A Secure QR Code Scheme

Julian Brackins [1] and Mengyu Qiao [2]

[1,2]Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
501 E St Joseph St, Rapid City, SD 57701
[1]*Julian.Brackins@mines.sdsmt.edu*
[2]*Mengyu.Qiao@sdsmt.edu*

Abstract

Quick Response code, an industrial standard originated from manufacturing informatization, has been getting growing attention and utilization in the past few years, due to the deep penetration of smart phones and the advancement of wireless networks. QR codes outperform other barcode standards because of their high capacity, fast readability, strong error tolerance, and flexible encoding options, all of which enable QR codes to be used as a convenient method of information sharing and storage for various computer and mobile applications including URL sharing, mobile app installation, social networks, electronic payment, ticketing and shipping services. The increasing popularity also make QR codes a potential target of traditional and emerging threats. One of the security risks present with QR codes is the existence of phishing schemes where attackers direct users to fake pages with the intention to intercept secret information. This can be translated into the QR code realm when hackers modify or print new QR codes present on posters, business cards, or other medium, allowing the scanned QR codes to redirect a user to a dangerous web page. Since security has always been a critical issue for mobile computing and communications, authentication methods are needed in order to protect the data and origin integrity of QR codes.

The aim of this project is to develop a scheme along with a proof of concept prototype of a security countermeasure that authenticates QR codes using build-in security features. This scheme involves the modification of QR code generation to embed message authentication code. Upon scanning a secure QR code, the user will receive the encoded message on the barcode as usual. However, there will also be an added message validation layer involving the use of cryptographic systems. This implementation is expected to be non-intrusive and additive to existing QR code applications. This means that if the scanning device were to be used on an old QR code without the digital signature information, the scanner would still return the intended payload. Meanwhile, the scanner would inform the user that the particular QR code lacks the security feature. The finalized prototype is expected to be deployed to popular mobile platforms to exemplify the anticipated functionalities.

1 Introduction

Quick Response (QR) code is a barcode standard developed by Japanese company Denso Wave in the 1990s[1]. Compared to traditional barcodes, QR codes are 2-dimensional, allowing for a greater amount of information storage. QR codes have a wide variety of uses, including easy sharing of URLs, digital currency exchange, as well as video games and other virtual entertainment. One recent example of the presence of QR codes is the photo-sharing mobile app Snapchat[6], which allows users to send additional text or web links in their snaps by including a QR code in their picture.

Figure 1: QR Code

With the popularity of these codes, it is important to identify any potential security vulnerabilities. Since QR codes are not (easily) human-readable, users have to rely on the scanning software to extract the message. Scammers can exploit this opportunity to tamper with existing QR codes. In the case of URLs, phishing scams can be implemented in the way that an attacker replaces publicly available QR codes on posters, signs, or business cards. This could redirect users to fake websites with the intention of collection personal credentials [5]. In 2011, a QR code containing a malicious application was circulated [7]. If scanned, this code would install a trojanized chat application, which would automatically send several premium-rate SMS messages at a rate of $6 per text. Because of these potential attack vectors, there is a strong demand to investigate effective security features for QR codes.

1.1 Overview

This paper presents an novel scheme to improve the security of QR codes by adding an additional layer of security features into QR codes.

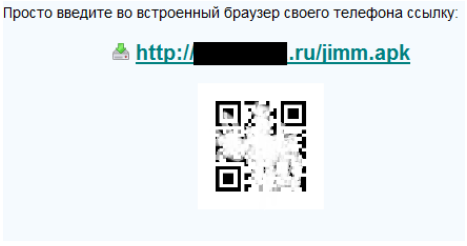


Figure 2: Malicious QR Code [7]

The following sections of this paper cover both a validation layer to be implemented in QR code scanning applications, as well as provide a modification to the encoding of QR codes in order to generate the codes with security features. Digital signatures, a commonly used cryptographic tool, are a system given the same legal standing as handwritten signature [2]. Two features of digital signatures, authentication and integrity, are crucial to the research detailed in this paper. Authentication of a digitally signed mes-

sage indicates the message was created by a trusted sender. Alterations of the message will cause a discrepancy between the message and the original signature, so that digital signature insures that a digitally signed message has not been edited in transit. Digital signatures have been used to authenticate emails, software, and other electronic messages, and the intent of this work is to implement a system of embedded authentication code for QR codes using digital signatures.

A successful envisioning of this implementation will allow for these security features to also be completely optional. For some QR codes, such as those which contain simple and noncritical messages, the possibility of message compromisation is not a pressing issue. If a standard QR code containing none of the proposed security features is scanned by the prototype application, the app will still return the intended QR code message payload. However, the app will inform the user that the message does not contain any security features, tasking the user to proceed with discretion when handling the message results.

The new, secure QR codes should also still be readable by currently existing QR code applications, without any destructive changes to the message structure. An incredibly trivial iteration of security feature would be to simply append the digital signature of a QR code to the end of the message itself during encoding, then parse the signature from the message when decoding. However, this type of implementation would be incompatible with other scanning apps, which would not anticipate separating the digital signature from the intended message payload.

The rest of this paper is organized as follows. Section 2 gives a general overview of QR code structure and the process of both generating and scanning the codes. Section 3 provides a discussion of the methods used to convert standard QR codes to digitally signed QR codes, including depictions of the prototype phone application. Section 4 describes an analysis of the digitally signed QR codes and possible insight on how to scale up this protocol's implementation beyond proof-of-concept, followed by conclusions in Section 5.

2 QR Code Standard Overview

Figure 3 gives a simplified description of the standard process of generating a QR code from a provided message and scanning the code with a smart phone or a comparable device.

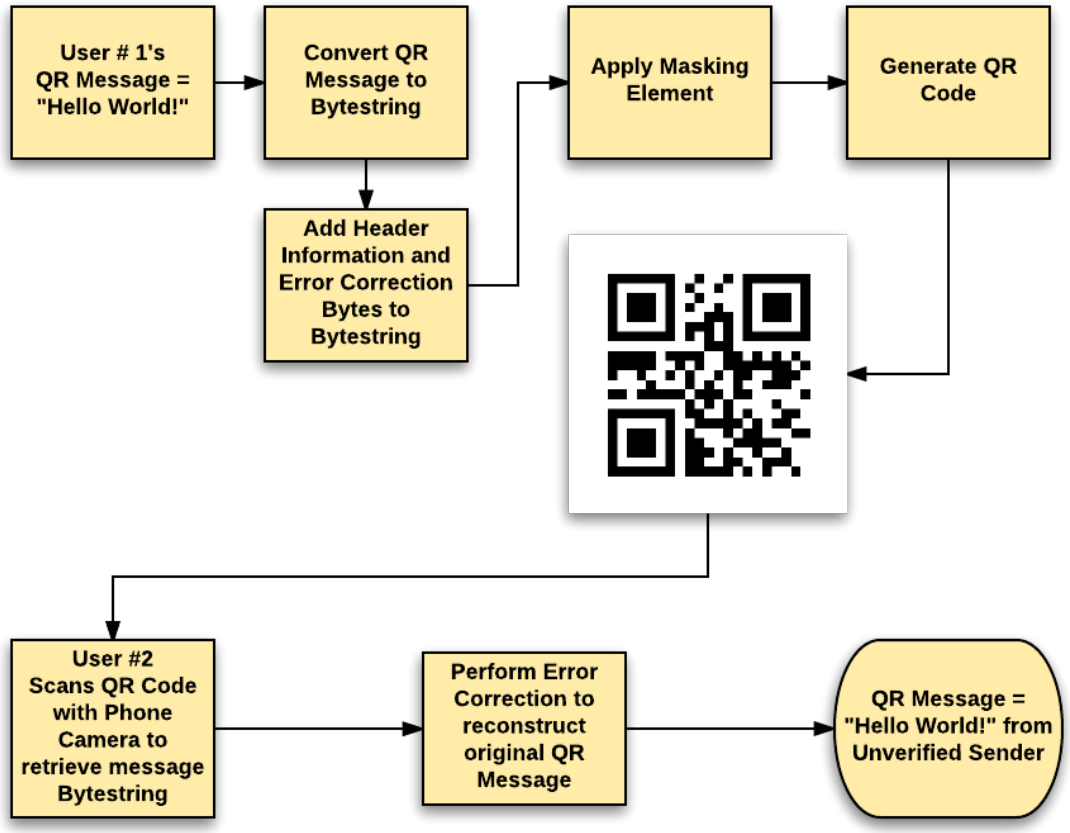


Figure 3: Simplified QR Code process. Details the generation & scanning of a QR code.

When provided a message string, the encoder converts the message into a byte string interleaved with general QR header information, error correction bytes, and a masking element. This modified byte string is then converted to a 2-dimensional matrix of 1's (white) and zeroes (black) which can be synthesized into an image. When this image is scanned by a phone camera, the byte string is retrieved and converted to the intended message, viewable by the person who initiated the scan. In order to implement a security protocol for QR codes, both the encoding and decoding procedures must be developed.
The following sections detail a number of QR code features that will be utilized or modified in this project.

2.1 QR Version Information

Each QR version has a distinct amount of black and white dots for composing the QR code. Versions range from Version 1 (21×21 dots) to Version 40 (177×177 dots) [4]. The QR code Version information can be identified in the sections of a QR code highlighted in blue in Figure 4.

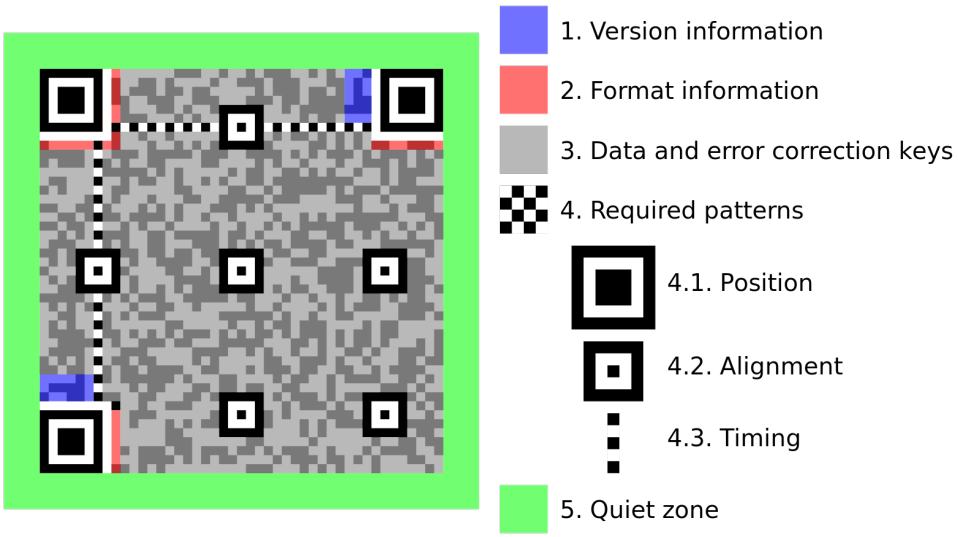


Figure 4: QR Code Structure. Image credit Bobmath on Wikipedia[3].

2.2 QR Format Information

QR code header information includes both Version Information and Format Information regarding the specific QR code. This is detailed in Figure 5 and identifies the masking pattern and level of Error Correction observed in the QR code.

2.3 Error Correction

QR codes have available functionality to restore data if the message is corrupted or damaged [4]. There are four levels of Error Correction possible for QR codes. Level L, the lowest Error Correction rate, can reconstruct a damaged message with up to 7% corruption. Level Q, the highest Error Correction rate, will reconstruct a message containing 30% corruption. Level M, which has an error tolerance of 15%, is the most commonly used level. QR codes with higher levels of Error Correction require more data bits, increasing the code size. Error Correction is implemented according to the Reed-Solomon algorithm. Reed-Solomon is a non-binary cyclic error correction method initially designed to reduce communication noise for artificial satellites [8]. The error correcting method is capable of performing corrections in data at the byte level, which has come in handy products such as CDs, Blu-Ray Discs, and QR codes.

2.4 Masking Element

Figure 5 details all of the available masking patters, as well as identifies where in the QR code this mask pattern specifications are located. Masking helps to produce an even distribution of white and black dots in a QR code and reduces the possibility of large blank spots in the barcode.

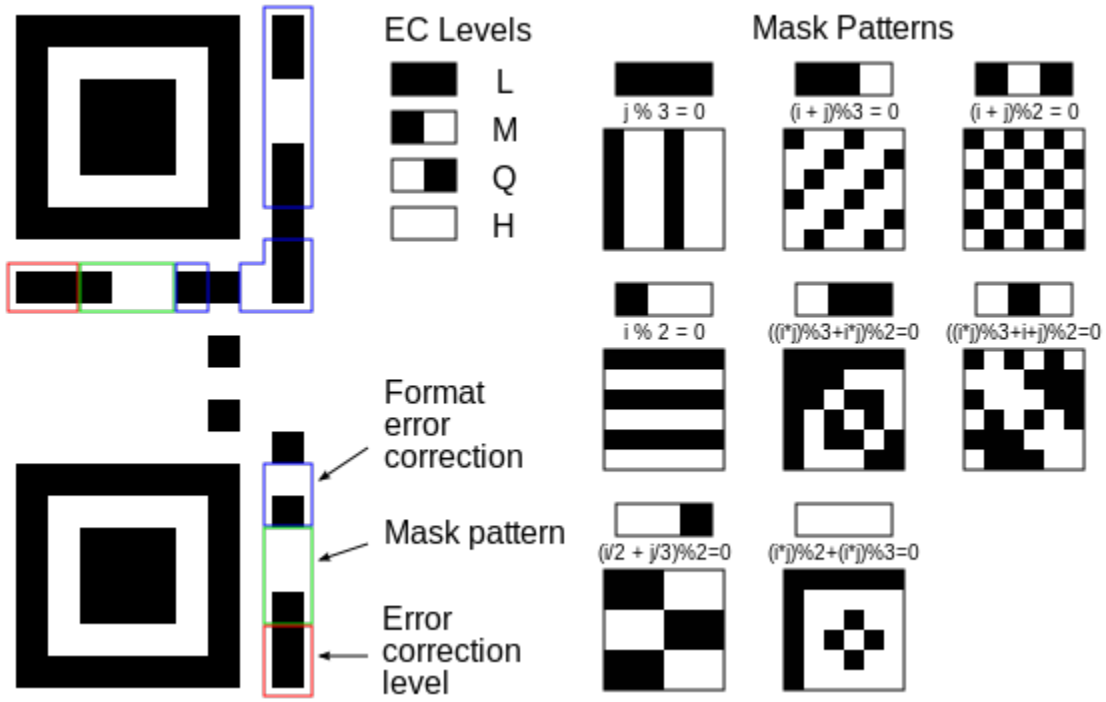


Figure 5: QR Format Information. Image credit Bobmath on Wikipedia[3].

3 Proposed Scheme

Figure 6 is an adaptation of Figure 3 detailing the added steps within the QR encoding / decoding process. The sections in yellow are the previously existing steps in the procedure. The pink sections are added methods in the encoding process, which involves signing the QR message and embedding the signature in the QR code.

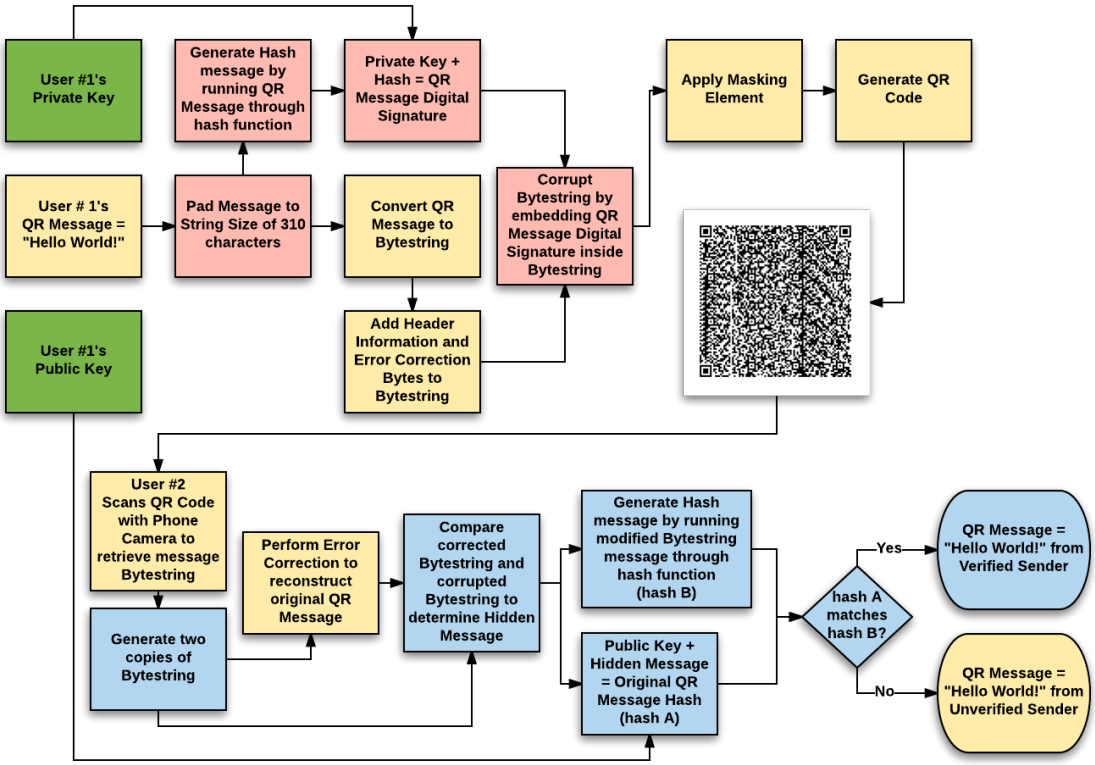


Figure 6: QR Code process from Figure 3 with proposed security features implemented.

The sections in blue detail the methods added in the decoding process, which would typically occur on a smart phone device. These methods involve converting the scanned QR code to a byte string, extracting the digital signature, and comparing the signature and QR message to determine if the message has been sent unaltered.
The encoder and decoder were both developed and executed on a MacBook Pro using Java. The decoder was later ported to a Google Nexus tablet running Android OS.

3.1 Encoding QR Codes with an embedded hidden message

With the current implementation, each character of an embedded digital signature is stored in the corrupted message bit string as noted in Algorithm 1. Each of these characters is represented by a byte in the bit string. The first 7 bits of this byte represent the character in the embedded message, translating to an ASCII value between 0 and 127. The 8th bit in the byte is always flipped to guarantee a detectable difference within each corrupted

Algorithm 1 Embed Message in Corrupted Bits

```
procedure EMBEDMSG(message, qrBitString)
    i ← 0
    for each asciiCharacter in message do
        secret ← Integer value of asciiCharacter
        for each byte in qrBitString (8 bits) do
            bit ← MSB of secret
            if bit != qrBitString[i] then
                flip qrBitString[i]
            i++
            shift secret 1 bit to the right
        flip the first bit in current byte of qrBitString
        move 8 bits over to the next byte in qrBitString
```

byte of an embedded message. For instance, if a particular byte in the message bit string was ”01000001” and the character to be embedded within that byte were to be a capital ”A”, the resultant byte would be ”01000001”, and would appear uncorrupted during the decoding procedure. As a result, the most significant bit would be flipped, generating the string ”11000001” instead.

3.1.1 Padding the Messages

In initial phases of the decoder, it was noted that the hidden messages were out of order in the byte string retrieved by the decoder.
As a result, the QR message length is inflated to 310 characters, and the hidden message length is inflated to 293 characters. Each message cannot exceed their respective predefined length. These values were selected through an iterative process to ensure an adequate length for the hidden message to hide a meaningful message, an adequate length for the standard QR message to contain nearly any typical URL, and to guarantee the hidden message length never exceeds the length of the standard QR message.

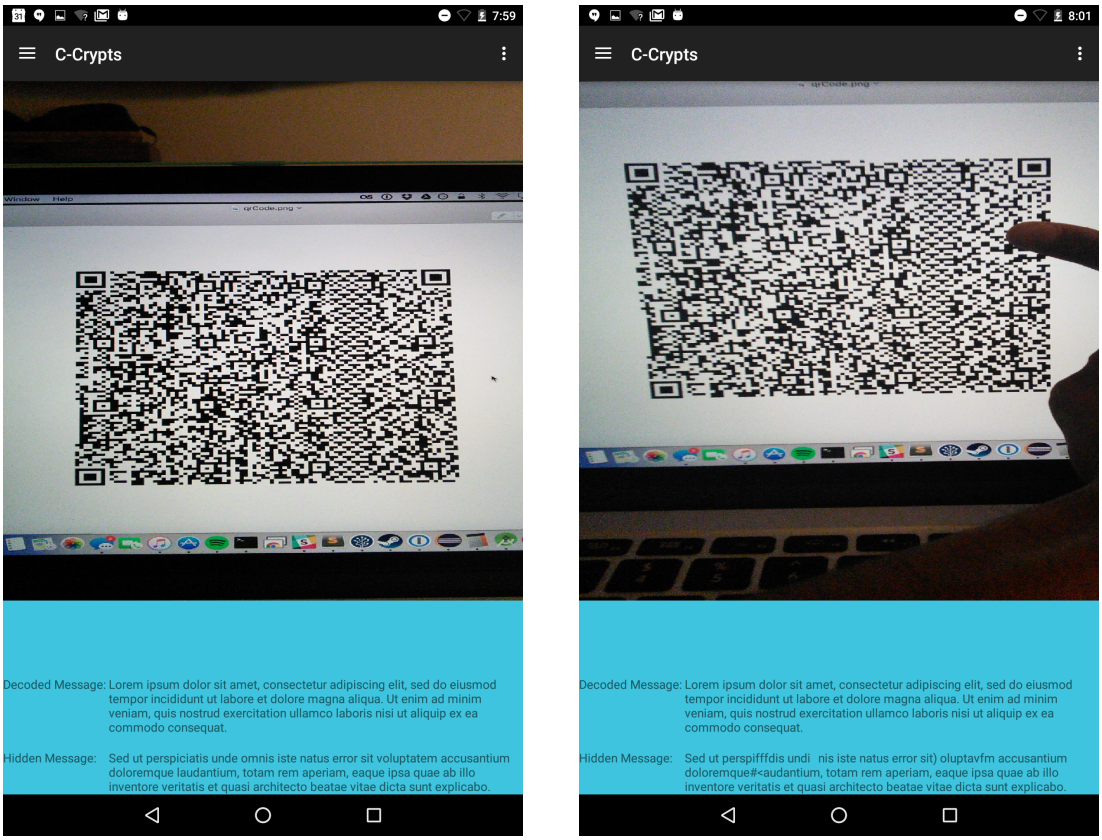
3.2 Decoding Secure QR Codes with a Mobile App

Decoding of the Secure QR Codes is detailed in Figure 6, with the blue sections indicating the additional features. When a Secure QR Code is scanned, the application will first generate two copies of the scanned byte string. One of these byte strings will be run through the Reed-Solomon Error Correction method as before, restoring the standard QR code message. This restored byte string will be compared byte-by-byte with the unchanged corrupted byte string. Any bytes that have been altered between the two strings are the bytes corrupted during encryption to embed the hidden message. Converting the first 7 bits of each of these bytes into the ASCII character representation of the number will return the hidden message. The following segment details the implementation of sending both the standard QR code message and hidden QR code message both as plaintext in order to verify this method can work in a real application.

3.2.1 Initial Development

The initial iteration of this research application sought out to determine the viability of embedding a hidden message into a standard QR code, ignoring any actual cryptographic protocol tools. If the application were to be successful in extracting a hidden message generated into a displayable QR code through a camera scanning feature, then the program could be later enhanced to support a verification system by embedding an ASCII representation of a digital signature in the hidden message. This will be demonstrated in section 3.2.2.

The next few scans are an overview of a Lorem Ipsum QR code. The lorem ipsum text was selected for this example as a general long form text string for testing out the retrieval of both a standard QR code payload as well as a hidden message of comparable length. The standard message is the first two sentences of the standard lorem ipsum text, with a length of 232 ASCII characters. The hidden message is sentence from "De finibus bonorum et malorum" by Marcus Tullius Cicero, with a length of 216, only a few characters shy of the standard message. The following subsection details the results of this test, as well as some observations of errors present.



(a) Successful Scan

(b) Corrupted Scan

Figure 7: Successful and Corrupted Scans of a QR code containing plaintext.

Figure 7 is an example of a scan of the lorem ipsum QR code. In Subfigure 7a a successful scanning of the QR code is observed. The decoded message contains the first paragraph

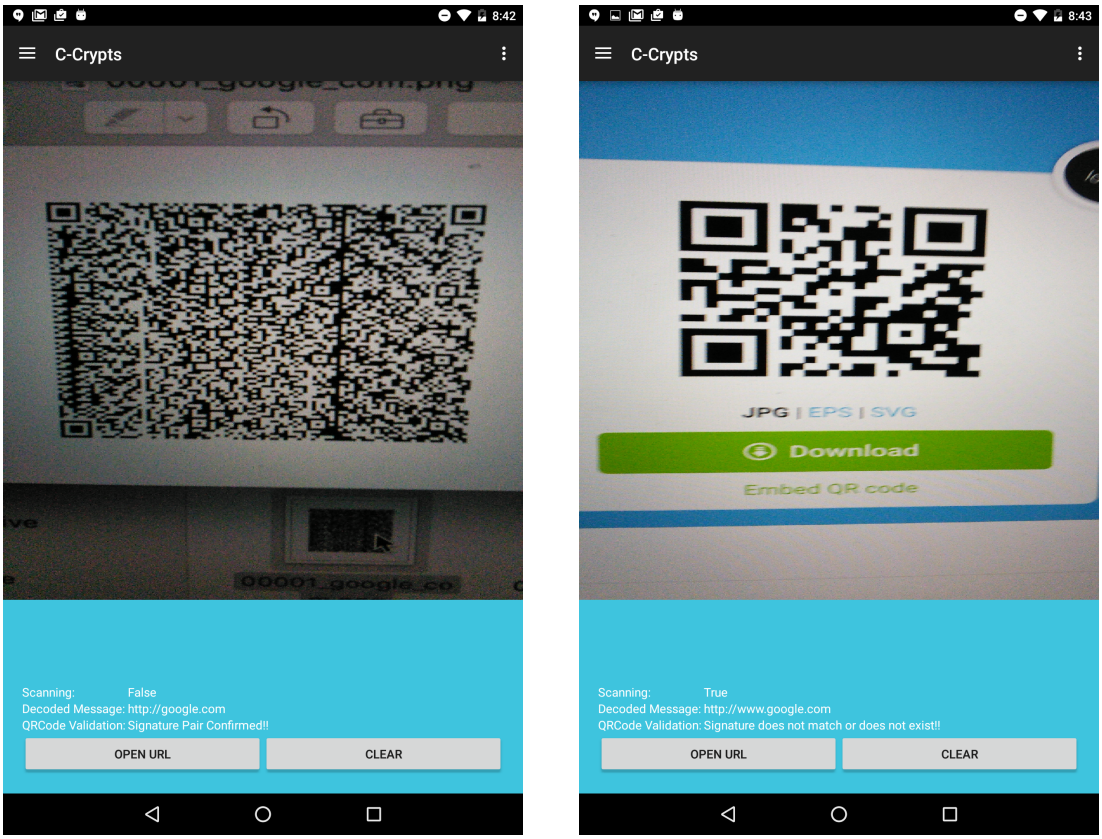
of the lorem ipsum text intact, and the hidden message is retrieved. If scanned by a stock QR code scanner, only the first paragraph will be displayed. Subfigure 7b demonstrates an example of the message being slightly obstructed, in this case by a finger. Because the retrieval of the hidden message relies on the differences between incorrect and correct bytes in a given QR code message, the hidden message is corrupted in this example. The first paragraph, which is the main decoded message, remains intact.

This concept, when extended to a verification process detailed later, will not explicitly prevent a user from scanning a QR code that does not successfully complete the verification step, either from a lack of a hidden message or the obstruction of the hidden message similar to this example. In all cases, the standard message in the QR code will be made available. However, the application will inform the user of the QR code's lack of successfully completing a verification of the QR code, allowing them to process the message at their discretion.

The lorem ipsum text was used in order to identify a rough estimate of message length, both for the standard message and the hidden message. The goal of this security feature is to primarily verify QR codes containing URLs to webpages. Most standard web pages fall under the character count displayed in the decoded message, and with the presence of URL shortener sites such as bit.ly, nearly all web addresses would be able to fit into the modified QR codes.

3.2.2 Verification

The final iteration of the prototype can be observed in Figure 9, where the application is used on both a Secure QR code and a standard QR code containing the URL for Google. Pressing the "Open URL" button will take the user to the intended website if the QR code is a URL. When scanning the secure QR code, an additional message appears for the user, indicating the QR contains an embedded digital signature. The app, which contains the public key corresponding with the private key used to generate the digital signature, verifies the signature of the QR code which will return a success if the standard message within the QR code has remained unchanged. If the digital signature validation is unsuccessful, or if the QR code does not contain a digital signature, the user will be informed by a message that indicates that the code is not secure.



(a) Successful Scan, Verified Sender

(b) Successful Scan, Unverified Sender

Figure 8: Prototype App scanning a secure QR code (Left) and a standard QR code (Right).

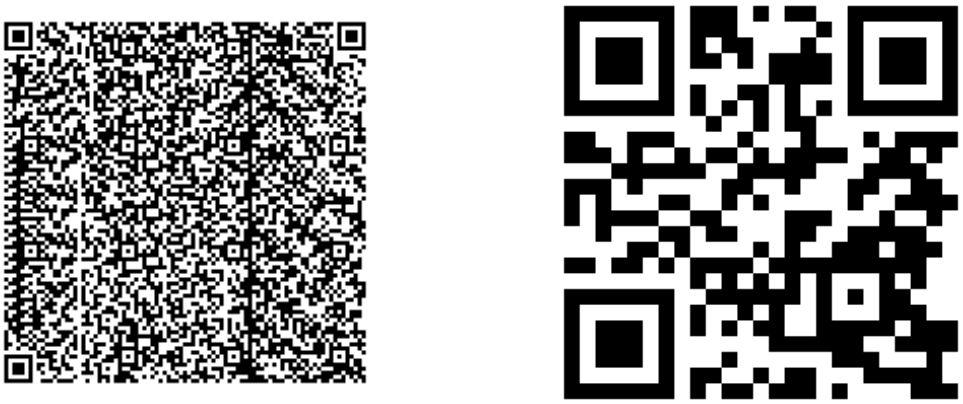


Figure 9: Secure QR code (Left) and Standard QR code (Right)

4 Additional Work

4.1 Improving Readability

The length of the QR message is dictated by the length of the hidden message. What this means is that the hidden message cannot ever exceed the length of the actual QR message, and therefore the QR message must be long enough to accommodate this. If the digital signature can be embedded in the QR code with a smaller byte footprint, both the hidden message and QR message could be shortened, potentially increasing the readability of the secure QR codes.

4.2 Error Correction

As noted in Figure 7b, any obstructions present in the secure QR code will modify the hidden message. Because of this, in order to verify a particular QR code, the entire code must remain completely intact. This is an unfortunate side-effect from the implementation's abuse of the pre-existing error correction functionality of the standard QR code. On a positive note, the standard message will still remain intact as long as the combination of the obstruction and the previously corrupted portion of the code used for the hidden message do not take up more than 30% of the QR code data area. Implementing another error correction feature, such as parity bits, in the hidden message could compensate for any unintentional flaws. This would require the hidden messages to be longer.

5 Conclusion

The QR code has many applications in marketing, commerce, and entertainment, and with this ubiquity comes a multitude of security risks. The security features presented in this research intend to mitigate the threats of malicious QR codes by establishing an authentication layer to help signify scanned codes were generated by trusted sources. By taking advantage of the Error Correction capabilities of the QR codes, digital signatures can be embedded in a QR code without modifying the original message by corrupting a number of bytes within the Error Correction threshold. These digital signatures can then be subsequently extracted during the scanning phase by comparing the corrupted byte string to the recovered byte string.

Standard QR codes will still return their intended messages when using this modified scanning method, meaning currently existing QR code generation sites and services will not be forced to change or adopt this security feature. However, the security features can be used for messages where origin integrity and message authentication are a high priority, including URLs and electronic payments. This added layer of security can help eliminate many of the existing attack vectors present in standard QR codes.

References

[1] History of QR code. http://www.qrcode.com/en/history/ (Retrieved: 3/16/2017).

[2] *Electronic Signatures in Global and National Commerce Act*. the U.S. Government Printing Office, 2000.

[3] BOBMATH. QR format information, 2011. [Online; accessed March 17, 2017].

[4] DENSO WAVE. QR Code, howpublished = "http://www.qrcode.com (Retrieved: 3/16/2017)", journal=DENSO WAVE.

[5] KIESEBERG, P., LEITHNER, M., MULAZZANI, M., MUNROE, L., SCHRITTWIESER, S., SINHA, M., AND WEIPPL, E. QR code security. *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia - MoMM '10* (2010).

[6] KOKALITCHEVA, K. Here's a secret secondary use for snapchat, May 2016.

[7] MASLENNIKOV, D. Malicious QR codes pushing android malware. https://securelist.com/blog/virus-watch/31386/malicious-qr-codes-pushing-android-malware/ (Retrieved: 3/16/2017), Sep 2011.

[8] REED, I. S., AND SOLOMON, G. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics 8*, 2 (1960), 300304.