

6. Decoding Generalized Reed-Solomon Codes

Decoding Generalized Reed-Solomon Codes

- We consider \mathcal{C}_{GRS} over \mathbb{F}_q with PCM

$$H_{\text{GRS}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^\ell & \alpha_2^\ell & \dots & \alpha_n^\ell \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \dots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} v_1 & & & 0 \\ & v_2 & & \\ & & \ddots & \\ 0 & & & v_n \end{pmatrix}$$

with $\alpha_1, \alpha_2, \dots, \alpha_n \in \mathbb{F}_q^*$ distinct, and $v_1, v_2, \dots, v_n \in \mathbb{F}_q^*$ (recall that $r = n - k = d - 1$).

- Codeword \mathbf{c} transmitted, word \mathbf{y} received, with error vector

$$\mathbf{e} = (e_1 \ e_2 \ \dots \ e_n) = \mathbf{y} - \mathbf{c}.$$

- $J = \{\kappa : e_\kappa \neq 0\}$ set of *error locations*.
- We describe an algorithm that correctly decodes \mathbf{y} to \mathbf{c} , under the assumption $|J| \leq \frac{1}{2}(d-1)$.

Syndrome Computation

- First step of the decoding algorithm: *syndrome computation*

$$\mathbf{S} = \begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{r-1} \end{pmatrix} = H_{\text{GRS}} \mathbf{y}^T = H_{\text{GRS}} \mathbf{e}^T$$

ℓ th row of H_{GRS} :
 $[v_1 \alpha_1^\ell, v_2 \alpha_2^\ell, \dots, v_n \alpha_n^\ell]$

$$S_\ell = \sum_{j=1}^n y_j v_j \alpha_j^\ell = \sum_{j=1}^n e_j v_j \alpha_j^\ell = \sum_{j \in J} e_j v_j \alpha_j^\ell, \quad \ell = 0, 1, \dots, r-1.$$

Example: For conventional RS codes, we have $\alpha_j = \alpha^{j-1}$ and $v_j = \alpha^{b(j-1)}$, so

$$S_\ell = \sum_{j=1}^n y_j \alpha^{(j-1)(b+\ell)} = y(\alpha^{b+\ell}), \quad \ell = 0, 1, \dots, r-1$$

(recall $\mathbf{c} \in \mathcal{C}_{\text{RS}} \Leftrightarrow c(\alpha^{b+\ell}) = 0, \ell = 0, 1, \dots, r-1$).

- Syndrome polynomial:*

$$S(x) = \sum_{\ell=0}^{r-1} S_\ell x^\ell = \sum_{\ell=0}^{r-1} x^\ell \sum_{j \in J} e_j v_j \alpha_j^\ell = \sum_{j \in J} e_j v_j \sum_{\ell=0}^{r-1} (\alpha_j x)^\ell.$$

A Congruence for the Syndrome Polynomial

$$S(x) = \sum_{j \in J} e_j v_j \sum_{\ell=0}^{r-1} (\alpha_j x)^\ell .$$

- We have

$$(1 - \alpha_j x) \sum_{\ell=0}^{r-1} (\alpha_j x)^\ell = 1 - (\alpha_j x)^r \equiv 1 \pmod{x^r} .$$

Therefore, we can write

$$\sum_{\ell=0}^{r-1} (\alpha_j x)^\ell \equiv \frac{1}{1 - \alpha_j x} \pmod{x^r}$$

$$\Rightarrow \boxed{S(x) \equiv \sum_{j \in J} \frac{e_j v_j}{1 - \alpha_j x} \pmod{x^r}}$$
$$\left(\sum_{\text{empty}} \square \triangleq 0 \right)$$

More Auxiliary Polynomials

- *Error locator polynomial (ELP)*

$$\Lambda(x) = \prod_{j \in J} (1 - \alpha_j x) \quad \left(\prod_{\text{empty}} \square \triangleq 1 \right)$$

- *Error evaluator polynomial (EEP)*

$$\Gamma(x) = \sum_{j \in J} e_j v_j \prod_{m \in J \setminus \{j\}} (1 - \alpha_m x)$$

- $\Lambda(\alpha_\kappa^{-1}) = 0 \iff \kappa \in J$ *roots of EEP point to error locations*

- $\Gamma(\alpha_\kappa^{-1}) = e_\kappa v_\kappa \prod_{m \in J \setminus \{\kappa\}} (1 - \alpha_m \alpha_\kappa^{-1}) \neq 0$

\implies

$$\gcd(\Lambda(x), \Gamma(x)) = 1$$

- The degrees of ELP and EEP satisfy

$$\deg \Lambda = |J| \quad \text{and} \quad \deg \Gamma < |J|$$

Of course, we don't know $\Lambda(x)$, $\Gamma(x)$: our goal is to find them

Key Equation of GRS Decoding

Since $|J| \leq \frac{1}{2}(d-1)$, from $\deg \Lambda = |J|$, $\deg \Gamma < |J|$ we get

$$(1) \quad \deg \Lambda \leq \frac{1}{2}(d-1)$$

and

$$(2) \quad \deg \Gamma < \frac{1}{2}(d-1)$$

The ELP and the EEP are related by

$$\Gamma(x) = \sum_{j \in J} e_j v_j \prod_{m \in J \setminus \{j\}} (1 - \alpha_m x) = \sum_{j \in J} e_j v_j \frac{\Lambda(x)}{1 - \alpha_j x} = \Lambda(x) \left(\sum_{j \in J} \frac{e_j v_j}{1 - \alpha_j x} \right)$$

$S(x) \bmod x^{d-1}$
(recall $d-1 = r$)

$$\Rightarrow (3) \quad \Lambda(x) S(x) \equiv \Gamma(x) \pmod{x^{d-1}}$$

(1)+(2)+(3): *key equation of GRS decoding*

We have $S(x)$, and we know d . We want to solve for $\Lambda(x)$ and $\Gamma(x)$ satisfying (1)+(2)+(3).

Key Equation of GRS Decoding (cont.)

$$(1) \quad \deg \Lambda \leq \frac{1}{2}(d-1)$$

$$(2) \quad \deg \Gamma < \frac{1}{2}(d-1)$$

$$(3) \quad \Lambda(x)S(x) \equiv \Gamma(x) \pmod{x^{d-1}}$$

The coefficients of $\Lambda(x)$ and $\Gamma(x)$ solve the system of linear equations

$$\begin{array}{c} \uparrow \\ d-1 \\ \downarrow \end{array}
 \begin{pmatrix}
 S_0 & 0 & 0 & \cdots & 0 \\
 S_1 & S_0 & 0 & \cdots & 0 \\
 \vdots & \vdots & \ddots & \ddots & \vdots \\
 S_{\tau-1} & S_{\tau-2} & \cdots & S_0 & 0 \\
 \hline
 S_{\tau} & S_{\tau-1} & \cdots & S_1 & S_0 \\
 S_{\tau+1} & S_{\tau} & \cdots & S_2 & S_1 \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 S_{d-2} & S_{d-3} & \cdots & S_{d-1-\tau} & S_{d-2-\tau}
 \end{pmatrix}
 \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{\tau} \end{pmatrix}
 =
 \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{\tau-1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}
 \quad \left(\tau \triangleq \lfloor \frac{d-1}{2} \rfloor \right)$$

$\leftarrow \tau+1 \rightarrow$

- a set of $r = d - 1$ linear equations in the coefficients of Λ and Γ
- the last $\lfloor \frac{1}{2}(d-1) \rfloor$ equations depend only on Λ
- we can solve for Λ , find its root set J , then solve *linear* equations for e_j
- straightforward solution leads to $O(d^3)$ algorithm — we'll present an $O(d^2)$ one

The Extended Euclidean Algorithm for polynomials

Given $a(x), b(x)$ over a field \mathbb{F} , with $a(x) \neq 0$ and $\deg a > \deg b$, the algorithm computes sequences of *remainders* $r_i(x)$, *quotients* $q_i(x)$, and *auxiliary polynomials* $s_i(x)$, $t_i(x)$

```

$$\begin{aligned} r_{-1}(x) &\leftarrow a(x); & r_0(x) &\leftarrow b(x); \\ s_{-1}(x) &\leftarrow 1; & s_0(x) &\leftarrow 0; \\ t_{-1}(x) &\leftarrow 0; & t_0(x) &\leftarrow 1; \\ \text{for } (i \leftarrow 1; & r_{i-1}(x) \neq 0; i++) \{ \\ & q_i(x) \leftarrow r_{i-2}(x) \operatorname{div} r_{i-1}(x); \\ & r_i(x) \leftarrow r_{i-2}(x) - q_i(x) r_{i-1}(x); \\ & s_i(x) \leftarrow s_{i-2}(x) - q_i(x) s_{i-1}(x); \\ & t_i(x) \leftarrow t_{i-2}(x) - q_i(x) t_{i-1}(x); \\ & \} \end{aligned}$$

```

- Let ν = largest i such that $r_i \neq 0$. Then, $r_\nu(x) = \gcd(a(x), b(x))$.
- We also know that $s_\nu(x)a(x) + t_\nu(x)b(x) = \gcd(a(x), b(x))$ (often used to compute modular inverses).

Properties of the Euclidean Algorithm Sequences

Proposition (E1)

The following relations hold:

$$(i) \text{ For } i = -1, 0, \dots, \nu + 1: \quad s_i(x)a(x) + t_i(x)b(x) = r_i(x)$$

$$(ii) \text{ For } i = 0, 1, \dots, \nu + 1: \quad \deg t_i + \deg r_{i-1} = \deg a$$

Proof. By induction on i . \square

Proposition (E2)

Suppose that $t(x), r(x) \in \mathbb{F}[x] \setminus \{0\}$ satisfy the following conditions:

$$(C1) \quad \gcd(t(x), r(x)) = 1$$

$$(C2) \quad \deg t + \deg r < \deg a$$

$$(C3) \quad t(x)b(x) \equiv r(x) \pmod{a(x)}$$

Then, for some $h \in \{0, 1, \dots, \nu + 1\}$ and a constant $c \in \mathbb{F}$, we have

$$t(x) = c \cdot t_h(x) \quad \text{and} \quad r(x) = c \cdot r_h(x).$$

Proof. Standard polynomial manipulations, Proposition (E1), and recalling that the sequence $\deg r_i$ is strictly decreasing. \square

Solving the Key Equation

- Apply the Euclidean algorithm with $a(x) = x^{d-1}$ and $b(x) = S(x)$.
 - Let $\Lambda(x)$ and $\Gamma(x)$ play the roles of $t(x)$ and $r(x)$, respectively, in Proposition (E2). *The definitions of Λ and Γ , and the key equation, guarantee that conditions (C1)–(C3) are satisfied.*
 - (C1) $\gcd(t(x), r(x)) = \gcd(\Lambda(x), \Gamma(x)) = 1$
 - (C2) $\deg t + \deg r = \deg \Lambda + \deg \Gamma < \deg a = d - 1$
 - (C3) $t(x)b(x) \equiv r(x) \pmod{a(x)} \Leftrightarrow \Lambda(x)S(x) \equiv \Gamma(x) \pmod{x^{d-1}}$
 - By Proposition (E2), we have $\Lambda(x) = c \cdot t_h(x)$ and $\Gamma(x) = c \cdot r_h(x)$ for some index h and scalar constant c .

How do we find index h ?

Theorem

*The solution to the key equation is unique up to a scalar constant, and it is obtained with the Euclidean algorithm by stopping at the **unique** index h such that*

$$\deg r_h < \frac{1}{2}(d-1) \leq \deg r_{h-1}$$

Proof. Such an h exists because r_i is strictly decreasing. The degree properties follow from the propositions. \square

Finding the Error Values

- *Formal derivatives* in finite fields: $\left[\sum_{i=0}^s a_i x^i\right]' = \sum_{i=1}^s i a_i x^{i-1}$
 $(a(x)b(x))' = a'(x)b(x) + a(x)b'(x)$ (not surprising)
- For the ELP, we have

$$\Lambda(x) = \prod_{j \in J} (1 - \alpha_j x) \quad \implies \quad \Lambda'(x) = \sum_{j \in J} (-\alpha_j) \prod_{m \in J \setminus \{j\}} (1 - \alpha_m x),$$

and, for $\kappa \in J$,

$$\Lambda'(\alpha_\kappa^{-1}) = -\alpha_\kappa \prod_{m \in J \setminus \{\kappa\}} (1 - \alpha_m \alpha_\kappa^{-1}),$$

$$\Gamma(\alpha_\kappa^{-1}) = e_\kappa v_\kappa \prod_{m \in J \setminus \{\kappa\}} (1 - \alpha_m \alpha_\kappa^{-1})$$

- Therefore, for all error locations $\kappa \in J$, we obtain

$$e_\kappa = -\frac{\alpha_\kappa}{v_\kappa} \cdot \frac{\Gamma(\alpha_\kappa^{-1})}{\Lambda'(\alpha_\kappa^{-1})}$$

Forney's algorithm for error values

Summary of GRS Decoding

Input: received word $(y_1 \ y_2 \ \dots \ y_n) \in \mathbb{F}_q^n$.

Output: error vector $(e_1 \ e_2 \ \dots \ e_n) \in \mathbb{F}_q^n$.

- ① *Syndrome computation:* Compute the polynomial $S(x) = \sum_{\ell=0}^{d-2} S_\ell x^\ell$ by

$$S_\ell = \sum_{j=1}^n y_j v_j \alpha_j^\ell, \quad \ell = 0, 1, \dots, d-2.$$

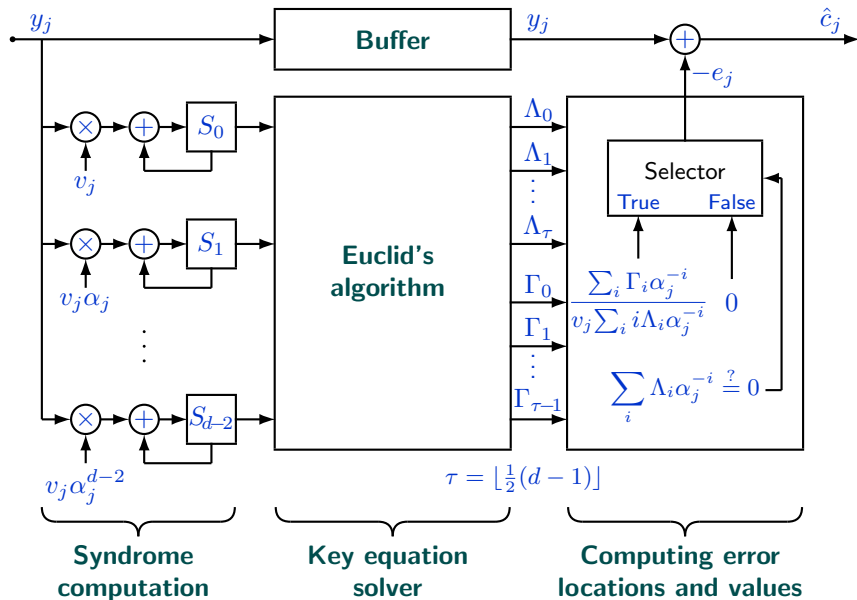
- ② *Solving the key equation:* Apply Euclid's algorithm to $a(x) \leftarrow x^{d-1}$ and $b(x) \leftarrow S(x)$ to produce $\Lambda(x) \leftarrow t_h(x)$ and $\Gamma(x) \leftarrow r_h(x)$, where h is the smallest index i for which $\deg r_i < \frac{1}{2}(d-1)$.

- ③ *Forney's algorithm:* Compute the error locations and values by

$$e_j = \begin{cases} -\frac{\alpha_j}{v_j} \cdot \frac{\Gamma(\alpha_j^{-1})}{\Lambda'(\alpha_j^{-1})} & \text{if } \Lambda(\alpha_j^{-1}) = 0 \\ 0 & \text{otherwise} \end{cases}, \quad j = 1, 2, \dots, n.$$

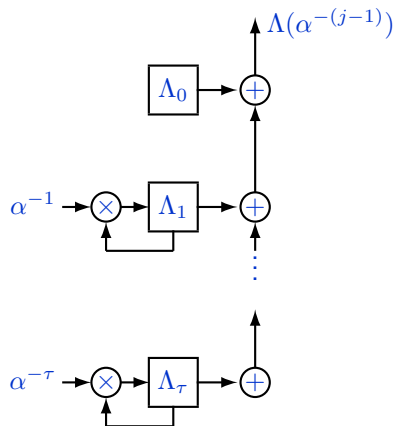
Complexity: 1. $O(dn)$ 2. $O((|J|+1)d)$ 3. $O((|J|+1)n)$

Schematic for GRS Decoder



Finding Roots of the ELP (RS Codes)

Chien search for RS codes ($\alpha_j = \alpha^{j-1}$, $1 \leq j \leq n$)



At clock cycle $\#j$, the cell labeled Λ_i contains

$$\Lambda_i \alpha^{-i(j-1)}, \quad 0 \leq i \leq \tau,$$

and the output of the circuit is

$$\begin{aligned} & \sum_{i=0}^{\tau} \Lambda_i \alpha^{-i(j-1)} \\ &= \Lambda(\alpha^{-(j-1)}) = \Lambda(\alpha_j^{-1}), \quad 1 \leq j \leq n. \end{aligned}$$


Other Decoding Algorithms

Many decoding algorithms and variants have been developed over the years. We mention a few of the most important ones.

- *Berlekamp algorithm* [1967] (also referred to as *Berlekamp-Massey* due to a clearer description and improvements by Massey [1969]): first efficient solution of the key equation, using Newton's identities and solving for shortest recurrence that generates the syndrome sequence. Complexity comparable to the Euclidean algorithm.
- *Welch-Berlekamp* [1986]: Solves key equation starting from *remainder syndrome* $y(x) \pmod{g(x)}$, without computing power sums. Akin to continued fractions and Padé approximations.
- *List decoding*: Decodes beyond $\tau = \lfloor \frac{1}{2}(d-1) \rfloor$ errors, producing a list of candidate decoded codewords. Very often, the coset leader is unique even beyond τ . Dates back to the '50s, but has gotten recent focus due to elegant and efficient algorithms by Sudan ['97], Guruswami-Sudan ['99] and others.
- *Soft decoding*: Information on the *reliability* of the symbols is available. Can lead to significant gains in decoding performance.

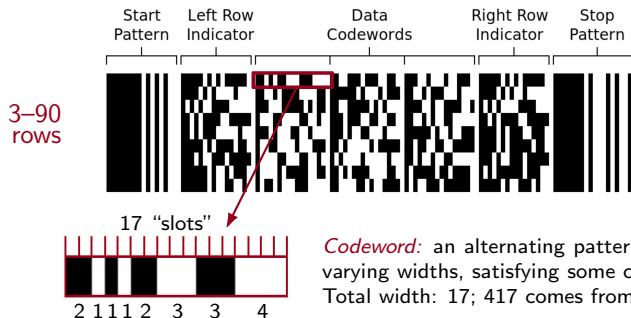
Applications: PDF417 bar code

[REDACTED]		PREFERACCESS		TSA PRE	
FREQUENT FLYER	FLIGHT	FROM	TO		
	CM 283	PTY	MVD		
ORDER ID: ACV1A1		PANAMA CITY	MONTEVIDEO		
ETICKET: 2302182845474	DATE: 22OCT	DEP: 15:43	ARR: 00:54		
SEQ: 92					
TERMINAL	GATE ****	GROUP 2	SEAT 2E	BOARDING BEGINS AT 14:43	
**GATE CLOSES 10 MINUTES PRIOR TO DEPARTURE// CIERRE DE PUERTA 10 MINUTOS ANTES DE LA SALIDA DEL VUELO **					



PDF417: A multi-row, 1D bar code (PDF: Portable Data File).

PDF417 bar code structure



In bar-code jargon, the whole array is referred to as a *symbol*

Codeword: an alternating pattern of 4 *bars* and 4 *spaces*, of varying widths, satisfying some constraints (e.g. width ≤ 6). Total width: 17; 417 comes from $4+17$.

- Basic global parameters (height, width, ECC level, etc.) are encoded in the *left* and *right row indicators*. A form of *repetition coding* (one copy per row).
- Consecutive rows use different sets of bar/space patterns (codewords). Each set has 929 codewords; 3 disjoint sets are used cyclically.
- Number of rows: $3 \leq h \leq 90$. Number of codewords per row: $1 \leq w \leq 30$ (all rows have the same number of codewords).
- Total number of codewords (all rows): $n \leq 928$.
- Using fixed tables, each codeword is mapped to a number in $\{0, 1, \dots, 928\}$, and *interpreted as an element of $GF(929)$* (929 is prime).

Table H1. The Bar-Space Sequence Table. Cluster 0

<u>bsbsbsbs</u>	<u>val</u>	<u>bsbsbsbs</u>	<u>val</u>	<u>bsbsbsbs</u>	<u>val</u>	<u>bsbsbsbs</u>	<u>val</u>	<u>bsbsbsbs</u>	<u>val</u>	<u>bsbsbsbs</u>	<u>val</u>
31111136	0	41111144	1	51111152	2	31111235	3	41111243	4	51111251	5
21111326	6	31111334	7	21111425	8	11111516	9	21111524	10	11111615	11
21112136	12	31112144	13	41112152	14	21112235	15	31112243	16	41112251	17
11112326	18	21112334	19	11112425	20	11113136	21	21113144	22	31113152	23
11113235	24	21113243	25	31113251	26	11113334	27	21113342	28	11114144	29
21114152	30	11114243	31	21114251	32	11115152	33	51116111	34	31121135	35
41121143	36	51121151	37	21121226	38	31121234	39	41121242	40	21121325	41
31121333	42	11121416	43	21121424	44	31121432	45	11121515	46	21121523	47
11121614	48	21122135	49	31122143	50	41122151	51	11122226	52	21122234	53
31122242	54	11122325	55	21122333	56	31122341	57	11122424	58	21122432	59
11123135	60	21123143	61	31123151	62	11123234	63	21123242	64	11123333	65
21123341	66	11124143	67	21124151	68	11124242	69	11124341	70	21131126	71
31131134	72	41131142	73	21131225	74	31131233	75	41131241	76	11131316	77
.
.

- An *error correction level*, s , $0 \leq s \leq 8$, is defined.
- The sequence of codewords (all rows) is interpreted as a *code block* in a $[k + r, k, r + 1]$ shortened Reed Solomon code over $\text{GF}(929)$, where
 - k is the number of codewords used for actual data.
 - Raw data is mapped to codewords using various efficient modes depending on whether the data is numeric, text, binary, or mixed.
 - One bar code can encode more than 1100 raw bytes, 1800 ASCII characters, or 2700 decimal digits, depending on the mode.
 - $r = 2^{s+1}$, so $r \in \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$.
 - $k + r \leq 928$.
- 2 check digits are reserved for *detection*; the rest (if any) are used for *erasure* and *full error* correction.
- The *generator polynomial* of the RS code is

$$g(x) = \prod_{i=1}^r (x - 3^i),$$

3 is primitive in $\text{GF}(929)$.

Application: QR codes



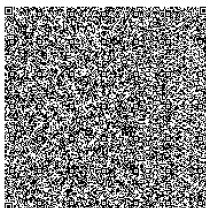
Version 1: 21×21



Version 3: 29×29



Version 10: 57×57



Version 40: 177×177

A truly 2D, highly versatile bar code (array referred to as a *symbol*)

Application: QR codes

Widespread use

- Product or part tracking (original motivation)
- Web links
- Restaurant menus
- Tickets
- Document verification
- ... etc.

Robust ECC allows for data recovery under significant damage, and also for graphic art customization.



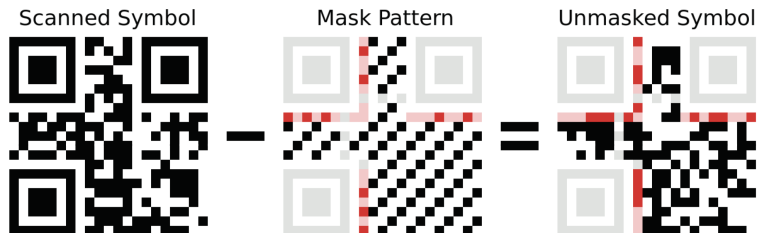
Fully recoverable symbols

QR codes: Versions (= Sizes)

Version	Size	Capacity	Version	Size	Capacity	Version	Size	Capacity	Version	Size	Capacity
<u>M1</u>	11	4½	<u>8</u>	49	242	<u>19</u>	93	991	<u>30</u>	137	2185
<u>M2</u>	13	10	<u>9</u>	53	292	<u>20</u>	97	1085	<u>31</u>	141	2323
<u>M3</u>	15	16½	<u>10</u>	57	346	<u>21</u>	101	1156	<u>32</u>	145	2465
<u>M4</u>	17	24	<u>11</u>	61	404	<u>22</u>	105	1258	<u>33</u>	149	2611
<u>1</u>	21	26	<u>12</u>	65	466	<u>23</u>	109	1364	<u>34</u>	153	2761
<u>2</u>	25	44	<u>13</u>	69	532	<u>24</u>	113	1474	<u>35</u>	157	2876
<u>3</u>	29	70	<u>14</u>	73	581	<u>25</u>	117	1588	<u>36</u>	161	3034
<u>4</u>	33	100	<u>15</u>	77	655	<u>26</u>	121	1706	<u>37</u>	165	3196
<u>5</u>	37	134	<u>16</u>	81	733	<u>27</u>	125	1828	<u>38</u>	169	3362
<u>6</u>	41	172	<u>17</u>	85	815	<u>28</u>	129	1921	<u>39</u>	173	3532
<u>7</u>	45	196	<u>18</u>	89	901	<u>29</u>	133	2051	<u>40</u>	177	3706

Capacity = number of main data bytes (including ECC)

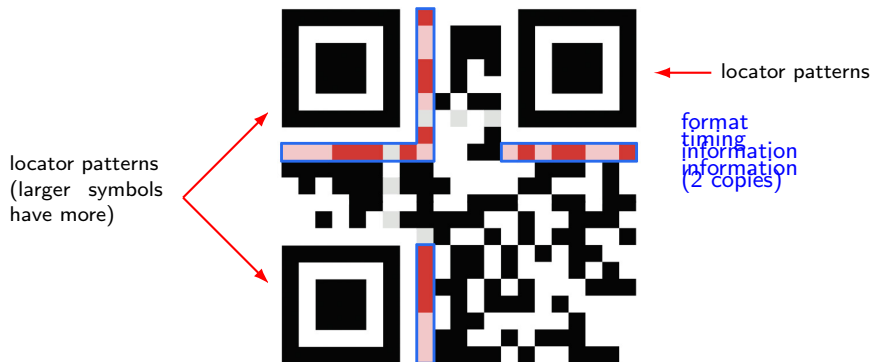
QR codes: masking



- An XOR mask is applied by the encoder to the raw data to minimize undesirable features (large areas of the same color, etc.).
- Several masks are tried, and the resulting array is scored for bad features. Mask with the best score is chosen.
- The choice is encoded in the symbol.

QR codes: structure

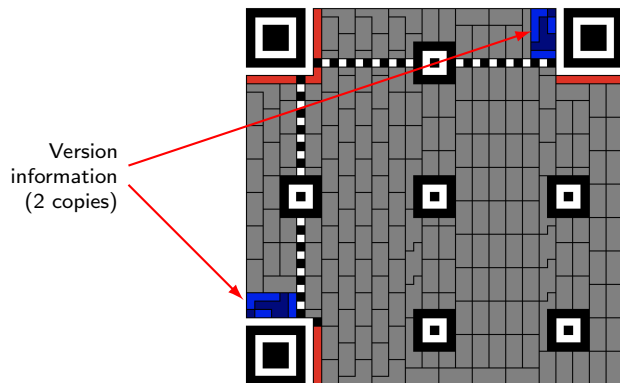
Version 1 symbol: 21×21



Format areas (2 copies): 5 bits of information, encoded with a $[15, 5, 7]$ binary BCH code (small code, exhaustive decoding possible). Format info (5 bits):

- 2 bits: error correction level (4 levels: L, M, Q, H).
- 3 bits: masking pattern.

QR codes: structure

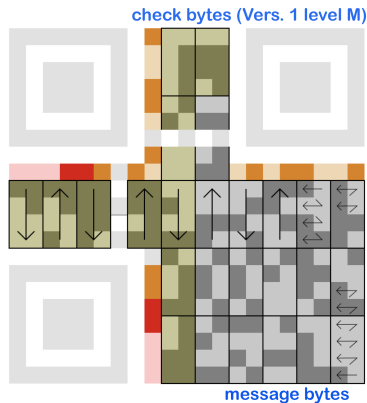


Larger symbols (Version 7: 45×45 and higher) also carry *version information*: 6 bits, encoded with a binary $[18, 6, 8]$ code.

The code is derived from the $[23, 12, 7]$ (perfect) Golay code by taking the even codewords ($[23, 11, 8]$) and shortening.

As with format information, two copies are written.

QR codes: main data with error correction



Data is encoded using *shortened RS codes* over $GF(256)$.

ECC Level	$n, n - k$ for 21×21 symbol	redundancy in general case
L	26, 7	$\approx 14\%$
M	26, 10	$\approx 30\%$
Q	26, 13	$\approx 50\%$
H	26, 17	$\approx 60\%$

For larger symbols:

- Data is broken up into multiple RS blocks (41×41 and larger)
- RS block length is limited so that $n - k \leq 30$ (complexity)
- RS blocks are *interleaved*

Examples:

vers.	array size	ECC level	message bytes	num. blocks $\times (n, n - k)$	ECC level	message bytes	num. blocks $\times (n, n - k)$
10	57×57	L	274	$2 \times (86, 18)$ $2 \times (87, 18)$	Q	154	$6 \times (43, 24)$ $2 \times (44, 24)$
40	177×177	L	2956	$19 \times (148, 30)$ $6 \times (149, 30)$	Q	1666	$34 \times (54, 30)$ $34 \times (55, 30)$