Computer Science Fundamentals

Channel Coding – Reed-Solomon Codes

Technische Hochschule Rosenheim
Winter 2021/22
Prof. Dr. Jochen Schmidt

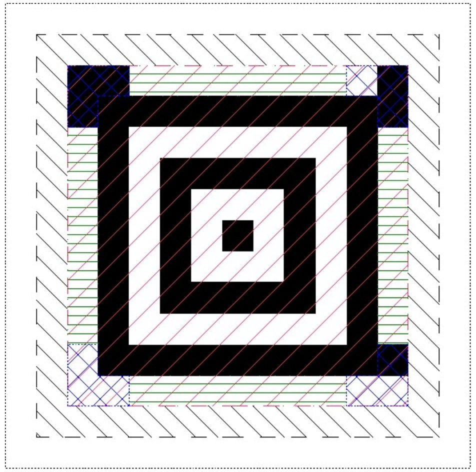
2-D Barcodes

- Many different variants
- Typical:
  - dots/lines of different widths
  - gaps in between ⟶ high contrast for reading (e.g., with laser scanner or camera)

Aztec-Code DataMatrix-Code MaxiCode QR-Code

Aztec Code

- Developed 1995, standardized in ISO/IEC 24778
- Usage: Online-tickets
  - German/Swiss/Austrian/… railways
  - many airlines
- Encodes 12 – 3000 characters
- Reed-Solomon code for error correction
  - still decodable in case of destruction of up to 25%
- Center: Marking with orientation marks



DataMatrix Code

- Developed 1980s, standardized in ISO/IEC 16022
- Usage:
  - Labeling of products with laser (permanent)
  - German/Swiss Post (clearing without stamp)
- Encodes up to approx. 3000 characters
- in the past: CRC code
- now: Reed-Solomon code
- rectangular border for finding the code and timing of the reader

STAMPIT 0,55 EUR
A00000CEE1 01.01.08

Maxicode

- 1989, standardized in ISO/IEC 16023
- Usage: UPS for parcel data
- Encodes 93 characters
  - up to 8 codes can be combined ($\longrightarrow$ 744 characters)
- Reed-Solomon code for error correction
- Marking in the center
- Hexagonal dots

QR Code

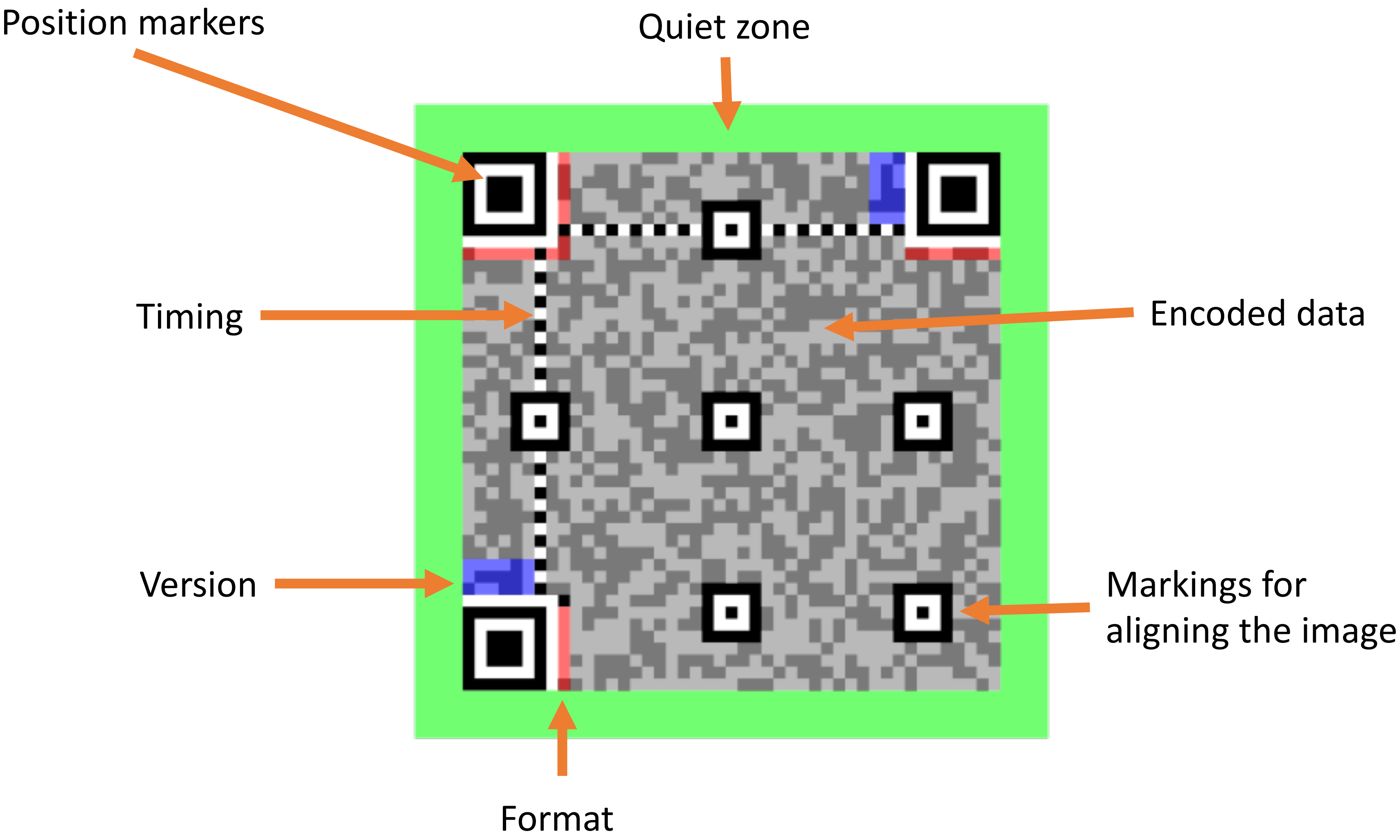
Technische Hochschule Rosenheim

- QR = Quick Response
- 1994, originally developed for automotive sector
- Standardized in ISO/IEC 18004
- Usage:
  - originally industrial applications
  - now widespread use for smartphone apps

- encodes approx. 1800 – 7000 characters
  - depending on the mode (only numbers, Latin letters, whole bytes, …)
  - and the desired robustness against errors
  - with more data: can be divided into up to 16 individual codes

- Reed-Solomon code for error correction
  - depending on the code, 7% – 30% of the data can be reconstructed
  - the more robust the less user data can be stored

QR Code – Structure



QR Code

Design QR codes like this one …

… work only because of good error correction mechanisms
⟶ Reed-Solomon codes

Reed-Solomon Codes (RS)

Technische Hochschule Rosenheim

- Irving S. Reed and Gustave Solomon, 1960
- Properties:
  - Detection and correction of
    - random multiple errors
    - burst errors
    - erasures (= missing data)
  - non-binary code
    - e.g., used on ASCII characters directly
    - is of course converted to binary for the actual transfer
  - linear cyclic block code
- Usage, e.g.,
  - QR codes, audio CD, DVD, Blu-ray, RAID 6, satellite communication, …

RS – Idea

- Interpret message as *coefficients of a polynomial* over a finite field
- Encoding: evaluate polynomial at n different positions
- Decoding: by interpolation

- Construction of code RS(q, m, n)
  - Choose finite field $\mathbb{F}_q$ with $q = p^l$ elements as alphabet, p prime, $l \in \{1, 2, 3, \dots\}$
    - for reasons of simplicity, we will only consider $l = 1$:
      - with q elements = calculations modulo q (only if q is prime)
      - coefficients can only take the values $0, 1, \dots, q-1$
    - for $l > 1$ the elements of the field are polynomials with coefficients from $\mathbb{F}_p$ and degree $< l$

  - Message (block of m symbols) $\boldsymbol{a} = (a_0, a_1, \dots, a_{m-1})$ interpreted as polynomial over $\mathbb{F}_q$:

$$P(x) = a_0 + a_1 x + a_2 x^2 + a_{m-1} x^{m-1}$$

  - Choose n pairwise distinct elements $(n \geq m)$ $u_0, u_1, \dots, u_{n-1} \in \mathbb{F}_q$
    - This is where we will evaluate the polynomial (i.e., the "x" values)

RS – Encoding

- Evaluate $P(x)$ at the n positions $u_0, u_1, \dots, u_{n-1}$
  - best to use Horner's method or discrete Fourier-Transform (DFT) as Fast Fourier-Transform (FFT)
- Code word $\boldsymbol{c} = \big(P(u_0), P(u_1), \dots, P(u_{n-1})\big)$

Example: RS(q, m, n) with $q = 5,\ m = 3,\ n = 5$

- Encode message $\boldsymbol{a} = (1, 2, 3)$ $\longrightarrow$ polynomial: $P(x) = 1 + 2x + 3x^2$
- Evaluate $P(x)$ at $n = 5$ positions
  - more are not possible anyway, since the field $\mathbb{F}_5$ has only 5 elements

$$
\begin{aligned}
P(0) &= 1 + 0 + 0 && = 1 && \text{(mod 5)}\\
P(1) &= 1 + 2 + 3 = 6 && = 1 && \text{(mod 5)}\\
P(2) &= 1 + 4 + 12 = 17 && = 2 && \text{(mod 5)}\\
P(3) &= 1 + 6 + 27 = 34 && = 4 && \text{(mod 5)}\\
P(4) &= 1 + 8 + 48 = 47 && = 2 && \text{(mod 5)}
\end{aligned}
$$

- Code word $\boldsymbol{c} = (1, 1, 2, 4, 2)$

RS – Decoding – Erasure

- RS(q, m, n) tolerates up to $n - m$ erasures
  - Erasure:
    - Part of the code was not received or could not be read
    - Positions of failures are known
  - so, we assume that at least m data points of the code word were received

- Polynomial $P(x)$ has degree $m - 1$
  - from m data points we can reconstruct $P(x)$
  - and therefore, the original message message (= coefficients of $P(x)$)
  - $\longrightarrow$ Lagrange interpolation

RS – Decoding – Erasure – Lagrange Interpolation

- Given: at least m data points $\left(u_i, P(u_i)\right)$
  - to simplify notation: Assume that the first m have been received
- Let $g_i(x) = \prod_{j=0, j\neq i}^{m-1}\left(x - u_j\right), i = 0, \dots, m-1$
- It holds: $g_i\left(u_j\right) = 0, j \neq i$
- We obtain $P(x)$ from

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

Technische Hochschule Rosenheim

- RS(q, m, n) with $q = 5,\ m = 3,\ n = 5$ as before
- $P(x)$ was evaluated at the positions $u_i = 0, 1, 2, 3, 4$
- Sent code word was $\boldsymbol{c} = (1, 1, 2, 4, 2)$
  - the last two values were erased $\longrightarrow$ received: $(1, 1, 2, \varepsilon, \varepsilon)$

- Determine polynomials $g_i(x)$:

mod 5 !

$$
\begin{aligned}
g_0(x) &= (x-1)(x-2) = x^2 - 3x + 2 &&= x^2 + 2x + 2 \\
g_1(x) &= x(x-2) = x^2 - 2x &&= x^2 + 3x \\
g_2(x) &= x(x-1) = x^2 - x &&= x^2 + 4x
\end{aligned}
$$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

Evaluate the $g_i(u_i)$ at $u_i = 0, 1, 2$

$$g_0(x) = x^2 + 2x + 2$$
$$g_0(0) = 2$$

$$g_1(x) = x^2 + 3x$$
$$g_1(1) = 1 + 3 = 4$$

$$g_2(x) = x^2 + 4x$$
$$g_2(2) = 4 + 8 = 12 = 2$$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

- Determinate multiplicative inverses $g_i^{-1}(u_i)$
  - they always exist because we have a field
  - use, e.g., extended Euclidean algorithm

$g_0(0) = 2 \longrightarrow g_0^{-1}(0) = 3$ (Test: 2 · 3 = 6 = 1)
$g_1(1) = 4 \longrightarrow g_1^{-1}(1) = 4$ (Test: 4 · 4 = 16 = 1)
$g_2(2) = 2 \longrightarrow g_2^{-1}(2) = 3$ (Test: 2 · 3 = 6 = 1)

- Product $P(u_i)g_i^{-1}(u_i)$

$$P(0)g_0^{-1}(0) = 1 \cdot 3 = 3$$
$$P(1)g_1^{-1}(1) = 1 \cdot 4 = 4$$
$$P(2)g_2^{-1}(2) = 2 \cdot 3 = 6 = 1 \pmod 5$$

$(1, 1, 2, \varepsilon, \varepsilon)$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

Plug-in everything:

$$P(x) = \sum_{i=0}^{2} \frac{P(u_i)}{g_i(u_i)} g_i(x) = 3g_0(x) + 4g_1(x) + 1g_2(x)$$

$$= 3(x^2 + 2x + 2) + 4(x^2 + 3x) + (x^2 + 4x)$$
$$= 8x^2 + 22x + 6$$

$$= 3x^2 + 2x + 1$$
$$= 1 + 2x + 3x^2$$

$\longrightarrow$ original message was $(1, 2, 3)$

RS – Decoding – Error Correction

- RS(q, m, n) has a Hamming distance of $n - m + 1$
- therefore, $(n - m)/2$ errors can be corrected

Proof: For $n \geq m$ two polynomials can only have the same value at $m - 1$ positions

- otherwise, they would be identical (and the messages too)
- the values of the polynomials therefore differ at $n - m + 1$ positions (= minimal distance between code words)

RS – Decoding – Error Correction

- Take two polynomials with yet unknown coefficients:
  - $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots$ of degree $\left\lceil \frac{n-m}{2} \right\rceil$
  - $g(x) = g_0 + g_1 x + g_2 x^2 + \cdots$ of degree $\left\lceil \frac{n-m}{2} \right\rceil + m - 1$
- Construct a new polynomial from these: $p(x, y) = y f(x) + g(x)$
- Determine the coefficients of $p(x, y)$ such that $p(u_i, y_i) = 0$,
  where $y_i = P(u_i)$ is the received (erroneous) code word
- The originally sent message results from the coefficients of the polynomial

$$-\frac{g(x)}{f(x)}$$

RS – Decoding – Error Correction – Example

- RS(q, m, n) with $q = 5,\ m = 3,\ n = 5$ as before
  - $(n-m)/2 = (5-3)/2 = 1$ error can be corrected
- $P(x)$ was evaluated at the positions $u_i = 0, 1, 2, 3, 4$
- Sent code word was $\boldsymbol{c} = (1, 1, 2, 4, 2)$
  - one position incorrect $\longrightarrow$ received: $(1, 1, \mathbf{0}, 4, 2)$

- Polynomials:
  - $f(x) = f_0 + f_1 x$ of degree $\left\lceil \frac{n-m}{2} \right\rceil = 1$
  - $g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3$ of degree $\left\lceil \frac{n-m}{2} \right\rceil + m - 1 = 3$
- result:

$$p(x, y) = yf(x) + g(x) = f_0 y + f_1 xy + g_0 + g_1 x + g_2 x^2 + g_3 x^3$$

RS – Decoding – Error Correction – Example

- $p(x,y) = yf(x) + g(x) = f_0 y + f_1 xy + g_0 + g_1 x + g_2 x^2 + g_3 x^3$
- received code word: $(1,1,0,4,2)$ $\longrightarrow$ data points (u_i, y_i): (0,1), (1,1), (2,0), (3,4), (4,2)
- Plug-in $p(u_i, y_i)$ and set to zero $\longrightarrow$ (homogenous) linear system of equations:

$$
\begin{aligned}
&f_0 + g_0 = 0 \longrightarrow g_0 = -f_0 = 4f_0\\
&f_0 + f_1 + g_0 + g_1 + g_2 + g_3 = 0\\
&g_0 + 2g_1 + 4g_2 + 8g_3 = 0\\
&4f_0 + 12f_1 + g_0 + 3g_1 + 9g_2 + 27g_3 = 0\\
&2f_0 + 8f_1 + g_0 + 4g_1 + 16g_2 + 64g_3 = 0
\end{aligned}
$$

plug-in to remaining equations + reduce mod 5

Caution: All calculations mod 5!

RS – Decoding – Error Correction – Example

- After plugging-in:

$$
\begin{aligned}
f_1 + g_1 + g_2 + g_3 &= 0\\
4f_0 + 2g_1 + 4g_2 + 3g_3 &= 0\\
3f_0 + 2f_1 + 3g_1 + 4g_2 + 2g_3 &= 0\\
f_0 + 3f_1 + 4g_1 + g_2 + 4g_3 &= 0
\end{aligned}
$$

- Solve the system of equations
  - e.g., using Gaussian elimination
  - 5 unknowns, 4 equations $\longrightarrow$ one unknown can be chosen freely ($\neq 0$)
  - note: finite field, inverses regarding multiplication:
    $1 \leftrightarrow 1$, $2 \leftrightarrow 3$, $3 \leftrightarrow 2$, $4 \leftrightarrow 4$

- Result (with $g_2 = 1$):
  $f_0 = 2,\ f_1 = 4,\ g_0 = 3,\ g_1 = 2,\ g_2 = 1,\ g_3 = 3$

RS – Decoding – Error Correction – Example

- Polynomials:
  - $f(x) = f_0 + f_1 x = 2 + 4x$
  - $g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 = 3 + 2x + x^2 + 3x^3$

- Calculate $\frac{g(x)}{f(x)}$

$$
\begin{array}{l}
(3x^3 + x^2 + 2x + 3) : (4x + 2) = 2x^2 + 3x + 4 \\
-\underline{(3x^3 + 4x^2)} \\
\qquad (2x^2 + 2x + 3) \\
\quad -\underline{(2x^2 + x)} \\
\qquad\qquad (x + 3) \\
\qquad\quad -\underline{(x + 3)} \\
\qquad\qquad ---------
\end{array}
$$

- Message = $-\frac{g(x)}{f(x)}$ = $-(2x^2 + 3x + 4) = 3x^2 + 2x + 1$

$\longrightarrow$ originally sent: (1, 2, 3)

RS – Notes

- Decoding in practice
  - with faster (and more complicated) methods, which typically:
    - locate error positions first,
    - treat these as erasures
    - reconstruct message
  - e.g., Berlekamp–Massey algorithm

- Examples for RS-Codes
  - Audio CD: two interleaved RS-Codes
    - CIRC: Cross-Interleaved Reed-Solomon Coding
    - Two RS codes over finite field with $2^8 = 256$ elements ($\longrightarrow$ 1 byte)
      - uses so-called shortened RS codes with resulting code lengths of 28 and 32 bytes
    - Burst errors up to 4000 bits (approx. 2.5mm scratch) can be corrected exactly, i.e., without any loss
    - Errors = erasures
  - DVD/Blu-ray: similar to audio CD, but longer codes
  - QR: code over finite field with $2^8 = 256$ elements ($\longrightarrow$ 1 byte)
    - unreadable parts of the code = erasures