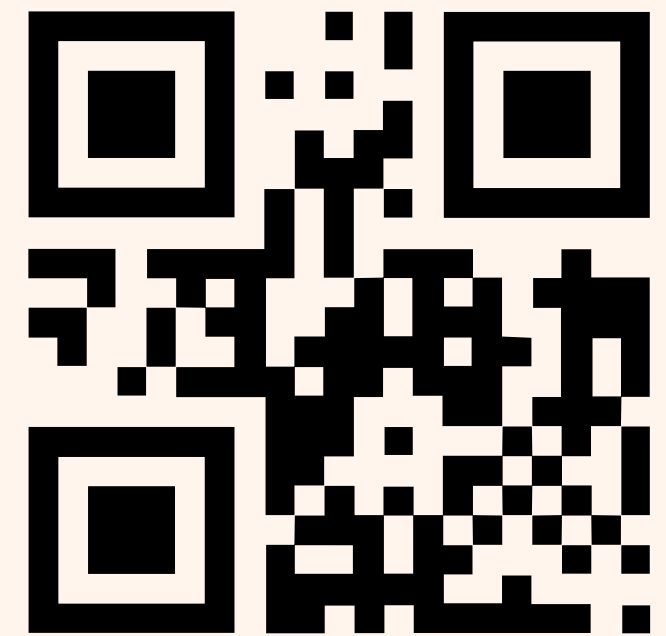


# Encoding Decoding Of QR Code

SC402 : Elements of  
Cryptography

Group Project



# Contribution

## Website Development:

- Preet Mevawalla

## PowerPoint Presentation:

- Prathav Kevadiya,
- Manan Parikh,
- Preet Mevawalla,
- Prayag Patel

## Report Preparation:

- Harsh Makwana,
- Aditya Shah

## System Requirement Specification:

- Manan Parikh,
- Prathav Kevadiya

## Video Creation:

- Prayag Patel

## Group Members:

Preet Mevawalla & 202003025

Manan Parikh & 202003005

Prathav Kevadiya & 202003020

Aditya Shah & 202003045

Prayag Patel & 202003048

Harsh Makwana & 202001264

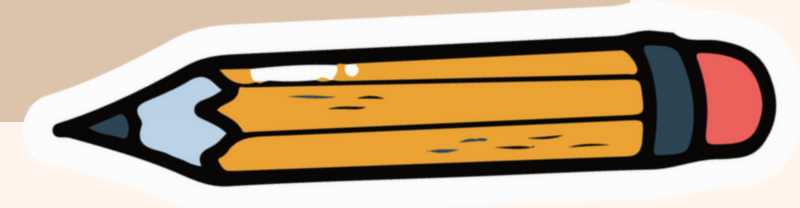
# Overview

- Data Placement in QR code.
- Reed Solomon code.
- Levels of error correction.
- QR code Encoding.
- QR Code Decoding.
- Example of Encoding and Decoding

# Introduction



- Cells, computers, and other gadgets are used to collect, analyze, and store data in diverse formats like text, numbers, photos, and videos in today's data-saturated digital domain.
- Despite this, the handling and representation of data underpin every digital system. This session will delve into concepts surrounding finite fields and data representation.



# Data Representation


- Representing and keeping data through the means of the digital format is widely practised. Digital data is represented using binary numbers, better known as bits, which come in either 0 or 1. These bits come together to form bytes, consisting of eight bits.
- The quantity of bits required to represent data depends on the kind of information. For example, a single ASCII character is represented using one byte, while a 32-bit integer requires the use of four bytes.

# Data Representation

- The idea of endianness is a critical component of data representation. Endianness refers to the arrangement of the bytes in memory. The least significant byte is saved at the lowest memory location in a little-endian system, but the most significant byte is placed there in a big-endian system.
- Data endianness can impact how it is read and written and create incompatibility when transferred across systems.

Example: Take the ASCII character 'A' for example. This character's binary representation is 01000001. This implies that one byte, or eight bits, is used in computers to represent the letter "A."

# Finite Fields:



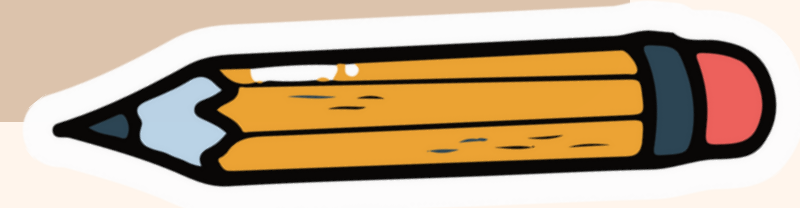
We can use the polynomial  $p(x) = x^8 + x^4 + x^3 + x^2 + 1$  for QC generation to study finite fields with  $2^8$  elements. All the elements that have 1 bit in them are written as polynomials as  $\text{bit}^*(x^i)$  where  $i$  is the position of the bit ; we can thus generate all the details from  $r, r^2, \dots, r^{255}$  in terms of our polynomial.

$$p(r) = r^8 + r^4 + r^3 + r^2 + 1 = 0$$

$$r = -(r^4 + r^3 + r^2 + 1)$$

$$\text{For example } r^9 = rr^8 = r^5 + r^4 + r^3 + r$$

Hence each polynomial will have a degree at most 7.





# QR-Generation

- A QR code can transmit information through a grid of pixels arranged over two dimensions. This encoding follows specific rules that depend on the message's character count and error correction level. The resulting grid size is determined by these factors and may vary between 21x21 to 177x177 pixels.
- Every new edition adds four more rows and columns, leading to larger grids. To guarantee accuracy when reading a QR code, error correction is incorporated into your data using the Reed-Solomon codes algorithm.

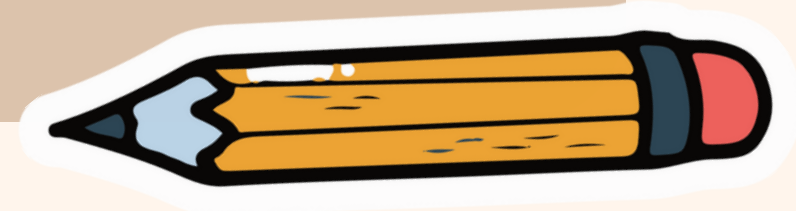


# Levels of error correction:



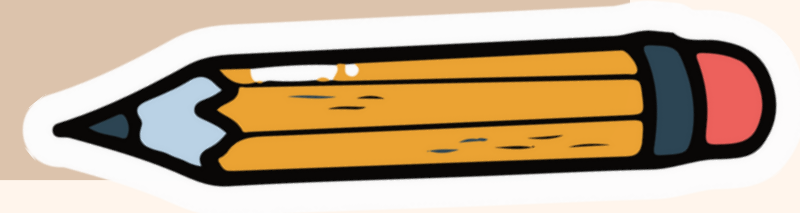
1. Level L: 7% maximum error correction rate
2. Level M: 15% maximum error correction efficiency
3. Level Q: a maximum error correction rate of 25%
4. Up to 30% mistake-correcting capabilities at Level H

The error correction level must first be chosen to produce a QR code. There are four distinct degrees of error correction, each offering a different amount of mistake correction.





# QR-Generation

- Following the error correction level selection, the smallest grid size that can encode the message with the chosen error correction level must be established. In byte mode with level H error correction, the amount of characters that can fit into each QR code version can be found out.
- Following selecting the proper grid size, the message is formatted by converting the characters to their decimal representations in ASCII code, which are then translated into binary and extended to the required length. After formatting, the message is added to the QR code grid.

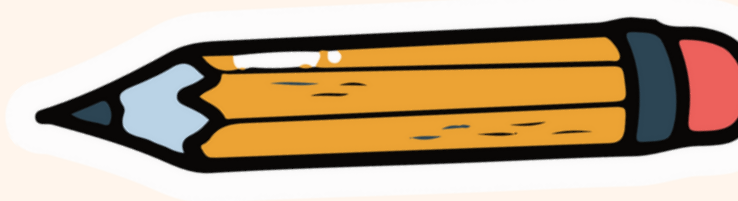


# Reed - Solomon Method

- 
- Reed solomon is basically an error correcting algorithm which is used in QR codes
  - It ensures that QR can be scanned even if it is damaged or blurred
- 

# Reed - Solomon Method

- The Reed Solomon method states that when we send data packages, some data packages might be lost during transmission.
- to retrieve the data we have lost, we send some extra(redundant) data packets along with the original data
- If we use  $k$  redundant data packets, it means we can retrieve lost data of maximum  $k$  data packets



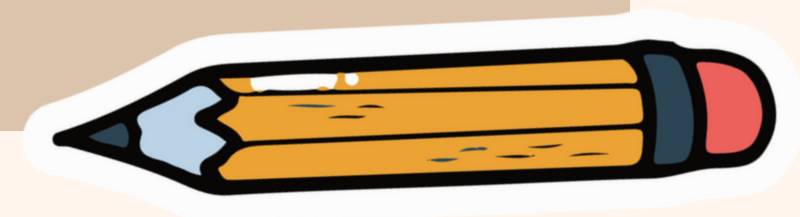
# Brief Idea about the Algorithm

Reed Solomon codes can correct the errors in finite order  $2^m$  that is true for any positive integer  $m$ . It is done by generating primitive polynomial  $p(x)$  of degree  $m$ , using primitive element  $a$ .

Suppose  $t$  is the maximum number of errors that need to correct, that is any positive integer  $t$ , given  $2t < n$ . Here  $n = 2^m - 1 =$  numbers of non-zero elements.

Codeword is constructed with polynomial :

$$m(x) = b_{k-1}x^{k-1} + \dots + b_1x + b_0$$



# Brief Idea about the Algorithm

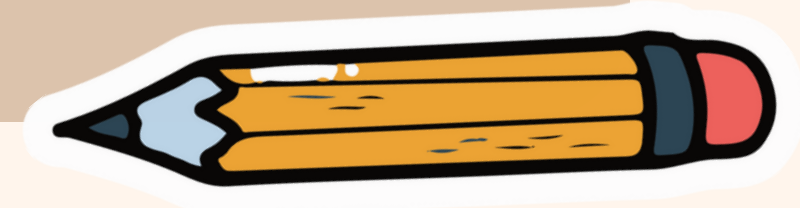
To form the prefix polynomial, we will use the polynomial:

$$g(x) = (x - 1)(x - a) \dots (x - a^{u-1}) \text{ where } u = 2^*t \text{ or } u = 2^*t + 1$$

Here,  $g(x)$  is known as generating a polynomial. Now let  $r(x)$  be the remainder polynomial then we have :


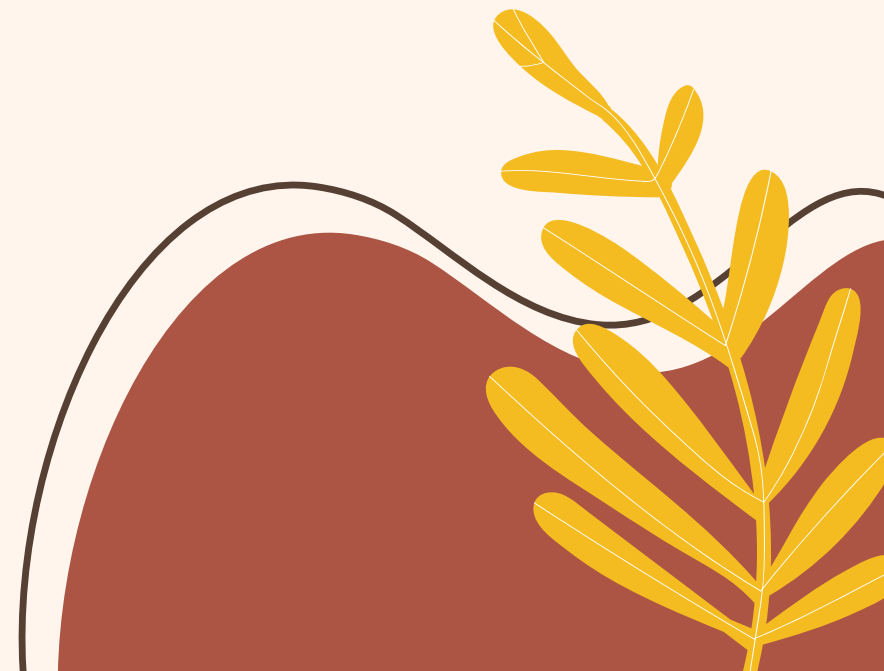
$$x^u m(x) = q(x) * g(x) + r(x)$$

$$\text{With } \text{degree}(r(x)) < \text{degree}(g(x)) \text{ or } s(x) = 0$$





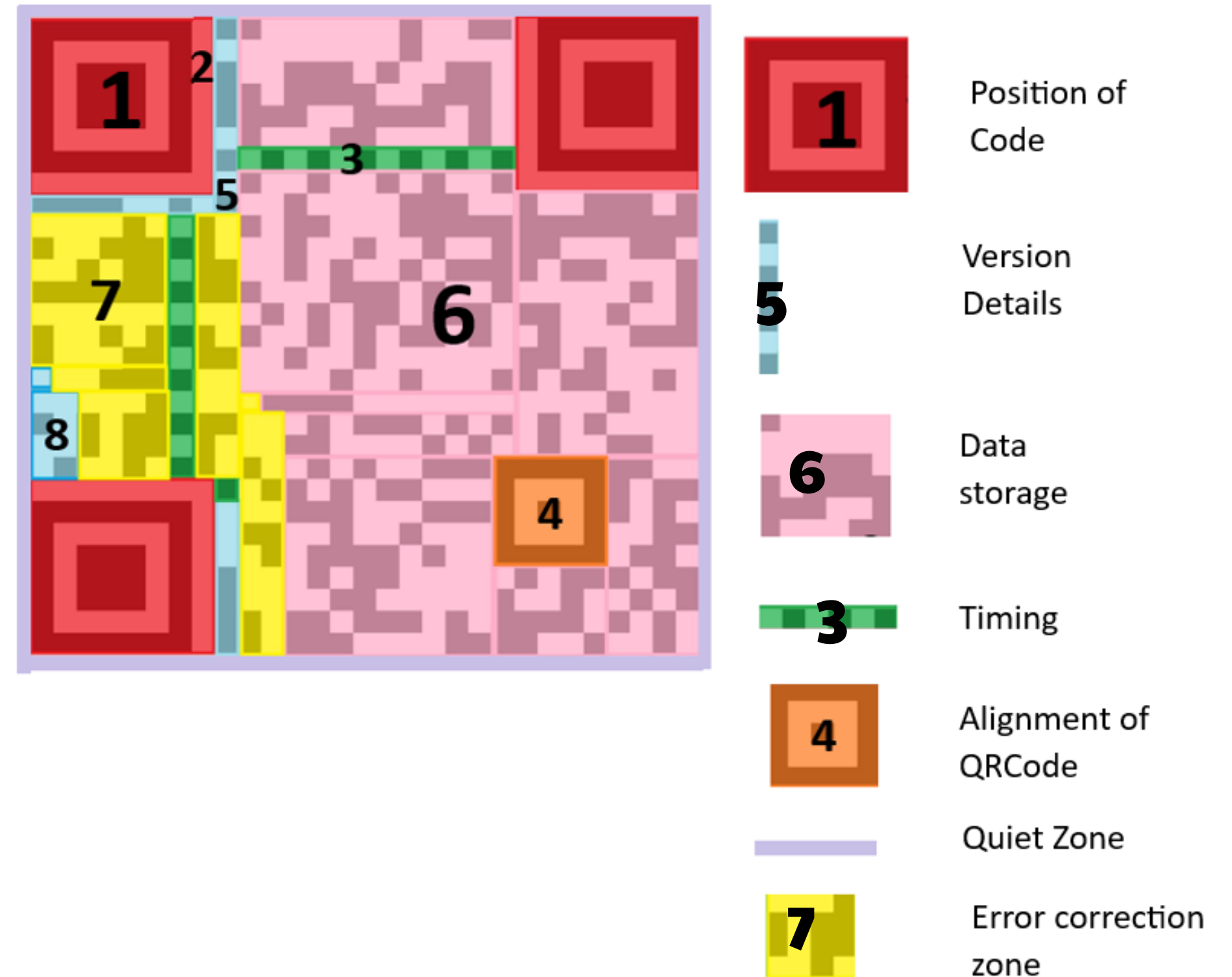
# Encoding and Decoding of QR Code

- How the data is stored in the QR code?
  - How long message can you store in a QR code?
  - How to decode QR Code?
- 
- 



# QR CODE STRUCTURE

- Here is the structure of the QR code version 2, which is divided into several subparts:
- There are a total of different 40 versions of QR codes available. Higher versions are having more pixels than smaller ones.



# QR Code Storage Capacity

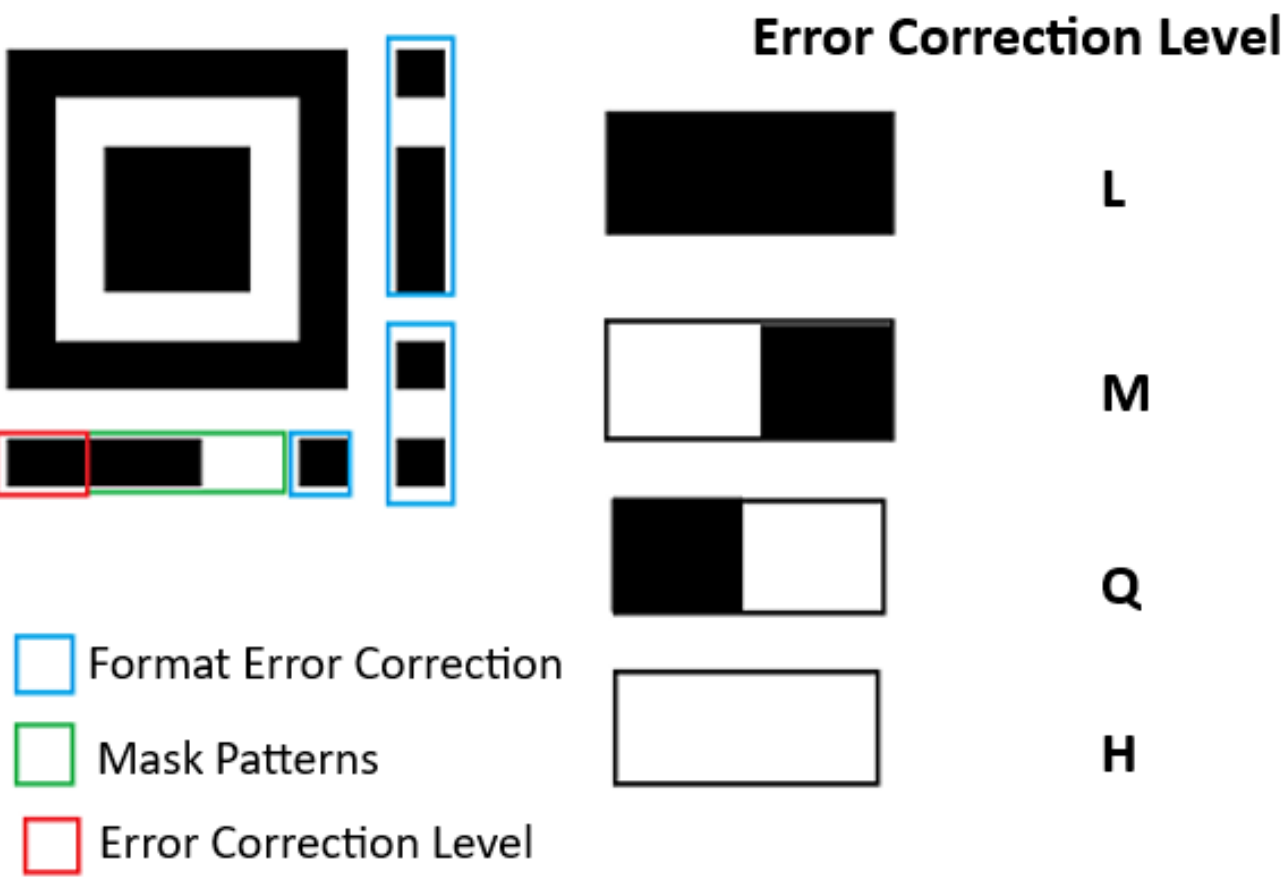
- QR code is finite square space therefore it must have certain limits on data that is encrypted. Following capacity for version 40 QRCode with low error correction capability(7%):

Input types	Max characters	Required bits/char
Number (0 to 9)	7,089	10/3
Alphabets and numbers (0 -9,A-Z(Upper case only), space,\$,%*+,-,.,/,,:)	4,296	11/2

# QR Code Encoding

- QR code primally stores two types of data:

- 1) Error correction and
- 2) Encrypted message.



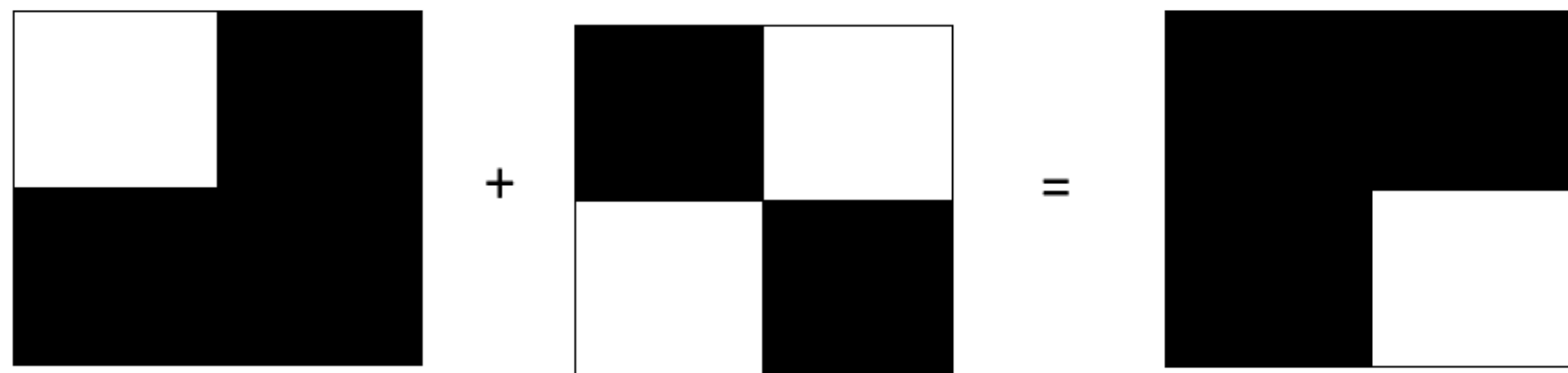
Error Correction level	Recovered Codewords (in percentage)
L (Low)	7%
M (Medium)	15%
Q (Quartile)	25%
H (High)	30%

# Make QRCode Patternless

- As we see that QR code is created based on the bits representation of any characters. For certain messages, we get specific patterns that may help to decode the QR code without a QR code reader.
- The Solution to this problem is to use the Masking technique over QRCode.
- Masking is applied on the data section on the QR code but sometimes it is also used on other symbols.

# Masking of Data

- Masking means converting something to different form original.
- In this case, we will do masking on QRCode pixels that will improve QRCode security directly. Following is the example :



Encoded Data

Version 0  
Masking

Encrypted QR  
Code

**Black(1) + Black(1) = White(0)**

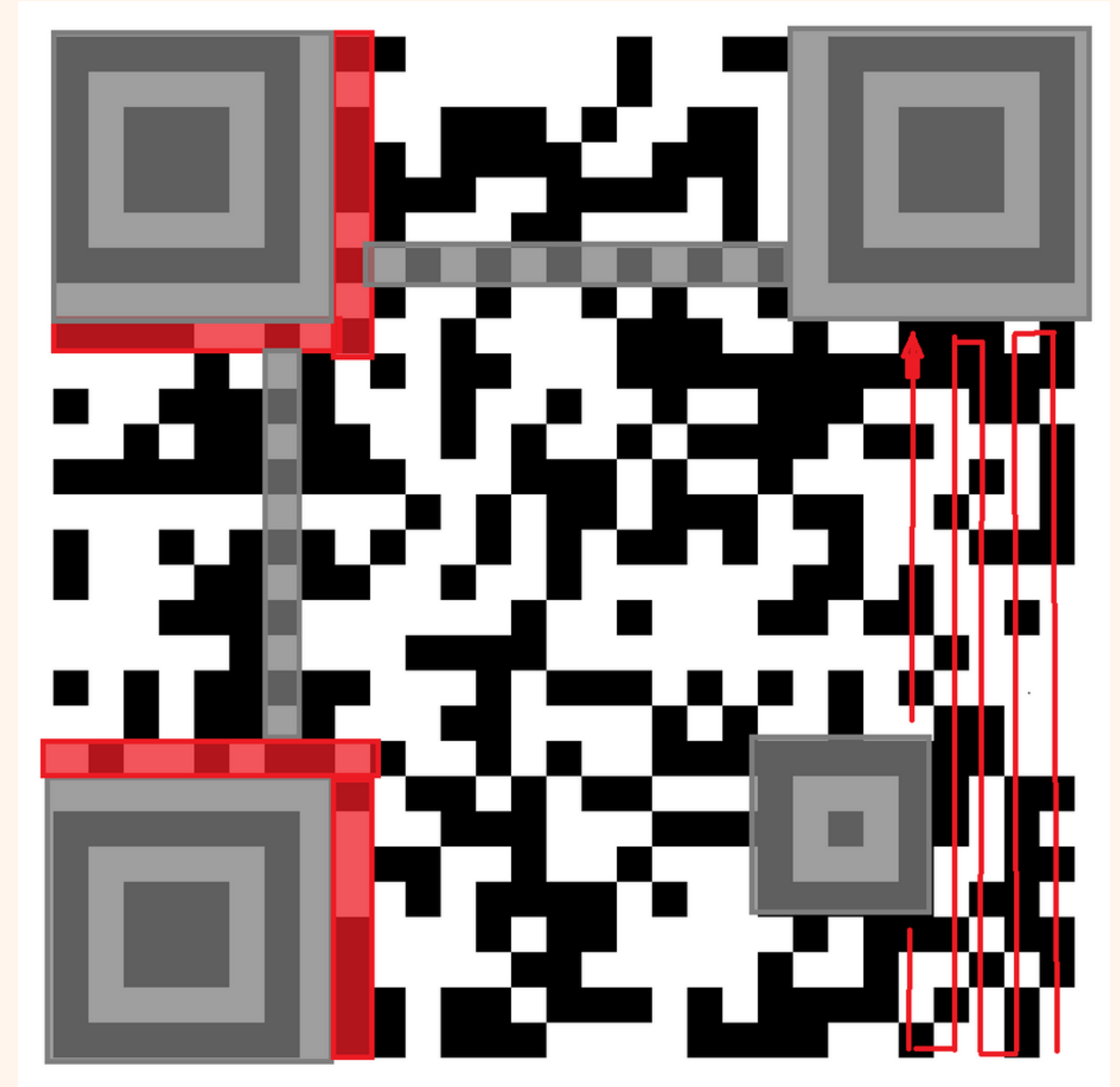
**Black(1) + White(0) = White(0) + Black(1) = Black(1)**

**White(0) + White(0) = White(0)**

**This can be modelled using bit XOR as well.**

# Data Filling in QR code

- Data filling starts from the bottom right corner of the QR code and then follows a zig-zag path on non-shaded pixels, here shaded parts are denoting fixed patterns of the QR code that is independent of the data that you want to encrypt.
- Here, data collectively mean error correction data and data to be stored. For non-shaded pixels are divided into two groups of type E (error) and D(data).



# Decoding of the Data section

- For the given QR code, firstly observe the right bottom pattern of the QR code
- Length field: It provides details of how many characters are encoded.
- Each length field uses a different number of bits based on the encoding type and version of the QR code.

Indicator	Meaning
0001	Numeric encoding
0010	Alphanumeric encoding
0100	Byte Encoding
0000	End of Message

Number of bits in length field for numeric and alphanumeric types

Encoding	ver 1- 9	ver 10-26	ver 27-30
Numeric	10	12	14
Alphanumeric	9	11	13
Byte	8	16	16



# Indicator and Length Field

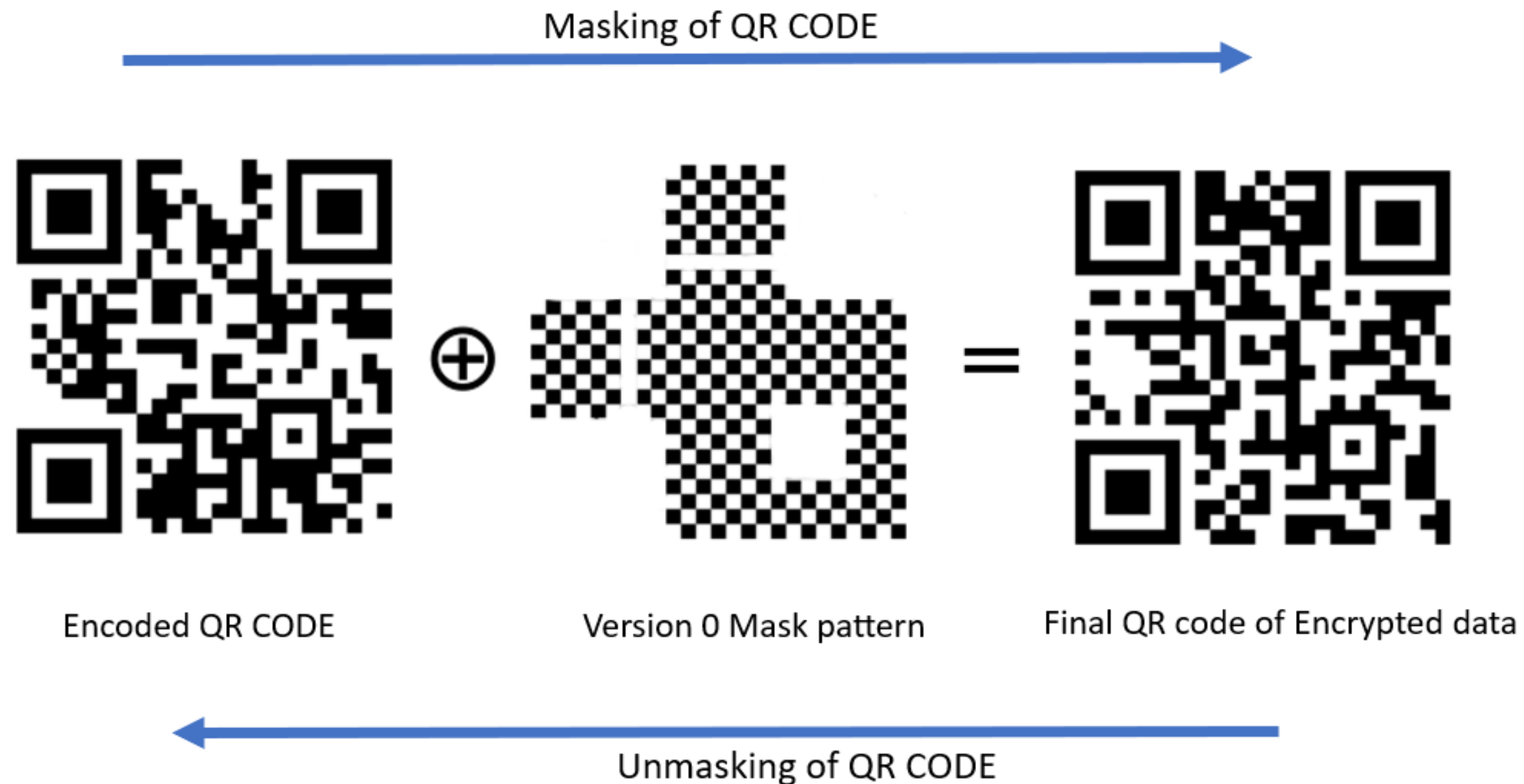


**Length field:** it is a 8 bits long message and is used to find encoded message length.

**Indicator:** it is a 4 bits long message and is used to find encoded message type.

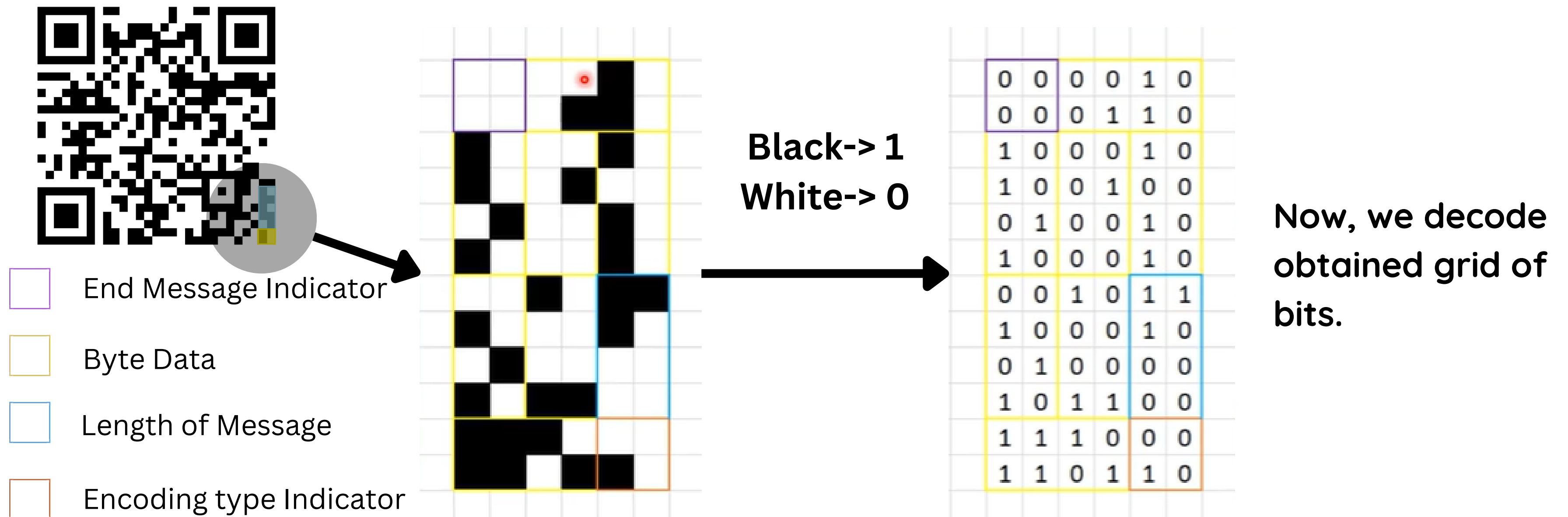
# Masking and Unmasking of QR Code

- Version 0 mask pattern is the simplest mask pattern, there are more complex patterns are also available.



# Decoding Unmasked QR code

- Suppose we have encrypted the message “QR Code”. Here, we don’t care about the fixed section(other than data sections).

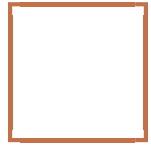


# Final Decoding of Bits message

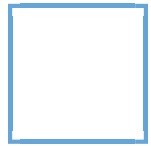
- Traverse zig-zag from the bottom right.

0	0	0	0	1	0
0	0	0	1	1	0
1	0	0	0	1	0
1	0	0	1	0	0
0	1	0	0	1	0
1	0	0	0	1	0
0	0	1	0	1	1
1	0	0	0	1	0
0	1	0	0	0	0
1	0	1	1	0	0
1	1	1	0	0	0
1	1	0	1	1	0

Start from Here



0100 => Byte encoding

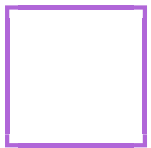


0000 0111 => 7 length field

Now, the main message to decode:  
(.) -> ASCII values



0101 0001 => Q(81)  
0101 0010 => R(82)  
0001 0000 => (32)  
0100 0011 => C(67)  
0110 1111 => o(111)  
0110 0100 => d(100)  
0110 0101 => e(101)



Message Over through:  
0000 => End of Message.

Hence, Decoded message is : "QR Code"

# References

- Sun, Jiabin, and Nan Zhang. "The Mobile payment based on public-key security technology." Journal of Physics: Conference Series. Vol. 1187. No. 5. IOP Publishing, 2019.
- <https://atcm.mathandtech.org/EP2021/invited/21891.pdf>

**Thank You!**