# Hadamard Encoder with GUI and QR Code Generation

**(Coding Theory and Cryptography)**

Submitted by:

Aditya (2203301) and Prithvi Raj(2203316)

Indian Institute of Technology Goa

Under the guidance of:

Prof. Rahul CS

April 10, 2025

# Contents

# Introduction

Hadamard codes are binary error-correcting codes known for their large minimum Hamming distance, making them ideal for robust transmission over noisy channels. This project implements a Hadamard encoder with support for text and image inputs and outputs encoded data in the form of scannable QR codes.

The encoder uses the extended Hadamard code of the form $[2^r, r+1, 2^{r-1}]_2$, and can work both with small parameters (e.g., $[16, 5, 8]_2$) suitable for hardware/breadboard implementation or larger parameters for software demonstrations.

# Objective

To build a flexible Hadamard encoder capable of encoding binary messages, text, and images into extended Hadamard codewords, and visualizing the output using QR codes. The system should support GUI-based interaction and allow user-specified parameters such as $r$ for encoding size.

# Tools and Technologies Used

- **Language:** Python 3
- **Matrix Operations:** NumPy
- **Image Processing:** OpenCV (cv2)
- **QR Code Generation:** qrcode (Python Library)
- **GUI Support:** Web based
- **File Handling:** os, uuid
- **Image Formats:** PNG, JPEG (via Pillow)
- **Pandas:** For handling data and generating CSV/Excel outputs.
- **JSON:** For handling data transfer and encoding formats.
- **ReportLab:** For generating PDF downloads.

# System Design

## Hadamard Encoder Class

The class `HadamardEncoder(K)` constructs an encoder that maps messages from 0 to $2^K - 1$ into codewords of length $2^{K-1}$ using a parity-based method. Each message bit is encoded into a binary vector of $+1/-1$, then mapped to 1/0 for digital storage.

## Encoding Functions

- `encode_text_string`: Encodes text using ASCII values.
- `encode_binary_string`: Encodes binary strings bit-by-bit.
- `encode_image`: Encodes images by treating each pixel as a message.

## QR Code Generation

- QR code size is dynamically adjusted to fit the encoded binary string.
- `generate_qr`: Saves each QR code as a PNG image.
- `process_data_into_qrcodes`: Splits long data into multiple QR codes.

# Experimental Setup

### Case A: Text Input

- Input: Any Text
- $K = 8$ (allows ASCII range 0–255)
- Result: Sequence of Hadamard-encoded binary chunks
- Output: Multiple QR codes for the encoded message

### Case B: Image Input

- Input: Image (e.g., 100×100)
- Each pixel is treated as a message and encoded
- Output: Encoded NumPy array and QR visualizations

# Results

- Successful encoding of text and images using Hadamard codes
- Output fits within dynamically-sized QR codes
- Modular code design supports future GUI expansion
- Encoder parameterizable by $K$ for various use cases

# Applications

- Error-resistant data transmission
- Secure QR-based file sharing
- Embedded systems encoding
- Education and demonstration of coding theory

# Conclusion

This project demonstrates the construction of a Hadamard encoder capable of transforming text and image data into highly redundant, QR-encoded output. By supporting modular input and QR output, this encoder offers both theoretical learning value and practical robustness, with clear room for feature expansion.

# References

# Bibliography

[1] *CS425 Notes*, Instructor: Dr. Rahul CS, Indian Institute of Technology Goa.

[2] *https://youtube.com/playlist?list=PLDu0JgProGz7qcN66T6-pe5627k0oUSH9si=Ih9m7-h6pEh01Zqn*