SQR Codes: A Secure Alternative to QR Codes

Gianni Hernandez
ghernandezdelapena@college.harvard.edu
Harvard University

Emeka Ezike
eezike@college.harvard.edu
Harvard University

Victor Goncalves
victorgoncalves@college.harvard.edu
Harvard University

Boluwaji Odufuwa
bodufuwa@college.harvard.edu
Harvard University

ABSTRACT

Quick Response (QR) codes are widely used for their quick and convenient data access, but they suffer from significant security vulnerabilities that expose users to QR phishing (quishing), unauthorized access, and malicious exploits. These issues stem from QR codes being machine-readable but not human-verifiable, leading to potential manipulations undetected by users until after scanning. To combat these risks, we propose Secure QR (SQR) codes, a robust alternative that enhances security while maintaining operational efficiency and standard compatibility. SQR codes make use of digital signatures and a third-party certification authority, drawing on security models from web browsers. This approach enables secure QR code generation and management, using the Elliptic Curve Digital Signature Algorithm (ECDSA) for authentication and integrity validation. Each SQR code embeds a URL and its entity's public key, making any content alterations detectable by our scanners.

KEYWORDS

QR codes, Phishing Attacks, Quishing, Digital Signatures, Certificate Authority, ECDSA

1 INTRODUCTION

Quick Response (QR) codes represent a significant evolution from traditional 1D barcodes. Their two-dimensional structure allows for considerably higher information density and enhances robustness through built-in error correction features. Today, they are widely used in various applications, including URL sharing, advertising, digital transactions (e.g., via Venmo and PayPal), authentication for access control, business card sharing, and social media interactions such as Snapchat friend requests [4]. QR codes are a popular tool for quickly and efficiently disseminating information due to their ease of deployment, particularly in public spaces. For example, in Cambridge, MA, the city uses QR codes on large advertisements to provide updates about the local MBTA transit system at bus stops, and on parking tickets to facilitate payment of fees.

Despite their convenience and utility, the intrinsic property of QR codes being machine-readable but not human-readable presents significant security challenges. Users are unable to visually distinguish between legitimate and maliciously altered QR codes, which places an inherent trust in the scanning software to accurately interpret the encoded message. This vulnerability makes QR codes particularly susceptible to "quishing" attacks, where malicious actors might embed harmful URLs within QR codes on seemingly realistic advertisements or posters, with the intention of harvesting user data.

Moreover, the potential for QR codes to facilitate malware distribution and unauthorized Wi-Fi connections—where users might unknowingly connect to malicious networks designed to intercept transmitted data—further exemplifies their security risks. Attackers can also alter the physical structure of a QR code by changing the color of specific modules (from white to black or vice versa), thereby redirecting users to unintended URLs [4]. These vulnerabilities emphasize the critical need for enhanced security measures when interacting with QR codes.

In this project, we introduce Secure QR (SQR) codes, an enhanced format of traditional QR codes that integrates robust security features. Built on the conventional monochrome dot matrix, our SQR codes are designed with an added layer of security through a centralized third-party certification authority inspired by web browser certificate authorities. This platform enables entities, such as large organizations, to generate and manage SQR codes securely. Entities first establish their identity with the certificate authority by generating a public/private key pair. They can then authorize URLs by creating digital signatures, embedding both the URL and the entity's public key within the SQR code.

This approach addresses key aspects of the CIA (Confidentiality, Integrity, and Authentication) triad. For integrity, we employ ECDSA digital signatures, allowing entities to validate URLs associated with them. This ensures that any alteration of the SQR code's content is detectable, as our scanners will detect differences between the altered code and will not allow its scan. In terms of authentication, our system verifies entities through the certificate authority using their public/private key pairs to authenticate the URLs they endorse. It's important to note that while confidentiality

is not a focus of our system, as it does not restrict data visibility. The primary goal is to ensure that users can confidently verify the authenticity of the content linked via SQR codes.

Our implementation of SQR codes maintains operational efficiency, presenting minimal latency overhead compared to traditional QR codes. However, it does require a larger payload for authentication and integrity purposes. Additionally, our implementation ensures that standard QR code scanners cannot interpret SQR codes, thereby reinforcing the uniqueness and security of our system. If an SQR code fails verification, the system prevents it from being scanned, thus upholding the security standards intended by the issuing entity.

2 MOTIVATION

There are a vast amount of flyers with QR codes present across Harvard University's campus. These flyers, found in classrooms, dormitories, libraries, and other public spaces, serve various purposes: they advertise club activities, political messages, surveys, events, and more. As these flyers are typically created by students for events and organizations, they do not undergo any formal verification process by the university. Therefore, it is difficult for viewers to distinguish these flyers as legitimate or malicious. Typically, students generally accept these flyers and scan them without caution of malicious websites. This behavior raises concerns, particularly in the context of potential quishing attacks.

Recognizing the potential risks associated with scanning malicious QR codes, we wanted to conduct a social study to examine engagement with these codes on campus. Our experiment involved 100 non-malicious flyers, designed to mimic a phishing attack while safely redirecting to a website that showed a counter of those who have been 'quished' to the website. These flyers promoted a fake event featuring a talk about elections with former President Barack Obama (Figure 2). We placed these posters at various high-traffic locations including: the Science Center, Lamont Library, undergraduate residential houses, public flyer boards, and the Science and Engineering Complex.

The response was tracked over a period of five days. Within the first 24 hours, the flyers had been scanned approximately 650 times. By the end of the fifth day, we recorded around 1200 unique scans (Figure 1). Note, by the 8th day, most of the flyers had been taken down, especially in the highest traffic locations. This high level of engagement underscores the effectiveness of QR codes in capturing attention and the potential risk if such tools were used maliciously. For instance, imagine a scenario where the QR code directed users to a website asking them to RSVP for the event by providing their first and last names, and their email address. Next, the site could prompt them to create an account, requiring them to set a password. In a theoretical malicious setup, such a site might store the user's credentials, including passwords in cleartext. This experiment sets the stage for further discussion on the security implications of QR codes and the need for increased awareness and scrutiny among the campus community.

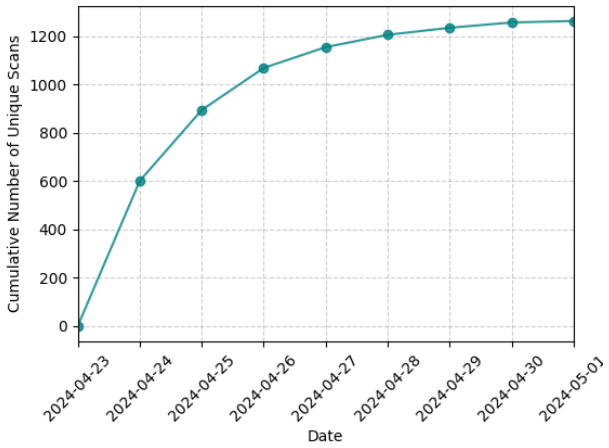

Figure 1: Cumulative number of unique scans over 8 days.

3 PRIOR WORK

3.1 QR Code Security Vulnerabilities and Attacks

The flexibility and ease-of-use of QR codes come with inherent security risks that have been exploited in real-world attacks compromising users' devices and privacy [1], [2], [3].

Previous research has identified several common vulnerabilities associated with QR codes:

(1) **Social Engineering Attacks**: Attackers employ techniques such as replacing entire QR codes or modifying individual modules to deceive users into accessing fraudulent websites and steal sensitive personal information [3]. This type of attack is known as "quishing" (QR code phishing).
(2) **Malicious Code Execution**: Certain codes like MMI codes embedded in QR codes can be used to execute commands on mobile devices, leading to data loss [3].
(3) **Web Application Attacks**: QR codes can facilitate attacks like SQL injection and browser-based exploits against web applications and sites [3].
(4) **Spreading Malware**: QR codes have been used to encode malicious URLs that lead to phishing attacks and inject malware onto users' devices [2].

3.2 Securing QR Codes with Cryptography

To address these security challenges, recent efforts have focused on integrating cryptographic techniques into QR codes.

Figure 2: Example QR Code Harvard Phishing Flyers

The study by Vidas et al. represents one of the first systematic evaluations of incorporating cryptography into QR codes following European Network and Information Security Agency (ENISA) guidelines [2].

This study evaluated the feasibility of using digital signatures (RSA, ECDSA), message authentication codes (HMAC), and symmetric encryption (AES) with QR codes on Android smartphones. ECDSA emerged as a promising option, providing robust protection while maintaining usability during scanning [2].

However, including certificates for establishing trust adds significant complexity that can hinder the scanning process. Statistics highlight potential tradeoffs between security and usability when using certificates, such as those between detection and speed, and ease-of-use. [2].

Building on this, Wahsheh and Luccio developed BarSec, a tool for easily generating and reading secure QR codes with selected cryptographic schemes while considering usability factors like algorithm selection and size overhead [5].

3.3 Machine Learning for QR Code Phishing Detection

Another line of research has applied machine learning techniques to detect QR code phishing attacks targeting the web. These studies developed models using methods like vectorization and classification algorithms (e.g. logistic regression, Naive Bayes) to distinguish between legitimate and phishing URLs in QR codes [1].

While promising for high phishing detection accuracy, the lack of comprehensive datasets for QR code phishing poses a challenge. User vigilance and scanning only trusted QR codes is still recommended to mitigate risks [1].

3.4 Towards Usable QR Code Security

As highlighted by Krombholz et al., maintaining a balance between security and usability is critical for QR code systems but has been largely overlooked [3]. Their study proposes guidelines like designing software to detect phishing QR codes, implementing secure requirements, using digital signatures, and content verification tools.

Understanding user concerns in different contexts is also emphasized, as research shows varying security vulnerability perceptions among internet users [3]. Investigating intercultural differences in security awareness could inform targeted strategies for raising QR code security awareness.

While cryptographic primitives and machine learning show promise for enhancing QR code security, open challenges remain in navigating the trade-offs with usability, acquiring robust datasets, and promoting widespread adoption of security practices. Prior work demonstrates that a systematic, lightweight approach is needed to make QR codes simultaneously secure and usable across diverse applications and contexts.

4 SQR CODES

4.1 Overview

With the security flaws of QR codes in mind, a solution must address the problems of tampering, social engineering, and lack of transparency without significantly impacting the usage of QR codes.

Like with most prior work on QR code security, a secure QR format need not fundamentally alter the monochrome dot matrix format of QR codes to achieve its goals. Since QR codes are capable of storing generic bits of data, the format is limited only by two factors: the amount of space required to store the data and the amount of time it takes to decode and/or retrieve data.

The first factor limits the level of error correction present in the QR code itself, with higher error correction levels providing more flexibility in the amount of bits that can be changed or corrupted, at the expense of providing less information per dot in the QR code dot matrix. Specifically, there are three levels of error correction in the QR code standard ("low", "medium", and "high"), with higher levels of error correction reserving more bits for error correction blocks, displayed as squares enclosing another square inside of them.

Meanwhile, the second factor limits where and how a QR code can be used: information stored away from the QR code requires extra latency to access via the internet, and requires the user to trust a third party as a reliable source of information.

Since most authentic websites already induce the costs and issues with third-party trust and telemetry via TCP/TLS/SSL handshakes, our attention is primarily directed towards the amount of information encoded in the SQR format. This limits usage in the general case where internet is not available, but the primary usage of QR codes continues to be for the sharing of websites and other services that require the internet (such as email).

Digital signatures using public key cryptography come to mind as a natural solution to the security flaws of QR codes when the aforementioned factors are considered, as discussed in Vidas et al. Digital signatures provide the user with two relevant guarantees for signed data: that the signed data is unaltered, and that the signed data was signed by the user with the public key provided.

However, a digital signature and public key pair are far too large to embed in a QR code practically (comprising hundreds of bytes in total for traditional schemes like RSA and ECDSA). Instead, we build upon this idea by storing only a 48 byte ECDSA P-192 in the QR code. The signature can be stored by an external, trusted third party, similarly to SSL.

Alternatively, one could elect to store the entire signature and public key inside of a modified dot matrix code format using multiple colors, such as HCC2D or a JAB code, which can encode up to three times the amount of data in the same surface area without losing any error correction properties. However, since HCC2D has not been standardized and the JAB code format was only recently standardized (and is rarely used), we have elected to work around the limitations of the QR code format instead of replacing it altogether.

Thus, the SQR format is simple: a 48-byte ECDSA P-192 public key followed by a link to a website embedded in a QR code. Though 48 bytes is not a negligible amount of overhead for the QR format, links can be shortened to fit comfortably within 30 bytes (e.g. via a service such as bit.ly). Normally, this might pose an issue as users may choose to ignore shortened links due to their opaque nature, but the identity guarantees of digital signatures should provide an extra layer of confidence for the user.

For example, suppose a user scans an SQR code on a parking meter attesting to a method of digital payment for parking fees. Once scanned, the user contacts a trusted authority and provides them with the public key and website pair contained within the SQR code.

If the pair has been previously registered by the user owning the corresponding public key, the SQR certificate authority returns the signature for the link, along with a user-readable identity (e.g. "Transportation Authority of Foobar City"), which is also signed with the private key of the code creator. The user can then say with a high degree of confidence that the link is safe and authentic regardless of the indirection and opacity involved with URL shorteners.

Naturally, this shifts some problems of trust onto the authority providing the user with the certificates. Similarly to SSL, we ensure a minimal attack surface, so that even in the case of a compromise, the minimum possible damage is incurred. Thus, SQR authorities storing the signatures do not store any private keys, only storing the websites and corresponding signatures created by an SQR code creator.

Crucially, this means that the SQR authority cannot impersonate an existing user by using their private key, although just as with SSL, a compromised SQR authority can create new identities for malicious purposes.

Potential attacks stemming from a compromised SQR authority and methods to alleviate these attacks are enumerated later on in our security analysis. In spite of these flaws, we expect that the SQR format should be a significant improvement over traditional QR codes from our addition of digital signatures.

4.2 Implementation

For our implementation, we focused on providing a minimal, but functional, version of the SQR format along with a website for scanning and creating SQR codes. Since there are no SQR certificate authorities running yet, we have also provided a user-friendly simulation of contact with an SQR certificate authority.

The details of the implementation follow directly from the design we have laid out in the previous section, but some details have been filled in. We have created a web-based front-end for the certificate authority which additionally generates QR codes for the user.

The website for the certificate authority additionally associates each public key with a username and password, allowing the user to sign multiple QR codes without re-entering their public key each time.

By default, the level of error correction in the QR code is medium (as specified by the QR code standard), which we have found to have an acceptable size for typical URLs. An

example is shown in Figure 3 with a target website pointing to a document hosted on Google Docs, which might be a typical use case for QR codes.

Note that QR codes are validated on a *per-user* rather than a *per-website* basis, since phishing can still take place through the use of otherwise safe websites. For example, a malicious user can use a Google Forms document to extract personal information through social engineering, even if Google Forms is a trustworthy website.

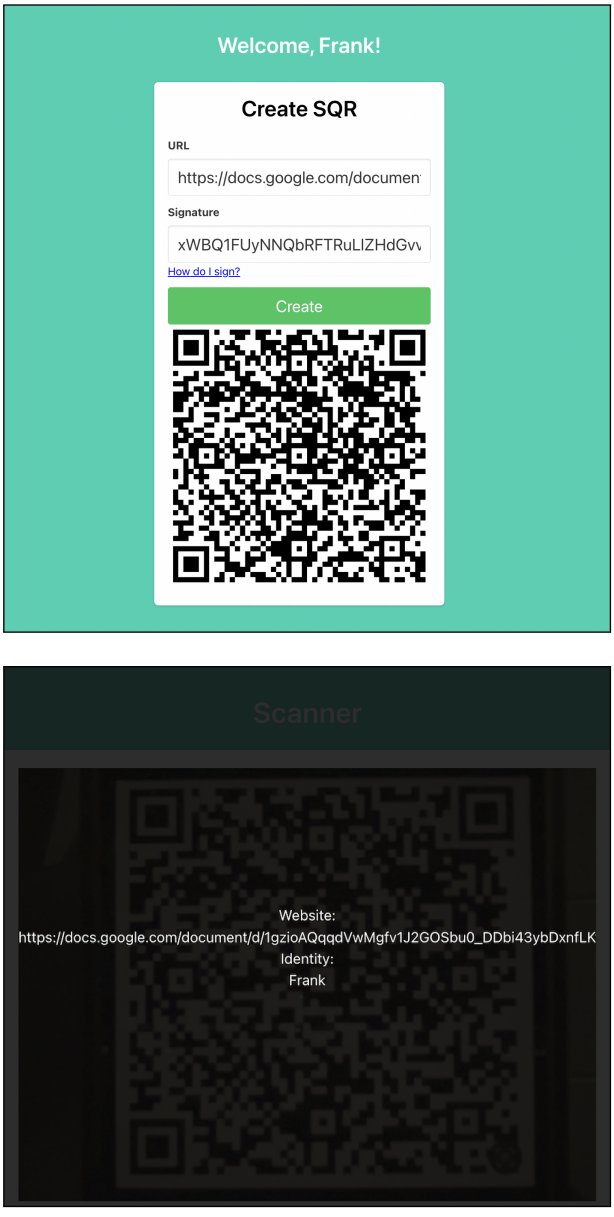

Figure 3: SQR for Google Docs

Website hosting services such as Netlify or Github Pages provide an additional vector of attack through otherwise-trustworthy websites. We also expect that most legitimate QR codes would be created by users other than the owner of the domain, so such a setup also increases the utility for the end user.

In addition to the SQR creation tool, we have provided a scanning utility on the same website, allowing users to scan and validate SQR codes. An example is shown in Figure 3 with the same QR code.

As laid out in the design, the site does not ask for or access the private key of the user at any point. Instead, the user is redirected to a site where they can run a script to generate their own ECDSA P-192 public and private keys.

The current implementation does not fully scope out the responsibilities of a certificate authority. For example, though users cannot create duplicate user-readable identifiers for their public key, the user can register any unused name as their identity, which might lead to untrustworthy users creating misleading identities to fool victims. As with the example of the parking meter from the previous section, a malicious user might register the name "Foobar City Transportation Authority", which closely mirrors the identity "Transportation Authority of Foobar City".

In a real world scenario, we would expect responsible certificate authorities to avoid storing signatures for untrustworthy clients. However, similarly to SSL, certificate authorities might eventually end up issuing signatures for untrustworthy clients.

If this became an issue, users could choose to whitelist only certain public keys on their own to avoid such attacks. An organization which wishes to avoid trusting any outside QR codes can give a whitelist to employees and prevent them from scanning any other QR codes. We leave this contingency plan unimplemented.

5 PERFORMANCE AND SECURITY ANALYSIS

5.1 SQR Speed and Size

As previously mentioned, the SQR format is built on top of the traditional QR format without altering its structure or introducing additional modules to any QR version. SQR simply encodes more information into the payload of a QR code. The resulting latency overhead of encoding a public key along with the url into a SQR code is incurred from the string concatenation of a url, a concatenator, and a public key. Such an operation is negligible in terms of added time overhead for the generation of SQR codes. However, scanning a SQR code does result in latency differences, as it necessitates payload parsing as well as authenticity and integrity verifications. Figure 4 illustrates the results of scanning done on SQR codes

and QR codes of varying URL length [1] and concludes that the SQR format adds an average of an 11.86 ms overhead to the traditional scanning process of QR codes. While this roughly represents a 52% increase over the average scanning time of 22.91 ms for QR codes, this overhead remains constant and does not scale with character count. Thus, an 11.86 ms overhead can be deemed negligible.

The primary overhead associated with SQR codes lies in the payload size of the encoding region. SQR codes must encode a trusted user's public key along with the URL they intend to encode. An ECDSA public key adds an additional 384 bits that need to be encoded. Depending on the size of the URL being encoded, this may result in fewer modules being available for error correction within the SQR code. We explore potential solutions to this issue in the Future Work section of this paper.

URL Character Count	SQR Scan Speed (ms)	QR Scan Speed (ms)
12	35.5	23.9
41	34	23.2
80	35.2	21
165	34.4	24
324	34	22.59
659	35.5	22.79

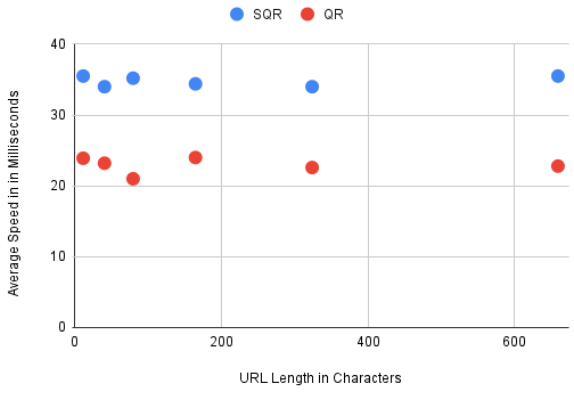

Figure 4: SQR and QR Code Scanning Speed

5.2 SQR Security

The main purpose of the SQR format is to provide added security on top of the traditional QR format to prevent previous and hypothetical attacks through QR codes. We go over the feasibility of these previously stated attacks when performed via SQR codes below and an additional attack on the SQR certificate authority.

[1]Scanning times represent average of scanning same URL over five trials

Attacker replaces whole SQR Code. This attack is usually performed in a physical setting as opposed to digitally. An attacker creates their own malicious QR code and then places this over a physical SQR code made for advertisement purposes by a trusted user via a physical vector such as a flyer or a pamphlet. This style of attack is preventable by SQR codes if the following conditions are met: the attacker is not a trusted user and does not have the ability to create SQR codes and the creator of the tampered SQR code advertises their trusted user identity on the physical vector. If these conditions are met, a user of a scanning device will not be presented with the identification of the trusted user when they scan the malicious QR code as QR codes do not provide this functionality. This should alert the user that the QR code they are scanning is not a SQR code and was not created by a trusted user. Thus, the code should not be trusted.

Attacker replaces encoding modules in SQR Code. This attack occurs when a malicious actor attempts to change the payload of a SQR code by changing its modules. This is not possible under the SQR format as the scanning device will verify the payload against the signature that corresponds to the specific SQR code for integrity. A false match will result in a failed scan.

Attacker compromises a certificate authority. This attack occurs when control of a certificate authority server is seized by an attacker. In this instance, an attacker may issue false identities and impersonate users using the user-readable identities (similarly to SSL). However, the compromised server cannot fully impersonate another user since they do not have control of their private key, and hence cannot sign QR codes using the same key.

The risks imposed by this attack cannot be fully mitigated, but we believe such an attack would be less damaging than an attack on an SSL certificate authority, since it does not allow for MITM attacks to be performed on a legitimate website. The risk imposed here is a reduction to the case of a normal QR code, for which it is already easy to impersonate a trustworthy party.

6 FUTURE WORK AND LIMITATIONS

The current implementation of SQR codes is still ripe with improvements that can further increase its security, efficiency, and breadth of application. We highlight these areas for improvement along with possible solutions below.

6.1 Decreasing Payload Size

As previously stated, SQR codes offer user authentication and payload integrity via a signed URL. This signature is authenticated by a user's public key, which is part of the SQR code payload along with the encoded URL. This addition of a ECDSA public key into the encoding area of a SQR code

results in a 384 bit overhead in the size of the SQR code payload as opposed to only encoding a URL into a traditional QR code. Most Search Engine Optimization guides recommend a URL of roughly 50-70 characters which corresponds to about 400-560 bits, thus, an overhead of 384 bits can increase a payload size by roughly 80%. Such an increase may result in less modules being available in a SQR code for error correction, which can lead to SQR codes that are unscannable when physically damaged or that are susceptible to physical attacks. This motivates some built-in payload compression for SQR codes.

We choose to focus on the compression of the URL within the payload and leave compression techniques for the public key to existing cryptographically strong techniques. Our proposed URL compression technique will make use of two rounds of compression, one via a conventional URL shortener and the second via a bit-encoding based approach. In the first round of compression, the raw URL to be encoded will be fed into a conventional URL shortener provider such as bit.ly. Alternatively, the SQR Certificate Authority may offer this URL shortening service.

We can hypothesize that this will reduce URL sizes to around 15 characters. This shortened URL will then be the input to the second round of compression that we will call Bit-Encoding (BE). BE makes use of the fact that URLs are made of four main parts: the protocol (e.g., HTTP or HTTPS), the domain/subdomain (e.g., docs.google), the top-level domain (e.g. ".com), and the path (e.g. "/document").

As there are a limited set of protocols, and top-level domains (TLDs), we can encode the protocol using 1 bit (to indicate HTTP[S]) and additionally encode 15 common TLDs for URL shorteners (such as .ly, or .le for bit.ly and goog.le) using 4 bits, with a sentinel value and variable-length encoding for uncommon TLDs.

Then, in the expected case, we can encode the path and protocol using only 5 bits. More bits can be allocated for the TLD if desired; a 16-bit value is more than enough to hold all currently supported TLDs, which would bring the bits required up to 17.

The domain, subdomain, and path can be compressed using a 6-bit encoding. The allowed characters in the domain and subdomain are as follows:

- Upper-case letters: [A-Z]
- Lower-case letters: [a-z]
- Digits: [0-9]
- Special characters: (18 total characters)

So, there are 80 total possible values. Though this is more than can be represented using 6 bits, special characters can be encoded using 64-bits as determined in RFC 4648 § 5, as commonly done in modern web browsers using percent-encoding.

Thus, for a URL such as https://bit.ly/Wn2Xdz, we can compress from a naive 147-bit ASCII representation (at 7 bits per character) down to a 73-bit BE representation, for a total size of 458 bits with an uncompressed key.

6.2 Developing Criteria for Trusted User

The SQR format relies on the assumption that only trusted, authenticated users can create SQR codes and that such users are less likely to engage in the creation of malicious or misleading SQR codes. The current SQR implementation is limited in that it does not take into account the characteristics of such a user and we leave this implementation of CA-led user-vetting to future work. To guide this implementation, we propose that a trusted user should fit the following criteria: *Established, Auditable* and *Secure*. This criteria borrows from elements used by web-based certificate authorities in their issuance of certificates to domains. We expand more below on the classification of each criterion.

Established. The most crucial characteristic of a trusted user in the SQR format is that a user be established. This trait refers to a user that is established as part of an official organization or institution, such as a university, a publicly traded company, or a municipality. Established status should also take into account the duration of existence of the organization or institution. Being established offers more sense of credibility to a trusted user and also ensures that all trusted users have some incentive to not engage in intentionally malicious actions to protect their organizational image. Most of these entities are also strongly obligated to abide by the law.

Auditable. Not only must a user be established, but they must also be auditable by the SQR certificate authority. Trusted user status should be revocable and may be canceled if a user creates a SQR code intentionally or unintentionally that results in malicious harm to scanners or if a user begins to engage in malicious actions through other forms of organizational operation. The latter case requires that a user be auditable and submit to investigation by the SQR certificate authority if deemed necessary. Not being auditable is a characteristic that may bar certain established users like private companies or confidential departments of government from being a trusted user.

Secure. As previously stated, intentional or unintentional malicious actions performed through SQR codes may result in the cancellation of a trusted user status. Unintentional malicious acts may result from security breaches of the operations of an established user. Therefore, an established user should also follow certain security standards in the operations they administer and these standards are subject to regular inspection by the SQR Certificate Authority.

6.3 SQR Codes For Any User

The current SQR format only allows SQR code creation by trusted users. Ideally, the SQR format should be modified so that payload integrity and user authentication is maintained regardless of who is able to create a SQR code. This is crucial future work as it allows SQR codes to have the same reach and application as QR codes and not only be available for creation by trusted users who will tend to be large, powerful organizations.

7 CONCLUSION

We have developed a method to prevent most "quishing" attacks via the use of embedded digital signatures and a trusted third party (a certificate authority). Although our method does not prevent all possible attacks, particularly in the case that the CA is compromised, we believe it will make quishing much more difficult and will prevent any QR code tampering or direct impersonation of a user with a given public key.

Further work on our method should look into reducing the size of the QR codes and formalizing a fair standard for acceptance of trusted organizations and individuals by CAs.

REFERENCES

[1] Adrian Dabrowski, Katharina Krombholz, Johanna Ullrich, and Edgar Weippl. 2014. QR Inception: Barcode-in-Barcode Attacks. *Proceedings of the ACM Conference on Computer and Communications Security* 2014. https://doi.org/10.1145/2666620.2666624

[2] Riccardo Focardi, Flaminia L. Luccio, and Heider A.M. Wahsheh. 2019. Usable security for QR code. *Journal of Information Security and Applications* 48 (2019), 102369. https://doi.org/10.1016/j.jisa.2019.102369

[3] Katharina Krombholz, Peter Fruehwirt, Peter Kieseberg, Ioannis Kapsalis, Markus Donko-Huber, and Edgar Weippl. 2014. QR Code Security: A Survey of Attacks and Challenges for Usable Security. 79–90. https://doi.org/10.1007/978-3-319-07620-1_8

[4] Katharina Krombholz, Peter Frühwirt, Peter Kieseberg, Ioannis Kapsalis, Markus Huber, and Edgar Weippl. 2014. QR Code Security: A Survey of Attacks and Challenges for Usable Security. In *Human Aspects of Information Security, Privacy, and Trust*. Springer International Publishing, Cham, 79–90.

[5] Heider Wahsheh and Flaminia Luccio. 2019. Evaluating Security, Privacy and Usability Features of QR Code Readers. 266–273. https://doi.org/10.5220/0007346202660273