**Reed-Solomon: Outline**

A \((n, k, n-k+1)\) Reed Solomon Code:

Consider the polynomial

\[ p(x) = a_{k-1}x^{k-1} + \ldots + a_1 x + a_0 \]

**Message:** \((a_{k-1}, \ldots, a_1, a_0)\)

**Codeword:** \((p(1), p(2), \ldots, p(n))\)

To keep the \(p(i)\) fixed size, we use \(a_i \in \text{GF}(p^r)\)

To make the \(p(i)\) distinct, \(n \times p^r\)

Any subset of size \(k\) of \((p(1), p(2), \ldots, p(n))\) is enough to reconstruct \(p(x)\).

**Reed Solomon: Outline**

A \((n, k, 2s +1)\) Reed Solomon Code:

- Can detect \(2s\) errors
- Can correct \(s\) errors

Generally can correct \(\alpha\) erasures and \(\beta\) errors if \(\alpha + 2\beta \leq 2s\)

**Correcting s errors:**

1. Find \(k + s\) symbols that agree on a polynomial \(p(x)\). These must exist since originally \(k + 2s\) symbols agreed and only \(s\) are in error
2. There are no \(k + s\) symbols that agree on the wrong polynomial \(p'(x)\)
   - Any subset of \(k\) symbols will define \(p'(x)\)
   - Since at most \(s\) out of the \(k+s\) symbols are in error, \(p'(x) = p(x)\)
Reed Solomon: Outline

Systematic version of Reed-Solomon
\[ p(x) = a_{k-1} x^{k-1} + \cdots + a_1 x + a_0 \]

Message: \((a_0, a_1, \ldots, a_{k-1})\)

Codeword: \((a_0, a_1, a_2, p(1), p(2), \ldots, p(2s))\)

This has the advantage that if we know there are no errors, it is trivial to decode.

Later we will see that version of RS used in practice uses something slightly different than \(p(1), p(2), \ldots\)

This will allow us to use the “Parity Check” ideas from linear codes (i.e. \(Hc^T = 0\)) to quickly test for errors.

RS in the Real World

*(204, 188, 17)\textsubscript{256}: ITU J.83(A)*

*(128, 122, 7)\textsubscript{256}: ITU J.83(B)*

*(255, 223, 33)\textsubscript{256}: Common in Practice*

- Note that they are all byte based (i.e. symbols are from \(GF(2^8)\)).

Performance on 600MHz Pentium (approx.):
- \((255,251) = 45\text{Mbps}\)
- \((255,223) = 4\text{Mbps}\)

Dozens of companies sell hardware cores that operate 10x faster (or more)
- \((204,188) = 320\text{Mbps (Altera decoder)}\)

Applications of Reed-Solomon Codes

- **Storage**: CDs, DVDs, “hard drives”.
- **Wireless**: Cell phones, wireless links
- **Satellite and Space**: TV, Mars rover, ...
- **Digital Television**: DVD, MPEG2 layover
- **High Speed Modems**: ADSL, DSL, ...

Good at handling burst errors.

Other codes are better for random errors.
- e.g. Gallager codes, Turbo codes

RS and “burst” errors

Let’s compare to Hamming Codes (which are “optimal”).

<table>
<thead>
<tr>
<th>Code</th>
<th>Code bits</th>
<th>Check bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS (255, 253, 3)\textsubscript{256}</td>
<td>2040</td>
<td>16</td>
</tr>
<tr>
<td>Hamming (2^{11}-1, 2^{11}-11-1, 3)\textsubscript{256}</td>
<td>2047</td>
<td>11</td>
</tr>
</tbody>
</table>

They can both correct 1 error, but not 2 random errors.

- The Hamming code does this with fewer check bits
- However, RS can fix 8 contiguous bit errors in one byte
- Much better than lower bound for 8 arbitrary errors

\[
\log \left( 1 + \binom{n}{1} + \cdots + \binom{n}{8} \right) > 8 \log(n - 7) = 88 \text{check bits}
\]
Discrete Fourier Transform

Another View of Reed-Solomon Codes

α is a primitive n\textsuperscript{th} root of unity \((\alpha^n = 1)\) - a generator

\[
T = \begin{pmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\
1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha^{n-1} & \alpha^{2(n-1)} & \ldots & \alpha^{(n-1)(n-1)}
\end{pmatrix}
\]

\[
\begin{pmatrix}
m_0 \\
m_1 \\
\vdots \\
m_{n-1}
\end{pmatrix} = T \cdot \begin{pmatrix}
c_0 \\
c_1 \\
\vdots \\
c_{n-1}
\end{pmatrix}
\]

\(m = T^{-1}c\)

Inverse DFT

The Discrete Fourier Transform

(DFT)

DFT Example

\(\alpha = x\) is 7\textsuperscript{th} root of unity in \(GF(2^3)/x^3 + x + 1\)

(ie, multiplicative group, which excludes additive inverse)

Recall \(\alpha = \alpha^2 = \alpha^4 = \ldots = \alpha^7 = 1 = \alpha^9\)

\[
T = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8 & \alpha^{10} & \alpha^{12} \\
1 & \alpha^3 & \alpha^6 & \alpha^{12} & \alpha^{15} & \alpha^{18} & \alpha^{21} \\
1 & \alpha^4 & \alpha^8 & \alpha^{16} & \alpha^{24} & \alpha^{32} & \alpha^{40} \\
1 & \alpha^5 & \alpha^{10} & \alpha^{20} & \alpha^{30} & \alpha^{40} & \alpha^{50} \\
1 & \alpha^6 & \alpha^{12} & \alpha^{24} & \alpha^{36} & \alpha^{48} & \alpha^{60}
\end{pmatrix}
\]

Should be clear that \(c = T \cdot (m_0, m_1, \ldots, m_{k-1}, 0, \ldots)^T\)

is the same as evaluating \(p(x) = m_0 + m_1x + \ldots + m_{k-1}x^{k-1}\)

at \(n\) points.

Decoding

Why is it hard?

Brute Force: try \(k+s\) choose \(k+2s\) possibilities and solve for each.

Cyclic Codes

A code is cyclic if:

\((c_0, c_1, \ldots, c_{n-1}) \in C \Rightarrow (c_{n-1}, c_0, \ldots, c_{n-2}) \in C\)

Both Hamming and Reed-Solomon codes are cyclic.

Note: we might have to reorder the columns to make the code "cyclic".

A linear code is cyclic if its "basis vectors" are cyclic.

We will only consider linear cyclic codes.

Motivation: They are more efficient to decode than general codes.
Generator and Parity Check Matrices

**Generator Matrix:**
A $k \times n$ matrix $G$ such that:
\[
C = \{ m \cdot G \mid m \in \Sigma^k \}
\]
Made from stacking the basis vectors

**Parity Check Matrix:**
A $(n - k) \times n$ matrix $H$ such that:
\[
C = \{ v \in \Sigma^n \mid H \cdot v^T = 0 \}
\]
Codewords are the nullspace of $H$

These always exist for linear codes
For the same code: $H \cdot G^T = 0$

---

Viewing $g$ as a matrix

If $g = g_0 + g_1 x + \ldots + g_n x^{n-k}$
We can put this generator in matrix form:
\[
G = \begin{pmatrix}
g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\
0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k}
\end{pmatrix}
\]

Write $m = m_0 + m_1 x + \ldots + m_{k-1} x^{k-1}$ as $(m_0, m_1, \ldots, m_{k-1})$
Then $c = mG$

---

Generator and Parity Check Polynomials

**Generator Polynomial:**
A degree $(n-k)$ polynomial $g$ such that:
\[
C = \{ m \cdot g \mid m \in \Sigma^k[x] \}
\]
such that $g \mid x^n - 1$

**Parity Check Polynomial:**
A degree $k$ polynomial $h$ such that:
\[
C = \{ v \in \Sigma^n[x] \mid h \cdot v = 0 \mod (x^n - 1) \}
\]
such that $h \mid x^n - 1$

These always exist for linear cyclic codes
For the same code: $h \cdot g = x^n - 1$

---

**$g$ generates cyclic codes**

\[
G = \begin{pmatrix}
g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\
0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k}
\end{pmatrix}
= \begin{pmatrix}
xg \\
g \\
\vdots \\
x^{k-1}g
\end{pmatrix}
\]

Codes are linear combinations of the rows.
All but last row is clearly cyclic (based on next row)
Shift of last row is $x^g \mod (x^n - 1)$
Consider $h = h_0 + h_1 x + \ldots + h_n x^n$ ($gh = x^n - 1$)
\[
h_0 g + (h_1 x)g + \ldots + (h_n x^{n-k})g + (h_n x^n)g = x^n - 1
\]
\[
x^g = -h_1 (h_0 g + h_1 (xg) + \ldots + h_n (x^{k-1}g)) \mod (x^n - 1)
\]
This is a linear combination of the rows.
**Viewing h as a matrix**

If \( h = h_0 + h_1 x + ... + h_k x^k \), we can put this parity check poly. in matrix form:

\[
H = \begin{pmatrix}
0 & \cdots & 0 & h_0 & \cdots & h_k \\
0 & \cdots & h_k & h_{k-1} & \cdots & 0 \\
\vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
h_k & \cdots & h_1 & h_0 & \cdots & 0
\end{pmatrix}
\]

\( H^T c = 0 \)

---

**Hamming Codes Revisited**

The Hamming \((7,4,3)\) code:

\[
g = 1 + x + x^3 \\
h = x^4 + x^2 + x + 1
\]

\[
G = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix}
\]

\[
H = \begin{pmatrix}
0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

\( gh = x^7 - 1, \quad G H^T = 0 \)

The columns are not identical to the previous example Hamming code.

---

**Factors of \( x^n - 1 \)**

These define the cyclic codes of length \( n \).

Let \( x^n - 1 \) can be factored into 4 irreducible polynomials.

- i.e. \( x^n - 1 = p_1(x)p_2(x)p_3(x)p_4(x) \)

How many cyclic codes of length \( n \) are there?

---

**\( g(x) \) when \( \Sigma = GF(p^r) \) and \( n = p^r \)**

Let \( g \) be a generator of \( GF(p^r) \).

Let \( n = p^r - 1 \) (the size of the multiplicative group)

Then \( g(x) = (x - \alpha)(x - \alpha^2) \cdots (x - \alpha^{n-k}) \) is a generator polynomial.

**Lemma:** \( g \mid x^n - 1 \) (\( a \mid b \), means \( a \) divides \( b \))

**Proof:**

- \( \alpha^n = 1 \) (because of the size of the group)
  - \( \alpha^n - 1 = 0 \)
  - \( \alpha \) root of \( x^n - 1 \)
  - \( (x - \alpha) \mid x^n - 1 \)
- similarly for \( \alpha, \alpha^2, \ldots, \alpha^{n-k} \)
- therefore \( x^n - 1 \) is divisible by \( (x - \alpha)(x - \alpha^2) \cdots \)
Back to Reed-Solomon

Consider a generator polynomial $g \in GF(p^r)[x]$, s.t. $g \mid (x^n - 1)$
Recall that $n - k = 2s$ (the degree of $g$)

**Encode:**
- $m' = m x^{2s}$ (basically shift by 2s)
- $b = m' \pmod{g}$
- $c = m' - b = (m_k, \ldots, m_0, -b_{2s-1}, \ldots, -b_0)$
- Note that $c$ is a cyclic code based on $g$
  - $m' = qg + b$
  - $c = m' - b = qg$

**Parity check:**
- $h \cdot c = 0$?

---

Example

Let's consider the $(7,3,5)_8$ Reed-Solomon code.
We use $GF(2^3)/x^3 + x + 1$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$x$</th>
<th>010</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^2$</td>
<td>$x^2$</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>$\alpha^3$</td>
<td>$x + 1$</td>
<td>011</td>
<td>4</td>
</tr>
<tr>
<td>$\alpha^4$</td>
<td>$x^2 + x$</td>
<td>110</td>
<td>5</td>
</tr>
<tr>
<td>$\alpha^5$</td>
<td>$x^2 + x + 1$</td>
<td>111</td>
<td>6</td>
</tr>
<tr>
<td>$\alpha^6$</td>
<td>$x^2 + 1$</td>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>$\alpha^7$</td>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Example RS (7,3,5)$_8$**

$g = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)$

$= x^4 + \alpha^2 x^3 + x^2 + \alpha x + \alpha^3$

$h = (x - \alpha^2)(x - \alpha^3)(x - \alpha^7)$

$= x^3 + \alpha^2 x^2 + \alpha^5 x + \alpha^4$

$gh = x^7 - 1$

Consider the message: 110 000 110

$m = \{x^4, 0, \alpha^4\} = \alpha^4 x^2 + \alpha^4$

$m' = x^4 m = \alpha^2 x^6 + \alpha^4 x^4$

$= (\alpha^4 x^2 + x + \alpha^2) g + (\alpha^3 x^3 + \alpha^5 x + \alpha^4)$

$c = \{x^4, 0, \alpha^4, \alpha^3, 0, \alpha^6, \alpha^5\}$

$= 110 000 110 011 000 101 101$

$\text{ch} = 0 \pmod{x^7 - 1}$

---

A useful theorem

**Theorem:** For any $\beta$, if $g(\beta) = 0$ then $\beta^{2s} m(\beta) = b(\beta)$

**Proof:**

$x^{2s} m(x) = g(x)q(x) + d(x)$

$\beta^{2s} m(\beta) = g(\beta)q(\beta) + b(\beta) = b(\beta)$

**Corollary:** $\beta^{2s} m(\beta) = b(\beta)$ for $\beta \in \{\alpha, \alpha^2, \ldots, \alpha^{2s}\}$

**Proof:**

$\{\alpha, \alpha^2, \ldots, \alpha^{2s}\}$ are the roots of $g$ by definition.
Fixing errors

**Theorem:** Any k symbols from c can reconstruct c and hence m

**Proof:**
We can write 2s equations involving m($c_{n-1}, ..., c_{2s}$) and b ($c_{2s+1}, ..., c_{n}$). These are

\[ \alpha^{2s} m(\alpha) = b(\alpha) \]
\[ \alpha^{4s} m(\alpha^2) = b(\alpha^2) \]

... \[ \alpha^{2s(2^s)} m(\alpha^{2^s}) = b(\alpha^{2^s}) \]

We have at most 2s unknowns, so we can solve for them. (I'm skipping showing that the equations are linearly independent).

---

Efficient Decoding

I don’t plan to go into the Reed-Solomon decoding algorithm, other than to mention the steps.