Improved Decoding of Quick Response Codes

Todd Mateer

Quick Response (QR) Code

Quick Response Code Versions

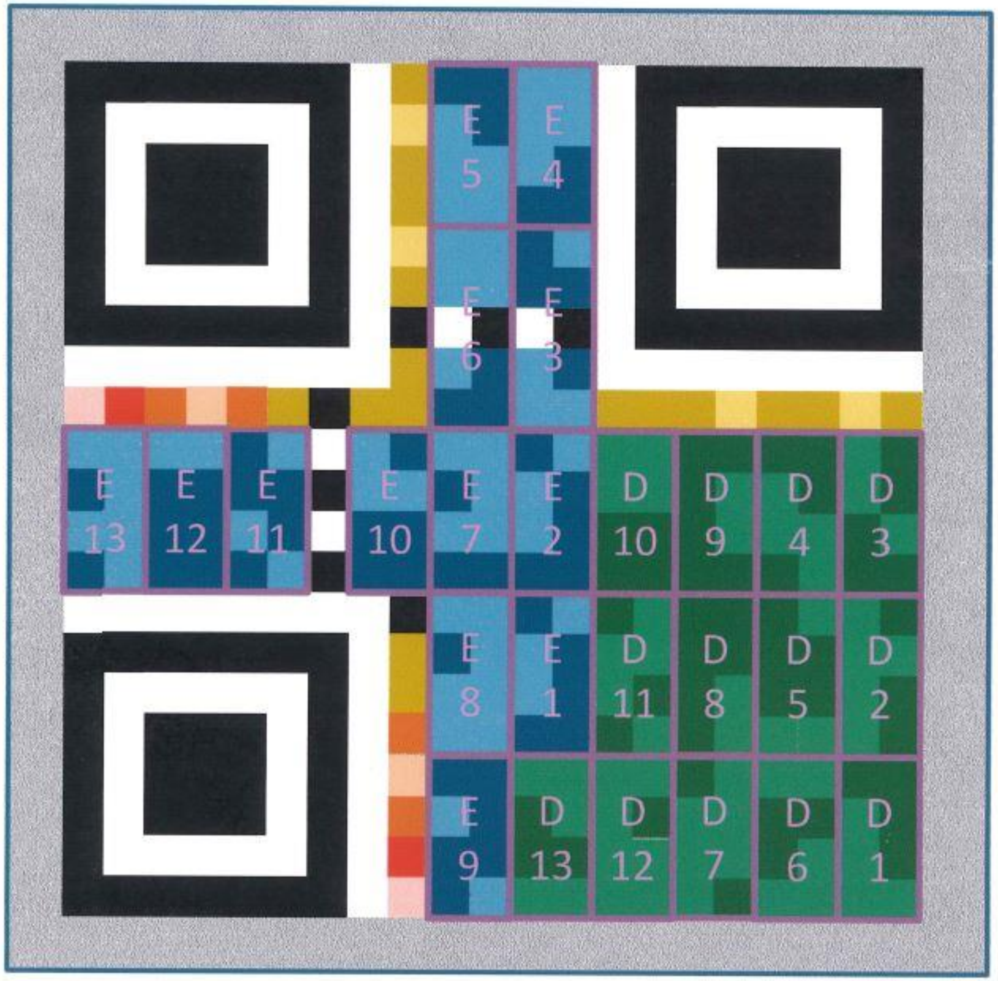
Version 1 Code

Version 40 Code

Quick Response Code Levels

Level	Error-correction capability
L (low)	7 percent of codewords can be restored
M (medium)	15 percent of codewords can be restored
Q (quarter)	25 percent of codewords can be restored
H (high)	30 percent of codewords can be restored

Quick Response Code Anatomy



Alignment and Timing Bits

Error Correction Level Bits

Version 1 QR Code Pattern "Q"

Masking Bits

Pattern uses mask

(15,5) BCH Code Bits

Data is Error Correction Level Bits and Masking Bits

Generator polynomial is $x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$

Data Bits

Error Correction Bits

Common QR Error Sources

Dirty Section

Missing Section

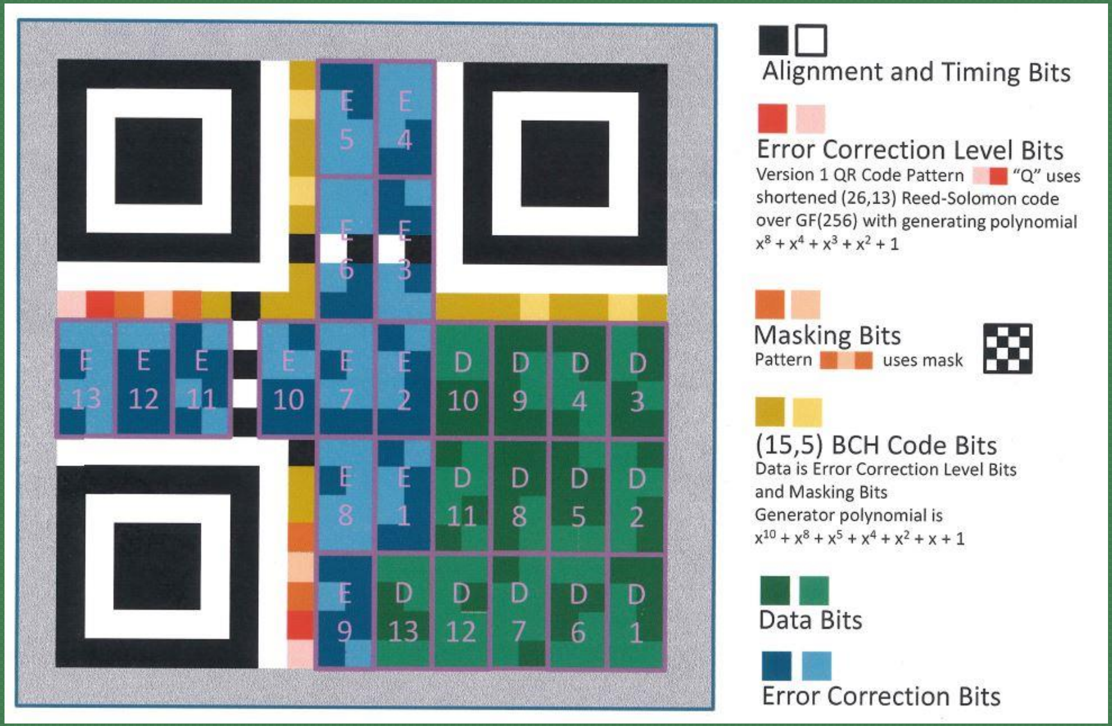
Advertising

Relation to Finite Fields?

- Error correction implemented through Reed-Solomon codes

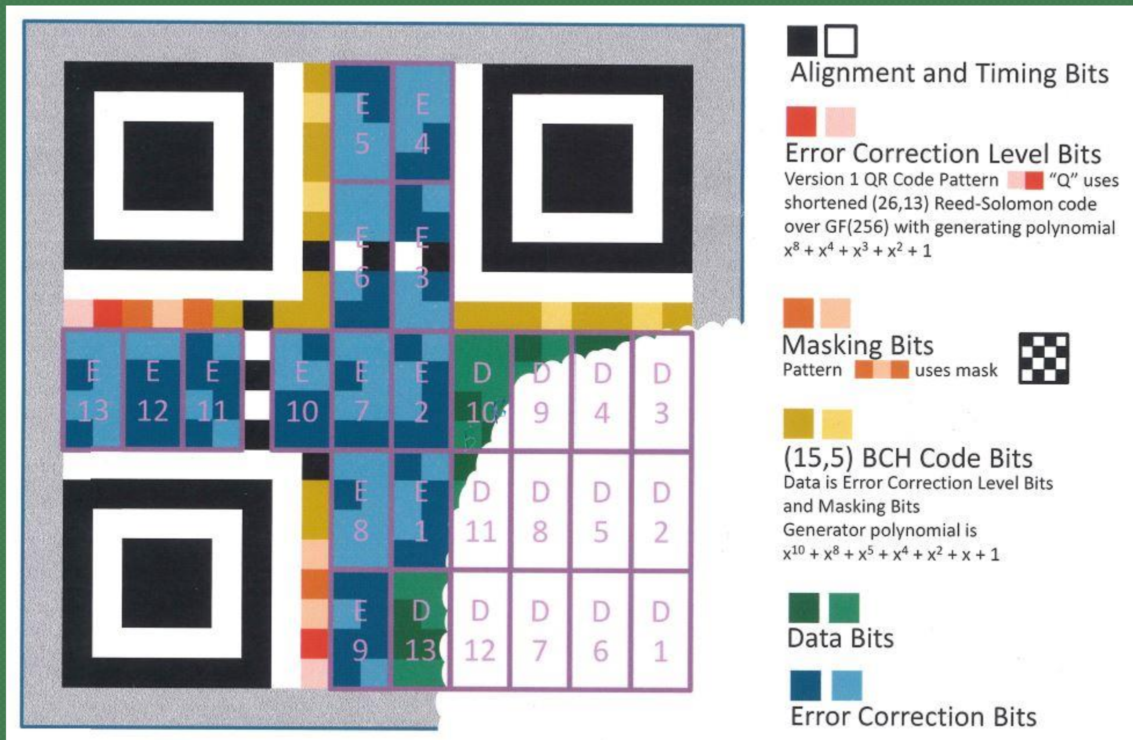
- All elements of the Reed-Solomon code used in QR codes are elements of GF(256) with generating polynomial $x^8 + x^4 + x^3 + x^2 + 1$

Quick Response Code Anatomy



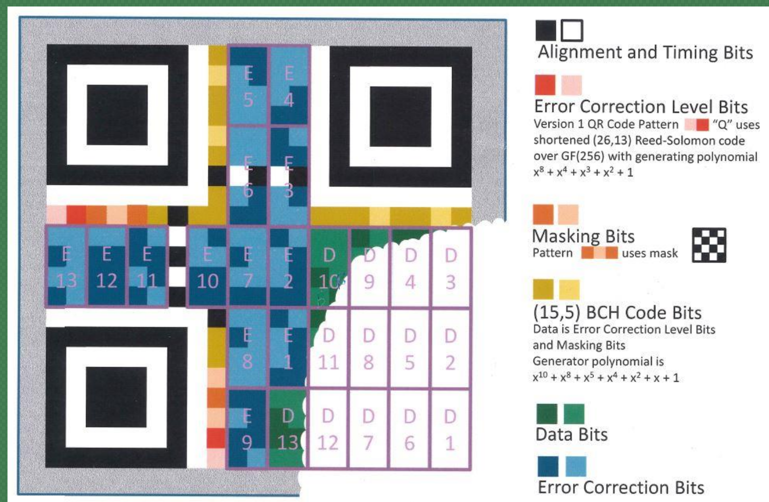
- Level Q code in this example based on (26, 13) Reed-Solomon code
- Code can correct up to 6 of the 26 blocks in this code which is about the advertised 25%

Quick Response Code Anatomy

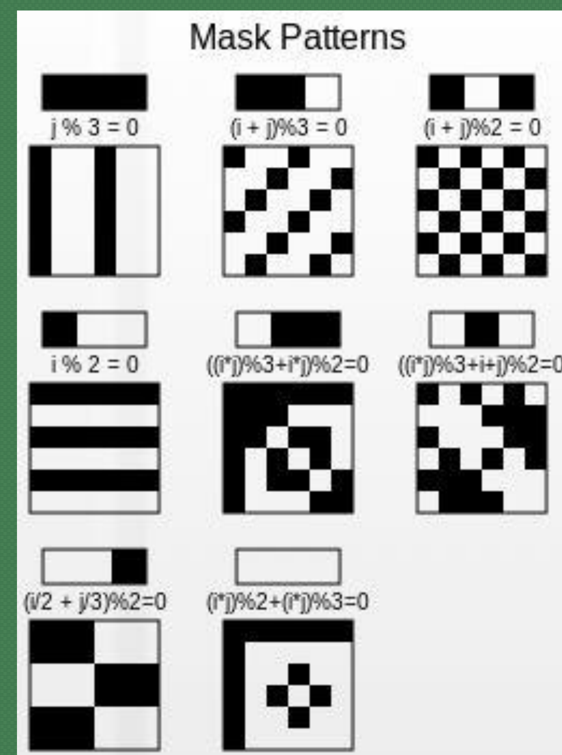


- More than 6 blocks are wrong here, so current QR decoders cannot handle this case.
- Can we take advantage of the fact that we know that certain blocks are known to be wrong to improve our decoding?
- How can a decoder decide whether or not a block is wrong without actually decoding?

Masking

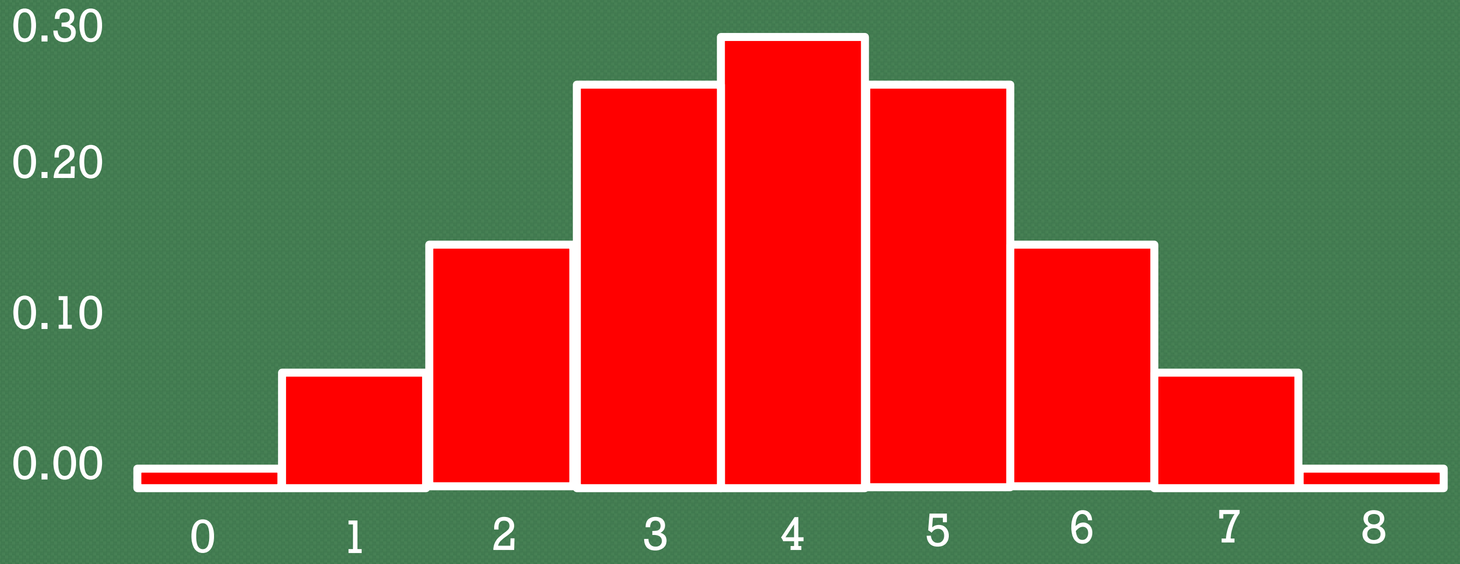


- A mask is applied to all of the original data before producing the actual QR code.
- An algorithm evaluates 8 different masks on the original data and determines which of them makes the bits in the final result appear the most “random”.



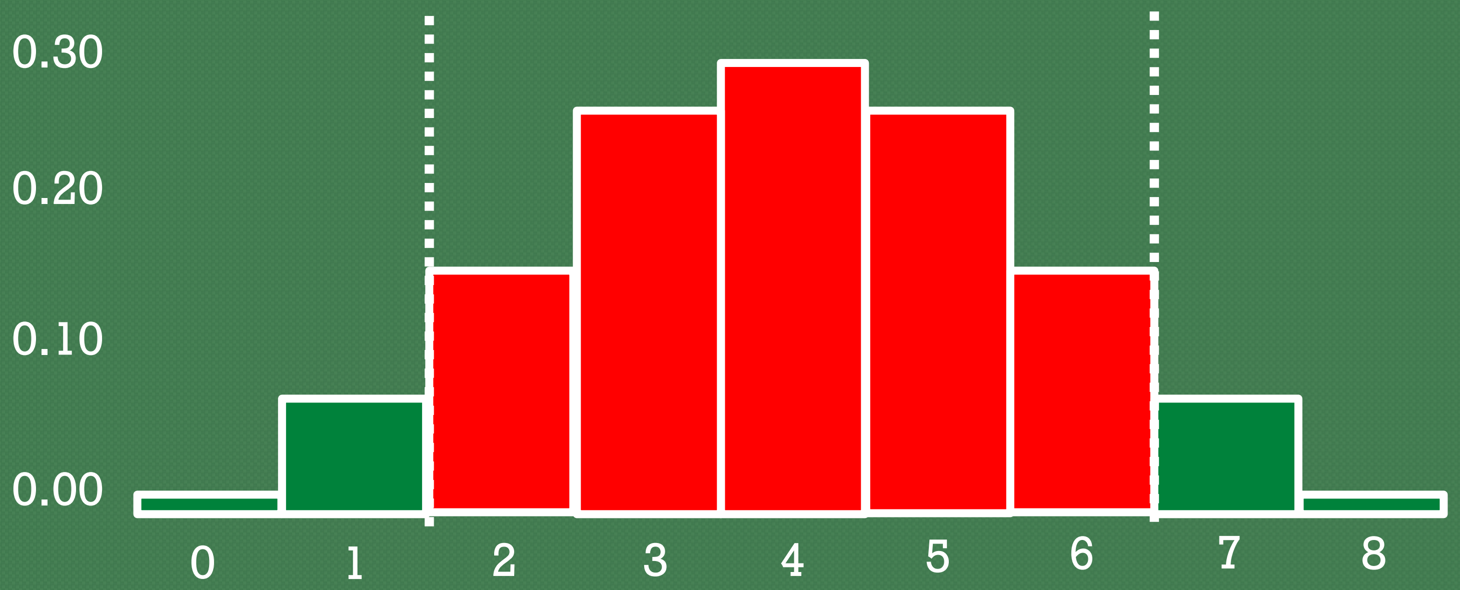
Random Data

- If truly random, each block in QR code would be modeled by 8 Bernoulli trials of p=0.50 (coin flip)
- Distribution of light and dark squares within a block would be governed by the following distribution:

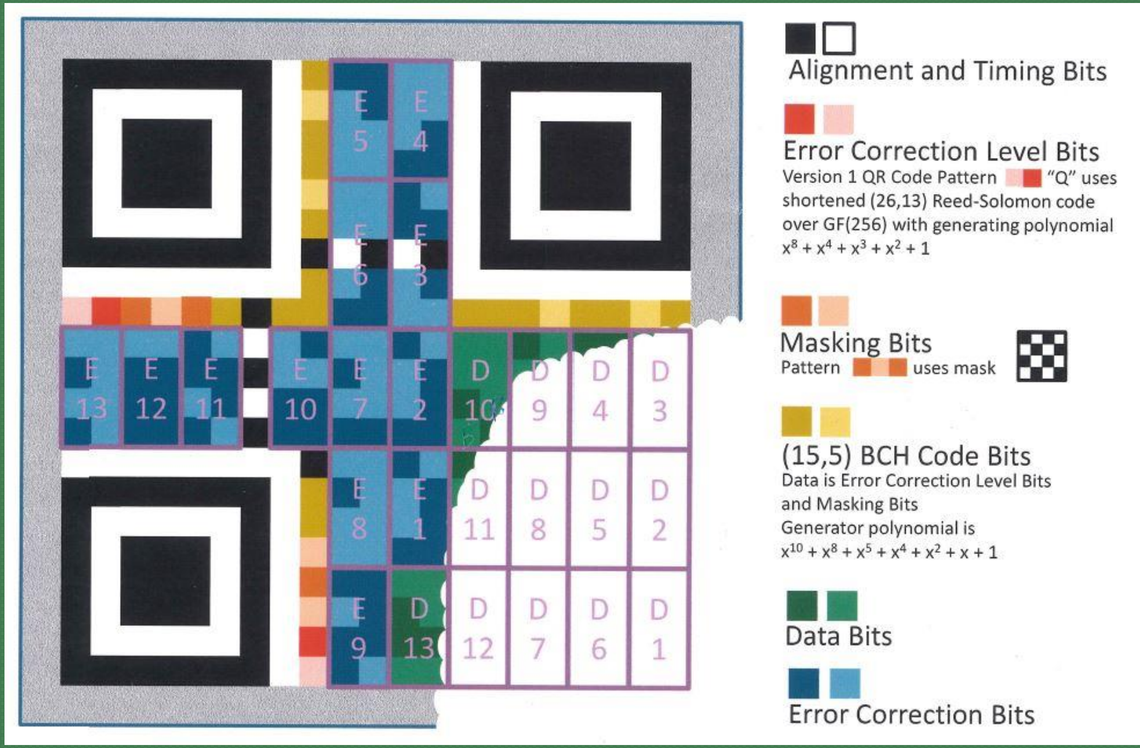


Random Data

- Use Neymann-Pearson Lemma from Statistics to construct a two tail rejection region with probability 90%
- If we observe 0, 1, 7, 8 squares of the same color in a block, we declare it unlikely to have come from the masking process and mark it as a mistake



Quick Response Code Anatomy



- So in this case, we can mark D1-D9, D11-D12 as mistakes.
- Now what can we do about them?

Erasure Decoding

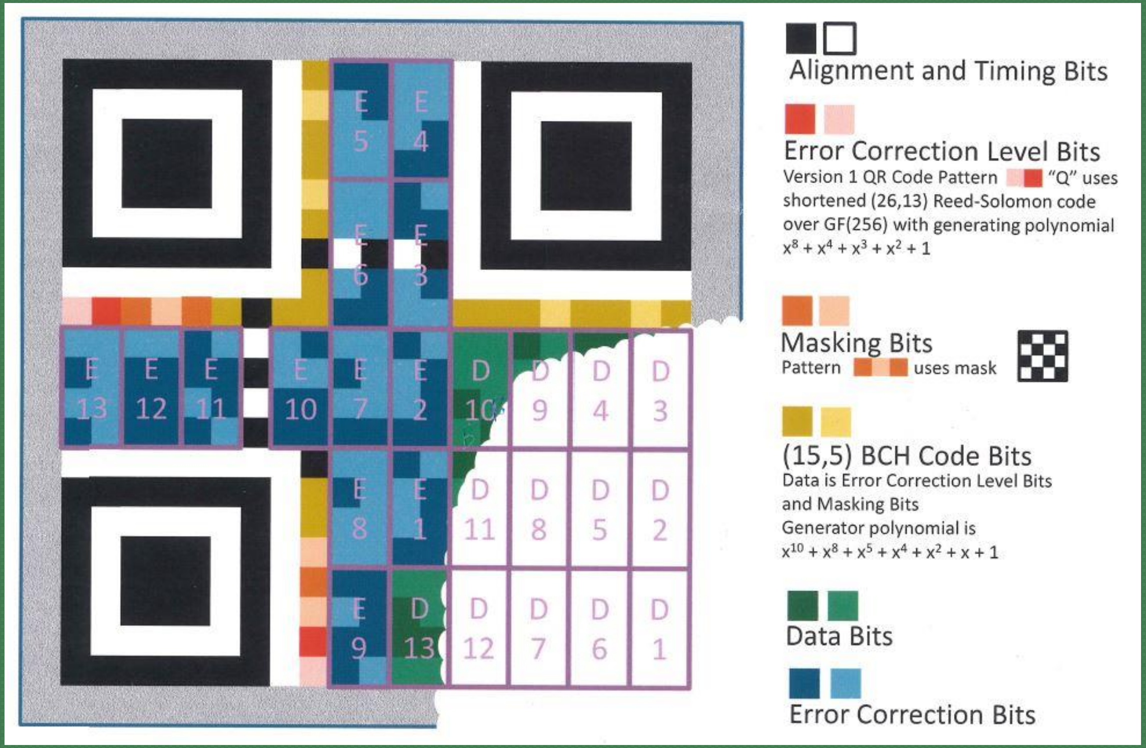
- These "known mistakes" are called "erasures" and can be corrected at "half the cost" of an error where the location is unknown.

- In the example of the (26,13) Level Q QR code, the underlying Reed-Solomon code can correct 6 errors where the location is unknown or up to 13 erasures or any combination that satisfies

  $2t + e \leq 13$ (t = #errors, e=#erasures)

- Python Code for Reed-Solomon decoding with erasures can be found at:
- https://en.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders/Additional_information

Quick Response Code Anatomy



- Here, we'll mark D1-D9, D11-D12 as erasures and correct up to one additional error in an unknown location

New Erasure Decoding Algorithm

- Based on results presented at the 2011 Canadian Workshop on Information Theory
- Main idea from 2011 is that the Berlekamp-Massey algorithm used to solve the "Key Equation" in traditional Reed-Solomon decoding algorithms is equivalent to the Expended Euclidean Algorithm for finding the greatest common divisor of two polynomials over finite fields
- Can also integrate ideas from Eastman (1988) who observed that this step can be computed without the need for any finite field inverses.

Key Equation Solver

Algorithm 4 : Efficient implementation of Key Equation solver

Input: The (possibly modified) syndrome polynomial $S^*(x) \in \mathbb{F}[x]$ for finite field $\mathbb{F}$;
Initialization polynomial $P(x)$ and optional second initialization polynomial $\Upsilon(x)$;
Starting step value K, stopping criteria Q, and integers $t, \mathrm{e} \geq 0$; Inverse flag (INV) equal to 0 or 1

Output: The polynomial $v(x)$ such that $r(x) = u(x) \cdot x^{2t+\mathrm{e}} + v(x) \cdot S^*(x)$
for some polynomials $u(x)$ and $r(x)$ where $\deg(r) < t$. [Optional: and polynomial $\Omega(x)$]

0. Allocate two arrays A and B each of size $2t + \mathrm{e}$ and initialized to all 0.
   [Optional: Allocate two arrays Y and Z each of size $2t + \mathrm{e}$ and initialized to all 0.]
   Set L be the degree of $P(x)$; Set $L_T = L$
   Copy $A[i] = P_i$ (the degree i coefficient of P) and $B[i] = P_i$ for each i in $0 \leq i \leq L$.
   [Opt: Copy $Y[i] = \Upsilon_i$ and $Z[i] = \Upsilon_i$ for each i in $0 \leq i < 2t + \mathrm{e}$.]
   Set pointer V to the starting address of A and T to the starting address of B
   [Opt: Set pointer Ω to the starting address of Y and Φ to the starting address of Z]
   Assign $\psi := 1$ and $\gamma := 1$
   If $(K - L \geq Q)$ then go to step 12
1. Assign $K := K + 1$.
2. Assign $D := \sum_{j=0}^{L} V[j] \cdot S^*[2t + \mathrm{e} + L - K - j]$.
   NOTE: $S^*[i]$ is the degree i coefficient of $S^*(x)$ for all $i \geq 0$
3. If $D = 0$, then go to step 11.
4. Set $C := \psi \cdot D$.
   If $2L < K$ then
5. Assign $T[j] := C \cdot T[j]$ for each j in $0 \leq j \leq L_T$
   [Opt: Assign $\Phi[j] := C \cdot \Phi[j]$ for each j in $0 \leq j < 2t + \mathrm{e}$]
   Then assign $T[j + K - 2L] := T[j + K + 2L] - \gamma \cdot V[j]$ for each j in $0 \leq j \leq L$.
   [Opt: and assign $\Phi[j+K-2L] := \Phi[j+K+2L] - \gamma \cdot \Omega[j]$ for each j in $0 \leq j < 2t+\mathrm{e}+2L-K$.
6. Swap pointers T and V. [Opt: Swap pointers Φ and Ω]. Assign $L_T = L$
   If INV=0, assign $\gamma := D$; If INV=1, assign $\psi := D^{-1}$.
7. Assign $L := K - L$.
8. else
9. If INV=0: Assign $V[j] := \gamma \cdot V[j]$ for each j in $0 \leq j \leq L$
   If INV=0: [Opt: Assign $\Omega[j] := \gamma \cdot \Omega[j]$ for each j in $0 \leq j \leq 2t + \mathrm{e}$]
   Assign $V[j + 2L - K] := V[j + 2L - K] - C \cdot T[j]$ for each j in $0 \leq j \leq L_T$
   [Opt: and $\Omega[j + 2L - K] := \Omega[j + 2L - K] - C \cdot \Phi[j]$ for each j in $0 \leq j < 2t + \mathrm{e} - 2L + K$]
10. end if
11. If $(K - L < Q)$ then go to step 1
12. Return $v(x) = \{V[0], V[1], \ldots, V[L]\}$ [opt: and $\Omega(x) = \{\Omega[0], \Omega[1], \cdots, \Omega[2t + \mathrm{e}]\}$.]

New Erasure Decoding Algorithm

Algorithm 5 : New algorithm for decoding systematic Reed-Solomon code with erasures

Input: The polynomial $r(x) \in \mathbb{F}[x]$ of degree less than n which represents
the received vector of a (n, k, d) Reed-Solomon codeword
transmitted through a noisy environment where $d = n - k + 1$;
the set $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_e\}$ of erasure positions in the received vector;
An integer b. Here, $\mathbb{F}$ is a finite field of characteristic 2.

Output: Either (1) a message polynomial $m(x) \in \mathbb{F}[x]$ of degree less than k
which can be encoded with the Reed-Solomon codeword $c(x) \in \mathbb{F}[x]$
where $c(x)$ and $r(x)$ differ in no more than $t + e$ positions,
(t is the error capacity, e is the number of erasures and $2t + e \leq n - k$)
or (2) "Decoding Failure".

0. Set $t = \lfloor (n - k - e)/2 \rfloor$.
1. Compute the syndrome
   $S(x) = S_{n-k-1} \cdot x^{n-k-1} + \cdots + S_1 \cdot x + S_0$ where $S_i = r(\alpha^{b+i})$.
2. Compute $\Lambda_2(x) := (\alpha^{\epsilon_1} \cdot x - 1) \cdot (\alpha^{\epsilon_2} \cdot x - 1) \cdot \cdots \cdot (\alpha^{\epsilon_e} \cdot x - 1)$.
   NOTE: If $e = 0$, then $\Lambda_2(x) := 1$.
3. Compute $H(x) = (S(x) \cdot \Lambda_2(x)) \bmod x^{2t+e}$ (ignore coefficients of degree $2t+e$ and higher)
4. Set $S^*(x) = H(x)$, $P(x) = 1$, (opt: $\Psi(x) = H(x)$), $K := 0$ and $Q := t$
5. Call Algorithm 4 to solve Key Equation with solution $\{V[0], V[1], \ldots, V[L]\}$.
6. Assign $\Lambda_1(x) := V[L] \cdot x^L + V[L-1] \cdot x^{L-1} + \cdots + V[1] \cdot x + V[0]$.
7. Determine the values $\{i_1, i_2, \ldots, i_\tau\}$ such that $\Lambda_1(\alpha^{-i_j}) = 0$
   for each $1 \leq j \leq \tau$. If $\tau \neq L$, then return "Decoding Failure";
8. If (τ is equal to L) then
9. Compute $\Lambda_1'(x)$ and $\Lambda_2'(x)$, the formal derivatives of $\Lambda_1(x)$ and $\Lambda_2(x)$ respectively.
10. Compute $\Omega(x) = \Lambda_1(x) \cdot H(x) \bmod x^{2t+e}$ (or add optional code of Algorithm 4)
11. Let $c(x) = r(x)$. For each $1 \leq j \leq \tau$, change
    $c_{i_j} = r_{i_j} + \Omega(\alpha^{-i_j}) / ((\alpha^{-i_j})^{1-b} \cdot \Lambda_1'(\alpha^{-i_j}) \cdot \Lambda_2(\alpha^{-i_j}))$
12. For each $1 \leq j \leq e$, change
    $c_{\epsilon_j} = r_{\epsilon_j} + \Omega(\alpha^{-\epsilon_j}) / ((\alpha^{-\epsilon_j})^{1-b} \cdot \Lambda_1(\alpha^{-\epsilon_j}) \cdot \Lambda_2'(\alpha^{-\epsilon_j}))$
13. End if
14. Extract $m(x)$ from the coefficients of $c(x)$ of degree $n - k$ and higher.
15. Return $m(x)$.

- This "Key Equation" solver can be inserted into a new Errors and Erasures Decoding Algorithm
- Error locator polynomial and Erasure polynomial computed separately

Blahut Traditional Algorithm

Algorithm 6 : Blahut algorithm for Reed-Solomon decoding (modified to use syndrome $\widehat{S}(x)$)

Input: The polynomial $r(x) \in \mathbb{F}[x]$ of degree less than n which represents
the received vector of a (n, k, d) Reed-Solomon codeword
transmitted through a noisy environment where $d = n - k + 1$;
the set $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_e\}$ of erasure positions in the received vector; An integer b.
Here, $\mathbb{F}$ is a finite field of characteristic 2.

Output: Either (1) a message polynomial $m(x) \in \mathbb{F}[x]$ of degree less than k
which can be encoded with the Reed-Solomon codeword $c(x) \in \mathbb{F}[x]$
where $c(x)$ and $r(x)$ differ in no more than $t + \mathrm{e}$ positions
(t is the error capacity, e is the number of erasures and $2t + \mathrm{e} \le n - k$),
or (2) "Decoding Failure".

0. Set $t = \lfloor (n - k - \mathrm{e})/2 \rfloor$.
1. Compute the syndrome
   $\widehat{S}(x) = \widehat{S}_{n-k-1} \cdot x^{n-k-1} + \cdots + \widehat{S}_1 \cdot x + \widehat{S}_0$ where $\widehat{S}_j = r(\alpha^{n-k-j+b-1})$.
2. Compute $W_2(x) := (x - \alpha^{\epsilon_1}) \cdot (x - \alpha^{\epsilon_2}) \cdot \cdots \cdot (x - \alpha^{\epsilon_e})$
   NOTE: If $\mathrm{e} = 0$, then $W_2(x) := 1$.
3. Set S^* to point to the degree e coefficient of $\widehat{S}(x)$.
   So $S^*[i]$ will be the degree $i + e$ coefficient of $\widehat{S}(x)$ for all $i \ge 0$.
4. Set $P := W_2(x)$
5. Set $K := 2e$ and $\mathrm{Q} := t + \mathrm{e}$
6. Call Algorithm 4 to solve Key Equation with solution $\{V[0], V[1], \ldots, V[L]\}$.
7. Assign $\Lambda(x) := V[0] \cdot x^L + V[1] \cdot x^{L-1} + \cdots + V[L-1] \cdot x + V[L]$.
8. Determine the positions $\{i_1, i_2, \ldots, i_\tau\}$ such that $\Lambda(\alpha^{-i_j}) = 0$ and $i_j \notin \{\epsilon_1, \epsilon_2, \ldots, \epsilon_e\}$
   for each $1 \le j \le \tau$. NOTE: The roots of $\Lambda(x)$ include both errors and erasures.
   If $\tau + \mathrm{e} \ne L$, then return "Decoding Failure";
9. If ($\tau + \mathrm{e}$ is equal to L) then
10. Compute $\Lambda'(x)$, the formal derivative of $\Lambda(x)$.
11. Compute $S(x) = \widehat{S}_0 \cdot x^{n-k-1} + \widehat{S}_1 \cdot x^{n-k-2} + \cdots + \widehat{S}_{n-k-2} \cdot x + \widehat{S}_{n-k-1}$.
12. Compute $\Omega(x) = \Lambda(x) \cdot S(x) \bmod x^{n-k}$
13. Let $c(x) = r(x)$. For each $1 \le j \le \tau$, change $c_{i_j} = r_{i_j} + \Omega(\alpha^{-i_j})/((\alpha^{-i_j})^{1-b} \cdot \Lambda'(\alpha^{-i_j}))$
14. For each $1 \le j \le \mathrm{e}$, change $c_{\epsilon_j} = r_{\epsilon_j} + \Omega(\alpha^{-\epsilon_j})/((\alpha^{-\epsilon_j})^{1-b} \cdot \Lambda'(\alpha^{-\epsilon_j}))$
15. End if
16. Extract $m(x)$ from the coefficients of $c(x)$ of degree $n - k$ and higher.
17. Return $m(x)$.

- Need to "reverse" all of the polynomials from Algorithm 5 to correspond.
- Error locator polynomial and Erasure polynomial computed together

Truong-Jeng-Chang Algorithm

Algorithm 7 : Truong-Jeng-Chang algorithm for decoding systematic Reed-Solomon code with erasures

Input: The polynomial $r(x) \in \mathbb{F}[x]$ of degree less than n which represents
the received vector of a (n, k, d) Reed-Solomon codeword
transmitted through a noisy environment where $d = n - k + 1$;
the set $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_e\}$ of erasure positions in the received vector;
An integer b. Here, $\mathbb{F}$ is a finite field of characteristic 2.

Output: Either (1) a message polynomial $m(x) \in \mathbb{F}[x]$ of degree less than k
which can be encoded with the Reed-Solomon codeword $c(x) \in \mathbb{F}[x]$
where $c(x)$ and $r(x)$ differ in no more than $t + e$ positions,
(t is the error capacity, e is the number of erasures and $2t + e \leq n - k$)
or (2) "Decoding Failure".

0. Set $t = \lfloor (n - k - e)/2 \rfloor$.
1. Compute the syndrome
   $S(x) = \bar{S}_{n-k-1} \cdot x^{n-k-1} + \cdots + \bar{S}_1 \cdot x + \bar{S}_0$ where $\bar{S}_i = r(\alpha^{b+i})$.
2. Compute $\Lambda_2(x) := (\alpha^{\epsilon_1} \cdot x - 1) \cdot (\alpha^{\epsilon_2} \cdot x - 1) \cdot \cdots \cdot (\alpha^{\epsilon_e} \cdot x - 1)$.
   NOTE: If $e = 0$, then $\Lambda_2(x) := 1$.
3. Compute $H(x) = (S(x) \cdot \Lambda_2(x)) \bmod x^{2t+e}$ (ignore coefficients of degree $2t + e$ and higher)
4. Set S^* to point to the degree e coefficient of $\widehat{S}(x)$.
   So $S^*[i]$ will be the degree $i + e$ coefficient of $\widehat{S}(x)$ for all $i \geq 0$.
5. Set $P(x) = 1$, (opt: $\Upsilon(x) = H(x)$), $K := 2e$ and $Q := t + e$
6. Call Algorithm 4 to solve Key Equation with solution $\{V[0], V[1], \ldots, V[L]\}$.
7. Assign $\Lambda(x) := V[0] \cdot x^L + V[1] \cdot x^{L-1} + \cdots + V[L-1] \cdot x + V[L]$.
8. Determine the positions $\{i_1, i_2, \ldots, i_\tau\}$ such that $\Lambda(\alpha^{-i_j}) = 0$ and $i_j \notin \{\epsilon_1, \epsilon_2, \ldots, \epsilon_e\}$
   for each $1 \leq j \leq \tau$. NOTE: Roots of $\Lambda(x)$ include both errors and erasures.
   If $\tau \neq L$, then return "Decoding Failure";
9. If ($\tau + e$ is equal to L) then
10. Compute $\Lambda'(x)$, the formal derivative of $\Lambda(x)$.
12. Compute $\Omega(x) = \Lambda(x) \cdot S(x) \bmod x^{n-k}$ (or use optional code of Algorithm 4)
13. Let $c(x) = r(x)$. For each $1 \leq j \leq \tau$, change $c_{i_j} = r_{i_j} + \Omega(\alpha^{-i_j})/((\alpha^{-i_j})^{1-b} \cdot \Lambda'(\alpha^{-i_j}))$
14. For each $1 \leq j \leq e$, change $c_{\epsilon_j} = r_{\epsilon_j} + \Omega(\alpha^{-\epsilon_j})/((\alpha^{-\epsilon_j})^{1-b} \cdot \Lambda'(\alpha^{-\epsilon_j}))$
15. End if
16. Extract $m(x)$ from the coefficients of $c(x)$ of degree $n - k$ and higher.
17. Return $m(x)$.

- Intermediate results of this algorithm should generally correspond to Algorithm 5
- Error locator polynomial and Erasure polynomial computed together

Performance Results

	Algorithm 5	Algorithm 6	Algorithm 7
8 errors, 0 erasures	70.07 microseconds	68.49 microseconds	70.13 microseconds
4 errors, 8 erasures	59.73 microseconds	82.45 microseconds	83.42 microseconds
1 error, 14 erasures	54.98 microseconds	92.87 microseconds	93.15 microseconds
0 errors, 16 erasures	54.14 microseconds	53.47 microseconds	53.71 microseconds

- All three algorithms perform about the same in the errors-only case and the erasures-only case
- New Algorithm (#5) performs better than the other two algorithms in the case where there is a mixture of errors and erasures.

Concluding Remarks

- Introduced new Reed-Solomon decoding algorithm advantageous in cases involving both errors and erasures
- Decoding of QR Codes with erasure locations that can be determined using statistics provides one application of this algorithm