|  |
| --- |
| University of Missouri – St. Louis |
| **Beyond Fobs: A Secure QR Code-Based Access System** |
|  |

|  |
| --- |
| Patrick Jennewein  5-1-2025 |

Table of Contents

[Abstract 2](#_Toc195431049)

[Introduction 2](#_Toc195431050)

[The Structure of QR Codes 2](#_Toc195431051)

[Data Masking and Reversing 4](#_Toc195431052)

[Comparing Traditional Barcodes to QR Codes 6](#_Toc195431053)

[The Rising Popularity of QR Codes 6](#_Toc195431054)

[QR Codes for Access Control 7](#_Toc195431055)

[Problem Statement 7](#_Toc195431056)

[Objective 8](#_Toc195431057)

[Literature Review 8](#_Toc195431058)

[Adaptive Thresholding and Deblurring Techniques in QR-Based Authentication 8](#_Toc195431059)

[Security Vulnerabilities in QR-Based Authentication 12](#_Toc195431060)

[Methodology and Implementation 12](#_Toc195431061)

[System Overview 12](#_Toc195431062)

[Technologies Used 12](#_Toc195431063)

[Enhanced QR Code Detection 12](#_Toc195431064)

[Time-Based QR Code Refresh System 12](#_Toc195431065)

[Challenges and Solutions 12](#_Toc195431066)

[Experimental Results and Evaluation 13](#_Toc195431067)

[Test Setup 13](#_Toc195431068)

[Performance Analysis 13](#_Toc195431069)

[Security Analysis 13](#_Toc195431070)

[User Experience 13](#_Toc195431071)

[Discussion 13](#_Toc195431072)

[Conclusion 13](#_Toc195431073)

[Works Cited 14](#_Toc195431074)

## Abstract

In this project, I propose a secure QR code authentication system that enhances traditional access methods.

While QR codes are widely used for authentication, conventional systems simply attempt to match a displayed QR code to a database. In the best case, these standard QR code detection methods are unreliable in challenging environments such as in low-light conditions with user-induced motion blur. In the worst case, standard QR code detection methods are subject to cybersecurity threats.

To address these issues, this project integrates two key components: (1) Enhanced QR code detection using adaptive thresholding, deblurring techniques, and homography transformations to improve robustness in real-world conditions; and (2) A time-based system that ensures QR codes dynamically refresh to prevent reuse or replay.

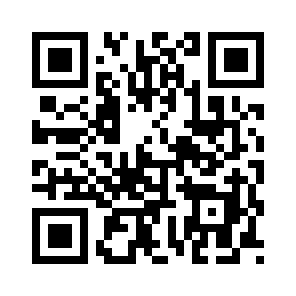
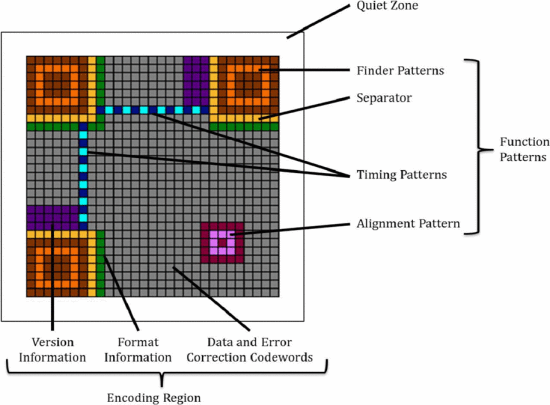
This approach provides a secure and user-friendly authentication solution that can be deployed in various environments, offering improved protection while maintaining ease of use.

## Introduction

“Quick Response” codes, otherwise known as “QR” codes, give users a two-dimensional matrix of white and black pixels (*Figure*) [1]. These codes – unlike more standardized one-dimensional barcodes – are capable of storing a larger amount of data, offer a much faster recognition, and can be read omni-directionally.

### The Structure of QR Codes

#### QR Code Patterns

QR code functionality is simple. These codes contain four types of patterns, giving it structural integrity and enabling accurate data retrieval [1]. The *finder pattern* encompasses three of the four corners of the QR code and correctly orients the decoder with its perfect ratio of a 7 x 7 outer dark square, 5 x 5 inner light square, and 3 x 3 inner dark square. *Separators* encompass the white space around each of the three finder patterns. The *timing patterns* are alternating black and white pixels that form horizontal and vertical lines in 6th row and 6th column, respectively, to ensure proper alignment. Finally, *alignment patterns* consist of a 5 x 5 dark module surrounding a 3 x 3 light module, and 1-pixel dark module. See *Figure right* for a visual description [1]. These patterns distinguish QR codes from other encoding methods.

#### The Basics of Reading QR Code Data

To understand how the error correction works one must understand how the QR is read from the camera:

1. The QR code image is scanned by the camera.
2. Each pixel[[1]](#footnote-1) of the QR code is interpreted as a bit. A black pixel is interpreted as a 1 and a white pixel is interpreted as a 0.
3. Bits are read following a zigzag pattern and grouped into 8-bit chunks – or bytes. For example, if the camera scans in an alternating set of black and white bits (10101010), this data is converted as the byte 170.
4. These bytes are then interpreted as a message. If some data is corrupted or missing, *Reed-Solomon* algorithms use the mathematical relationship between bytes to detect and correct the errors before the message is displayed [2]

A simple example can be seen in (*Figure*) below, a QR code holding a text-based URL [3]. Decoding a QR code starts from the bottom right corner, continuing up and around as per the orange arrows shown in a zigzag pattern. The first four bits, pictured as Enc below, gives the “encoding mode,” of which there are 10 different modes as of this publication.[[2]](#footnote-2) The second 8 bits, pictured as Len below, indicate how many characters should be read from the data stream.

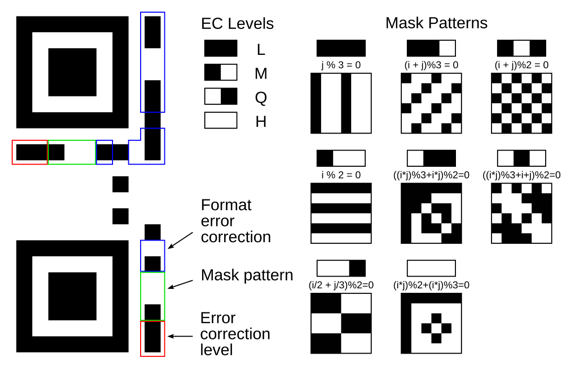
 

### Data Masking and Reversing

In analyzing the first 3 bytes (decoded as w, w, and w) from *Figure* or *Figure*, one may notice that these bytes seemingly do not match despite all bytes supposed to be representing the same data: w. From a cursory glance, the bytes are as follows:[[3]](#footnote-3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Original** | | **Segments** | | | |
| Decoding small QR codes by hand | | **Original Byte** | **Interpretation Order** | | **Interpretation** |
|  |  | | 10111011  (187) |
|  |  | | 10110100  (180) |
|  |  | | 01000100  (68) |
|  |  | | |  | |

This is because the *mask pattern* has not yet been applied. Masking occurs, “when an object (called the target) is affected by the presence of a second object (called the mask)” [4]. In this context, the *target* is the original byte of data while the mask is the predefined binary pattern applied to those bits (see *Figure*). This masking process alters the visual layout the QR code to prevent problematic patterns – such as large areas of uniform bits or sequences that may resemble finder patterns. In this QR code, the mask pattern in *Figure* is 001, and can be found from the format information in the QR code (*Figure*).



Thus, for every even-numbered row (starting from the top left corner of the QR code), the bit will be flipped. The even-numbered rows are highlighted below and the corresponding encodings are also given.

|  |  |  |  |
| --- | --- | --- | --- |
| **Original** | **Segments** | | |
|  | **Original Byte** | **Update** | **Interpretation** |
|  |  | 10001000  (136) |
|  |  | 10001000  (136) |
|  |  | 10001000  (136) |

Though while each byte now matches 136, the ASCII value for w is 119. The discrepancy lies in that, after a QR code has been interpreted from its *masked* version, the reader must thereby *unmask* it – flipping the outputs once more. This means that each byte of 10001000 becomes flipped, and the reader finally interprets the byte as 01110111, or 119 (w).

#### Reed-Solomon Codes

An issue arises when a QR code is not read perfectly: perhaps the screen holding the QR code has a smudge or water droplet, or perhaps the user has a finger over a portion of the QR code. When there is an issue with the QR code, *Reed-Solomon* *Codes* alleviate this by providing redundancy.

Formally, a Reed-Solomon code is a BCH[[4]](#footnote-4) code where is the total number of bytes in a QR code block – including the actual message and the error correction bytes; is how much real data (text, a URL, etc.) is being sent; is the minimum number of symbol differences that can be detected and corrected, and is all possible values. [2]

1. Choose a message 🡪 17 characters is 17 bytes
2. Choose QR code version and error correction level
3. Convert message into bytes
4. Build the message polynomial
   1. [www.wikipedia.org](http://www.wikipedia.org) 🡪 119 + 119x + 119x^2 + 46x^3
5. Multiply by x^(n-k) to shift the polynomial
6. Divide the generator polynomial
7. Combine data and error correction bytes
8. Scan and error correction in action

### Comparing Traditional Barcodes to QR Codes

The previous analysis underscores how much more sophisticated that QR codes are versus traditional barcodes (Figure). Some features merit even further discussion.

A black background with numbers

AI-generated content may be incorrect.

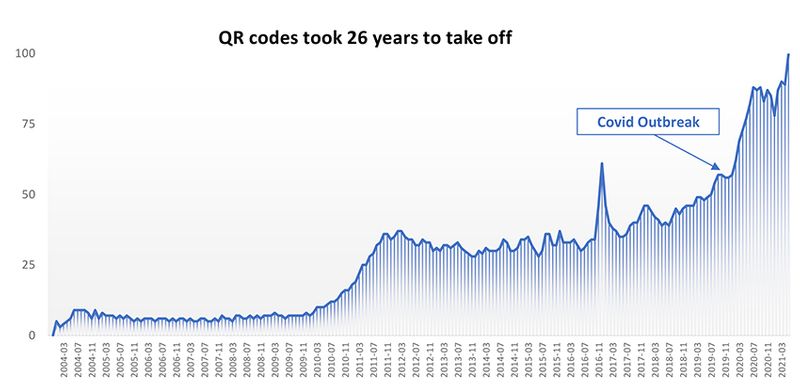
Traditional barcodes (e.g. a *universal product code* or *UPC*) can store only up to 20 digits. QR codes can store over 7,000 characters of data, including numeric data, alphanumeric data, binary data, and Kanji data. In addition, QR codes can encode URLs, e-mail addresses, and multimedia. As such, QR codes can encode 10 times more data than a barcode of the same size [5].

Furthermore, QR codes are “rapidly readable” from any direction, all made possible from the detection patterns in the 3 corners of the code. The structure of these codes also evades background interreference. UPCs do not have these features. [5]

### The Rising Popularity of QR Codes

While QR codes were created in 1994 as a means of tracking inventory, they become much more popular with the increasing use of smartphones.

According to a 2011 study [6], 20.1 million American smartphone owners used their device to scan a QR code in a 3-month average period. Among those tested, 44% scanned from a retail store and 59.4% scanned from home. This study hypothesized that the QR code was becoming the retailer’s “secret weapon.” For example, electronics retailer Best Buy adopted QR code integration on product tags to view and compare key features of the product.



But the popularity of QR codes in 2011 pales in comparison to what occurred during the COVID-19 outbreak (see *Figure*). Between March 2020 and December 2020, 8.74 million users began using QR codes for mobile payment, and QR code payments represented 85% of all mobile payments in 2020 [7]. In addition, there was a 110% increase in QR code usage in general from 2019 to 2020, and a 28% increase from 2020 to 2021. Overall, the pandemic was a catalyst for the widespread adoption of QR codes, transforming them from a technology used in specific contexts to one that was much more ubiquitous.

### QR Codes for Access Control

Most think about QR codes in the context of the user scanning a QR code and receiving data (a website, an e-mail address, a restaurant menu, etc.). However, the use of QR codes has greatly expanded even past this. QR codes have been widely adopted for access control contexts such as ticketing and building entry, offering an efficient and contactless method for verifying user credentials. In addition, unlike a physical fob or key card, QR codes only exist digitally and can be deactivated and revoked without the need to recover an object. This streamlines management and administrative overhead.

This same principle of secure, digital access control is increasingly relevant in academic environments, where schools have traditionally relied on physical fob systems. These physical fob systems are a common method of access control in schools, allowing students to unlock doors or check in to secure areas.

Talk about Gen Z’s reliance on phones.

## Problem Statement

Physical fob systems are ineffective as students frequently forget or misplace their fobs. This makes access control unreliable and induces administrative burdens to continually issue and replace fobs. While students often forget or misplace their fobs, students almost certainly carry their smartphones.

Yet even if a fob-based system were to be replaced by a more robust QR code system for access control, this would also have significant shortcomings. Traditional QR code detection often fails in real-world environments, specifically in low light conditions or under motion blur. Static QR codes also present a security risk, as they can be easily copied, reused, or shared.

## Objective

The primary objective of this project is to develop a secure, robust QR code authentication system that can replace a fob system while addressing detection and security challenges.To achieve this, the system will integrate two features.

First, this project implements adaptive thresholding, deblurring techniques, and homography transformations to improve QR code readability under the real-world conditions including poor lighting and motion blur. Second, this project refreshes QR codes, allowing each QR code to be used only once.

Together, these improvements deliver a more secure, user-friendly, and resilient QR code authentication solution suitable for use in schools and other environments needing access control.

## Literature Review

### Adaptive Thresholding and Deblurring Techniques in QR-Based Authentication

A perhaps surprising amount of research has been conducted in making QR code and barcode reading more robust and reliable.

A close-up of barcode

AI-generated content may be incorrect.In an Institute of Electrical and Electronics Engineers conference in 2010 conducted by Yahyanejad, researchers proposed methods to remove motion blur from UPC images specifically [8]. To understand their methods, it is import to examine precisely the problem they aimed to fix. In (Figure), three UPCs are presented: (a) is the ideal barcode with perfect black and white modules. These barcodes are unrealistic in real-world applications. (c) is an actual image produced by a camera without any motion blur. This is how UPCs are typically read: there is varying illumination, differences in light reflection, imperfect black and white pixels, etc. (e) is the image obtained from motion blur. As one can see, a significant difference between image (c) and (e) are exposed in the histogram: (e)’s histogram is significantly less consistent.

The deblurring method proposed is to basically guess the kernel that was used to “smear” the barcode, deconvolute the image using that same kernel and mathematically reverse the effect of the blur to get a sharp image back, determine if that kernel is the same as the original barcode, and repeat with different kernels. This results in a simple 4-step process.

*First*, the UPC was converted from a 2-dimensional image to a 1-dimensional image. Because UPCs are simply a series of vertical white and black lines, whether or not the top, middle, or bottom of the vertical line is obtained is irrelevant. Therefore, researchers averaged the pixels of each vertical column to one pixel.[[5]](#footnote-5)

A number and mathematical symbols

AI-generated content may be incorrect.

As a consequence, the kernel used will also become 1-dimensional.

A mathematical equation with numbers and symbols

AI-generated content may be incorrect.

Notably, this is an optional step and, because QR codes are indeed 2-dimensional, this step would not need to be replicated if applied to a QR code.

*Second*, a kernel – which consists of the length of the blur – is guessed. Assuming a length of in the kernel, the researchers started with a uniform kernel, assuming that the blur was spread over evenly across pixels.

A black and white text

AI-generated content may be incorrect.

*Third*, how close this 1-dimensional image is to a real UPC is evaluated using a target function. Luckily, UPCs have specific conventions that make this possible. The histogram of a barcode without motion blur has a clear bimodal shape with one intensity modeling black and the other intensity modeling white. Though an ideal UPC will have a perfect bimodal distribution with one intensity being perfect white and the other intensity being perfect black, a UPC in the real-world will essentially never have this distribution and, instead, will have much more gray and an intensity distribution much closer to the mean. The target function should be a histogram with two peaks where the variance of each peak is small while the distance between them is large. The researchers present the following Bivar equation, which is designed to evaluate how similar a deblurred 1-dimensional signal is to a clean barcode.

A math equations and formulas

AI-generated content may be incorrect.

contains all values less than the mean (the darkest pixels) and contains all values greater than the mean (the lightest pixels). thereby calculates the sum of the variances between these two groups, demonstrating how tightly values are clustered around each peak. This sum is divided by the square of the differences between their means, to measure how far apart these peaks are. This is called a *Bivar* *score*. A lower score means that the barcode has a tight, well-separated set of bimodal peaks. A higher score means that the proposed kernel is invalid.

*Fourth*, a small, random noise vector[[6]](#footnote-6) is added to a new kernel. The new kernel’s Bivar score is compared to the previous kernel’s score, and the better of the two is kept. This process is repeated many times; the researchers completed this process 1,000 times in fact.

The results of this algorithm were largely successful and an example can be seen in (Figure). 138 images taken with a 3.2 megapixel smartphone camera; 45 images (32.6%) were decoded successfully without any deblurring applied, leaving 93 images unsuccessfully decoded without any deblurring applied. Of those 93 images, 44 were successfully decoded after deblurring. Moreover, the deblurring algorithm had no effect on images that would have otherwise successfully been decoded without deblurring, so the deblurring algorithm led to almost a 50% increase in successful decoding [8].[[7]](#footnote-7)

A close up of a bar code

AI-generated content may be incorrect.

In 2021, Rioux et al. explored more advanced deblurring techniques that rely on taking what we already know about QR codes, including the strict patterns and structure rules [9]. In other words, while many techniques such as those demonstrated in Yahyanejad guess the kernel somewhat randomly until it approximates correctness and then use it to unblur the image. Rioux proposes to augment this strategy by treating the code as a probabilistic object. In other words, it is a much more sophisticated trial and error process: rather than randomly selecting a fixed blur kernel, Rioux et al.’s algorithm uses a structured, iterative process — estimating the most likely QR code via probability, and refining the kernel through optimization.

One key conceptual shift is that Rioux’s algorithm does not attempt to uncover a single most likely QR code. Instead, it analyzes many valid QR codes that follow the symbology rules and assigns a probability to each one. This mitigates the risk of committing too early to a potentially incorrect guess and allows the algorithm to remain flexible. This algorithm is a four-step process.

*First*, the image must be modeled because, in order to understand how to reverse a blur, one must understand how it occurred in the first place. This modeling is by the following equation:

A black and white math equation

AI-generated content may be incorrect.

In this equation, is the original QR code, is the upsampling matrix, is the blur kernel that describes how the camera shook or how the image got smeared, is the convultion operation that spreads to each pixel to represent the smearing, and is the resulting blurry image.

*Second*, the algorithm defines a probability function over all possible QR codes and estimates the most likely QR code as a distribution. In simplest terms, the algorithm considers all possible QR codes that follow the QR code rules and figures out how likely each one is. This is formalized using *Kullback-Leibler divergence[[8]](#footnote-8)* to stay close to known QR code structure. In estimating the most likely QR code, the following equation is used:

A black text with black letters

AI-generated content may be incorrect.

*Third*, and because the algorithm cannot feasibly iterate through every possibility in general, the algorithm will solve a *dual* *problem* with a shortcut known as the *Fenchel-Rockafellar* duality. This mathematical tool elicits that one can solve a huge problem by solving a much smaller, easier version of the problem that leads to the same answer [10]. Here, the larger problem is iterating through every QR code possible in existence; the smaller problem that is *actually* solved in the algorithm is iterating over the possibilities for a single pixel. The problem goes from trillions of possibilities of QR codes to 2 possibilities for each pixel, repeated over however many pixels are in the QR code – from over a trillion problems to only a few thousand problems.

*Fourth*, once the probabilities of each pixel are uncovered, the entire QR code is thereby reconstructed. The algorithm reconstructs the entire QR code by combining these probabilities into a full image. This grayscale image is then thresholded — turning modules with values greater than 0.5 into white, and others into black — to produce a final binary QR code for scanning [9].

The results of this algorithm are impressive, as can be seen in (Figure) which represents a blurred image and the unblurred result.

A qr code with a qr code

AI-generated content may be incorrect.

#### Discussion of Yahyanejad, et al. and Rioux, et al.

While extensive research has been conducted, issues arise when utilizing previous techniques for this project. An issue with the Yahyanejad research is its speed. The computation time for a UPC with a width of 1,000 pixels was approximately 8 seconds for 1,000 iterations and 2 seconds with 10 iterations. In fact, “*No attempts* to optimize the code for speed were made” [8]. Rioux suffers similar problems for its application to the problem at hand. According to the authors themselves, the algorithm is computationally heavy and “any implementation would require a significant amount of preprocessing” [9]. Moreover, both authors assume a uniform blur.

For the purposes of my project, speed is essential, the amount of processing power available is relatively low, and the likelihood of a uniform blur is small. This project is meant to be tested when perhaps an entire line of students is waiting to gain access to a building. In addition, this project will not be used with a high-performing computer, but rather a consumer-grade computer available on the budget of a school.

#### Reading Blurred QR Codes with Speed and Accuracy

[11]

### Security Vulnerabilities in QR-Based Authentication

What approaches have been attempted? What were the results?

## Methodology and Implementation

### System Overview

High level description of the approach with a flowchart.

### Technologies Used

What were all the technologies used?

### Enhanced QR Code Detection

#### Adaptive Thresholding

Text here.

#### Homography Transformations

Text here.

#### Deblurring Techniques

Text here.

### Time-Based QR Code Refresh System

Talk about the implementation of dynamic QR codes that prevent reuse or replay.

### Challenges and Solutions

What sorts of issues arose and how were they solved?

## Experimental Results and Evaluation

### Test Setup

How did I test my system, including what dataset and parameters?

### Performance Analysis

How well did the system hold up?

### Security Analysis

How well did the system hold up?

### User Experience

How well did the system hold up?

## Discussion

How should the results be interpreted?

## Conclusion

Summary of key findings and final remarks.

## Works Cited

|  |  |
| --- | --- |
| [1] | S. Tiwari, An Introduction to QR Code Technology, Bhubaneswar: 2016 International Conference on Information Technology (ICIT), 2016, pp. 39-44. |
| [2] | B. S. Bajaj, "Reliability on QR Codes and Reed-Solomon Codes," Canary Islands, 2025. |
| [3] | Wikipedia, "QR Code," 14 October 2011. [Online]. Available: https://commons.wikimedia.org/wiki/File:QR\_Character\_Placement.svg. [Accessed 27 March 2025]. |
| [4] | J. Zhang, D. Li, J. Jia, W. Sun and G. Zhai, "Protection and Hiding Algorithm of QR Code Based on Multi-channel Visual Masking," *2019 IEEE Visual Communications and Image Processing (VCIP),* pp. 1-4, 2019. |
| [5] | J. H. Chang, "An introduction to using QR codes in scholarly journals," *Sci Ed,* vol. 1, no. 2, pp. 113-117, 2014. |
| [6] | Retail TouchPoints, "QR Code Trends And Early Adopters," 14 February 2012. [Online]. Available: https://www.retailtouchpoints.com/topics/digital-marketing/mobile-marketing/qr-code-trends-and-early-adopters?utm\_source=chatgpt.com. [Accessed 8 April 2025]. |
| [7] | M. Tu, L. Wu, H. Wan, Z. Din, Z. Guo and J. Chen, "The Adoption of QR Code Mobile Payment During COVID-19: A Social Learning Perspective," *Frontiers in Psychology,* 2022. |
| [8] | S. Yahyanejad and J. Strom, "Removing motion blur from barcode images," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, c2010. |
| [9] | G. Rioux, C. Scarvelis, R. Choksi, T. Hoheisel and P. Marechal, "Blind Deblurring of Barcodes via Kullback-Leibler Divergence," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* vol. 43, no. 1, pp. 77-78, 2021. |
| [10] | E. Dohmatob, "Fenchel-Rockafellar duality theorem, one ring to rule'em all!," 31 October 2019. [Online]. Available: https://dohmatob.github.io/research/2019/10/31/duality.html. [Accessed 13 April 2025]. |
| [11] | J. Li, Z. Dong, M. Zhou and C. Zhengai, "A motion blur QR code identification algorithm based on feature," *Neurocomputing,* pp. 351-361, 2022. |

1. A *pixel* can be more appropriately referred to as a *module* [↑](#footnote-ref-1)
2. Modes include 0001 for numeric encoding, 0010 for alphanumeric encoding, 0100 for byte encoding, 1000 for Kanji encoding, etc. [3] [↑](#footnote-ref-2)
3. The three figures are recreations of the QR code for “www.wikipedia.com.” Each figure is a “w” in “www”. [↑](#footnote-ref-3)
4. Explain BCH. [↑](#footnote-ref-4)
5. More specifically, researchers averaged approximately 5% of each vertical line, in the middle of the UPC. [↑](#footnote-ref-5)
6. This vector is controlled by a variable that shrinks over time, so the guesses become better with each attempt. [↑](#footnote-ref-6)
7. 45 images were decoded successfully without deblurring and an additional 44 images were decoded successfully with deblurring (without any effect on the original 45). Thus, 89 of the 138 images (64.4%) could be decoded with deblurring versus only 45 (32.6%) being able to be decoded without deblurring. [↑](#footnote-ref-7)
8. In simple terms, Kullback-Leibler divergence is a way to measure how different two probability distributions are from each other. In the context of QR codes, this is used to ensure that the probabilities uncovered in the algorithm approximate what we know about QR code structure. This ensures that the probabilities are not unrealistic. [↑](#footnote-ref-8)