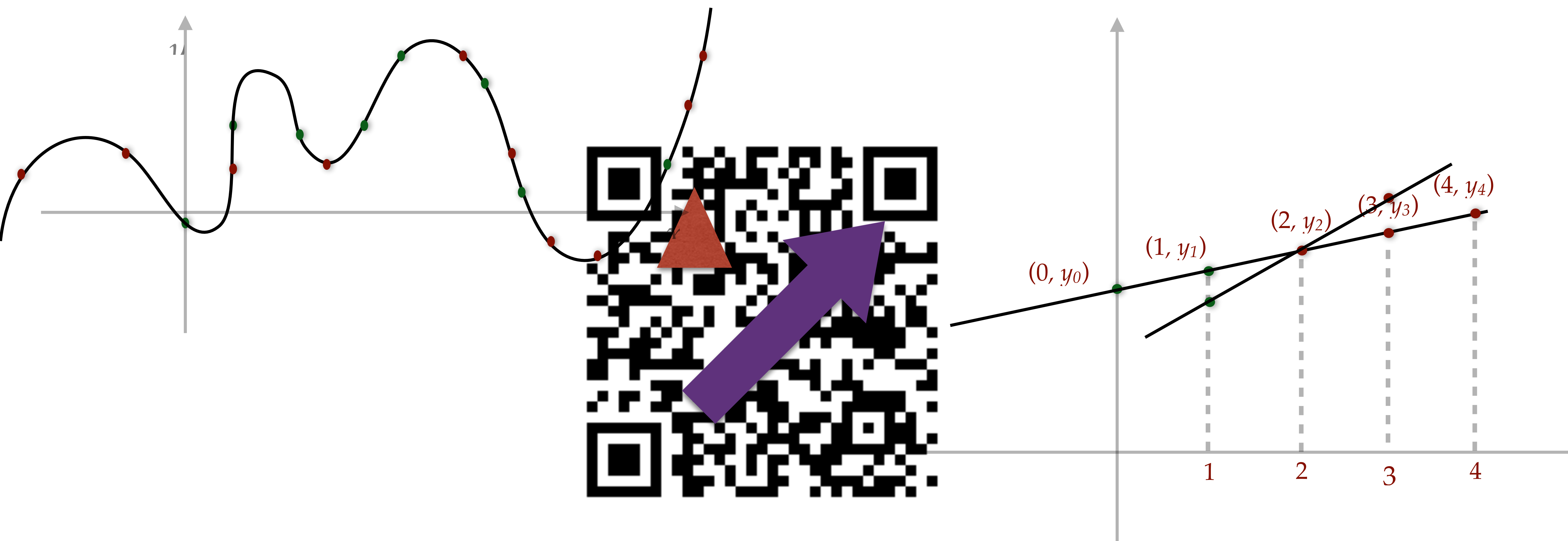
Algebra in Computer Science: QR codes

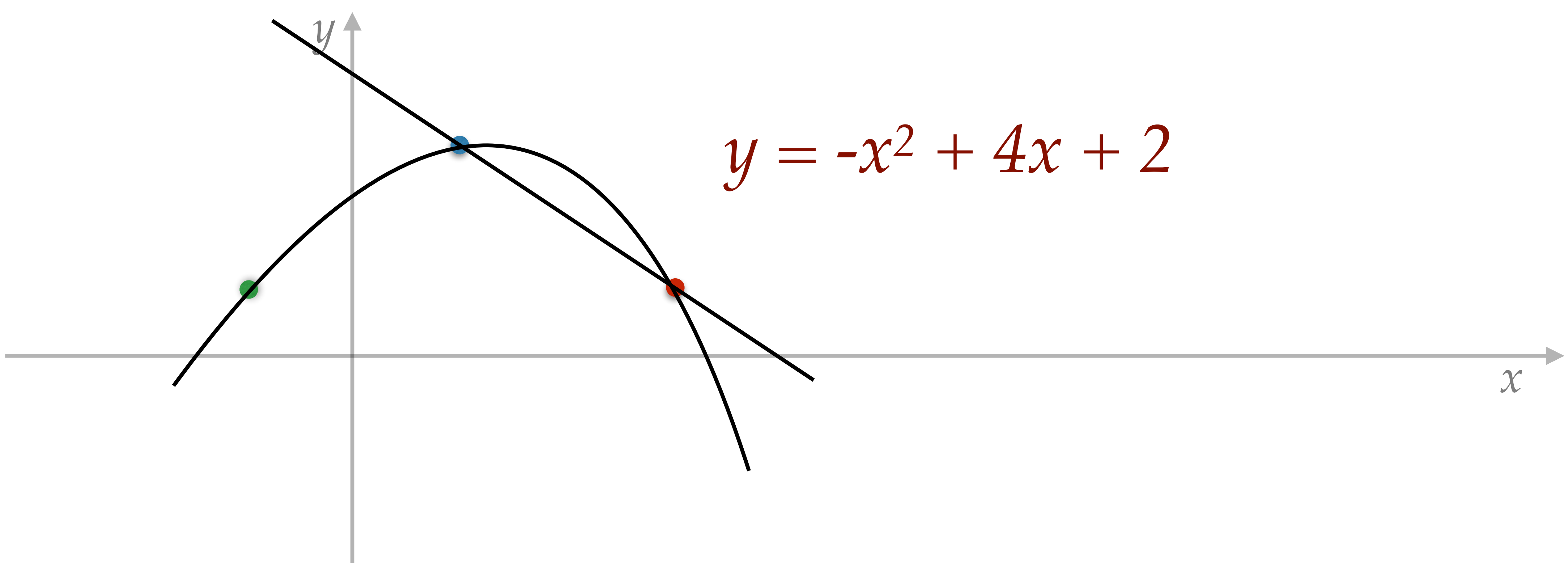


Algebra

- Addition, multiplication, division
- Polynomials, roots, evaluations
- Modular arithmetic
  - $9 + 4 \times 7 \equiv 13 \ (\mathit{mod}\ 24)$
  - $5 \times 4 \equiv -1 \ (\mathit{mod}\ 7)$
- 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.

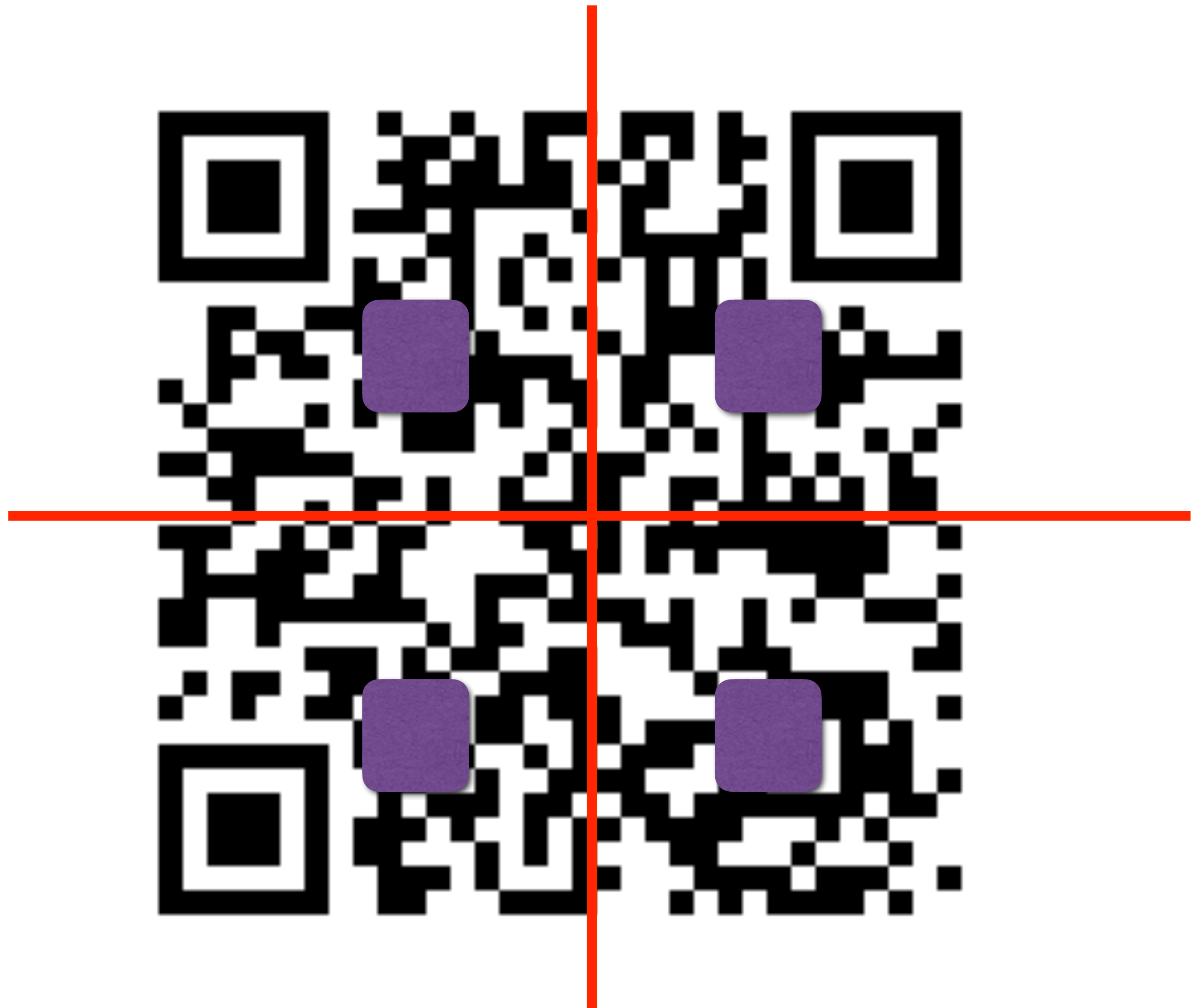


$y = -x^2 + 4x + 2$

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)

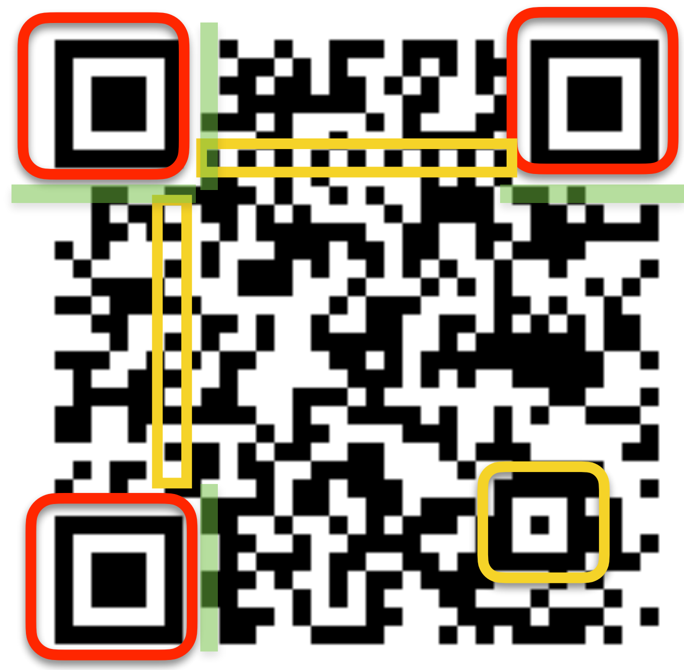
Redundancy in QR codes

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

Redundancy in QR codes

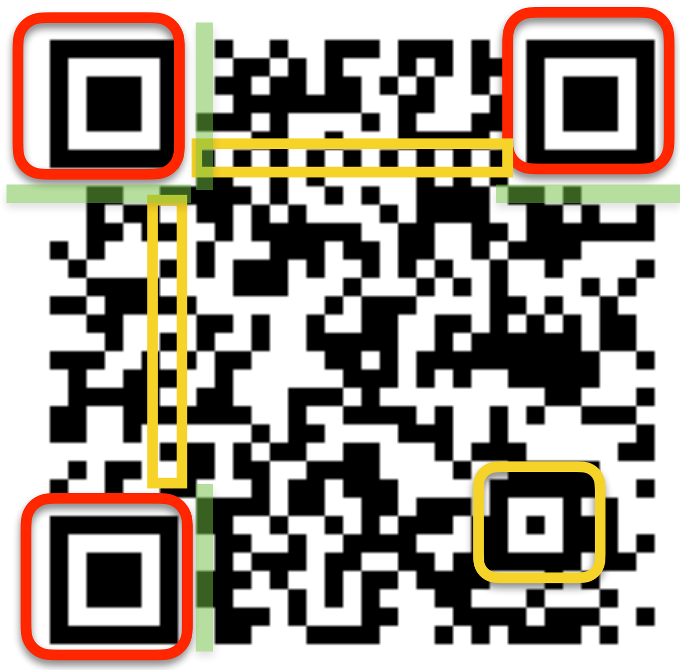
- Too much erased, cannot be read

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

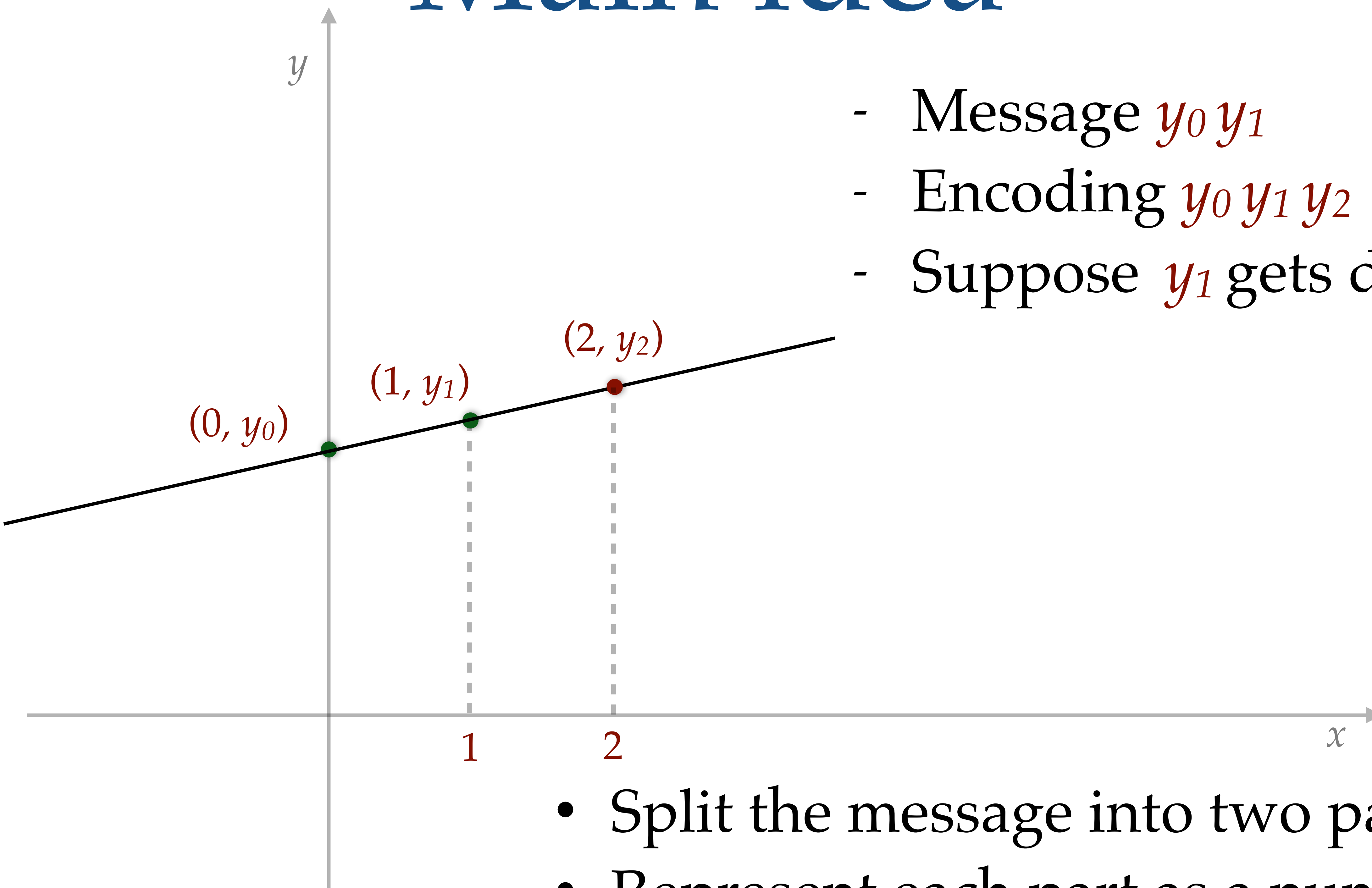


- 100 bytes of data can be stored.
- www.cse.iitb.ac.in/~risc2024/
  29 characters
- Could have stored at most 3 copies
- Level H error correction: guaranteed to work even if any 32 bytes are deleted or modified

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

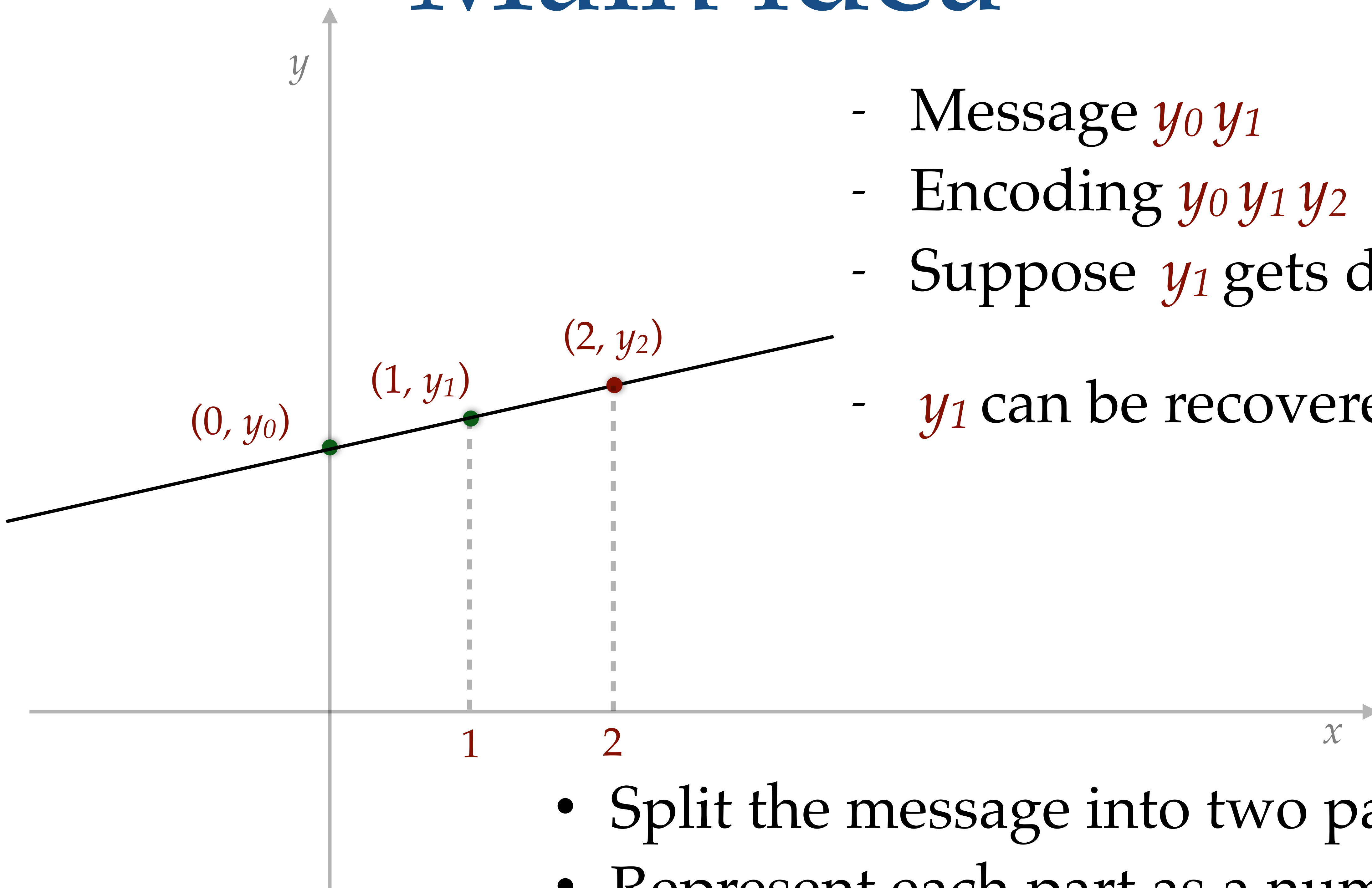
Main idea



- 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

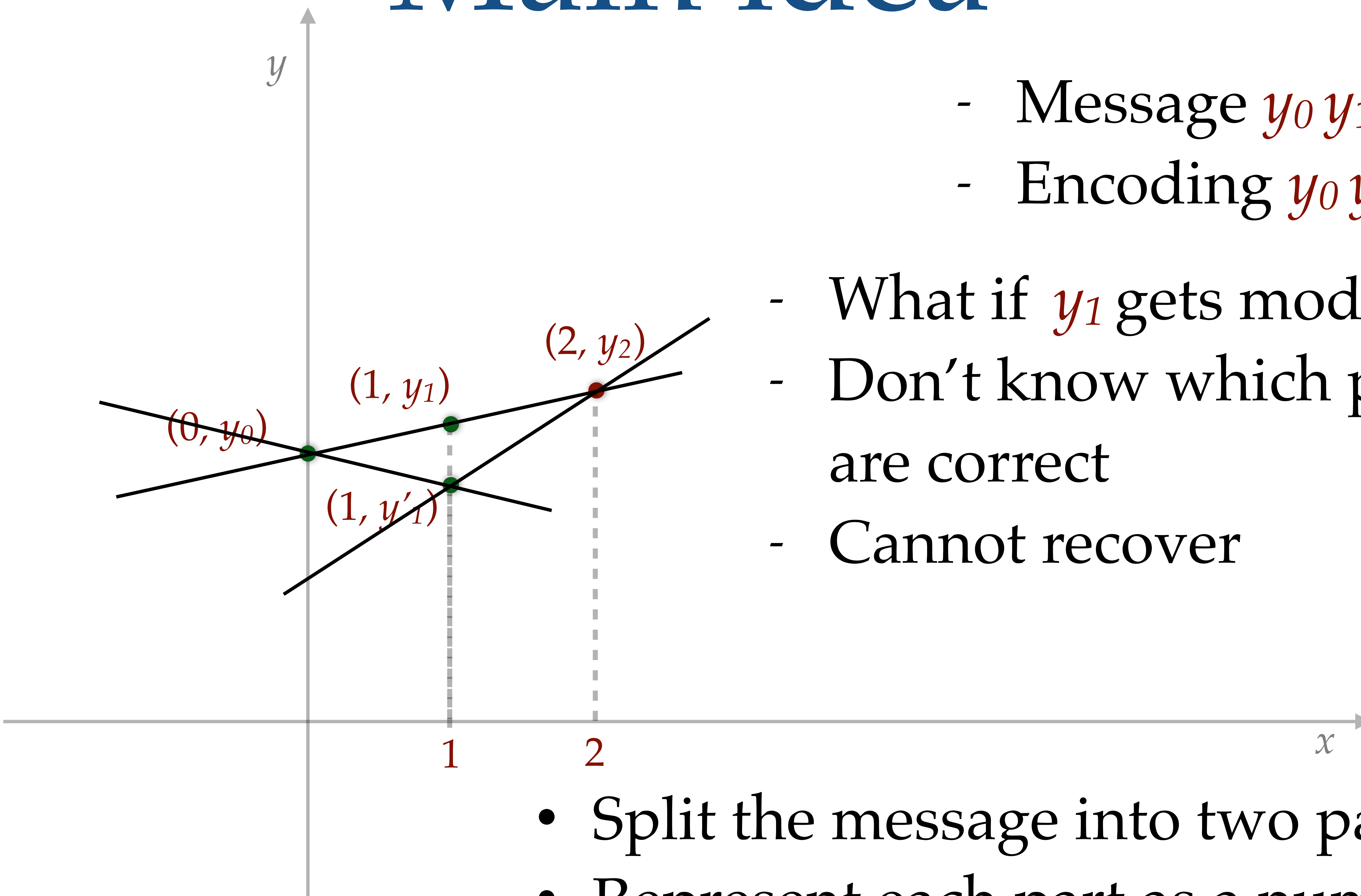


- Message $y_0\, y_1$
- Encoding $y_0\, y_1\, y_2$
- Suppose y_1 gets deleted

- 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

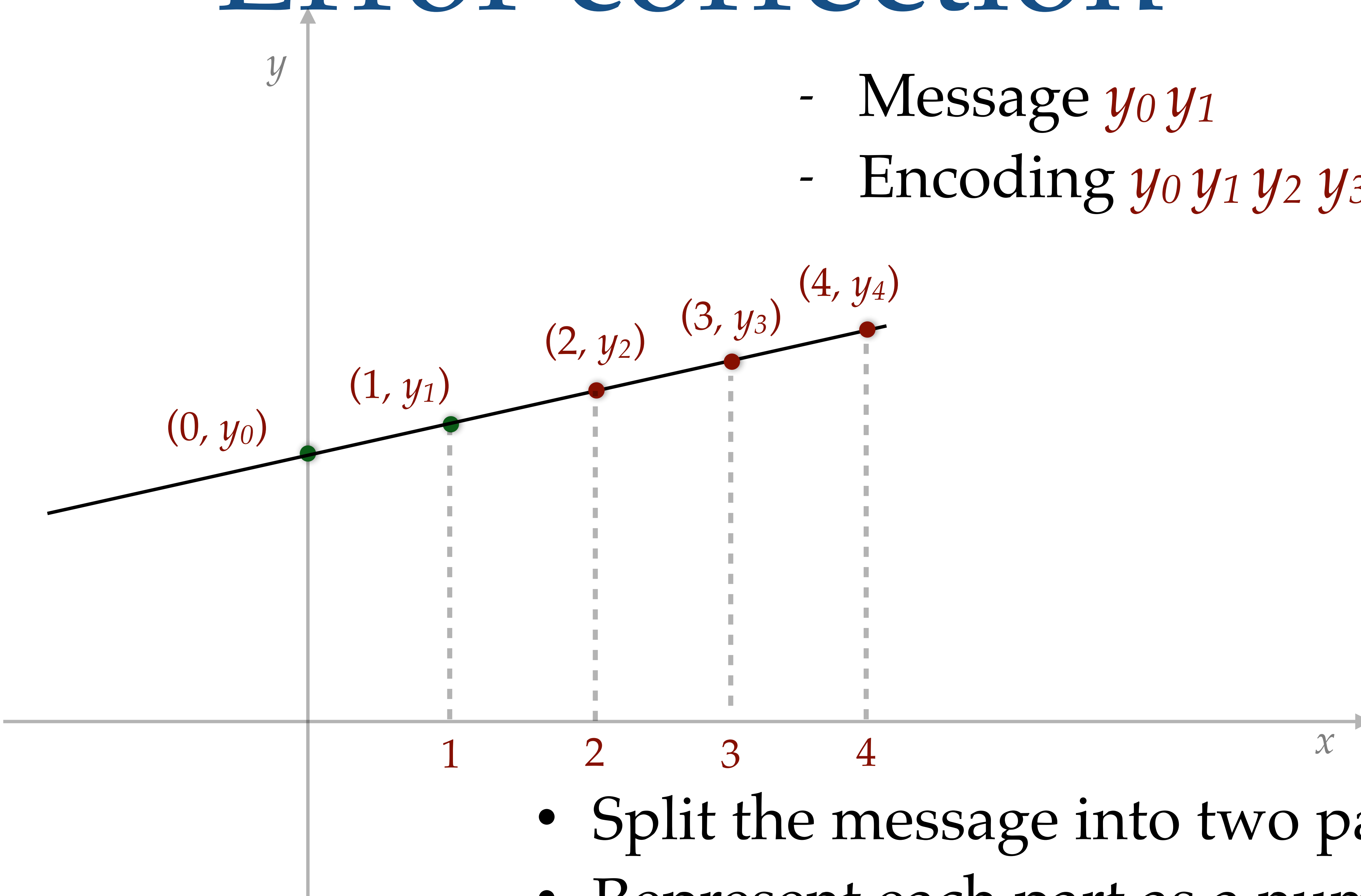


- Message $y_0\,y_1$
- Encoding $y_0\,y_1\,y_2$

- 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

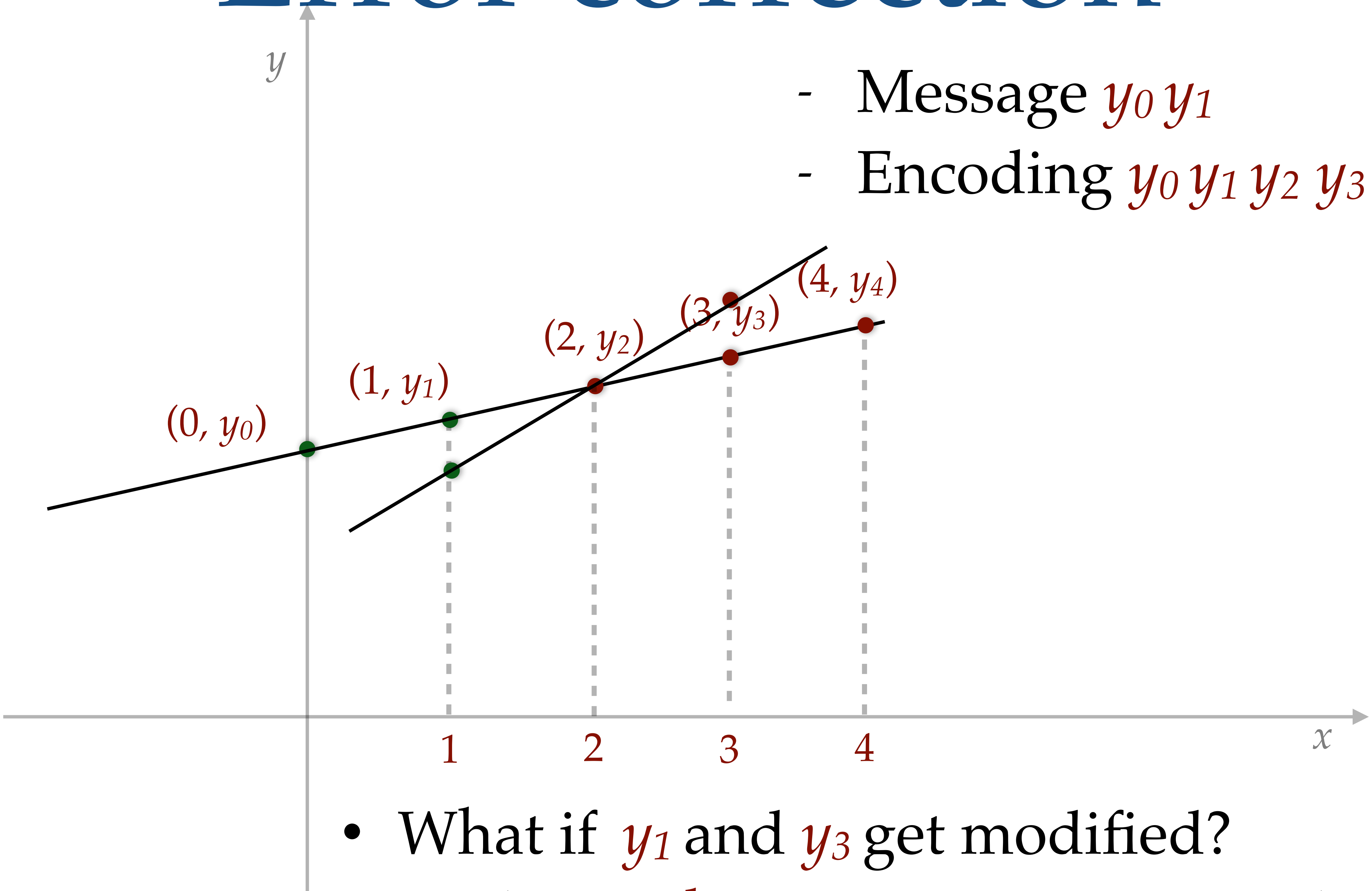
Error correction



- 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

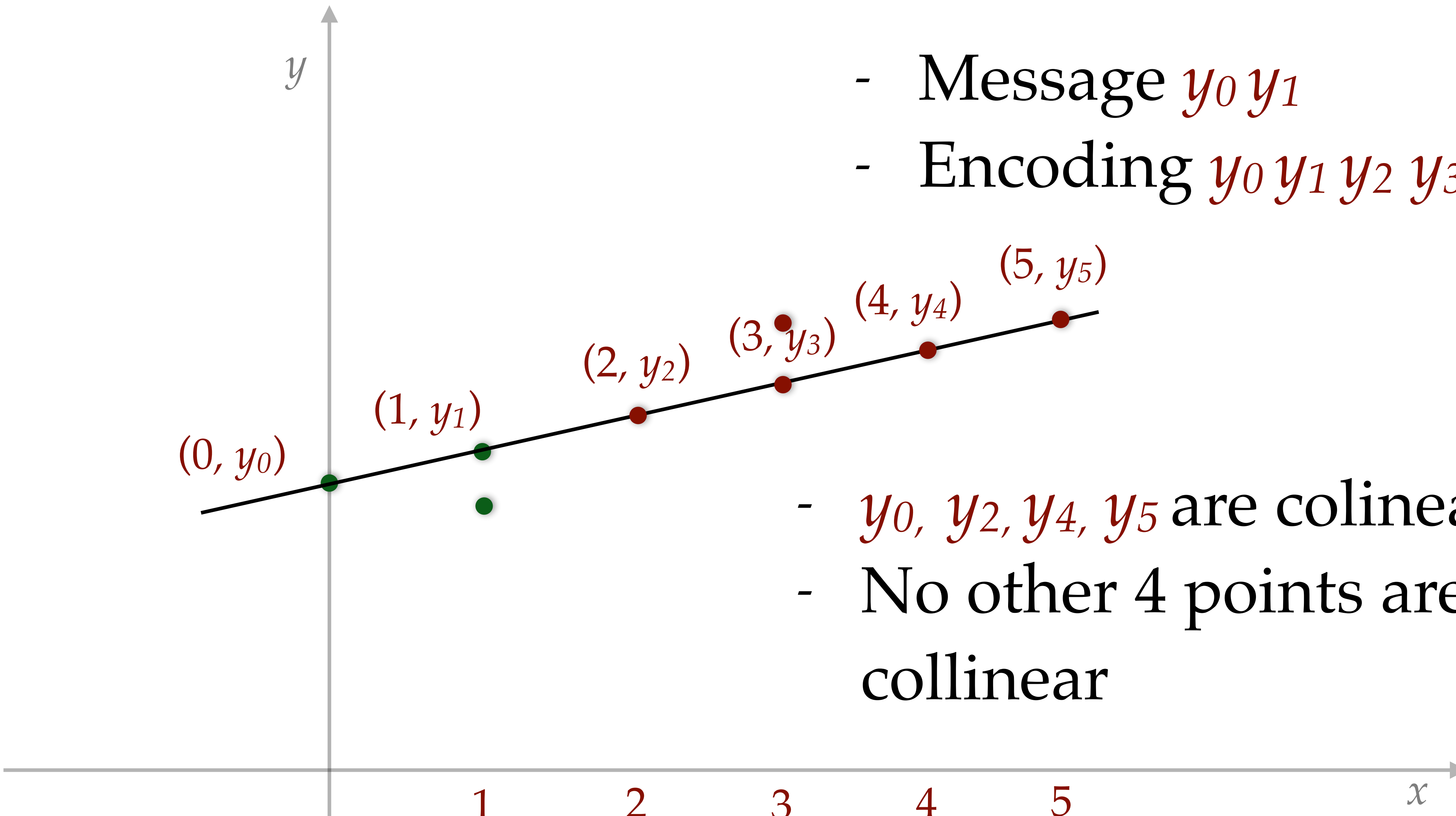


- Message $y_0\, y_1$
- Encoding $y_0\, y_1\, y_2\, y_3\, y_4$

- 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

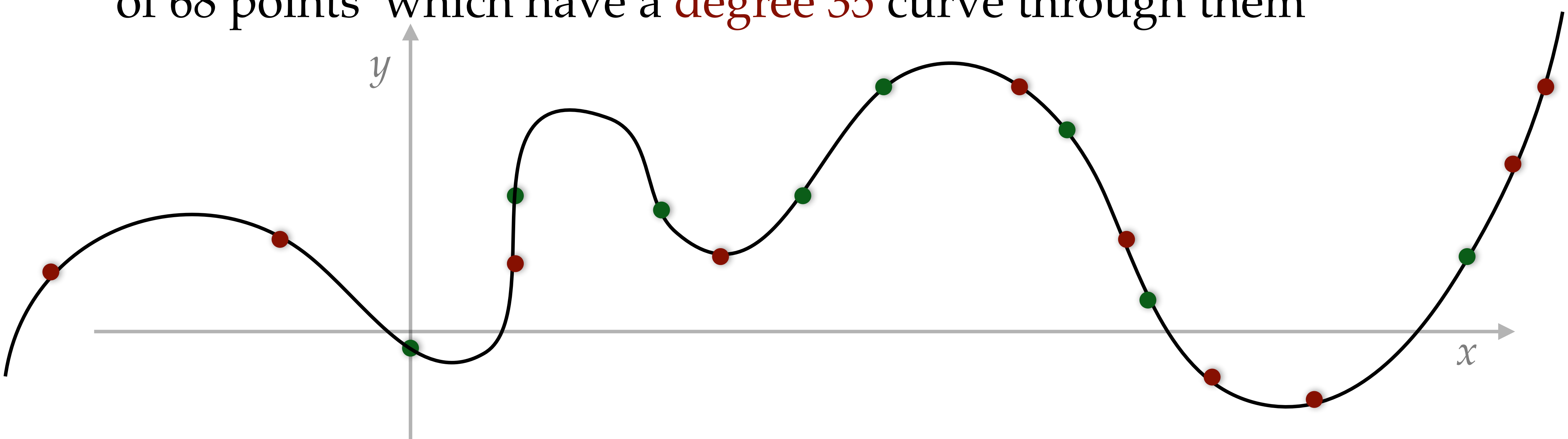
- What if y_1 and y_3 get modified?
- We know four points are correct, but don't know which four.

~~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