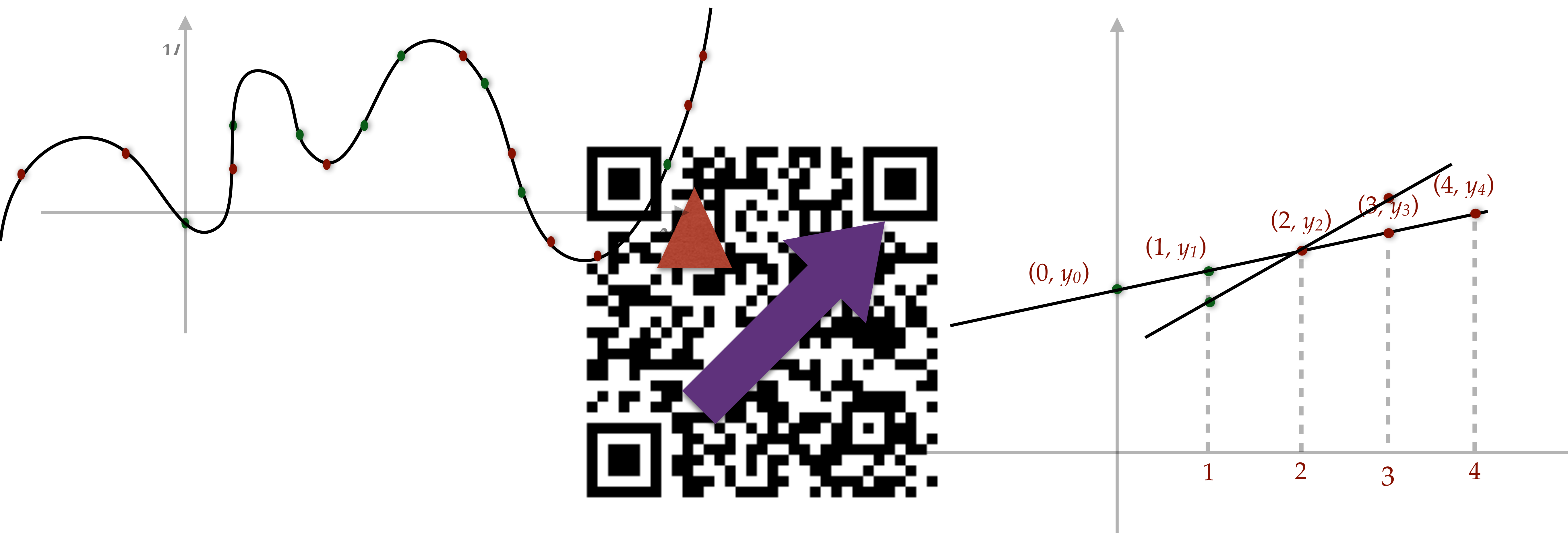
# Algebra in Computer Science: QR codes

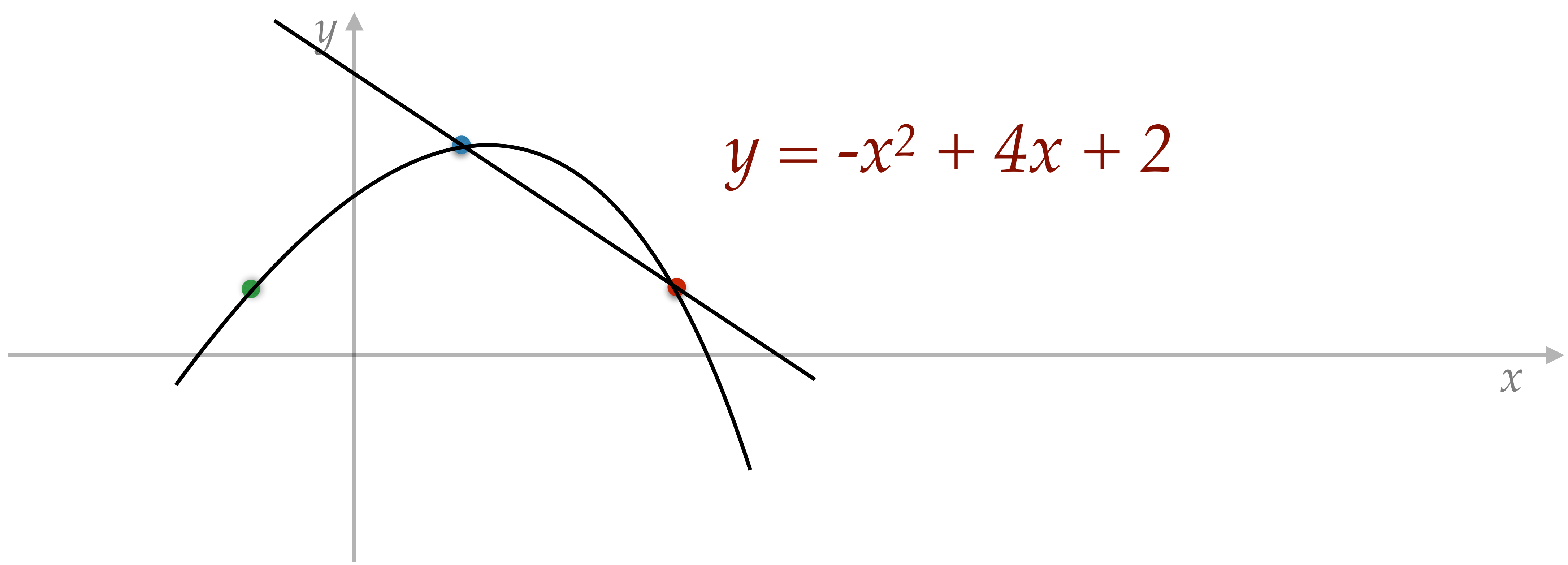


# Algebra

- Addition, multiplication, division
- Polynomials, roots, evaluations
- Modular arithmetic
  - $9 + 4 \times 7 \equiv 13 \pmod{24}$
  - $5 \times 4 \equiv -1 \pmod{7}$
- Algebra has wide range of applications in computer science
  - Data compression
  - Reliable and secure communication
  - Efficient verification of computation
  - Software verification

# Basic fact from Algebra

- A polynomial $f(x)$ of degree $d$ has at most $d$ roots.
- Equivalently,
- Given $d+1$ points in the plane, there is a unique degree $d$ curve passing through them.



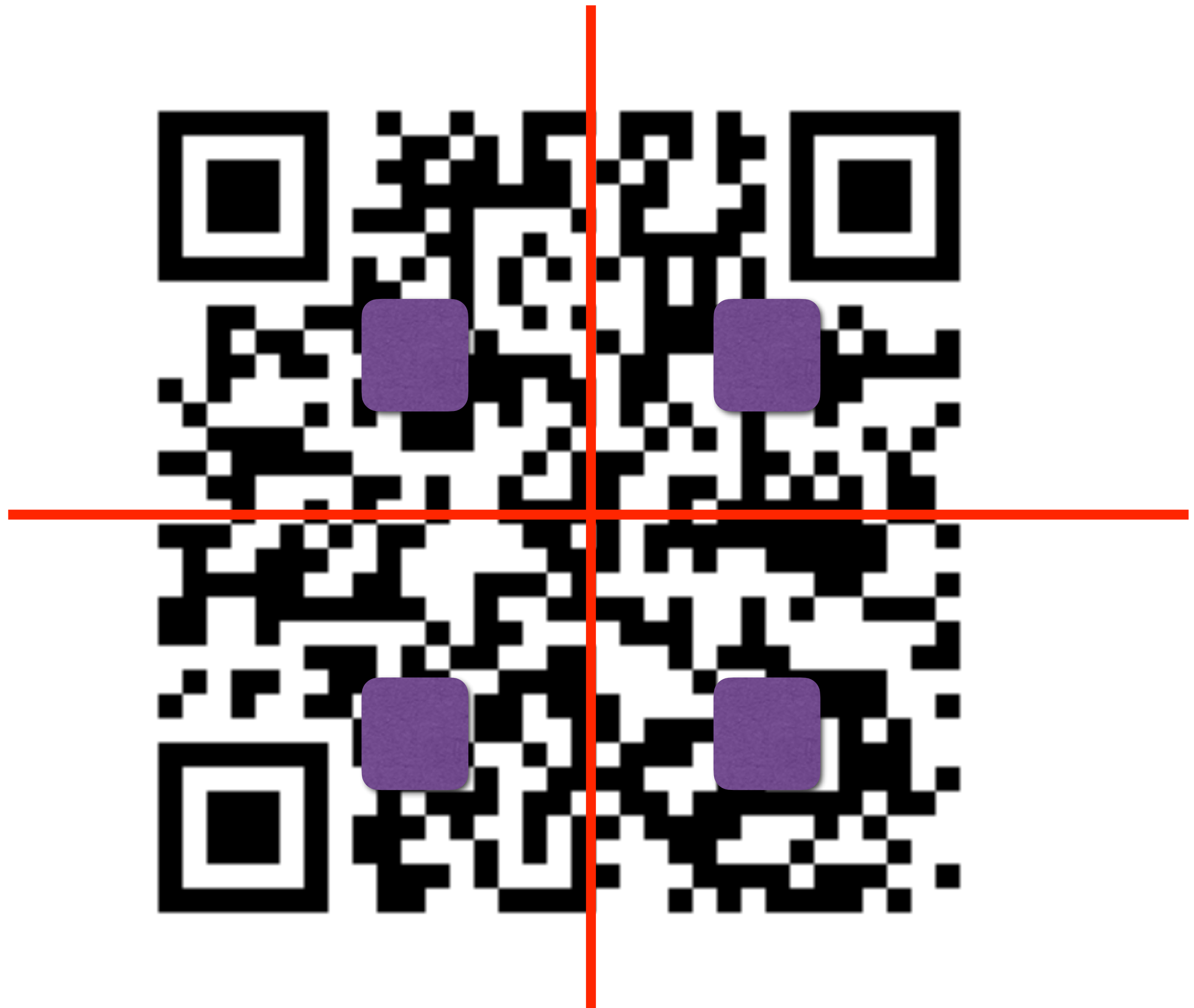
# Basic facts from Linear Algebra

- $n$ linear equations in $n$ variables
- Unique solution if they are linearly independent
- If the RHS is zero, then there is no nonzero solution.
- The facts about roots of a polynomial and solutions for system of linear equations hold true over modular arithmetic (modulo a prime).
- More generally over Galois Fields.

# QR codes

- Can be read, even when partially occluded / erased

# Redundancy in QR codes



- Can be read, even when partially erased or modified
- Guess: the information is copied multiple times
- Possibly, the same bit gets erased from each copy

# Redundancy in QR codes

- Can be read, even when 30% portion from anywhere is erased or modified (Level H)

# Redundancy in QR codes

- Can be read, even when 30% portion from anywhere is erased or modified (Level H)

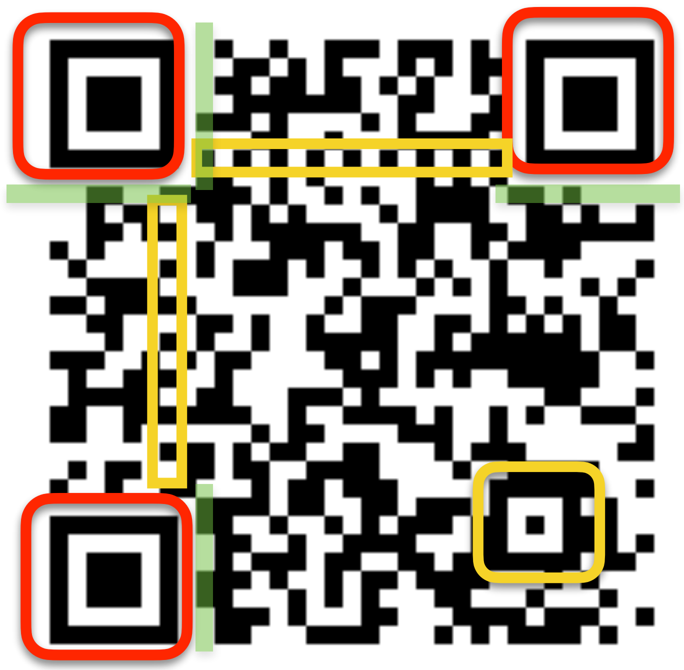
# Redundancy in QR codes

- Can be read, even when 30% portion from anywhere is erased or modified (Level H)

# Redundancy in QR codes

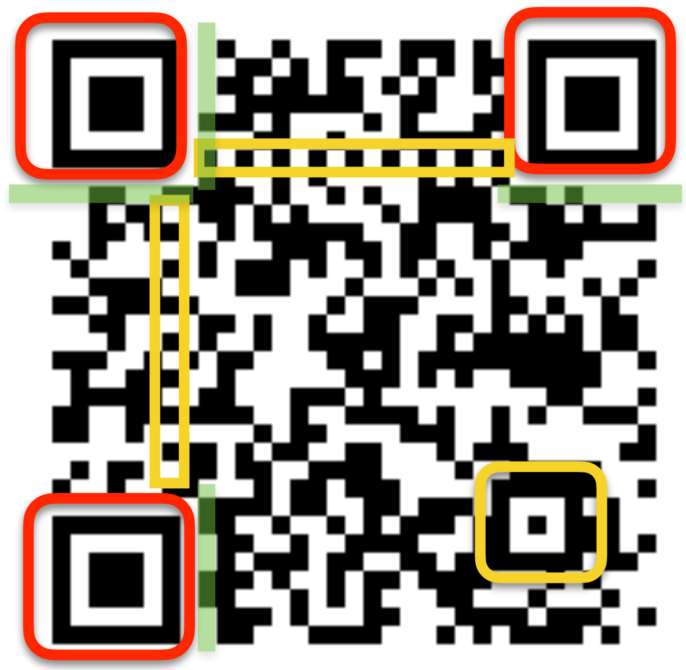
- Too much erased, cannot be read

# Pixel pattern



- 33 × 33 grid of pixels
- 1089 bits or ~136 bytes
- Some pixel patterns are used for position and alignment detection
- Some pixels encode format information, like error correction level
- 100 bytes of data can be stored.

# Information redundancy

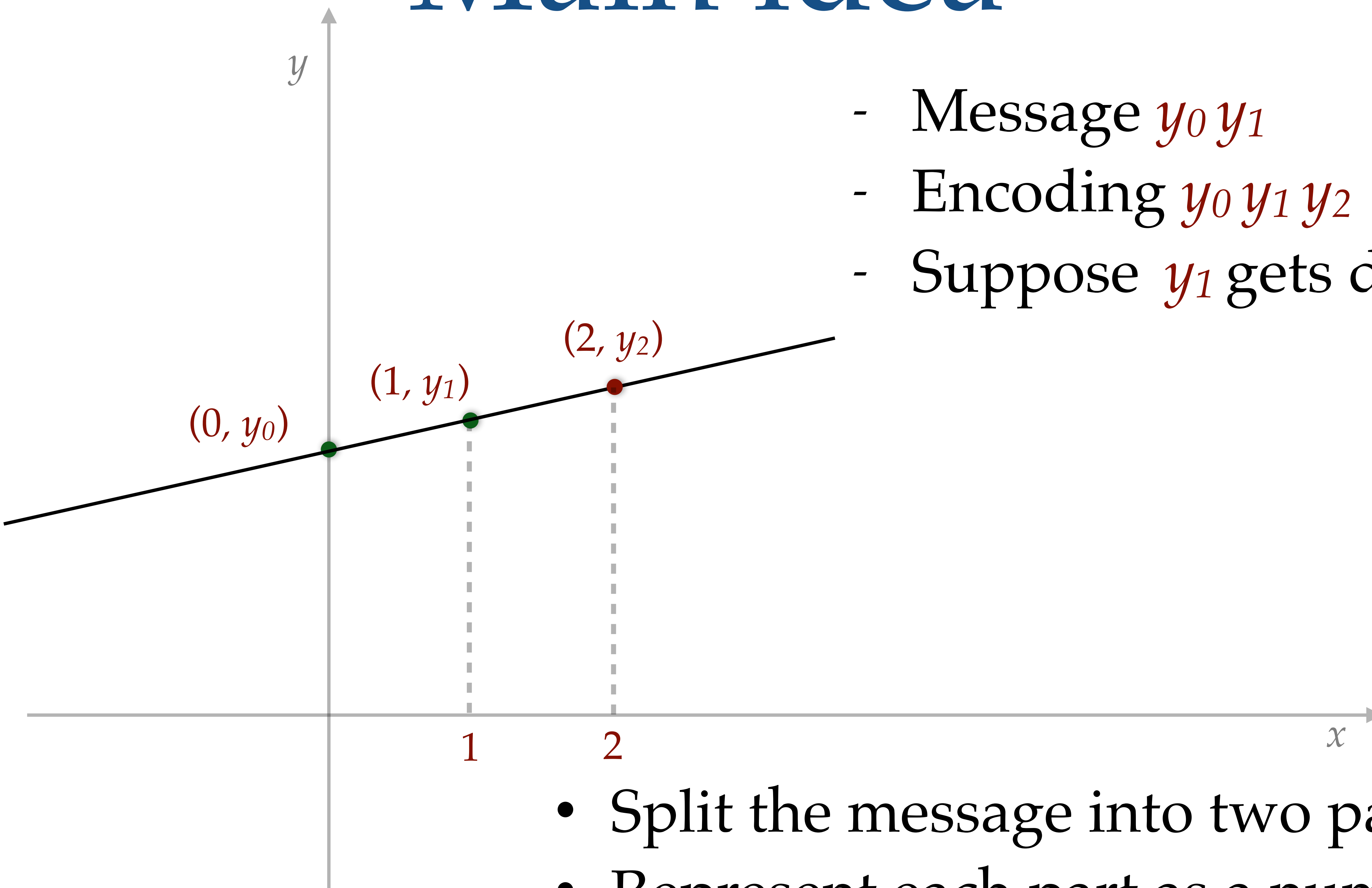


- 100 bytes of data can be stored.
- www.cse.iitb.ac.in/~risc2024/
  29 characters
- Could have stored at most 3 copies
- Level H error correction: guaranteed to work even if any 32 bytes are deleted or modified

# Challenge

- 36 bytes of information.
- Store it using 100 bytes.
- Recover after any 32 bytes are deleted or modified.
- Simply duplicating the data will not work
- Coding theory: algebra and geometry

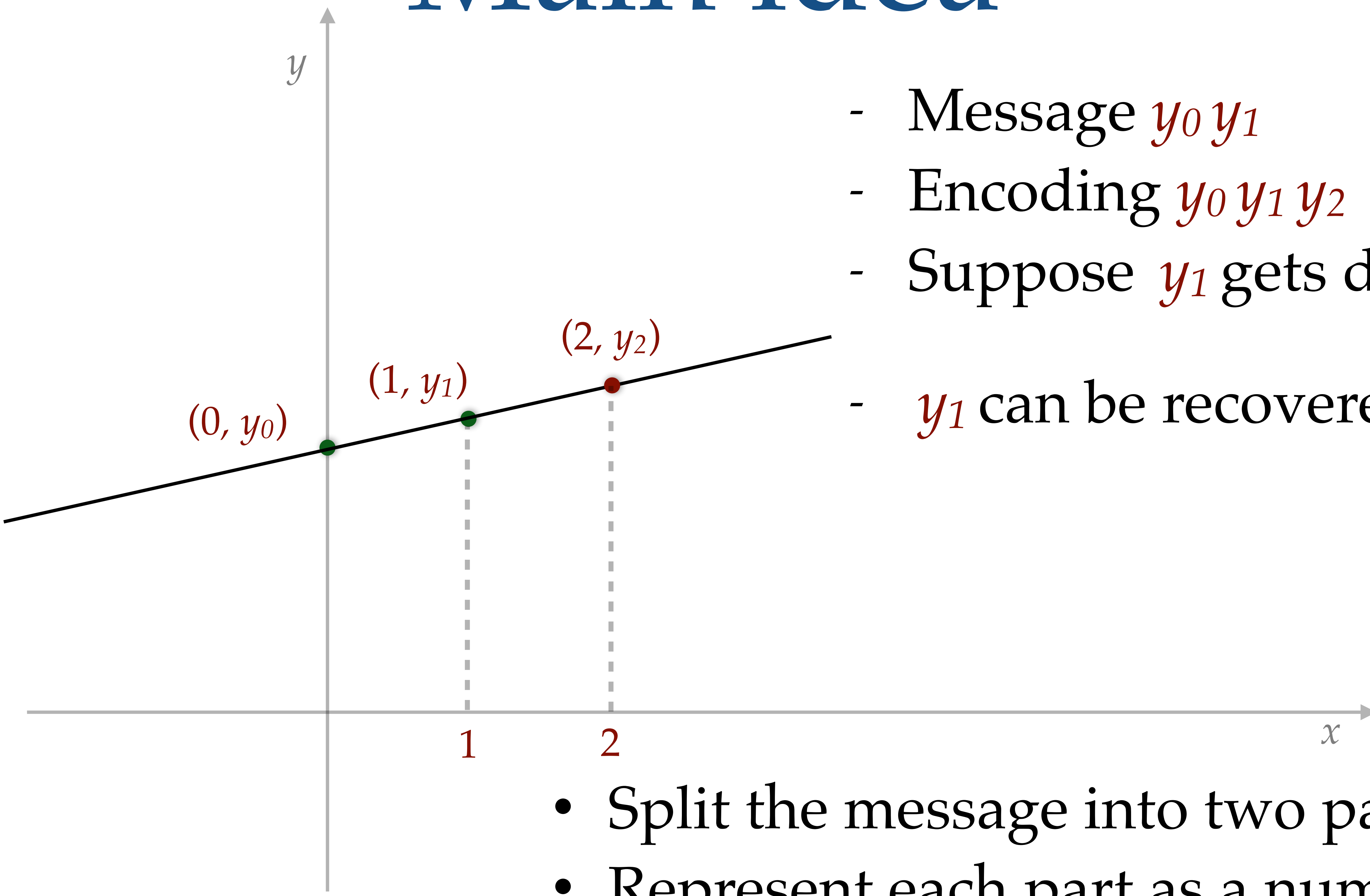
# Main idea




- Message $y_0\, y_1$
- Encoding $y_0\, y_1\, y_2$
- Suppose $y_1$ gets deleted

- Split the message into two parts
- Represent each part as a number, say $y_0$ and $y_1$

# Main idea

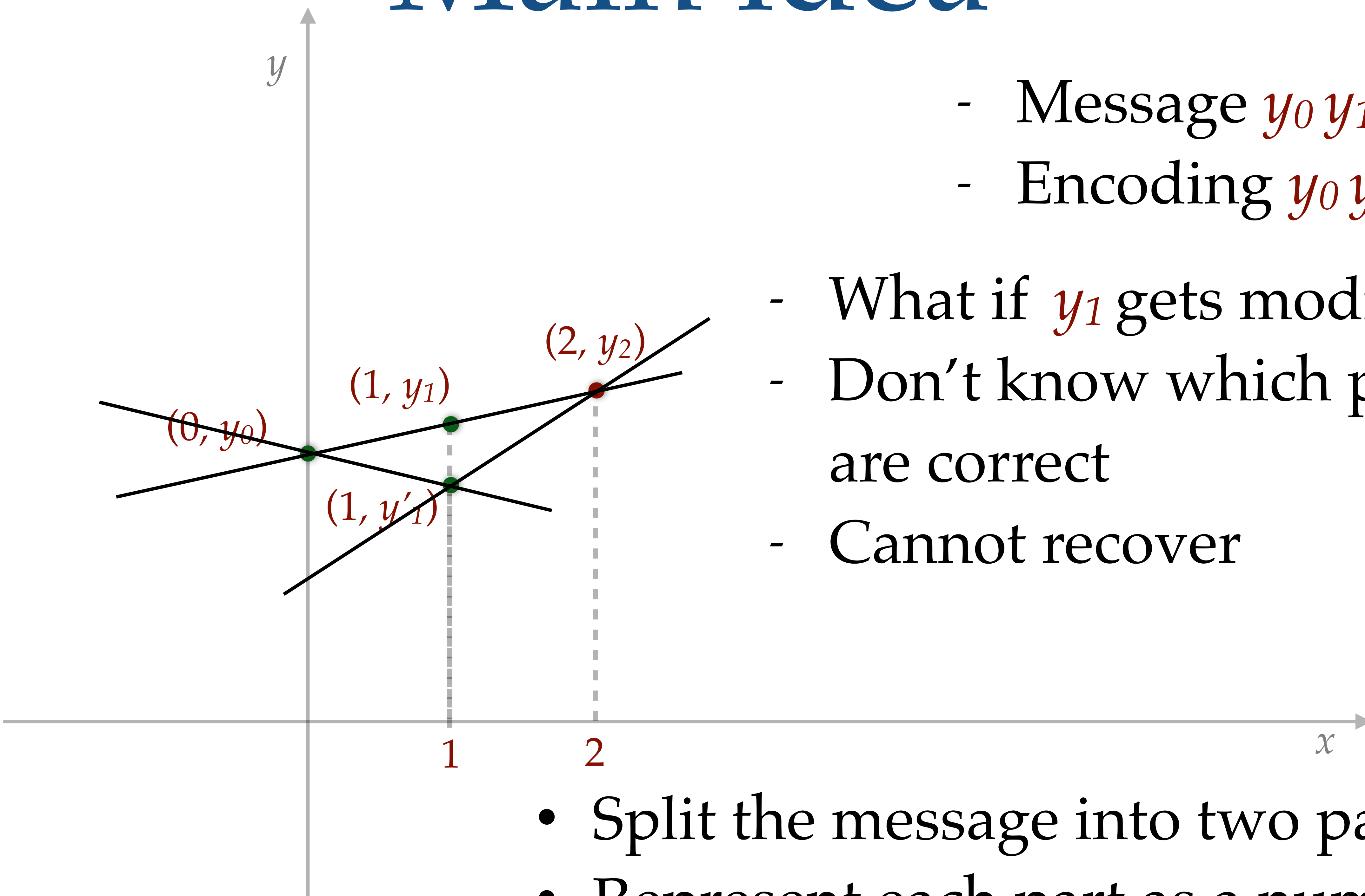



- Message $y_0\,y_1$
- Encoding $y_0\,y_1\,y_2$
- Suppose $y_1$ gets deleted

- $y_1$ can be recovered

• Split the message into two parts
• Represent each part as a number, say $y_0$ and $y_1$

# Main idea



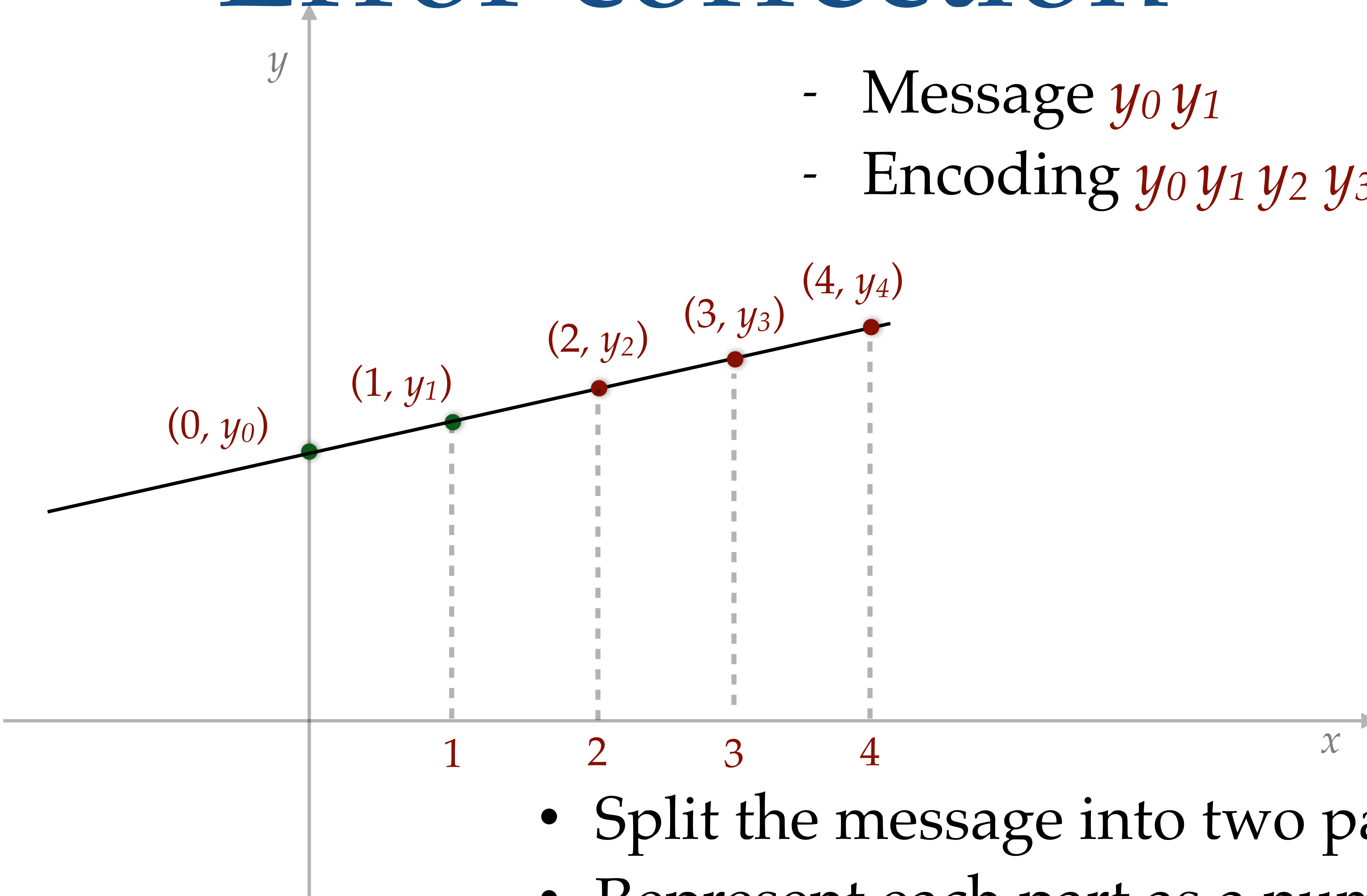

- Message $y_0\, y_1$
- Encoding $y_0\, y_1\, y_2$

- What if $y_1$ gets modified?
- Don't know which parts are correct
- Cannot recover

- Split the message into two parts
- Represent each part as a number, say $y_0$ and $y_1$

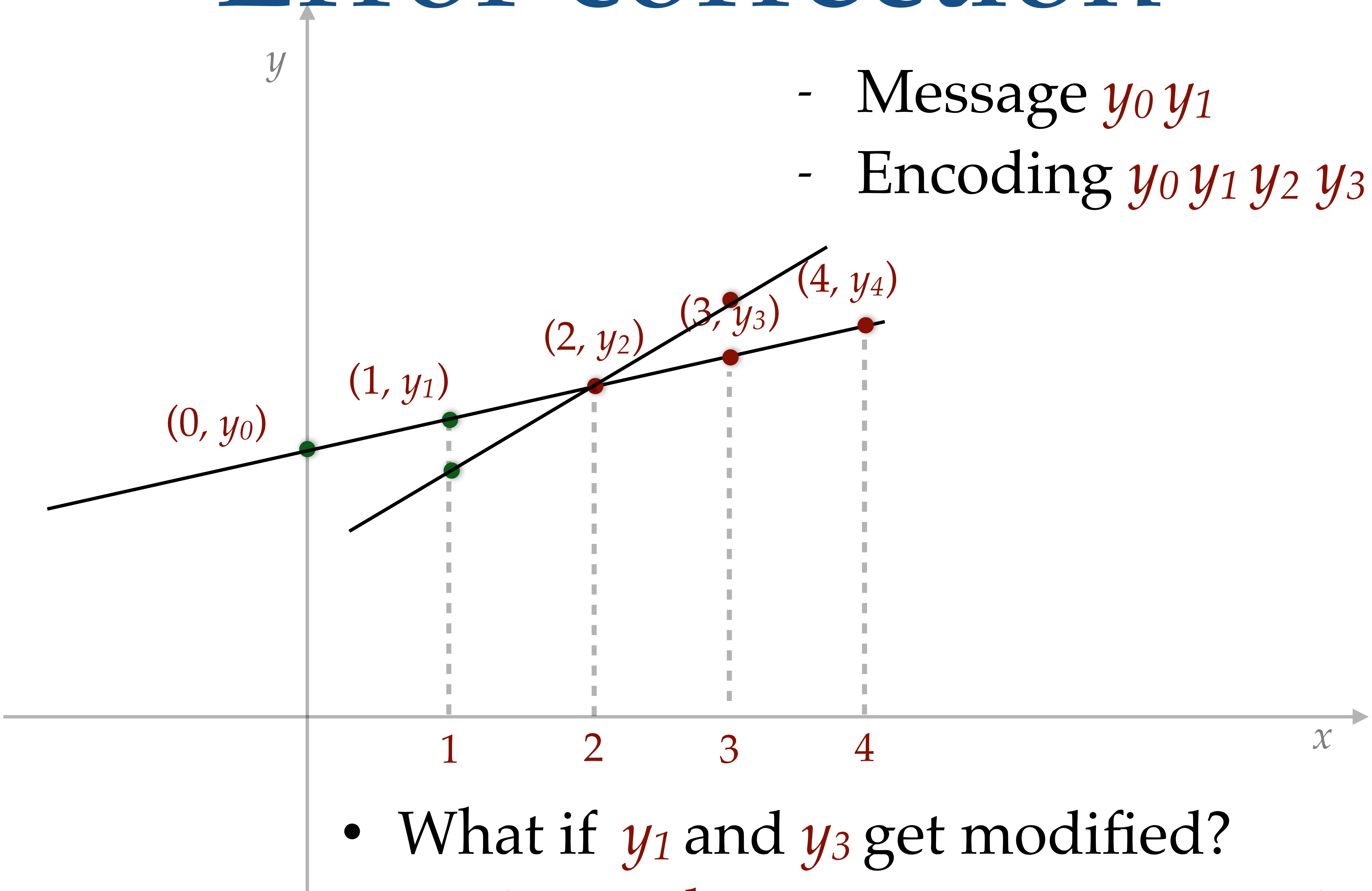
# Error correction

- Message $y_0\,y_1$
- Encoding $y_0\,y_1\,y_2\,y_3\,y_4$




- Split the message into two parts
- Represent each part as a number, say $y_0$ and $y_1$

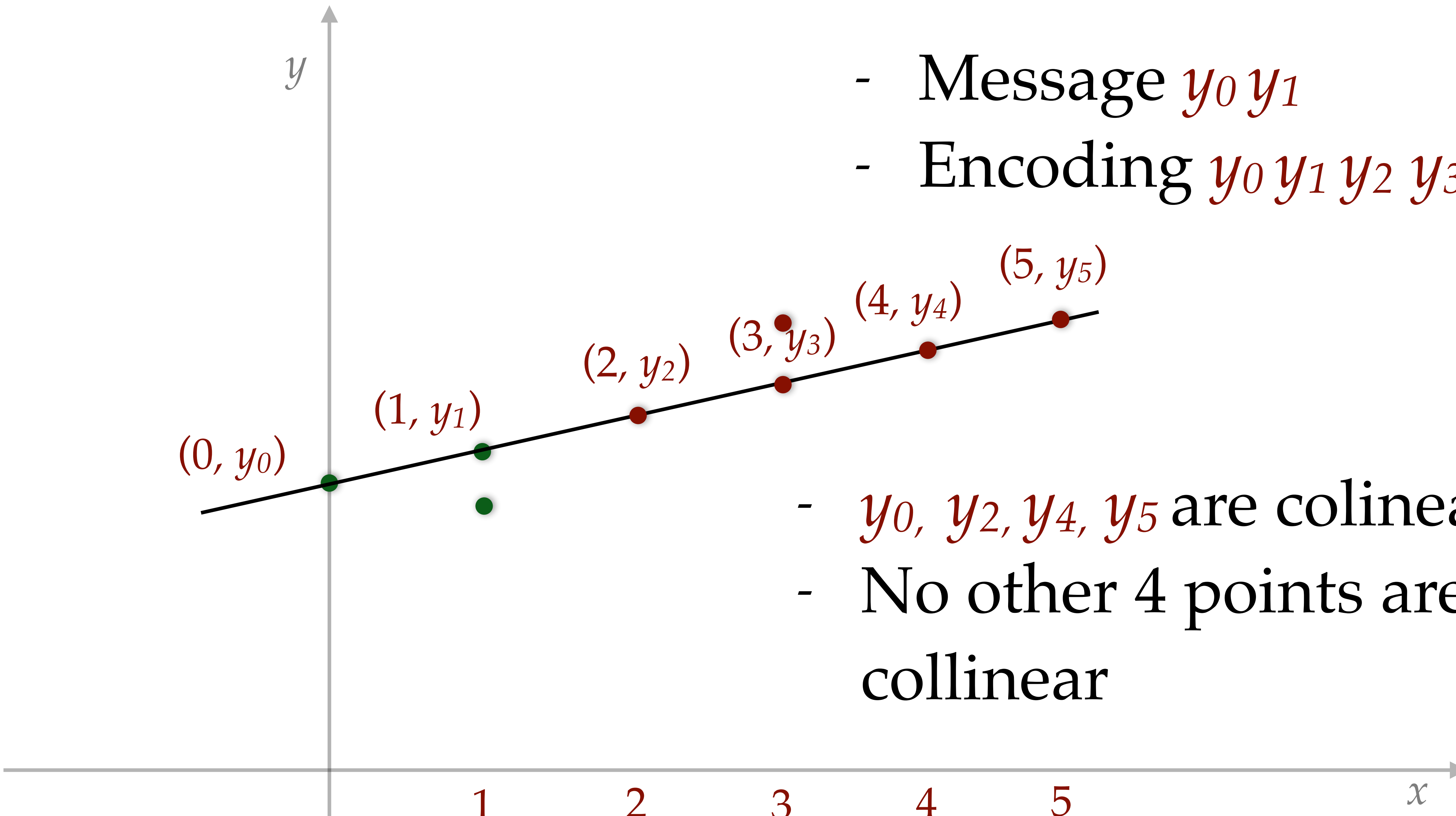
# Error correction




- Message $y_0\, y_1$
- Encoding $y_0\, y_1\, y_2\, y_3\, y_4$

- What if $y_1$ and $y_3$ get modified?
- We know three points are correct, but don't know which three.

# Error correction




- Message $y_0\, y_1$
- Encoding $y_0\, y_1\, y_2\, y_3\, y_4\, y_5$

- $y_0,\ y_2, y_4,\ y_5$ are colinear
- No other 4 points are collinear

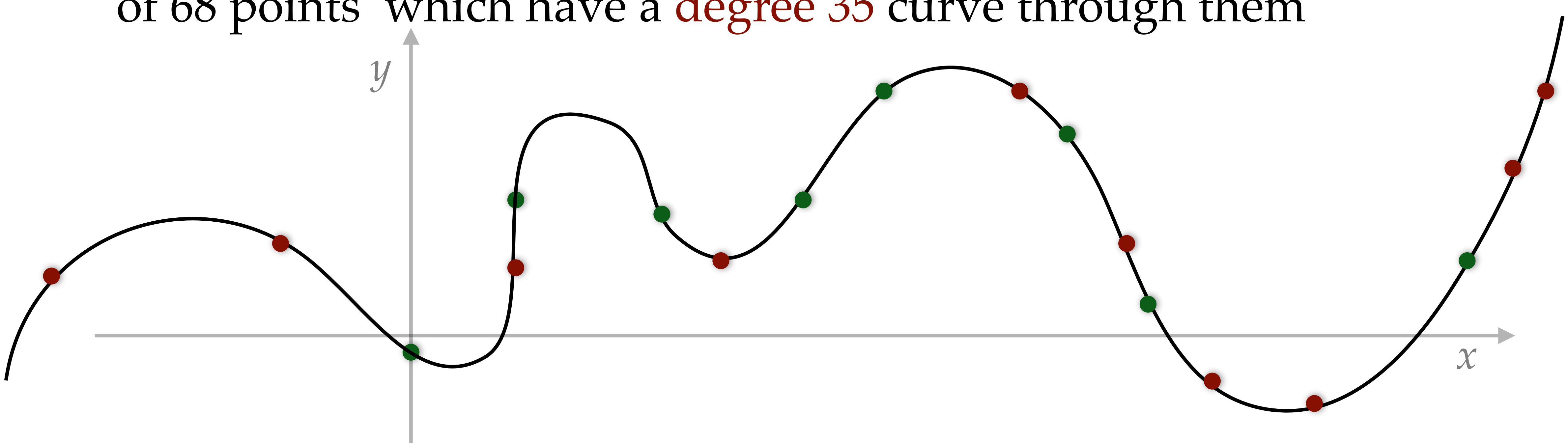
- What if $y_1$ and $y_3$ get modified?
- We know four points are correct, but don't know which four.

# ~~Challenge solved~~

- Message split into 2 blocks.
- converted into 6 blocks.
- Can recover after any 2 blocks are deleted or modified.
- We can handle 33% errors in our data.
- Does it solve what we wanted?
- Not really.
  What if a small portion from every block is modified?

# Towards challenge

- Split the 36 byte message into many blocks, say *36* blocks
- Each block is one byte
- Visualize them as 36 points in the plane.
- Pass a unique degree 35 curve through them
- Take 64 other points on the curve (total 100 points)
- Homework: Even if any 32 points are modified, there is only one set of 68 points which have a degree 35 curve through them



# Coding theory

- This construction is called
  - Reed Solomon (RS) codes [1960]
  - Bose–Chaudhuri–Hocquenghem (BCH) codes [1959/60]
- Need modular arithmetic, so that numbers don't blow up.
- Used in all kinds of communications, data storage
  - wired, wireless, satellite, CD, hard disks, servers
- Other questions people study:
  - Fast error correction / Local error correction

# Coding theory

- QR codes use “dual” of Reed Solomon Code.
- Which is a special case of Bose–Chaudhuri–Hocquenghem (BCH) codes

# Galois Field

- QR codes: each byte (8 pixels) is viewed as a number in {0,1, …, 255}.
- GF(256)
  - Addition: bitwise XOR
  - Multiplication: a multiplication table (256 × 256) such that
    - Identity. For any $a \in \{0,1,\ldots, 255\}$, $a \times 1 = a$
    - Commutative. $a \times b = b \times a$
    - Associative. $a(bc) = (ab)c$
    - Distributive. $a(b+c) = ab + ac$
    - Inverse. For any $a \in \{0,1,\ldots, 255\}$, there exists $a^{-1}$ $s.t.$ $a^{-1} a = 1$

# Error correction bytes

- Suppose the message is 2 bytes: $m_0$, $m_1$
- Suppose we want to add 4 error correction bytes: $m_2$, $m_3$, $m_4$, $m_5$
- Typically, the error correction bytes are a linear combination of the message bytes
- Example 1: parity checksum is used for data transmission
- Example 2: debit card's last digit is some kind of linear combination of remaining digits (*mod 10*)

# Error correction bytes

- Suppose the message is 2 bytes: $m_0, m_1$
- Suppose we want to add 4 error correction bytes: $m_2, m_3, m_4, m_5$
- Error correction bytes defined via 4 linear equations, for example:
  - $m_0 + 2m_1 + m_2 + m_3 - 5m_4 + 7m_5 = 0$
  - $m_0 - 3m_1 - m_2 - m_3 + 2m_4 + 2m_5 = 0$
  - $m_0 + 5m_1 + 3m_2 + m_3 - m_4 - 3m_5 = 0$
  - $2m_0 - m_1 + 4m_2 - 3m_3 - 6m_4 + m_5 = 0$
- Given $m_0, m_1$, such four linear equations will uniquely determine $m_2, m_3, m_4, m_5$
- Encoded message will be $m_0\ m_1\ m_2\ m_3\ m_4\ m_5$

# Deletion

- $m_0 + 2m_1 + m_2 + m_3 - 5m_4 + 7m_5 = 0$
- $m_0 - 3m_1 - m_2 - m_3 + 2m_4 + 2m_5 = 0$
- $m_0 + 5m_1 + 3m_2 + m_3 - m_4 - 3m_5 = 0$
- $2m_0 - m_1 + 4m_2 - 3m_3 - 6m_4 + m_5 = 0$

- Suppose 2 bytes get deleted.
- Say, $m_1, m_3$.
- We can solve these equations to recover $m_1, m_3$.
- 4 equations, 2 unknowns.
- In fact, we can recover four deleted bytes.

# Corruption/Modification

- $m_0 + 2m_1 + m_2 + m_3 - 5m_4 + 7m_5 = 0$
- $m_0 - 3m_1 - m_2 - m_3 + 2m_4 + 2m_5 = 0$
- $m_0 + 5m_1 + 3m_2 + m_3 - m_4 - 3m_5 = 0$
- $2m_0 - m_1 + 4m_2 - 3m_3 - 6m_4 + m_5 = 0$

- Suppose 2 bytes get corrupted.
- Say, $m_1, m_3$.
- Then the right hand side will not be zero.
- That will tell us something is wrong.
- If we know which bytes are correct and which are corrupted, then we can find the correct values by solving this system.

# Corruption / Modification

- $m_0 + 2m_1 + m_2 + m_3 - 5m_4 + 7m_5 = 0$
- $m_0 - 3m_1 - m_2 - m_3 + 2m_4 + 2m_5 = 0$
- $m_0 + 5m_1 + 3m_2 + m_3 - m_4 - 3m_5 = 0$
- $2m_0 - m_1 + 4m_2 - 3m_3 - 6m_4 + m_5 = 0$

- Suppose 2 bytes get corrupted. Say, $m_1$, $m_3$.
- Then the right hand side will not be zero.
- But, we don't know which 2 bytes are corrupted.
- We can try (6 choose 2) possibilities, and see for which possibility gives a solvable system of equation.
- In general, that is exponential. Also, what if multiple possibilities are solvable.

# BCH Codes

- We choose equations in a clever way, so that we are able to recover the corrupted bytes.
- We see the encoded message as coefficients of a polynomial with 4 specified roots.
  - $1\, m_0 + 1\, m_1 + 1\, m_2 + 1\, m_3 + 1\, m_4 + 1\, m_5 = 0$
  - $\alpha^5 m_0 + \alpha^4 m_1 + \alpha^3 m_2 + \alpha^2 m_3 + \alpha\, m_4 + 1 m_5 = 0$
  - $\alpha^{10} m_0 + \alpha^8 m_1 + \alpha^6 m_2 + \alpha^4 m_3 + \alpha^2 m_4 + 1 m_5 = 0$
  - $\alpha^{15} m_0 + \alpha^{12} m_1 + \alpha^9 m_2 + \alpha^6 m_3 + \alpha^3 m_4 + 1 m_5 = 0$

# BCH Codes

- Suppose 2 bytes get corrupted.
- Intended message $m_0\ m_1\ m_2\ m_3\ m_4\ m_5$
- Received message $c_0\ \ c_1\ \ c_2\ \ c_3\ \ c_4\ \ c_5$
  - $1\ c_0 + 1\ c_1 + 1\ c_2 + 1\ c_3 + 1\ c_4 + 1\ c_5 = 2$
  - $\alpha^5 c_0 + \alpha^4 c_1 + \alpha^3 c_2 + \alpha^2 c_3 + \alpha\ c_4 + 1 c_5 = 5$
  - $\alpha^{10} c_0 + \alpha^8 c_1 + \alpha^6 c_2 + \alpha^4 c_3 + \alpha^2 c_4 + 1 c_5 = 6$
  - $\alpha^{15} c_0 + \alpha^{12} c_1 + \alpha^9 c_2 + \alpha^6 c_3 + \alpha^3 c_4 + 1 c_5 = 11$

# BCH Codes

- Intended message $m_0\ m_1\ m_2\ m_3\ m_4\ m_5$
- Received message $c_0\ \ c_1\ \ c_2\ \ c_3\ \ c_4\ \ c_5$
- Let us subtract the two set of equations
  - $1(c_0-m_0)+ 1(c_1-m_1) + 1(c_2-m_2) + 1(c_3-m_3) +1(c_4-m_4)+ 1(c_5-m_5) = 2$
  - $\alpha^5(c_0-m_0)+\alpha^4 (c_1-m_1) +\alpha^3(c_2-m_2)+\alpha^2(c_3-m_3)+ \alpha(c_4-m_4)+ 1(c_5-m_5) = 5$
  - $\alpha^{10}(c_0-m_0)+\alpha^8 (c_1-m_1) +\alpha^6(c_2-m_2)+\alpha^4(c_3-m_3)+ \alpha^2(c_4-m_4)+ 1(c_5-m_5) = 6$
  - $\alpha^{15}(c_0-m_0)+\alpha^{12} (c_1-m_1) +\alpha^9(c_2-m_2)+\alpha^6(c_3-m_3)+ \alpha^3(c_4-m_4)+ 1(c_5-m_5) = 11$
- Define $e_0 = c_0 - m_0$ ,

  $e_1 = c_1 - m_1$ ,

  $e_2 = c_2 - m_2$ ,....

# BCH Codes

- Intended message $m_0\ m_1\ m_2\ m_3\ m_4\ m_5$
- Received message $c_0\ \ c_1\ \ c_2\ \ c_3\ \ c_4\ \ c_5$
- Differences $e_0\ \ e_1\ \ e_2\ \ e_3\ \ e_4\ \ e_5$
  - $1\, e_0 + 1\, e_1 + 1\, e_2 + 1\, e_3 + 1\, e_4 + 1\, e_5 = 2$
  - $\alpha^5 e_0 + \alpha^4 e_1 + \alpha^3 e_2 + \alpha^2 e_3 + \alpha\, e_4 + 1 e_5 = 5$
  - $\alpha^{10} e_0 + \alpha^8 e_1 + \alpha^6 e_2 + \alpha^4 e_3 + \alpha^2 e_4 + 1 e_5 = 6$
  - $\alpha^{15} e_0 + \alpha^{12} e_1 + \alpha^9 e_2 + \alpha^6 e_3 + \alpha^3 e_4 + 1 e_5 = 11$
- Suppose $e_{5-i}$ , $e_{5-j}$ are nonzero. Rest are zero.

# BCH Codes

- Intended message $m_0\ m_1\ m_2\ m_3\ m_4\ m_5$
- Received message $c_0\ \ c_1\ \ c_2\ \ c_3\ \ c_4\ \ c_5$
- Differences $e_0\ \ e_1\ \ e_2\ \ e_3\ \ e_4\ \ e_5$
  - $1\, e_{5\text{-}i} + 1\, e_{5\text{-}j} = 2$
  - $\alpha^i e_{5\text{-}i} + \alpha^j e_{5\text{-}j} = 5$
  - $\alpha^{2i} e_{5\text{-}i} + \alpha^{2j} e_{5\text{-}j} = 6$
  - $\alpha^{3i} e_{5\text{-}i} + \alpha^{3j} e_{5\text{-}j} = 11$
- Suppose $e_{5\text{-}i}$ , $e_{5\text{-}j}$ are nonzero. Rest are zero.

# BCH Codes

- $1\, e_{5\text{-}i} + 1\, e_{5\text{-}j} = 2$
- $\alpha^i e_{5\text{-}i} + \alpha^j e_{5\text{-}j} = 5$
- $\alpha^{2i} e_{5\text{-}i} + \alpha^{2j} e_{5\text{-}j} = 6$
- $\alpha^{3i} e_{5\text{-}i} + \alpha^{3j} e_{5\text{-}j} = 11$

- We do not know $i, j$. We do not know $e_0\ e_1\ e_2\ e_3\ e_4\ e_5$
- We will first find $\alpha^i$ and $\alpha^j$. Then we will solve the system of equations.
- To find $\alpha^i$ and $\alpha^j$ find the coefficients of the polynomial
  - $(y- \alpha^i)(y-\alpha^j) = y^2 + a\, y + b$

# BCH Codes

  - $1\, e_{5-i} + 1\, e_{5-j} = 2$
  - $\alpha^i e_{5-i} + \alpha^j e_{5-j} = 5$
  - $\alpha^{2i} e_{5-i} + \alpha^{2j} e_{5-j} = 6$
  - $\alpha^{3i} e_{5-i} + \alpha^{3j} e_{5-j} = 11$
- $(y - \alpha^i)(y - \alpha^j) = y^2 + a\, y + b$
- To find $a$, $b$, multiply first three equations by $b$, $a$, $1$, respectively and add.
  - $(b + a\, \alpha^i + \alpha^{2i})\, e_{5-i} + (b + a\, \alpha^j + \alpha^{2j})\, e_{5-j} = 2b + 5a + 6$
- Note that $\alpha^i$ and $\alpha^j$ are roots of $y^2 + a\, y + b$.
- Thus above equation becomes
  - $(0)\, e_{5-i} + (0)\, e_{5-j} = 2b + 5a + 6$

# BCH Codes

  - $1\ e_{5-i} + 1\ e_{5-j} = 2$
  - $\alpha^i e_{5-i} + \alpha^j e_{5-j} = 5$
  - $\alpha^{2i} e_{5-i} + \alpha^{2j} e_{5-j} = 6$
  - $\alpha^{3i} e_{5-i} + \alpha^{3j} e_{5-j} = 11$
- $0 = 2b + 5a + 6$
- Similarly, multiply last three equations by $b, a, 1,$ respectively and add.
  - $(b\ \alpha^i + a\ \alpha^{2i} + \alpha^{3i})\ e_{5-i} + (b\ \alpha^j + a\ \alpha^{2j} + \alpha^{3j})\ e_{5-j} = 5b + 6a + 11$
  - $\alpha^i (b + a\ \alpha^i + \alpha^{2i})\ e_{5-i} + \alpha^j (b + a\ \alpha^j + \alpha^{2j})\ e_{5-j} = 5b + 6a + 11$
- Note that $\alpha^i$ and $\alpha^j$ are roots of $y^2 + a\ y + b.$
- Thus above equation becomes
  - $(0)\ e_{5-i} + (0)\ e_{5-j} = 5b + 6a + 11$

# BCH Codes

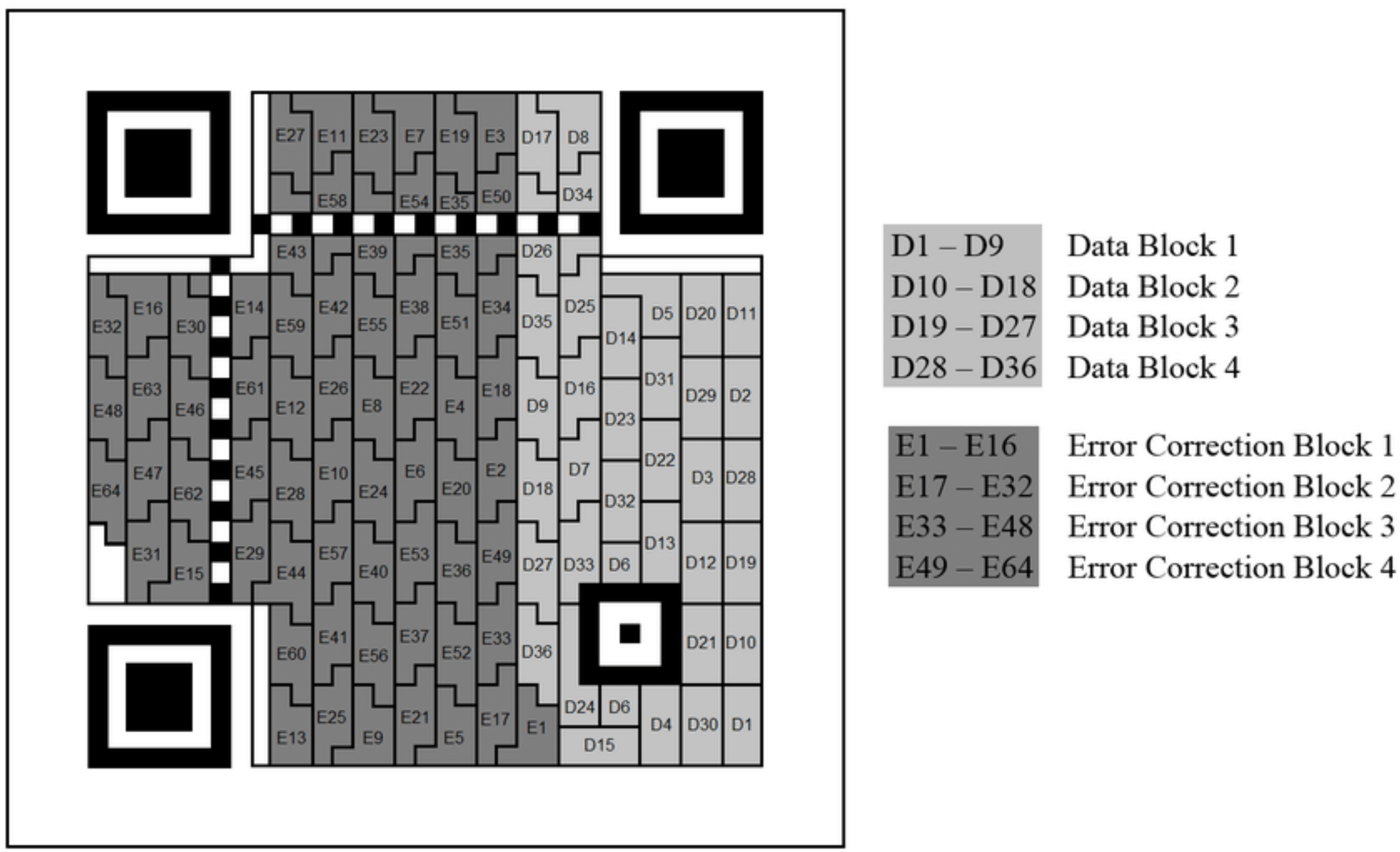
- We get two equations
  - $0 = 2b + 5a + 6$
  - $0 = 5b + 6a + 11$
- Solving these will give us $a, b$.
- Recall that $\alpha^i$ and $\alpha^j$ are roots of $y^2 + a\,y + b$.
- Check which of $\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5$ are roots of $y^2 + a\,y + b$.
- Once we know $\alpha^i$ and $\alpha^j$, we can solve the earlier system of equations to find $e_{5-i}$, $e_{5-j}$. Remaining four $e_k$'s are zero.
- Thus, we know all $e_0 = c_0 - m_0$, $e_1 = c_1 - m_1$, $e_2 = c_2 - m_2$ ,....
- And we can get the actual message $m_0\, m_1\, m_2\, m_3\, m_4\, m_5$

# QR code generation

- For version 4 QR code, with error level H (30%),
- 36 bytes of message and 64 additional bytes generated for error correction.
- The encoded message will have 100 bytes.
- These 100 bytes are the coefficients of a degree 99 polynomial with 64 specified roots $\alpha^0, \alpha^1, \ldots, \alpha^{63}$
- Where the highest 36 coefficients are the given message
- Guarantee: recover after any 32 bytes out of 100 get corrupted.

# Data arrangement

- Version 4 QR code, with error level H (30%)
- From DOI:10.1007/978-3-319-72359-4_42



# Thanks

- If you are interested in theory talks,
- subscribe to mailing list https://www.cse.iitb.ac.in/~theory/seminar.html
- 

# Redundancy in QR codes

- Too much erased, cannot be read