^{k_1}(\beta)x, 1 - \sigma^{k_2}(\beta)x, \ldots, 1 - \sigma^{k_\nu}(\beta)x]_r$$

For $1 \leq j \leq \nu$ we thus know that there is $p_j$ of degree $\nu - 1$ such that $\lambda = (1 - \sigma^{k_j}(\beta)x)p_j$. Define the *evaluator polynomial*

$$\omega = \sum_{j=1}^{\nu} e_j \sigma^{k_j}(\alpha) p_j \tag{1}$$

**Sugiyama's decoding scheme.**

If $\lambda$ is computed, then the error positions $k_1, \ldots, k_\nu$ are derived. With these at hand, we can compute $p_1, \ldots, p_\nu$. Finally, the error values $e_1, \ldots, e_\nu$ are computed by solving a linear system from (1), whenever $\omega$ is known.

# The non-commutative key equation

**Theorem**

*These polynomials satisfy the non-commutative key equation*

$$\omega = S\lambda + x^{2\tau}u,$$

*for some $u$ of degree smaller than $\nu$.*

# The non-commutative key equation

## Theorem

*These polynomials satisfy the non-commutative key equation*

$$\omega = S\lambda + x^{2\tau}u,$$

*for some $u$ of degree smaller than $\nu$.*

## Theorem

*The non-commutative key equation*

$$x^{2\tau}u + S\lambda = \omega \tag{2}$$

*is a right multiple of the equation*

$$x^{2\tau}u_I + Sv_I = r_I, \tag{3}$$

*where $u_I, v_I$ and $r_I$ are the Bezout coefficients returned by the REEA with input $x^{2\tau}$ and $S$, and $I$ is the index determined by the conditions $\deg r_{I-1} \geq \tau$ and $\deg r_I < \tau$. In particular, $\lambda = v_I g$ and $\omega = r_I g$ for some $g \in L[x;\sigma]$.*

# Sugiyama-like decoding

- If $(\lambda, \omega)_r = 1$, then the last theorem gives that the REEA serves for the computation of the locator polynomial $\lambda$ and the evaluator polynomial $\omega$.

# Sugiyama-like decoding

- If $(\lambda, \omega)_r = 1$, then the last theorem gives that the REEA serves for the computation of the locator polynomial $\lambda$ and the evaluator polynomial $\omega$.
- As mentioned above, once these polynomials are computed, the error polynomial is computed by a scheme similar to the Sugiyama algorithm for (commutative) RS codes.

# Sugiyama-like decoding

- If $(\lambda, \omega)_r = 1$, then the last theorem gives that the REEA serves for the computation of the locator polynomial $\lambda$ and the evaluator polynomial $\omega$.
- As mentioned above, once these polynomials are computed, the error polynomial is computed by a scheme similar to the Sugiyama algorithm for (commutative) RS codes.
- The condition $(\lambda, \omega)_r = 1$ is equivalent to

$$\deg v_I = \mathrm{Cardinal}\{0 \leq i \leq n-1 : 1 - \sigma^i(\beta)x \text{ left divides } v_I\}$$

# Sugiyama-like decoding algorithm

**Input:** A polynomial $y = \sum_{i=0}^{n-1} y_i x^i$ received from the transmission of a codeword $c$ in a skew RS code $\mathcal{C}$ generated by $g = [\{x - \sigma^i(\beta)\}_{i=0,\dots,n-k-1}]_\ell$ of error-correcting capacity $\tau = \lfloor \frac{n-k}{2} \rfloor$.

**Output:** A codeword $c'$, or *key equation failure*.

**for** $0 \leq i \leq 2\tau - 1$ **do**
  $S_i \leftarrow \sum_{j=0}^{n-1} y_j N_j(\sigma^i(\beta))$
$S \leftarrow \sum_{i=0}^{2\tau-1} \sigma^i(\alpha) S_i x^i$
**If** $S = 0$ **then**
  **Return** $y$
$\{u_i, v_i, r_i\}_{i=0,\dots,l} \leftarrow \mathrm{REEA}(x^{2\tau}, S)$
$I \leftarrow$ first iteration in REEA with $\deg r_i < \tau$, $pos \leftarrow \emptyset$
**for** $0 \leq i \leq n-1$ **do**
  **If** $1 - \sigma^i(\beta)x$ is a left divisor of $v_I$ **then**
    $pos = pos \cup \{i\}$
**If** $\deg v_I > \mathrm{Cardinal}(pos)$ **then**
  **Return** *key equation failure*
**for** $j \in pos$ **do**
  $p_j \leftarrow \text{right-quotient}(v_I, 1 - \sigma^j(\beta)x)$
Solve the linear system $r_I = \sum_{j \in pos} e_j \sigma^j(\alpha) p_j$
$e \leftarrow \sum_{j \in pos} e_j x^j$
**Return** $y - e$

# Key equation failures

How often occurs the key equation failure?

# Key equation failures

How often occurs the key equation failure? Well, it never happens if the error values are in "general position".

# Key equation failures

How often occurs the key equation failure? Well, it never happens if the error values are in "general position". More precisely,

**Theorem**

*The key equation failure happens if and only if* $e_1, \ldots, e_\nu$ *are linearly dependent over* $K = L^\sigma$.

# Key equation failures

How often occurs the key equation failure? Well, it never happens if the error values are in "general position". More precisely,

**Theorem**

*The key equation failure happens if and only if* $e_1, \ldots, e_\nu$ *are linearly dependent over* $K = L^\sigma$.

On the other hand, we have an algorithm to solve the key equation failure, if we insist.