Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

PART VIII

Outline

Why Looking for More Cyclic Codes?

Reed-Solomon codes can be seen as MDS cyclic codes. But over a fixed field $\mathbb{F}_q$, MDS codes of arbitrary length $n > q + 2$, are conjectured to exist only when $k = 1$ (the repetition code) or $k \geq n - 1$ (that is $d = 1$ or 2).

Why Looking for More Cyclic Codes?

Reed-Solomon codes can be seen as MDS cyclic codes. But over a fixed field $\mathbb{F}_q$, MDS codes of arbitrary length $n > q + 2$, are conjectured to exist only when $k = 1$ (the repetition code) or $k \geq n - 1$ (that is $d = 1$ or 2).

In both cases, either $R_{\mathcal{C}} \to 0$ or $\delta_{\mathcal{C}} \to 0$, when $n \to \infty$.

Why Looking for More Cyclic Codes?

Reed-Solomon codes can be seen as MDS cyclic codes. But over a fixed field $\mathbb{F}_q$, MDS codes of arbitrary length $n > q + 2$, are conjectured to exist only when $k = 1$ (the repetition code) or $k \geq n - 1$ (that is $d = 1$ or 2).

In both cases, either $R_{\mathcal{C}} \to 0$ or $\delta_{\mathcal{C}} \to 0$, when $n \to \infty$.

Problem 1: Find families of codes over $\mathbb{F}_q$ such that for arbitrarily large n, neither $R_{\mathcal{C}} \to 0$ nor $\delta_{\mathcal{C}} \to 0$.

Why Looking for More Cyclic Codes?

Reed-Solomon codes can be seen as MDS cyclic codes. But over a fixed field $\mathbb{F}_q$, MDS codes of arbitrary length $n > q + 2$, are conjectured to exist only when $k = 1$ (the repetition code) or $k \geq n - 1$ (that is $d = 1$ or 2).

In both cases, either $R_{\mathcal{C}} \to 0$ or $\delta_{\mathcal{C}} \to 0$, when $n \to \infty$.

Problem 1: Find families of codes over $\mathbb{F}_q$ such that for arbitrarily large n, neither $R_{\mathcal{C}} \to 0$ nor $\delta_{\mathcal{C}} \to 0$.
Very hard if we look for codes over the Gilbert-Varshamov bound.

Why Looking for More Cyclic Codes?

Reed-Solomon codes can be seen as MDS cyclic codes. But over a fixed field $\mathbb{F}_q$, MDS codes of arbitrary length $n > q + 2$, are conjectured to exist only when $k = 1$ (the repetition code) or $k \geq n - 1$ (that is $d = 1$ or 2).

In both cases, either $R_{\mathcal{C}} \to 0$ or $\delta_{\mathcal{C}} \to 0$, when $n \to \infty$.

Problem 1: Find families of codes over $\mathbb{F}_q$ such that for arbitrarily large n, neither $R_{\mathcal{C}} \to 0$ nor $\delta_{\mathcal{C}} \to 0$.
Very hard if we look for codes over the Gilbert-Varshamov bound.

Problem 2: Find interesting cyclic codes over $\mathbb{F}_q$, with $d \geq 3$, $k \geq 2$ and $n > q + 2$.

Why Looking for More Cyclic Codes?

Reed-Solomon codes can be seen as MDS cyclic codes. But over a fixed field $\mathbb{F}_q$, MDS codes of arbitrary length $n > q + 2$, are conjectured to exist only when $k = 1$ (the repetition code) or $k \geq n - 1$ (that is $d = 1$ or 2).

In both cases, either $R_{\mathcal{C}} \to 0$ or $\delta_{\mathcal{C}} \to 0$, when $n \to \infty$.

Problem 1: Find families of codes over $\mathbb{F}_q$ such that for arbitrarily large n, neither $R_{\mathcal{C}} \to 0$ nor $\delta_{\mathcal{C}} \to 0$.
Very hard if we look for codes over the Gilbert-Varshamov bound.

Problem 2: Find interesting cyclic codes over $\mathbb{F}_q$, with $d \geq 3$, $k \geq 2$ and $n > q + 2$.
E.g., BCH codes.

BCH Codes

Let $\beta \in \mathbb{F}_{q^e}$ be a n-th primitive root of unity, for a minimal e.
(Recall that $n \mid q^e - 1$, or $q^e = 1 \bmod n$.)

BCH Codes

Let $\beta \in \mathbb{F}_{q^e}$ be a n-th primitive root of unity, for a minimal e. (Recall that $n \mid q^e - 1$, or $q^e = 1 \bmod n$.)

For some integer $1 \leq m \leq n-1$ consider the minimal polynomial $g \in \mathbb{F}_q[X]$ of $\beta, \beta^2, \ldots, \beta^m$, that is the monic polynomial of minimal degree vanishing at $\beta, \beta^2, \ldots, \beta^m$.

BCH Codes

Let $\beta \in \mathbb{F}_{q^e}$ be a n-th primitive root of unity, for a minimal e. (Recall that $n \mid q^e - 1$, or $q^e = 1 \bmod n$.)

For some integer $1 \leq m \leq n-1$ consider the minimal polynomial $g \in \mathbb{F}_q[X]$ of $\beta, \beta^2, \ldots, \beta^m$, that is the monic polynomial of minimal degree vanishing at $\beta, \beta^2, \ldots, \beta^m$.

Since $g \in \mathbb{F}_q[X]$, all the conjugates of $\beta, \beta^2, \ldots, \beta^m$ are roots of g. Thus, $\deg g \leq em$.

BCH Codes

Let $\beta \in \mathbb{F}_{q^e}$ be a n-th primitive root of unity, for a minimal e. (Recall that $n \mid q^e - 1$, or $q^e = 1 \bmod n$.)

For some integer $1 \leq m \leq n-1$ consider the minimal polynomial $g \in \mathbb{F}_q[X]$ of $\beta, \beta^2, \ldots, \beta^m$, that is the monic polynomial of minimal degree vanishing at $\beta, \beta^2, \ldots, \beta^m$.

Since $g \in \mathbb{F}_q[X]$, all the conjugates of $\beta, \beta^2, \ldots, \beta^m$ are roots of g. Thus, $\deg g \leq em$.

The cyclic code $\mathcal{C} = (g) \subset \mathbb{F}_q[X]/(X^n - 1)$ is called a BCH code (from its inventors Bose, Ray-Chaudhuri and Hocquenghem).

BCH Codes

Let $\beta \in \mathbb{F}_{q^e}$ be a n-th primitive root of unity, for a minimal e. (Recall that $n \mid q^e - 1$, or $q^e = 1 \bmod n$.)

For some integer $1 \leq m \leq n-1$ consider the minimal polynomial $g \in \mathbb{F}_q[X]$ of $\beta, \beta^2, \ldots, \beta^m$, that is the monic polynomial of minimal degree vanishing at $\beta, \beta^2, \ldots, \beta^m$.

Since $g \in \mathbb{F}_q[X]$, all the conjugates of $\beta, \beta^2, \ldots, \beta^m$ are roots of g. Thus, $\deg g \leq em$.

The cyclic code $\mathcal{C} = (g) \subset \mathbb{F}_q[X]/(X^n - 1)$ is called a BCH code (from its inventors Bose, Ray-Chaudhuri and Hocquenghem).

$\mathcal{C}$ is a $[n, k, d]_q$-linear code with $d \geq m + 1$, and $k \geq n - em$.

BCH Codes

Let $\beta \in \mathbb{F}_{q^e}$ be a n-th primitive root of unity, for a minimal e. (Recall that $n \mid q^e - 1$, or $q^e = 1 \bmod n$.)

For some integer $1 \leq m \leq n-1$ consider the minimal polynomial $g \in \mathbb{F}_q[X]$ of $\beta, \beta^2, \ldots, \beta^m$, that is the monic polynomial of minimal degree vanishing at $\beta, \beta^2, \ldots, \beta^m$.

Since $g \in \mathbb{F}_q[X]$, all the conjugates of $\beta, \beta^2, \ldots, \beta^m$ are roots of g. Thus, $\deg g \leq em$.

The cyclic code $\mathcal{C} = (g) \subset \mathbb{F}_q[X]/(X^n - 1)$ is called a BCH code (from its inventors Bose, Ray-Chaudhuri and Hocquenghem).

$\mathcal{C}$ is a $[n, k, d]_q$-linear code with $d \geq m + 1$, and $k \geq n - em$.

If $e = 1$, then $n \mid q - 1$, $\deg g = m$, $d = m + 1$ and $\mathcal{C}$ is MDS.

Proof of the lower bound $d \geq m + 1$

Assume that there is a codeword of weight m (or less)
$w = \lambda_1 X^{i_1} + \ldots + \lambda_m X^{i_m}$, where $i_1 < \cdots < i_m$

Proof of the lower bound $d \geq m+1$

Assume that there is a codeword of weight m (or less) $w = \lambda_1 X^{i_1} + \ldots + \lambda_m X^{i_m}$, where $i_1 < \cdots < i_m$

Then, $w \in \mathcal{C}$ implies that $w(\beta^j) = 0$ for $j = 1, \ldots, m$.
This means that $(\lambda_1, \ldots, \lambda_m)$ is a solution to

$$\begin{pmatrix} \beta^{i_1} & \beta^{i_2} & \ldots & \beta^{i_m} \\ \beta^{2i_1} & \beta^{2i_2} & \ldots & \beta^{2i_m} \\ \vdots & \vdots & & \vdots \\ \beta^{mi_1} & \beta^{mi_2} & \ldots & \beta^{mi_m} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Proof of the lower bound $d \geq m+1$

Assume that there is a codeword of weight m (or less) $w = \lambda_1 X^{i_1} + \ldots + \lambda_m X^{i_m}$, where $i_1 < \cdots < i_m$

Then, $w \in \mathcal{C}$ implies that $w(\beta^j) = 0$ for $j = 1, \ldots, m$.
This means that $(\lambda_1, \ldots, \lambda_m)$ is a solution to

$$\begin{pmatrix} \beta^{i_1} & \beta^{i_2} & \ldots & \beta^{i_m} \\ \beta^{2i_1} & \beta^{2i_2} & \ldots & \beta^{2i_m} \\ \vdots & \vdots & & \vdots \\ \beta^{mi_1} & \beta^{mi_2} & \ldots & \beta^{mi_m} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

but the only possible solution is $\lambda_1 = \cdots = \lambda_m = 0$

Proof of the lower bound $d \geq m+1$

Assume that there is a codeword of weight m (or less) $w = \lambda_1 X^{i_1} + \ldots + \lambda_m X^{i_m}$, where $i_1 < \cdots < i_m$

Then, $w \in \mathcal{C}$ implies that $w(\beta^j) = 0$ for $j = 1, \ldots, m$.
This means that $(\lambda_1, \ldots, \lambda_m)$ is a solution to

$$\begin{pmatrix} \beta^{i_1} & \beta^{i_2} & \ldots & \beta^{i_m} \\ \beta^{2i_1} & \beta^{2i_2} & \ldots & \beta^{2i_m} \\ \vdots & \vdots & & \vdots \\ \beta^{mi_1} & \beta^{mi_2} & \ldots & \beta^{mi_m} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

but the only possible solution is $\lambda_1 = \cdots = \lambda_m = 0$

The same proof works for any set of roots of the form $\beta^a, \beta^{a+b}, \ldots, \beta^{a+bm}$ for $\gcd(b, n) = 1$.

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ $(255 = 17 \cdot 15)$

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ $(255 = 17 \cdot 15)$
Primitive 17-th root of unity: $\beta = t^{15} = t^5 + t^2 + t$ or 26 (in hexadecimal)

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ $(255 = 17 \cdot 15)$
Primitive 17-th root of unity: $\beta = t^{15} = t^5 + t^2 + t$ or 26 (in hexadecimal)
Powers of β: 26 60 C1 B9 0F DF 1A 3B A9 55 91 96 64 59 24 2C

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ $(255 = 17 \cdot 15)$
Primitive 17-th root of unity: $\beta = t^{15} = t^5 + t^2 + t$ or 26 (in hexadecimal)
Powers of β: 26 60 C1 B9 0F DF 1A 3B A9 55 91 96 64 59 24 2C
Conjugate classes: $\{1\}$ $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{15}, \beta^{13}, \beta^9\}$
$\{\beta^3, \beta^6, \beta^{12}, \beta^7, \beta^{14}, \beta^{11}, \beta^5, \beta^{10}\}$

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ ($255 = 17 \cdot 15$)
Primitive 17-th root of unity: $\beta = t^{15} = t^5 + t^2 + t$ or 26 (in hexadecimal)
Powers of β: 26 60 C1 B9 0F DF 1A 3B A9 55 91 96 64 59 24 2C
Conjugate classes: $\{1\}$ $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{15}, \beta^{13}, \beta^9\}$
$\{\beta^3, \beta^6, \beta^{12}, \beta^7, \beta^{14}, \beta^{11}, \beta^5, \beta^{10}\}$

Taking $m = 2$: (The **design distance** is $d^* = m + 1 = 3$)
$g = (X - \beta)(X - \beta^2)(X - \beta^4)(X - \beta^8)(X - \beta^{16})(X - \beta^{15})(X - \beta^{13})(X - \beta^9) =$
$= X^8 + X^7 + X^6 + X^4 + X^2 + X + 1$
Dimension: $k = n - \deg g = 9$.
Minimum distance: $d \geq d^* = 3$

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ ($255 = 17 \cdot 15$)
Primitive 17-th root of unity: $\beta = t^{15} = t^5 + t^2 + t$ or 26 (in hexadecimal)
Powers of β: 26 60 C1 B9 0F DF 1A 3B A9 55 91 96 64 59 24 2C
Conjugate classes: $\{1\}$ $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{15}, \beta^{13}, \beta^9\}$
$\{\beta^3, \beta^6, \beta^{12}, \beta^7, \beta^{14}, \beta^{11}, \beta^5, \beta^{10}\}$

Taking $m = 2$: (The **design distance** is $d^* = m + 1 = 3$)
$g = (X - \beta)(X - \beta^2)(X - \beta^4)(X - \beta^8)(X - \beta^{16})(X - \beta^{15})(X - \beta^{13})(X - \beta^9) =$
$= X^8 + X^7 + X^6 + X^4 + X^2 + X + 1$
Dimension: $k = n - \deg g = 9$.
Minimum distance: $d \geq d^* = 3$

Computing actual minimum distance: $d = 5$

Example: $[17, 9, 5]_2$ BCH Code

$\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t^2 + 1)$
Primitive element: t

$n = 17$ ($255 = 17 \cdot 15$)
Primitive 17-th root of unity: $\beta = t^{15} = t^5 + t^2 + t$ or 26 (in hexadecimal)
Powers of β: 26 60 C1 B9 0F DF 1A 3B A9 55 91 96 64 59 24 2C
Conjugate classes: $\{1\}$ $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{15}, \beta^{13}, \beta^9\}$
$\{\beta^3, \beta^6, \beta^{12}, \beta^7, \beta^{14}, \beta^{11}, \beta^5, \beta^{10}\}$

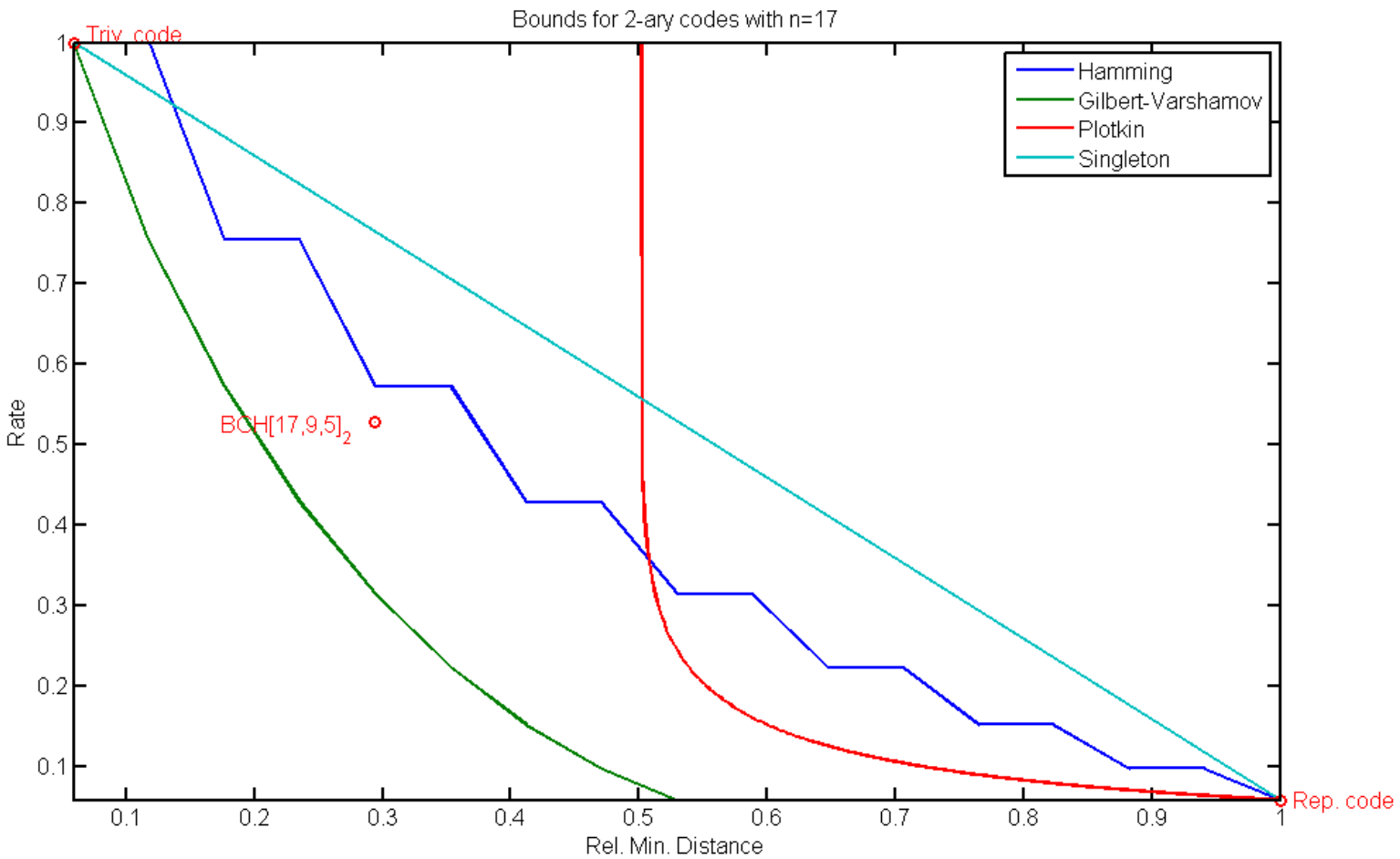
Taking $m = 2$: (The **design distance** is $d^* = m + 1 = 3$)
$g = (X - \beta)(X - \beta^2)(X - \beta^4)(X - \beta^8)(X - \beta^{16})(X - \beta^{15})(X - \beta^{13})(X - \beta^9) =$
$= X^8 + X^7 + X^6 + X^4 + X^2 + X + 1$
Dimension: $k = n - \deg g = 9$.
Minimum distance: $d \geq d^* = 3$

Computing actual minimum distance: $d = 5$

$\mathcal{C}$ is a $[17, 9, 5]_2$ BCH code, with an optimal dimension.

The sphere packing bound reads here $k \leq \left\lfloor 17 - \log_2(1 + 17 + \binom{17}{2}) \right\rfloor = 9$

Example: $[17, 9, 5]_2$ BCH Code



Further Comments About Reed-Solomon Codes

The length of a cyclic Reed-Solomon $[n, k, n-k+1]_q$-code $\mathcal{C}$ fulfils $n \mid q-1$.

Further Comments About Reed-Solomon Codes

The length of a cyclic Reed-Solomon $[n, k, n-k+1]_q$-code $\mathcal{C}$ fulfils $n \mid q-1$.

However, the cyclic Reed-Solomon code can be shortened to any length $n' \geq k$ by defining $\mathcal{C}' = (\mathbb{F}_q[X]_{\leq k-1})\, g$, where

$$g = \prod_{i=1}^{n'-k} (X - \beta^{-i})$$

and $\beta = \alpha^{\frac{q-1}{n}}$, for a primitive element α.

Further Comments About Reed-Solomon Codes

The length of a cyclic Reed-Solomon $[n, k, n-k+1]_q$-code $\mathcal{C}$ fulfils $n \mid q-1$.

However, the cyclic Reed-Solomon code can be shortened to any length $n' \geq k$ by defining $\mathcal{C}' = (\mathbb{F}_q[X]_{\leq k-1})\, g$, where

$$g = \prod_{i=1}^{n'-k} (X - \beta^{-i})$$

and $\beta = \alpha^{\frac{q-1}{n}}$, for a primitive element α.

- $\mathcal{C}'$ is not cyclic (because $g \nmid X^{n'} - 1$), but

Further Comments About Reed-Solomon Codes

The length of a cyclic Reed-Solomon $[n, k, n-k+1]_q$-code $\mathcal{C}$ fulfils $n \mid q-1$.

However, the cyclic Reed-Solomon code can be shortened to any length $n' \geq k$ by defining $\mathcal{C}' = (\mathbb{F}_q[X]_{\leq k-1})\, g$, where

$$g = \prod_{i=1}^{n'-k} (X - \beta^{-i})$$

and $\beta = \alpha^{\frac{q-1}{n}}$, for a primitive element α.

- $\mathcal{C}'$ is not cyclic (because $g \nmid X^{n'} - 1$), but
- $\mathcal{C}'$ is an MDS code,

Further Comments About Reed-Solomon Codes

The length of a cyclic Reed-Solomon $[n, k, n-k+1]_q$-code $\mathcal{C}$ fulfils $n \mid q-1$.

However, the cyclic Reed-Solomon code can be shortened to any length $n' \geq k$ by defining $\mathcal{C}' = (\mathbb{F}_q[X]_{\leq k-1})\, g$, where

$$g = \prod_{i=1}^{n'-k} (X - \beta^{-i})$$

and $\beta = \alpha^{\frac{q-1}{n}}$, for a primitive element α.

- $\mathcal{C}'$ is not cyclic (because $g \nmid X^{n'} - 1$), but
- $\mathcal{C}'$ is an MDS code,
- encoding in $\mathcal{C}'$ can be computed by division by g,

Further Comments About Reed-Solomon Codes

The length of a cyclic Reed-Solomon $[n, k, n-k+1]_q$-code $\mathcal{C}$ fulfils $n \mid q-1$.

However, the cyclic Reed-Solomon code can be shortened to any length $n' \geq k$ by defining $\mathcal{C}' = (\mathbb{F}_q[X]_{\leq k-1})\, g$, where

$$g = \prod_{i=1}^{n'-k} (X - \beta^{-i})$$

and $\beta = \alpha^{\frac{q-1}{n}}$, for a primitive element α.

- $\mathcal{C}'$ is not cyclic (because $g \nmid X^{n'} - 1$), but
- $\mathcal{C}'$ is an MDS code,
- encoding in $\mathcal{C}'$ can be computed by division by g,
- error correction in $\mathcal{C}'$ works the same way as in the original cyclic Reed-Solomon code $\mathcal{C}$.

Outline

Dealing With Error Bursts

Formerly, we only considered random independent errors.

Dealing With Error Bursts

Formerly, we only considered random independent errors.

But in some real applications correlated errors occur:

Dealing With Error Bursts

Formerly, we only considered random independent errors.

But in some real applications correlated errors occur:

- Errors occurs more frequently at adjacent positions.
  Communication channel suffers a transient malfunction or interference.

Dealing With Error Bursts

Formerly, we only considered random independent errors.

But in some real applications correlated errors occur:

- Errors occurs more frequently at adjacent positions.
  Communication channel suffers a transient malfunction or interference.
- Errors tend to follow some predefined timing pattern.
  Because symbols are stored into a two dimensional structure, and there are new adjacency relations.

Dealing With Error Bursts

Formerly, we only considered random independent errors.

But in some real applications correlated errors occur:

- Errors occurs more frequently at adjacent positions. Communication channel suffers a transient malfunction or interference.
- Errors tend to follow some predefined timing pattern. Because symbols are stored into a two dimensional structure, and there are new adjacency relations.

This makes errors accumulate in a single codeword, typically exceeding the error correction capability of the code.

Concatenation of Codes

Concatenation of codes: Building codes with a large length from two smaller codes.

Build a complex code from two simpler ones by composition of the encoding functions. Then apply the two layers of error correction.

Concatenation of Codes

Concatenation of codes: Building codes with a large length from two smaller codes.

Build a complex code from two simpler ones by composition of the encoding functions. Then apply the two layers of error correction.

The resulting encoding function is simpler than the encoding function of a directly designed large code.

Correcting a large number of errors is also more efficient.

Adds protection against some specific error patterns.

Concatenation of Linear Codes (I)

Let $\mathcal{C}_1 \subset (\mathbb{F}_{q_1})^{n_1}$ be a $[n_1, k_1, d_1]_{q_1}$-linear code, called the **inner code**, and let $\mathcal{C}_2 \subset (\mathbb{F}_{q_2})^{n_2}$ be a $[n_2, k_2, d_2]_{q_2}$-linear code, called the **outer code**.

Concatenation of Linear Codes (I)

Let $\mathcal{C}_1 \subset (\mathbb{F}_{q_1})^{n_1}$ be a $[n_1, k_1, d_1]_{q_1}$-linear code, called the **inner code**, and let $\mathcal{C}_2 \subset (\mathbb{F}_{q_2})^{n_2}$ be a $[n_2, k_2, d_2]_{q_2}$-linear code, called the **outer code**.

We assume the existence of a q-linear bijection $\psi : \mathbb{F}_{q_2}^{n_2} \to \mathbb{F}_{q_1}^{k_1}$
Thus, $q_1 = q^{e_1}$, $q_2 = q^{e_2}$ for some q, e_1, e_2 and $e_1 k_1 = e_2 n_2$.
We also use natural q-linear bijections $\mathbb{F}_{q^{e_1}} \cong \mathbb{F}_q^{e_1}$ and $\mathbb{F}_{q^{e_2}} \cong \mathbb{F}_q^{e_2}$.

Concatenation of Linear Codes (I)

Let $\mathcal{C}_1 \subset (\mathbb{F}_{q_1})^{n_1}$ be a $[n_1, k_1, d_1]_{q_1}$-linear code, called the **inner code**, and let $\mathcal{C}_2 \subset (\mathbb{F}_{q_2})^{n_2}$ be a $[n_2, k_2, d_2]_{q_2}$-linear code, called the **outer code**.

We assume the existence of a q-linear bijection $\psi : \mathbb{F}_{q_2}^{n_2} \to \mathbb{F}_{q_1}^{k_1}$
Thus, $q_1 = q^{e_1}$, $q_2 = q^{e_2}$ for some q, e_1, e_2 and $e_1 k_1 = e_2 n_2$.
We also use natural q-linear bijections $\mathbb{F}_{q^{e_1}} \cong \mathbb{F}_q^{e_1}$ and $\mathbb{F}_{q^{e_2}} \cong \mathbb{F}_q^{e_2}$.

The **concatenated code** $\mathcal{C} \subset (\mathbb{F}_q)^{e_1 n_1}$ is defined as $\mathcal{C} = Enc_1 \circ \psi \circ Enc_2((\mathbb{F}_{q^{e_2}})^{k_2})$, which is a $[e_1 n_1, e_2 k_2, d]_q$-linear code for some $d \geq d_1$.

Concatenation of Linear Codes (I)

Let $\mathcal{C}_1 \subset (\mathbb{F}_{q_1})^{n_1}$ be a $[n_1, k_1, d_1]_{q_1}$-linear code, called the **inner code**, and let $\mathcal{C}_2 \subset (\mathbb{F}_{q_2})^{n_2}$ be a $[n_2, k_2, d_2]_{q_2}$-linear code, called the **outer code**.

We assume the existence of a q-linear bijection $\psi : \mathbb{F}_{q_2}^{n_2} \to \mathbb{F}_{q_1}^{k_1}$ Thus, $q_1 = q^{e_1}$, $q_2 = q^{e_2}$ for some q, e_1, e_2 and $e_1 k_1 = e_2 n_2$. We also use natural q-linear bijections $\mathbb{F}_{q^{e_1}} \cong \mathbb{F}_q^{e_1}$ and $\mathbb{F}_{q^{e_2}} \cong \mathbb{F}_q^{e_2}$.

The **concatenated code** $\mathcal{C} \subset (\mathbb{F}_q)^{e_1 n_1}$ is defined as $\mathcal{C} = Enc_1 \circ \psi \circ Enc_2((\mathbb{F}_{q^{e_2}})^{k_2})$, which is a $[e_1 n_1, e_2 k_2, d]_q$-linear code for some $d \geq d_1$.

Observe that the encoding map for $\mathcal{C}$ is the composition:

$$\mathbb{F}_q^{e_2 k_2} \cong (\mathbb{F}_{q^{e_2}})^{k_2} \xrightarrow{Enc_2} (\mathbb{F}_{q^{e_2}})^{n_2} \cong (\mathbb{F}_{q^{e_1}})^{k_1} \xrightarrow{Enc_1} (\mathbb{F}_{q^{e_1}})^{n_1} \cong \mathbb{F}_q^{e_1 n_1}$$

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$s_{11} \cdots s_{1k_1'} s_{21} \cdots s_{2k_1'} \cdots s_{k_2 1} \cdots s_{k_2 k_1'}$

Source sequence in $\mathbb{F}_q^{k_1' k_2}$

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$$\underbrace{s_{11} \cdots s_{1k_1'}}_{\mathbf{s}_1} \underbrace{s_{21} \cdots s_{2k_1'}}_{\mathbf{s}_2} \cdots \underbrace{s_{k_2 1} \cdots s_{k_2 k_1'}}_{\mathbf{s}_{k_2}}$$

Source sequence in $(\mathbb{F}_q^{k_1'})^{k_2} \cong (\mathbb{F}_{q^{e_2}})^{k_2}$

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$$s_{11} \cdots s_{1k_1'} s_{21} \cdots s_{2k_1'} \cdots s_{k_2 1} \cdots s_{k_2 k_1'}$$

$$\begin{array}{cccccc} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_{k_2} & & \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{k_2} & \cdots & \mathbf{x}_{n_2} \end{array}$$

Outer codeword in $\mathcal{C}_2 \subset (\mathbb{F}_{q^{e_2}})^{n_2} \cong (\mathbb{F}_q^{k_1'})^{n_2}$

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$s_{11} \cdots s_{1k_1'} s_{21} \cdots s_{2k_1'} \cdots s_{k_2 1} \cdots s_{k_2 k_1'}$

$\mathbf{s}_1$	$\mathbf{s}_2$	$\cdots$	$\mathbf{s}_{k_2}$		
$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_{k_2}$	$\cdots$	$\mathbf{x}_{n_2}$
$\mathbf{y}_1$	$\mathbf{y}_2$	$\cdots$	$\mathbf{y}_{k_2}$	$\cdots$	$\mathbf{y}_{n_2}$

Inner codeword sequence in $(\mathcal{C}_1')^{n_2} \subset (\mathbb{F}_q^{n_1'})^{n_2}$

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$$
\begin{array}{llll}
s_{11} \cdots s_{1k_1'} & s_{21} \cdots s_{2k_1'} & \cdots s_{k_2 1} \cdots s_{k_2 k_1'} & \\
\mathbf{s}_1 & \mathbf{s}_2 & \cdots \quad \mathbf{s}_{k_2} & \\
\mathbf{x}_1 & \mathbf{x}_2 & \cdots \quad \mathbf{x}_{k_2} & \cdots \quad \mathbf{x}_{n_2} \\
\mathbf{y}_1 & \mathbf{y}_2 & \cdots \quad \mathbf{y}_{k_2} & \cdots \quad \mathbf{y}_{n_2} \\
y_{11} \cdots y_{1k_1'} \cdots y_{1n_1'} & y_{21} \cdots y_{2k_1'} \cdots y_{2n_1'} & \cdots y_{k_2 1} \cdots y_{k_2 k_1'} \cdots y_{k_2 n_1'} & \cdots y_{n_2 1} \cdots y_{n_2 n_1'}
\end{array}
$$

Final encoded sequence in $\mathbb{F}_q^{n_1' n_2}$

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$$
\begin{array}{cccccccccc}
s_{11} \cdots s_{1k_1'} & & s_{21} \cdots s_{2k_1'} & & \cdots & s_{k_2 1} \cdots s_{k_2 k_1'} & & & & \\
\mathbf{s}_1 & & \mathbf{s}_2 & & \cdots & \mathbf{s}_{k_2} & & & & \\
\mathbf{x}_1 & & \mathbf{x}_2 & & \cdots & \mathbf{x}_{k_2} & & \cdots & \mathbf{x}_{n_2} & \\
\mathbf{y}_1 & & \mathbf{y}_2 & & \cdots & \mathbf{y}_{k_2} & & \cdots & \mathbf{y}_{n_2} & \\
y_{11} \cdots y_{1k_1'} & \cdots y_{1n_1'} & y_{21} \cdots y_{2k_1'} & \cdots y_{2n_1'} & \cdots & y_{k_2 1} \cdots y_{k_2 k_1'} & \cdots & y_{k_2 n_1'} & \cdots y_{n_2 1} \cdots y_{n_2 n_1'} &
\end{array}
$$

The simplest example is taking $\mathcal{C}_1'$ the trivial code $\mathbb{F}_q^{k_1'}$.

Concatenation of Linear Codes (II)

As a particular case, $e_1 = 1$, $e_2 = k_1'$ and $\mathcal{C}_1$ consists of n_2 copies of a $[n_1', k_1', d_1']$ code $\mathcal{C}_1'$ (Each symbol in the outer codeword is encoded separately with $\mathcal{C}_1'$)
Then, $\mathcal{C}_1$ is $[n_1' n_2, k_1' k_2, d_1']_q$ and $d \geq d_1' d_2$.

$$
\begin{array}{cccccccccc}
s_{11} \cdots s_{1k_1'} & & s_{21} \cdots s_{2k_1'} & & \cdots & s_{k_2 1} \cdots s_{k_2 k_1'} & & & & \\
\mathbf{s}_1 & & \mathbf{s}_2 & & \cdots & \mathbf{s}_{k_2} & & & & \\
\mathbf{x}_1 & & \mathbf{x}_2 & & \cdots & \mathbf{x}_{k_2} & & \cdots & \mathbf{x}_{n_2} & \\
\mathbf{y}_1 & & \mathbf{y}_2 & & \cdots & \mathbf{y}_{k_2} & & \cdots & \mathbf{y}_{n_2} & \\
y_{11} \cdots y_{1k_1'} & \cdots & y_{1n_1'} & y_{21} \cdots y_{2k_1'} \cdots y_{2n_1'} & \cdots & y_{k_2 1} \cdots y_{k_2 k_1'} & \cdots & y_{k_2 n_1'} & \cdots & y_{n_2 1} \cdots y_{n_2 n_1'}
\end{array}
$$

The simplest example is taking $\mathcal{C}_1'$ the trivial code $\mathbb{F}_q^{k_1'}$.

This is just looking to a $[n, k, d]_{q^e}$ code as a $[en, ek, ed]_q$ code, which has the same rate, the same random error correction capability, but it can correct up to $\lfloor \frac{d_2 - 1}{4} \rfloor$ bursts of length e.

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Inner code: 15 copies of the previous example of $[17, 9, 5]_2$ BCH code

- it corrects up to 2 errors (each copy)
- it detects up to 4 errors (each copy)

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Inner code: 15 copies of the previous example of $[17, 9, 5]_2$ BCH code

- it corrects up to 2 errors (each copy)
- it detects up to 4 errors (each copy)

Outer code: A shortened $[15, 11, 5]_{2^9}$ Reed-Solomon code

- it corrects up to 2 errors
- it corrects up to 1 error and 2 erasures
- it corrects up to 4 erasures

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Inner code: 15 copies of the previous example of $[17, 9, 5]_2$ BCH code

- it corrects up to 2 errors (each copy)
- it detects up to 4 errors (each copy)

Outer code: A shortened $[15, 11, 5]_{2^9}$ Reed-Solomon code

- it corrects up to 2 errors
- it corrects up to 1 error and 2 erasures
- it corrects up to 4 erasures

The concatenated code is a $[255, 99, d]_2$ code with $d \geq 25$.

- it corrects at least any 12 errors
- it can handle more errors if they follow some pattern...

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Decoding Strategy 1: Use the inner code to correct up to 2 errors per inner codeword. If error correction is impossible, mark the inner codeword as erasure.

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Decoding Strategy 1: Use the inner code to correct up to 2 errors per inner codeword. If error correction is impossible, mark the inner codeword as erasure.

Bad inner decoding can happen from 3 errors.
Bad outer decoding typically happens when there are more than 2 errors in more than 2 inner codewords.

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Decoding Strategy 1: Use the inner code to correct up to 2 errors per inner codeword. If error correction is impossible, mark the inner codeword as erasure.

Bad inner decoding can happen from 3 errors.
Bad outer decoding typically happens when there are more than 2 errors in more than 2 inner codewords.

Decoding Strategy 2: Use the inner code to correct up to 1 error. If error correction is impossible, mark the inner codeword as erasure.

Example: Inner $15 \times [17, 9, 5]_2$, Outer $[15, 11, 5]_{2^9}$

Decoding Strategy 1: Use the inner code to correct up to 2 errors per inner codeword. If error correction is impossible, mark the inner codeword as erasure.

Bad inner decoding can happen from 3 errors.
Bad outer decoding typically happens when there are more than 2 errors in more than 2 inner codewords.

Decoding Strategy 2: Use the inner code to correct up to 1 error. If error correction is impossible, mark the inner codeword as erasure.

Bad inner decoding can only happen from 4 errors. All inner codeword with 2 or 3 errors are marked as erasures.
Bad outer decoding typically happens when there are more than 1 error in more than 4 inner codewords.

Concatenation of Linear Codes (III)

Another approach is using $e_1 = e_2 = 1$ and then $n_2 = k_1$. The encoding sequence is just

$$\mathbb{F}_q^{k_2} \xrightarrow{Enc_2} (\mathbb{F}_q)^{n_2} = (\mathbb{F}_q)^{k_1} \xrightarrow{Enc_1} (\mathbb{F}_q)^{n_1}$$

Concatenation of Linear Codes (III)

Another approach is using $e_1 = e_2 = 1$ and then $n_2 = k_1$. The encoding sequence is just

$$\mathbb{F}_q^{k_2} \xrightarrow{Enc_2} (\mathbb{F}_q)^{n_2} = (\mathbb{F}_q)^{k_1} \xrightarrow{Enc_1} (\mathbb{F}_q)^{n_1}$$

For instance, we can use a $[28, 24, 5]_{2^8}$ shortened Reed-Solomon code as outer code and a $[32, 28, 5]_{2^8}$ shortened Reed-Solomon code as inner code.

Each layer adds four redundancy symbols (1024 bits) and can correct up to 2 errors or 4 erasures.

Concatenation of Linear Codes (III)

Another approach is using $e_1 = e_2 = 1$ and then $n_2 = k_1$. The encoding sequence is just

$$\mathbb{F}_q^{k_2} \xrightarrow{Enc_2} (\mathbb{F}_q)^{n_2} = (\mathbb{F}_q)^{k_1} \xrightarrow{Enc_1} (\mathbb{F}_q)^{n_1}$$

For instance, we can use a $[28, 24, 5]_{2^8}$ shortened Reed-Solomon code as outer code and a $[32, 28, 5]_{2^8}$ shortened Reed-Solomon code as inner code.

Each layer adds four redundancy symbols (1024 bits) and can correct up to 2 errors or 4 erasures.

Reinterpreting the code as a binary code (i.e., concatenating again with (32 copies of) the trivial code $\mathbb{F}_2^8$) we add some error burst resilience.

Concatenation of Linear Codes (III)

Another approach is using $e_1 = e_2 = 1$ and then $n_2 = k_1$. The encoding sequence is just

$$\mathbb{F}_q^{k_2} \xrightarrow{Enc_2} (\mathbb{F}_q)^{n_2} = (\mathbb{F}_q)^{k_1} \xrightarrow{Enc_1} (\mathbb{F}_q)^{n_1}$$

For instance, we can use a $[28, 24, 5]_{2^8}$ shortened Reed-Solomon code as outer code and a $[32, 28, 5]_{2^8}$ shortened Reed-Solomon code as inner code.

Each layer adds four redundancy symbols (1024 bits) and can correct up to 2 errors or 4 erasures.

Reinterpreting the code as a binary code (i.e., concatenating again with (32 copies of) the trivial code $\mathbb{F}_2^8$) we add some error burst resilience.

. . . but the benefits come when concatenation os combined with interleaving. . .

Outline

Interleaving

Interleaving is just shuffling the symbols in the codewords in order to break the adjacency relations.

Interleaving

Interleaving is just shuffling the symbols in the codewords in order to break the adjacency relations.

This transforms error bursts into errors at “randomly looking” positions, which typically are better distributed among the different interleaved codewords.

Interleaving

Interleaving is just shuffling the symbols in the codewords in order to break the adjacency relations.

This transforms error bursts into errors at "randomly looking" positions, which typically are better distributed among the different interleaved codewords.

For instance, the simplest way to interleave m codewords $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbf{F}_q^n$ is writing them as rows in a matrix and reading the matrix columnwise.

$$\begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix} \longrightarrow (x_{11}, x_{21}, \ldots, x_{m1}, \ldots, x_{mn})$$

Interleaving

Interleaving is just shuffling the symbols in the codewords in order to break the adjacency relations.

This transforms error bursts into errors at "randomly looking" positions, which typically are better distributed among the different interleaved codewords.

For instance, the simplest way to interleave m codewords $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbf{F}_q^n$ is writing them as rows in a matrix and reading the matrix columnwise.

$$\begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix} \longrightarrow (x_{11}, x_{21}, \ldots, x_{m1}, \ldots, x_{mn})$$

Then, an error burst of length $\leq r$ can only cause up to $\lceil \frac{r}{m} \rceil$ errors in the same codeword.

Cross Interleaving

If the source information is a data stream, interleaving can be based on delays.

Cross Interleaving

If the source information is a data stream, interleaving can be based on delays.

Starting from a sequence of codewords $\mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathcal{C} \subset \mathbb{F}_q^n$, or the corresponding sequence of symbols in $\mathbb{F}_q$

$$\ldots x_{1,1} x_{1,2} \ldots x_{1,n}\ x_{2,1} x_{2,2} \ldots x_{2,n}\ x_{3,1} x_{3,2} \ldots x_{3,n} \ldots$$

we construct the interleaved sequence of symbols in $\mathbb{F}_q$

$$\ldots x_{n,1} x_{n-1,2} \ldots x_{1,n}\ x_{n+1,1} x_{n,2} \ldots x_{2,n}\ x_{n+2,1} x_{n+1,2} \ldots x_{3,n} \ldots$$

which can be seen as applying delays $0, +1, +2, \ldots, +(n-1)$ to the symbols in the codeword.

Cross Interleaving

If the source information is a data stream, interleaving can be based on delays.

Starting from a sequence of codewords $\mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathcal{C} \subset \mathbb{F}_q^n$, or the corresponding sequence of symbols in $\mathbb{F}_q$

$$\ldots x_{1,1} x_{1,2} \ldots x_{1,n}\ x_{2,1} x_{2,2} \ldots x_{2,n}\ x_{3,1} x_{3,2} \ldots x_{3,n} \ldots$$

we construct the interleaved sequence of symbols in $\mathbb{F}_q$

$$\ldots x_{n,1} x_{n-1,2} \ldots x_{1,n}\ x_{n+1,1} x_{n,2} \ldots x_{2,n}\ x_{n+2,1} x_{n+1,2} \ldots x_{3,n} \ldots$$

which can be seen as applying delays $0, +1, +2, \ldots, +(n-1)$ to the symbols in the codeword.

An error burst of length $n+1$ is deinterleaved into single errors affecting $n+1$ codewords, which will be easily corrected.

Example of Cross Interleaving

For example, consider $n = 4$, the codeword $\mathbf{x}_i$ is represented as (a_i, b_i, c_i, d_i), and undefined symbols are set to 0 (this only applies to the start and end of the stream).

Example of Cross Interleaving

For example, consider $n = 4$, the codeword $\mathbf{x}_i$ is represented as (a_i, b_i, c_i, d_i), and undefined symbols are set to 0 (this only applies to the start and end of the stream).

$$\begin{array}{l} \cdots 00 a_1 a_2 a_3 a_4 a_5 \cdots \\ \cdots 00 b_1 b_2 b_3 b_4 b_5 \cdots \\ \cdots 00 c_1 c_2 c_3 c_4 c_5 \cdots \\ \cdots 00 d_1 d_2 d_3 d_4 d_5 \cdots \end{array} \rightarrow \quad a_1 000 a_2 b_1 00 a_3 b_2 c_1 0 a_4 b_3 c_2 d_1 a_5 b_4 c_3 d_2 \cdots$$

Example of Cross Interleaving

For example, consider $n = 4$, the codeword $\mathbf{x}_i$ is represented as (a_i, b_i, c_i, d_i), and undefined symbols are set to 0 (this only applies to the start and end of the stream).

$$\begin{array}{l} \cdots 00 a_1 a_2 a_3 a_4 a_5 \cdots \\ \cdots 00 b_1 b_2 b_3 b_4 b_5 \cdots \\ \cdots 00 c_1 c_2 c_3 c_4 c_5 \cdots \\ \cdots 00 d_1 d_2 d_3 d_4 d_5 \cdots \end{array} \longrightarrow a_1 000 a_2 b_1 00 a_3 b_2 c_1 0 a_4 b_3 c_2 d_1 a_5 b_4 c_3 d_2 \cdots$$

An error burst of length 5 is transformed into single errors in 5 consecutive codewords.

Concatenation and Interleaving (I)

In practice, interleaving and concatenation are combined to achieve resilience to long error bursts.

Concatenation and Interleaving (I)

In practice, interleaving and concatenation are combined to achieve resilience to long error bursts.

Consider now cross interleaving after encoding with a $[6, 4, 3]$ inner code.

$$
\begin{array}{l}
a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9 \cdots \\
b_1\, b_2\, b_3\, b_4\, b_5\, b_6\, b_7\, b_8\, b_9 \cdots \\
c_1\, c_2\, c_3\, c_4\, c_5\, c_6\, c_7\, c_8\, c_9 \cdots \\
d_1\, d_2\, d_3\, d_4\, d_5\, d_6\, d_7\, d_8\, d_9 \cdots \\
e_1\, e_2\, e_3\, e_4\, e_5\, e_6\, e_7\, e_8\, e_9 \cdots \\
f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7\ f_8\ f_9 \cdots
\end{array}
$$

Concatenation and Interleaving (I)

In practice, interleaving and concatenation are combined to achieve resilience to long error bursts.

Consider now cross interleaving after encoding with a $[6, 4, 3]$ inner code.

$$
\begin{array}{l}
a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9 \cdots \\
b_1\, b_2\, b_3\, b_4\, b_5\, b_6\, b_7\, b_8\, b_9 \cdots \\
c_1\, c_2\, c_3\, c_4\, c_5\, c_6\, c_7\, c_8\, c_9 \cdots \\
d_1\, d_2\, d_3\, d_4\, d_5\, d_6\, d_7\, d_8\, d_9 \cdots \\
e_1\, e_2\, e_3\, e_4\, e_5\, e_6\, e_7\, e_8\, e_9 \cdots \\
f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7\ f_8\ f_9 \cdots \\
\\
1\ 2\ 2\ 2\ 3\ 2\ 1\ 0\ 0 \cdots
\end{array}
$$

An error burst of length 13 produces up to three errors in the same inner codeword.

Concatenation and Interleaving (I)

In practice, interleaving and concatenation are combined to achieve resilience to long error bursts.

Consider now cross interleaving after encoding with a $[6, 4, 3]$ inner code.

$$\begin{array}{l} a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9 \cdots \\ b_1\, b_2\, b_3\, b_4\, b_5\, b_6\, b_7\, b_8\, b_9 \cdots \\ c_1\, c_2\, c_3\, c_4\, c_5\, c_6\, c_7\, c_8\, c_9 \cdots \\ d_1\, d_2\, d_3\, d_4\, d_5\, d_6\, d_7\, d_8\, d_9 \cdots \\ e_1\, e_2\, e_3\, e_4\, e_5\, e_6\, e_7\, e_8\, e_9 \cdots \\ f_1\ \ f_2\ \ f_3\ \ f_4\ \ f_5\ \ f_6\ \ f_7\ \ f_8\ \ f_9 \cdots \\ \\ 1\ \ 2\ \ 2\ \ 2\ \ 3\ \ 2\ \ 1\ \ 0\ \ 0 \cdots \end{array} \quad \rightarrow \quad \begin{array}{l} a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9 \cdots \\ b_1\, b_2\, b_3\, b_4\, b_5\, b_6\, b_7\, b_8\, b_9 \cdots \\ c_1\, c_2\, c_3\, c_4\, c_5\, c_6\, c_7\, c_8\, c_9 \cdots \\ d_1\, d_2\, d_3\, d_4\, d_5\, d_6\, d_7\, d_8\, d_9 \cdots \end{array}$$

An error burst of length 13 produces up to three errors in the same inner codeword.

Concatenation and Interleaving (II)

With another layer of interleaving (e.g., double delay cross interleaving) after encoding with an outer code of minimum distance 4 (e.g., the repetition code $[4, 1, 4]$) all errors and erasures are successfully corrected!

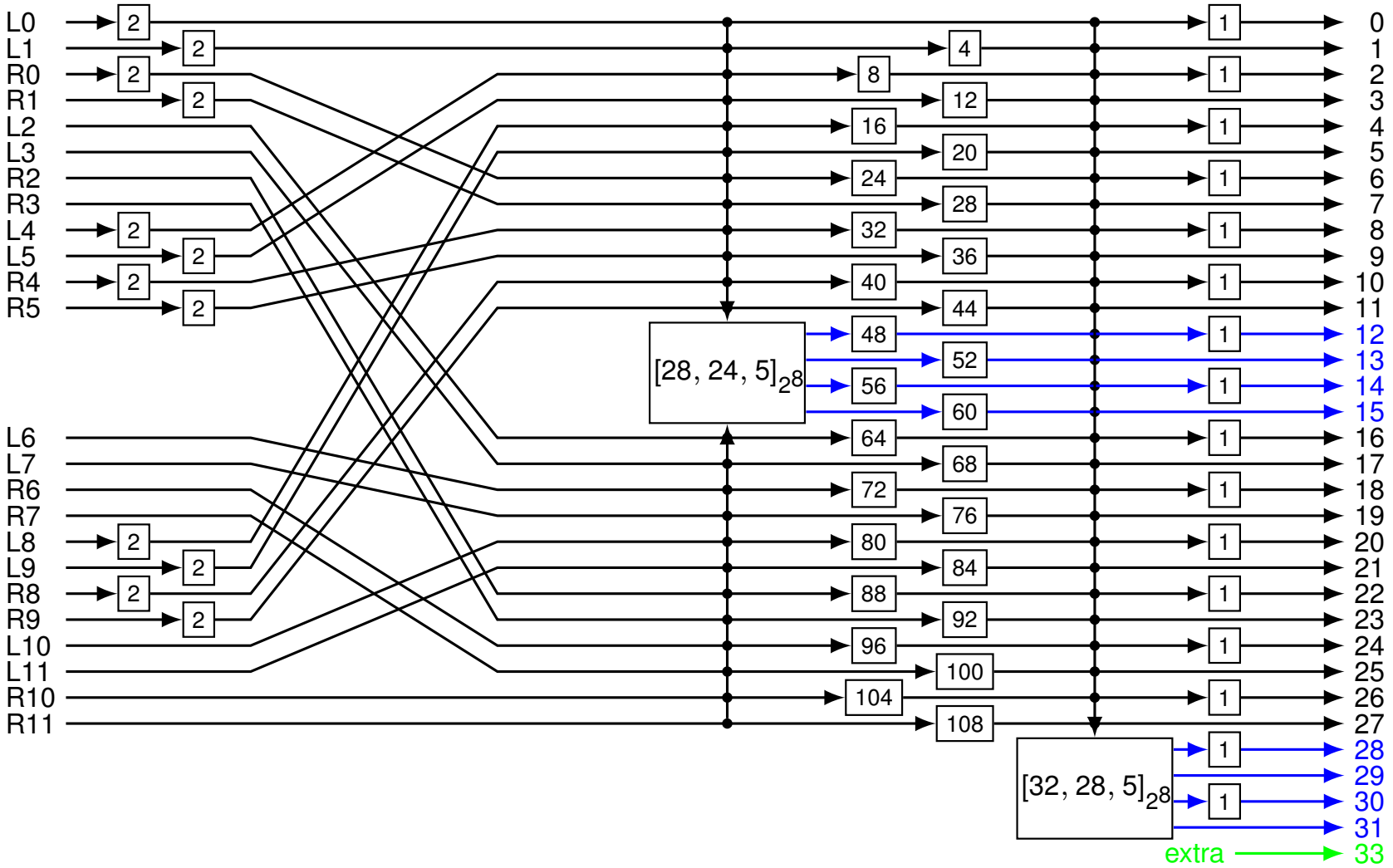
Concatenation and Interleaving (II)

With another layer of interleaving (e.g., double delay cross interleaving) after encoding with an outer code of minimum distance 4 (e.g., the repetition code $[4, 1, 4]$) all errors and erasures are successfully corrected!

$$
\begin{array}{llllllllllll}
\cdots & a_{-4} & a_{-3} & a_{-2} & a_{-1} & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \cdots \\
\cdots & b_{-2} & b_{-1} & b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \cdots \\
\cdots & c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} \cdots \\
\cdots & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 & d_8 & d_9 & c_{10} & c_{11} & c_{12} \cdots \\
\cdots & 1 & 1 & 2 & 1 & 3 & 1 & 3 & 1 & 2 & 0 & 1 \cdots \quad \text{(erasures)} \\
\cdots & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \cdots \quad \text{(errors)}
\end{array}
$$

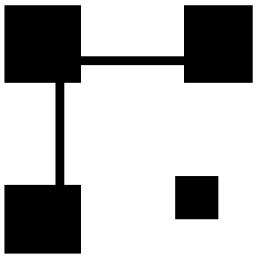
Outline

Compact Disk Interleaving Diagram

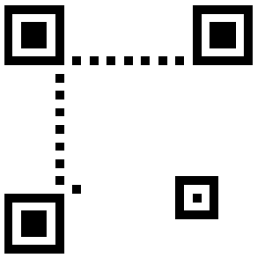


QR Barcodes Interleaving (I)

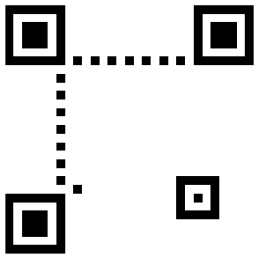
Reserved Areas



Alignement Marks



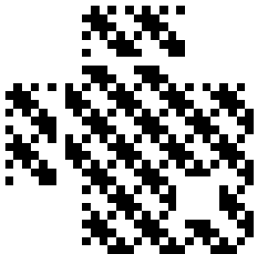
No Version Info Added



QR Unmasked

with Format Info

Mask



QR Code (w/o border)

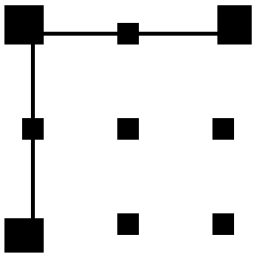
QR Code (Final)

Interleaving Map

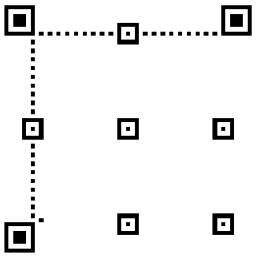
QR Code version number: 3
QR Code size (w/o border): 29×29
Available space: 70 bytes
Error correction level: H
Using field $\mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X^2 + 1)$ for data encoding
Using 2 blocks of shortened RS code $[35, 13, 23]_{256}$
Generating polynomial coeffs. (in hexadecimal)
01 59 B3 83 B0 B6 F4 13 BD 45 28 1C 89 1D 7B 43 FD 56 DA E6 1A 91 F5
Total raw capacity: 26 bytes
Total parity bytes: 44
Data preencoded in 'binary' format
Encoded blocks (in hexadecimal):
40 E4 86 56 C6 C6 F2 05 76 F7 26 C6 42 D8 BB 91 43 DE F7 47 6F AE 88 42 42 A4 4B A5 5C E5 25 81 3C 55 08
12 12 10 EC 11 EC 11 EC 11 EC 11 EC 11 F6 7B 07 19 07 03 1E 79 F5 84 9C 12 31 DB 03 D3 FA 66 C8 A4 86 AE
Interleaved blocks (in hexadecimal):
40 12 E4 12 86 10 56 EC C6 11 C6 EC F2 11 05 EC 76 11 F7 EC 26 11 C6 EC 42 11 D8 F6 BB 7B 91 07 43 19 DE 07 F7 03 47 1E 6F 79 AE F5 88 84 42 9C 42 12 A4 31 4B DB A5 03 5C D3 E5 FA 25 66 81 C8 3C A4 55 86 08 AE

QR Barcodes Interleaving (II)

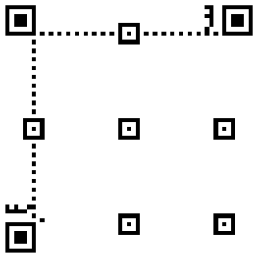
Reserved Areas



Alignement Marks



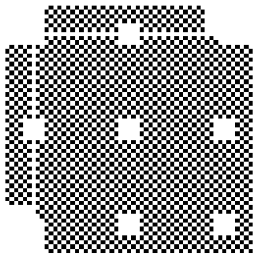
Version Info Added



QR Unmasked

with Format Info

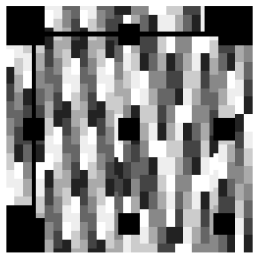
Mask



QR Code (w/o border)

QR Code (Final)

Interleaving Map



QR Code version number: 10
QR Code size (w/o border): 57×57
Available space: 346 bytes
Error correction level: H
Using field $\mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X^2 + 1)$ for data encoding
Using 6 blocks of shortened RS code $[43, 15, 29]_{256}$ and 2 blocks of $[44, 16, 29]_{256}$
Generating polynomial coeffs. (in hexadecimal)
01 FC 09 1C 0D 12 FB D0 96 67 AE 64 29 A7 0C F7 38 75 77 E9 7F B5 64 79 93 B0 4A 3A C5
Total raw capacity: 122 bytes
Total parity bytes: 224
Data preencoded in 'binary' format
Encoded blocks (in hexadecimal):
40 00 E4 86 56 C6 C6 F2 05 76 F7 26 C6 42...
12 10 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11...
11 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11 EC...
EC 11 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11...
11 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11 EC...
EC 11 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11...
11 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11 EC...
11 EC 11 EC 11 EC 11 EC 11 EC 11 EC 11 EC...
Interleaved blocks (in hexadecimal):
40 12 11 EC 11 EC 11 11 00 10 EC 11 EC 11 EC EC
E4 EC 11 EC 11 EC 11 11 86 11 EC 11 EC 11 EC EC
56 EC 11 EC 11 EC 11 11 C6 11 EC 11 EC 11 EC EC
C6 EC 11 EC 11 EC 11 11 F2 11 EC 11 EC 11 EC EC
05 EC 11 EC 11 EC 11 11 76 11 EC...

Real Systems (I)

Additional layers of encoding are necessary in a real system:

Real Systems (I)

Additional layers of encoding are necessary in a real system:

Source data preencoding: Special compression codes can be applied to make the symbols close to the uniform distribution.

Real Systems (I)

Additional layers of encoding are necessary in a real system:

Source data preencoding: Special compression codes can be applied to make the symbols close to the uniform distribution.

E.g., decimal characters can be grouped into triplets and then converted into binary strings of length 10. Most text documents can be converted into binary with a 6 bits per symbol encoding.

Real Systems (I)

Additional layers of encoding are necessary in a real system:

Source data preencoding: Special compression codes can be applied to make the symbols close to the uniform distribution.

E.g., decimal characters can be grouped into triplets and then converted into binary strings of length 10. Most text documents can be converted into binary with a 6 bits per symbol encoding.

Extra error correction can be performed if the source is redundant (and not compressed). E.g., interpolation of missing samples in digital audio produces reasonable results.

Real Systems (II)

Final reencoding: Encoded data encoded must be prepared for the physical (storage or transmission) media.

Real Systems (II)

Final reencoding: Encoded data encoded must be prepared for the physical (storage or transmission) media.

E.g., at reception, synchronization information about the starting of a codeword is necessary, as well as other system information. The positioning marks in a QR barcode are an example.

There can be physical limitations that forces structuring the encoded data in a particular way.

Real Systems (II)

Final reencoding: Encoded data encoded must be prepared for the physical (storage or transmission) media.

E.g., at reception, synchronization information about the starting of a codeword is necessary, as well as other system information. The positioning marks in a QR barcode are an example.

There can be physical limitations that forces structuring the encoded data in a particular way.

The information stored in a Compact Disk can be modelled by a binary string in which ones are always separated by a number of zeros between 2 and 10. Thus, codewords must be reencoded with a special code (EFM or Eight-To-Fourteen Modulation) to fulfil these requirements.

Eight-To-Fourteen Modulation

It is a nonlinear binary code of length 14 and size 257 (not intended for error correction).

Eight-To-Fourteen Modulation

It is a nonlinear binary code of length 14 and size 257 (not intended for error correction).

We assume the extra restriction that there are no more than 8 leading or trailing zeros in any codeword.

Eight-To-Fourteen Modulation

It is a nonlinear binary code of length 14 and size 257 (not intended for error correction).

We assume the extra restriction that there are no more than 8 leading or trailing zeros in any codeword.

The code is

00000000100000 00000000100001 00000000100010 00000000100100 00000001000000 00000001000001

00000001000010 00000001000100 00000001001000 00000001001001 00000010000000 00000010000001

00000010000010 00000010000100 00000010001000 00000010001001 ... 10010010010010

Eight-To-Fourteen Modulation

It is a nonlinear binary code of length 14 and size 257 (not intended for error correction).

We assume the extra restriction that there are no more than 8 leading or trailing zeros in any codeword.

The code is

00000000100000 00000000100001 00000000100010 00000000100100 00000001000000 00000001000001

00000001000010 00000001000100 00000001001000 00000001001001 00000010000000 00000010000001

00000010000010 00000010000100 00000010001000 00000010001001 ... 10010010010010

Only 256 codewords are used (to map every element in $\mathbb{F}_{256}$ to the code).

Eight-To-Fourteen Modulation

It is a nonlinear binary code of length 14 and size 257 (not intended for error correction).

We assume the extra restriction that there are no more than 8 leading or trailing zeros in any codeword.

The code is

00000000100000 00000000100001 00000000100010 00000000100100 00000001000000 00000001000001

00000001000010 00000001000100 00000001001000 00000001001001 00000010000000 00000010000001

00000010000010 00000010000100 00000010001000 00000010001001 ... 10010010010010

Only 256 codewords are used (to map every element in $\mathbb{F}_{256}$ to the code).

Codewords are separated by three extra binary digits (to avoid breaking the rules about the number of separating zeros).

Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

END OF PART VIII