Riccardo Focardi, Flaminia L. Luccio, Heider A.M. Wahsheh,
Usable security for QR code,
Journal of Information Security and Applications, Volume 48, 2019,
102369, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2019.102369

Usable Security for QR Code ☆

Riccardo Focardi[a], Flaminia L. Luccio[a,*], Heider A. M. Wahsheh[a]

[a]*DAIS, Università Ca' Foscari Venezia, via Torino 155, 30170, Venezia, Italy,*

Abstract

QR codes are widely used in various settings such as consumer advertising, commercial tracking, ticketing and marketing. People tend to scan QR codes and trust their content, but there exists no standard mechanism for providing authenticity and confidentiality of the code content. Attacks such as the redirection to a malicious website or the infection of a smartphone with a malware are realistic and feasible in practice. In this paper, we present the first systematic study of *usable* state-of-the-art cryptographic primitives inside QR codes. We select standard, popular cryptographic schemes and we compare them based on performance, size and security. We conduct tests that show how different usability factors impact on the QR code scanning performance and we evaluate the usability/security trade-off of the considered cryptographic schemes. Interestingly, we find out that in some cases security breaks usability and we provide recommendations for the choice of secure and usable cryptographic schemes.

Keywords: QR codes; Usable Security; Cryptography; Digital Signature; HMAC.

1. Introduction

A barcode is a machine-readable image that represents data in parallel lines (one-dimensional, 1D, barcode), or as dots or lines that are arranged

☆This paper is an extended and revised version of Focardi et al. (2018b).

*Corresponding author

Email addresses: `focardi@unive.it` (Riccardo Focardi), `luccio@unive.it` (Flaminia L. Luccio), `heider.wahsheh@unive.it` (Heider A. M. Wahsheh)

URL: `http://dsi.unive.it/~focardi` (Riccardo Focardi), `http://dsi.unive.it/~luccio` (Flaminia L. Luccio)

in matrix form (two-dimensional, 2D, barcode). Quick Response (QR) codes are the most widely used 2D barcodes in the marketing world, in education and in public services. Users believe that they are simple to use and useful (Vidas et al., 2013). They are also the barcodes with higher data capacity (Dabrowski et al., 2014), and may store different types of data, such as numeric, alphanumeric, binary and Kanji characters (ISO/IEC Standard, 2015).

Barcodes are used in various scenarios for different purposes. A typical application is to encode a URL that links to a related Web page containing detailed information about a product or a service. When the barcode is scanned, the link is usually shown to the user that can decide whether to open it or not in the browser. Barcodes are also used for physical access control, identification and logistics. In these cases, they contain data that are given as input to back-end applications, which interpret them and act consequently.

In general, barcodes are just a way to provide input to users or applications and, since they do not offer any standard way to guarantee content authentication, the input they provide is in fact, *untrusted*. Potential security risks regard the encoding of malicious URLs that look similar to the honest ones and the encoding of data that trigger vulnerabilities in the back-end applications. Moreover, the barcode reader application may become a point of attack since, independently of the use case, the barcode content passes through it and might trigger vulnerabilities directly on the user device.

In September 2011 the first malicious usage of QR code was detected by the KasperSky Lab (Kaspersky Lab, 2011). The attack was performed using a malicious link that was encoded in a QR code: the users were obliviously directed to a Web page, where a malware was unconsciously downloaded in the connecting device. In general, attacks can target barcode scanning devices (e.g., smartphones) by reaching sensitive information such as passwords, contact information, photos, videos, credit card numbers, etc., and can thus violate the users' privacy. Attackers may also take full control of mobile devices by, e.g., accessing E-mail, SMS, etc. (Kieseberg et al., 2012).

In the last years there has been a large increase in the use of barcodes in everyday life, thus preventing attacks is a necessary and challenging issue. In the literature, there are some proposals and tools to improve QR code security but none of them justifies the architectural choices and the usage of the underlying cryptographic scheme, and often the adopted schemes are vulnerable or deprecated. Focardi et al. (2018a) provides a comprehensive

discussion on existing QR codes security mechanisms.

In this paper, we present the first systematic study of *usable* state-of-the-art cryptographic primitives inside QR codes. Following European Union Agency for Network and Information Security (ENISA) (2014) guidelines, we select well studied and standard cryptographic schemes and we compare them based on performance, size and security. We conduct tests that show how different usability factors impact on the QR code scanning performance, and we evaluate the usability/security trade-off of the considered cryptographic schemes. Our results show that secure QR codes can be used in practice, but schemes with big size overhead might rise usability issues. Moreover, secure QR codes are denser and cannot be printed on small areas without compromising usability. In particular, we show that in some cases providing a high degree of security breaks usability, and we provide recommendations for the choice of secure and usable cryptographic schemes. We have implemented a proof-of-concept Android app that confirms our findings. In particular, when the scheme and the printing size are chosen appropriately with respect to usability constraints, the QR codes can be scanned without affecting the user experience.

1.1. Contributions

Our contributions can be summarized as follows: (*i*) we survey attacks on QR codes and discuss the potential benefits of enhancing them with cryptography; (*ii*) we present the results of extensive experiments to determine the impact of usability factors on QR code scanning; (*iii*) we analyze the time and space overhead of a selected set of cryptographic schemes, with various key sizes and formats; (*iv*) we evaluate the cryptographic schemes with respect to the usability analysis.

1.2. Related work

In Ishihara and Niimi (2014), it is proposed a tamper detection system for QR codes based on steganographically embedding a digital signature into the error correcting area. However, the paper appears preliminary as it only considers small sizes for signatures x and the embedding of actual signatures is left as a future work. In Razzak (2012), the Elliptic Curve Digital Signature Algorithm (ECDSA) is used to digitally sign barcodes. Experimental results on different key lengths and hash functions for ECDSA show a reasonable space overhead but, with respect to our paper, no comparison with

other signature schemes is done and there are no considerations about usability. The reported time overhead might also, by itself, break usability. We show that with modern smartphones time is not anymore, an issue. In Peng et al. (2014), a group of students from MIT have performed preliminary experiments about enhancing QR codes with cryptography and digital signature.

We also find proposals that are not based on security enhanced QR codes. For example, the study of Yao and Shin (2013) investigates the security features of existing QR code scanners for preventing phishing and malware propagation. Authors propose a new scanner, named SafeQR, based on two existing Web services: the Google Safe Browsing API (Google, 2019) and the Phishtank API (Phishtank, 2019). There exist some commercial solutions for secure QR codes, e.g. (2D Technology Group Inc., 2016; Yakshtes and Shishkin, 2012), but there is no publicly available description of the proprietary technology which prevents us to perform any security analysis.

1.3. Paper Structure

The rest of the paper is organized as follows: in Section 2 we present well known attacking scenarios and discuss which are prevented by adopting digitally signed QR codes. Section 3 analyzes the performance of QR code scanning with respect to different usability factors, and discusses the results of different usability tests. In Section 4, we analyze the overhead of time and space introduced by the addition of cryptographic primitives inside QR codes, using different standard formats, and we evaluate the cryptographic primitives with respect to usability. Section 5 draws some concluding remarks.

2. Security of cryptographic QR codes

In the following, we consider the most prominent attack scenarios for QR codes and discuss in which extent cryptographic QR codes prevent them.

Phishing. In a barcode phishing attack, the attacker tries to get sensitive information such as the login details and the credit card number of a user by, e.g., encoding a malicious Web address inside the barcode that redirects the user to a fake Web page which appears very similar to the legitimate one (Kieseberg et al., 2012; Krombholz et al., 2014; Vidas et al., 2013). This is a typical case of social engineering attack (Krombholz et al., 2013).

Malware propagation. In Kharraz et al. (2014) it is discussed how QR codes can be used by attackers to redirect users to malicious sites that silently install a malware by exploiting vulnerable applications on the device. This is typically done through an exploit kit that fingerprints the device and selects the *appropriate* exploit and malware.

Barcode tampering and counterfeiting. Since QR codes can be used to provide information about a good, an attacker can benefit by pasting a fake QR code so to advertise false products information or false special offers in which the adversary will sell another product to the victims (Dabrowski et al., 2014; Kieseberg et al., 2012). When barcodes are used to identify and track items, such as in a supply chain, barcode tampering might enable an attacker to counterfeit a product.

Barcode-in-barcode attacks. It is a special case of tampering in which some pattern in a barcode image corresponds to another valid barcode (Dabrowski et al., 2014). The QR code will be scanned normally most of the times but in some cases the secondary barcode will be decoded, triggering potentially dangerous situations. This is somehow related to decoding a file format differently in two different applications (Albertini, 2014). Interestingly, we discovered that this barcode-in-barcode decoding might happen by chance, especially when barcodes become dense (cf. *Wrong* scans of section 3).

SQL and command injections. The study of Kieseberg et al. (2012) refers to automated systems using the information encoded in the barcodes to access a relational database. If the string in the barcode is appended to the query without proper sanitization, the attacker may easily trigger a SQL injection attack.

Cross-site scripting attacks (XSS). Mobile apps are often based on Web technology and this may allow malicious JavaScript code to be injected into trusted HTML pages, and executed in the app, for example when the server does not sanitize the user data that is rendered in a page (Jin et al., 2014).

Reader applications attacks. During the installation process, many barcode reader applications ask for full permissions to access user's resources such as the device location, the contact list and the photos. In case of a vulnerability that can be triggered by a suitable crafted barcode, the attacker would get access to private user's data Kieseberg et al. (2010).

2.1. Discussion

Enhancing QR codes with cryptography may prevent the above attacks, assuming that the attacker cannot forge the cryptographic material. For example, a digital signature or a Message Authentication Codes (MAC) of the QR code content would allow to verify its origin. In both cases, it is necessary that the verification key is known to the barcode reader. In an open environment this can be hard to achieve, since a Public Key Infrastructure (PKI), similar to the one for the HTTPS protocol, would be vulnerable to the "HTTPS phishing problem", i.e., attackers that have a valid certificate and use names similar to the one of legitimate entities (Wired, 2017). Moreover, MACs are based on symmetric keys that are known to be hard to deal with in an open environment. In fact, the reader should share the MAC symmetric key with the QR code generator. However, in a closed/controlled environment, the reader might be configured to only recognize internal certificates or MAC keys, and verifying the signature/MAC would prove the trustworthiness of the QR code content. For example, entities of a supply chain might share their public keys in order to check the trustworthiness of the QR codes used to identify and track products. This would prevent counterfeiting and any other type of attack targeting reader and back-end applications that might disrupt and/or take control of the system as, e.g., reader application attacks and SQL injections.

In many situations it is useful to encrypt the QR code content so to protect its confidentiality. Consider again the example of a supply chain. If QR codes contain the product ID which is also physically stamped on the product, an attacker might reuse a signed barcode (e.g., collected from a box) by simply stamping the expected ID on the counterfeited product. A solution to this problem is to encrypt the barcode payload so that it does not leak any sensitive information. Interestingly, this would implicitly prevent the above-mentioned attacks as barcode content decryption would give random bytes in case of an attacker trying to forge a valid encrypted QR code without knowing the encryption key. Using authenticated encryption schemes (such as the Galois Counter Mode, GCM) would add the additional feature of making decryption fail in case of a forging attack.

3. Usability of QR code scanning

In order to evaluate the usability of QR codes, we have developed an Android app, called *BarTest*, that automatically collects user feedback about

the scanning experience on a proposed set of test barcodes.

After each successful scan, the user can express her subjective satisfaction:

Questionnaire 1 (User evaluation). *User expresses her subjective evaluation about successful scans, through the three-level scale:*

1. *Excellent;*
2. *Good;*
3. *Bad.*

Notice that, our scale differs from a standard Likert scale as it does not have a neutral point and it is asymmetric providing only one negative option. In particular: (*i*) we preferred not to have a neutral point in order to force users to express an opinion; (*ii*) we did not want users to choose between different negative evaluations, as negative would be equivalent to unusable, but we kept two positive evaluations in order to judge the degree of usability.

In case the answer is different from *Excellent*, the user is also asked to provide more feedback.

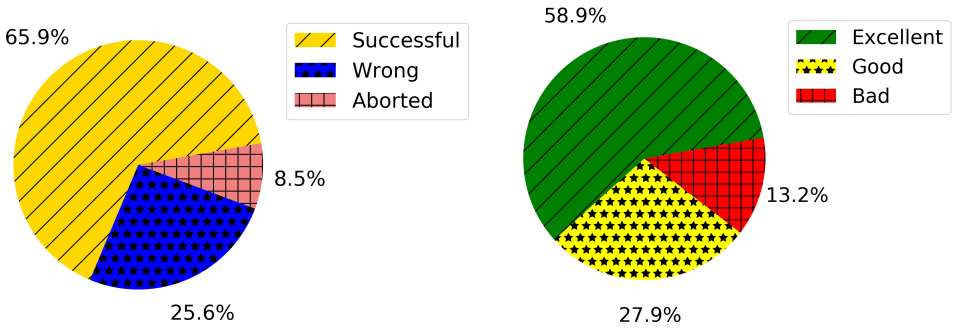
Questionnaire 2 (User feedback). *When user satisfaction is different from Excellent, the user is also asked to provide more feedback by selecting one or more of the following choices:*

1. *I had to move the phone too much;*
2. *Scanning took too much time;*
3. *I had to rescan the barcode too many times;*
4. *Other (filled by the user).*

There might be a risk that not collecting feedback for excellent answers would incentive users to pick that choice in order to simplify their task. We believe that this has not affected the study for two reasons: (*i*) users did not know in advance that feedback was not asked in that case; (*ii*) feedback was relatively simple to give, except for option 4 (*Other, filled by users*), and lazy users had the chance to answer quickly just by selecting options 1, 2 or 3 with a simple click. Moreover, we believe that asking a feedback for excellent answers would have looked artificial and superfluous with the risk of loosing users' engagement in the task.

Together with the subjective user evaluation, *BarTest* also records information about the scanning outcome:

***Successful* scan** the barcode is successfully decoded, and user satisfaction is collected through Questionnaire 1;



(a) Scanning outcome of experiments. (b) User evaluation for *Successful* scans.

Figure 1: Summary of outcome and evaluation.

***Wrong* scan** the barcode is incorrectly decoded, e.g., when some pattern in a barcode image corresponds to another valid barcode (Dabrowski et al., 2014). This is automatically detected by *BarTest* since test QR codes contain a known payload together with the barcode ID. Whenever the scanned payload does not match the expected one the application declares it as *Wrong*. In this case, the user is offered to rescan the barcode or abort (see below);

***Aborted* scan** the user cancels the scan, e.g., when scanning takes too much time or gives wrong results. The user is asked for a feedback, through Questionnaire 2

Interestingly, *BarTest* automatically collects statistics about scanning and abort time for each barcode. This improves on previous work (Focardi et al., 2018b) by providing data that can be used to correlate scanning efficiency and effectiveness with respect to the user subjective evaluation and feedback (cf. Section 3.7).

3.1. Experimental Setup

The barcodes proposed to users are of four different sizes: 200×200, 300×300, 400×400, 500×500 pixels that, visualized on a 96 DPI screen, correspond to 5.29×5.29, 7.93×7.93, 10.58×10.58, 13.22×13.22 centimeters. The idea is to cover both barcodes that can be printed on small areas, e.g., in products, and bigger ones that might be printed on advertisements, bus stops, etc. For readability, in the following we will always refer to the size

in pixel (assuming implicitly 96 DPI) but, of course, what matters in the experiment is the actual size. The generated barcodes contain random data of various sizes, multiple of 100, from 100 to 2000 bytes. We adopted the most recent version of QR codes (model 2) with an error correction capability of level M (approximatively 15%), which is the most frequently used.[1]

When conducting the experiments there was a risk of having the users judging app usability instead of scanning usability. For this reason, we carefully designed our experimental setup making the following choices: (*i*) we designed our app so to have a simple, intuitive control-path. We double-checked that all the obvious choices were covered (e.g., letting the users press the back button in order to abort a scanning); (*ii*) we conducted the experiments with first and second year computer science undergraduate students. This ensured that users did not encounter big issues while installing and using the app, letting them concentrate on the barcode scanning. It might be objected that computer science students could be more skilled at scanning barcodes than a normal user of the same age, but in fact we believe that there is no technical skill related to that, and any person confident with a smartphone camera would have accomplished the task in the same way. Moreover, in our opinion, this represents well the population of users that would mostly be interested in scanning barcodes; (*iii*) we asked students to run the experiments before class so that they had the chance to take their time and ask us questions in case they encountered technical difficulties about how to install or start the app. We also did not impose any time restriction.

The experiments were conducted with the help of 149 users,which is a big enough population size for usability studies. Each user was asked to scan four barcodes printed on A4 paper using the *BarTest* app, for a total of 596 QR codes. The four barcodes were picked at random from the above described set.

3.2. Scanning outcome

Figure 1a shows the overall outcome percentage of the scanning experiments. The users *Aborted* only in 8.5% of cases, while in the remaining 91.5% of the cases, they obtained 65.9% *Successful* and 25.6% *Wrong* scans. The high percentage of *Wrong* scans highlights the possible risk of the so-called *misleading* problem, i.e., patterns in a barcode image that correspond to another valid barcode (Dabrowski et al., 2014)).

[1] `https://www.qrcode.com/en/about/error_correction.html`

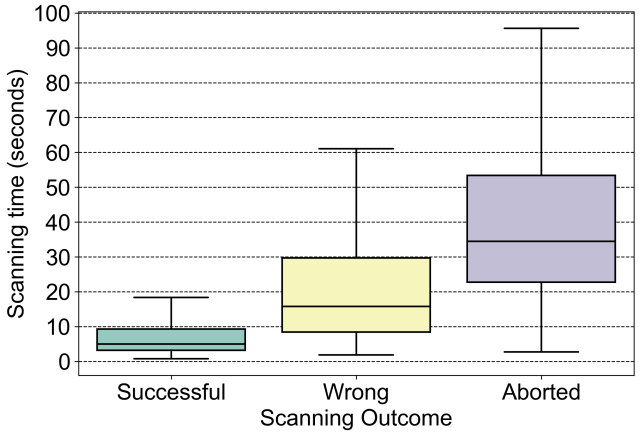


Figure 2: Scanning time for different outcomes.

The *Successful* percentage, which represents about two thirds of the global scans, is divided into: *Excellent* (58.9%), *Good* (27.9%) and *Bad* (13.2%) scans, as shown in Figure 1b.

3.3. Scanning time

Figure 2 presents the boxplot of the scanning time for the various outcomes. We can observe that, there is a clear correlation between the scanning time and the outcome. In particular, we notice that *Aborted* scans are related to long scanning time. This can be easily understood by considering that users explicitly abort scans when time becomes longer than their expectation. Moreover, *Wrong* scans take longer than *Successful* scans. A plausible justification is that when barcodes are complex and contain patterns that correspond to other valid barcodes, the reader first tries to decode the complex barcode and, only when it fails, looks for other types of barcodes.

The scanning time distribution for the three user evaluation levels is reported in the boxplots of Figure 3. It is evident how smaller scanning times are related to better user experiences. We notice, in particular, that *Excellent* readings are more centered around their median point 4.1 seconds and the majority of them recorded less than 10.6 seconds delay, with some outliers (16 scans out of 209) that exceed 10.6 seconds. Moreover, *Good* and *Bad* readings show minimum values close to the *Excellent* scans, but with a wider range of points distribution. The majority of *Good* scans recorded less than 28.9 seconds with some outliers (9 scans out of 99). The *Bad* maximum

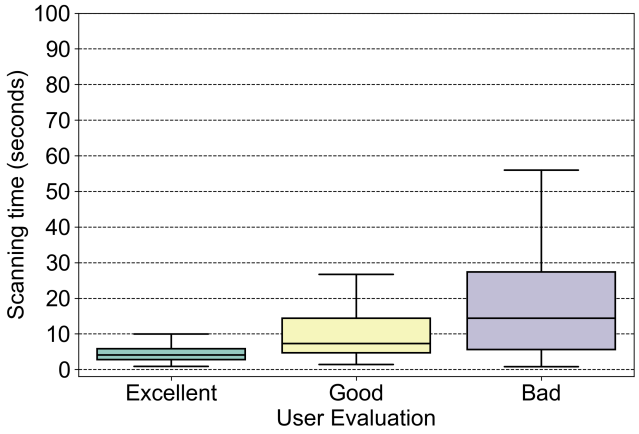


Figure 3: Scanning time for different user evaluations.

non-outlier reached 56 seconds with few outliers (2 scans out of 47). The distinction between *Good* and *Bad* evaluations is thus not so neat as the one between *Excellent* and *Good*.

Interestingly, the median scanning time for *Successful* scans (5 seconds, cf. Figure 2) is close to median of *Excellent* (4.1 seconds), and well below the median of *Good* (7.3 seconds). This is consistent with the intuition that users typically give *Excellent* or *Good* to scan that are immediately *Successful* and *Bad* when barcodes need to be re-scanned (e.g., with *Wrong* outcome).

3.4. Scanning outcome vs. data size

Figure 4 presents the frequency of the scanning outcomes for the 300×300 pixels image size. The data sizes are grouped into five ranges of 100-400, 500-800, 900-1200, 1300-1600 and 1700-2000 bytes, respectively. The X-axis represents the data size groups, while the Y-axis represents the outcome percentage. The summation of *Successful*, *Wrong* and *Aborted* percentages is 100%.

The figure clearly shows that the fraction of *Successful* outcomes decreases when the data size increases, in favor of increasing *Wrong* and *Aborted* outcomes. In particular, the *Successful* outcome percentage is the highest (100%) for the minimum data size group, i.e., barcodes with 100-400 bytes, while the maximum data size group, i.e., barcodes with 1700-2000 bytes, yields 18.2% *Successful*, 59.1% *Wrong* and 22.7% *Aborted* outcomes. We conclude that, when the barcode becomes denser the probability of getting

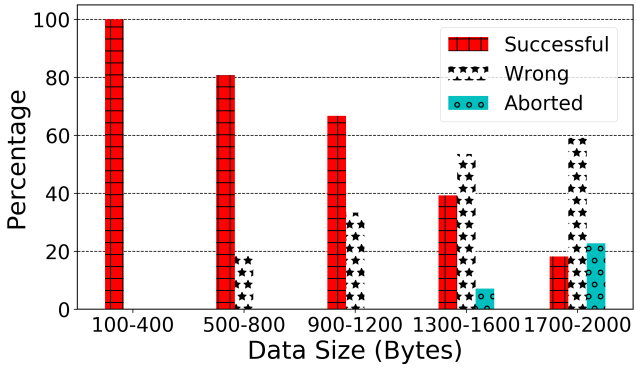


Figure 4: Outcome percentage for 300×300 pixels image size.

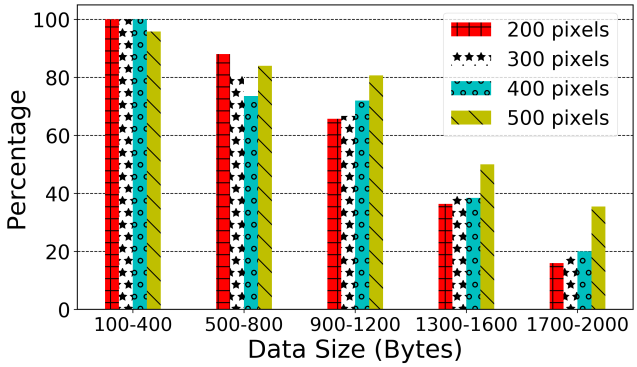


Figure 5: *Successful* percentage for all image sizes

a wrong result increases and, when getting the correct result becomes too hard, users start aborting the scanning. This, of course, points out a usability problem for dense barcodes.

Figure 5 shows how the fraction of *Successful* is affected by density for the different barcode sizes. We notice that for dense barcodes, bigger sizes increase the number of *Successful* outcomes. In fact, bigger barcodes will be less dense than smaller ones, when the data size is the same. For small data size groups (100-400 and 500-800) we notice that bigger sizes give less *Successful* scans in some cases. This might be justified by some external factors that make the scanning of bigger barcodes more difficult. For example, when the barcode is very big the user needs to scan it from a bigger distance so to capture the whole image. In fact, barcodes were provided on an A4 sheet of

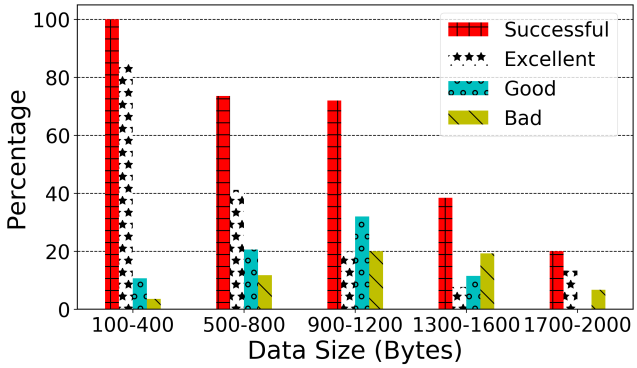


Figure 6: User evaluation of *Successful* outcomes for 400×400 pixels and different data size groups.

paper and scanning a big barcode might have required the user to stand-up in order to have a reliable scanning (cf. Section 3.6).

3.5. User evaluation vs. data size

Figure 6 shows the users' evaluation for barcodes with 400×400 pixels. The X-axis represents the data size groups, while the Y-axis represents the percentage. The *Successful* percentage (red bar) is divided into three subgroups, depending on user evaluation: *Excellent*, *Good* and *Bad*. For example, the group of 100-400 bytes recorded 85.7% *Excellent*, 10.7% *Good* and 3.6% *Bad*. In the group of 900-1200 bytes, instead, the percentage was 20%, 32% and 20% for *Excellent*, *Good* and *Bad*, respectively (out of 72% which is the *Successful* percentage). Generally, user evaluation is worse when data size increases and *Successful* rate decreases.

3.6. User feedback

We analyze user feedback for non-*Excellent* scans, collected through Questionnaire 2. We observe that:

- *I had to move the phone too much* was chosen in 26.4% of the answers. This resembles the *Readability Range*, introduced in (Focardi et al., 2018b) to measure the minimum and maximum distance inside which the QR code is readable: a small Readability Range requires users to excessively move the phone in order to scan the barcode. Figure 7 shows the measured Readability Range for 500 × 500 pixels (Focardi

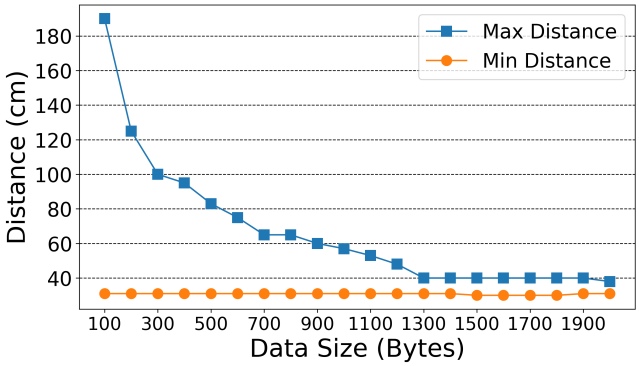


Figure 7: Measuring the Readability Range.

et al., 2018b), which decreases when the data size increases reducing usability. However, in the present study, we could not find a clear correlation between this answer and the image and data sizes. We speculate that the necessity of moving the phone might be influenced by additional external factors, such as light and physical position of the user with respect to the barcode.

- *Scanning took too much time* was chosen in 39.1% of the answers. It is interesting to analyze the user subjective evaluation of scanning time. Figure 8 plots the scanning time for users that did not complain about scanning time (boxplot on the left) with the median around 5.7 seconds, and users that answered with *Scanning took too much time* with the median around 23.34 seconds (boxplot on the right). Interestingly, the first quartile of the rightmost boxplot is around 10.13 seconds that corresponds to the 10 seconds threshold between what was considered reasonably readable and hardly readable in (Focardi et al., 2018b).

- *I had to rescan the barcode too many times* was chosen in 50.6% of the answers. We have analyzed the rescan results as shown in Figure 9. In total, 48.7% of rescan attempts ended with *Successful* decoding (even with several rescans), divided into *Excellent* (12.9%), *Good* (17.5%) and *Bad* (18.3%) scans. The remaining rescan attempts ended with *Wrong* scans in 42.4% of the cases, and with *Aborted* scans in 8.9% of the cases. Rescanning decreases by itself usability and our study shows that, in about half of the cases, it does not solve the scanning issue

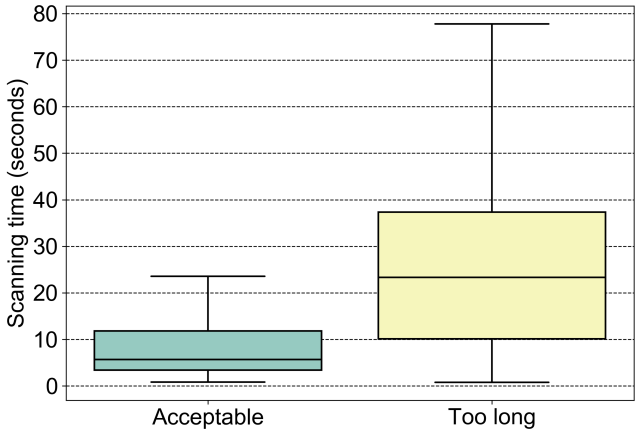


Figure 8: Acceptable scanning time, based on user's selection of *Scanning took too much time* in Questionnaire 2.

providing wrong results or aborts.

- Only 6.9% of the answers contained personal comments from users. Some regarded the environment, confirming that the light in the room can affect the scanning task and the surface where the QR code is printed can affect the scanning outcome, as already discussed in (Grover et al., 2010). Other comments regarded the difficulty in scanning very big barcodes that might require to move the phone far from the sheet of paper. Some users got confused by *Wrong* outcomes, confirming that this case reduces usability.

3.7. Barcode Usability

We define QR code usability based on ISO 9241 (ISO, 2018) through Effectiveness, Efficiency, Satisfaction, defined below.

Definition 1. *Effectiveness (Alturki and Gay, 2017) is the success ratio, defined formally as:*

$$Effectiveness = \frac{n.\ of\ successful\ tasks}{n.\ of\ undertaken\ tasks}$$

In our experiments, the successful tasks are the scans with *Successful* outcome and the total tasks undertaken are all the performed scans, i.e., *Successful*, *Wrong* and *Aborted* ones.

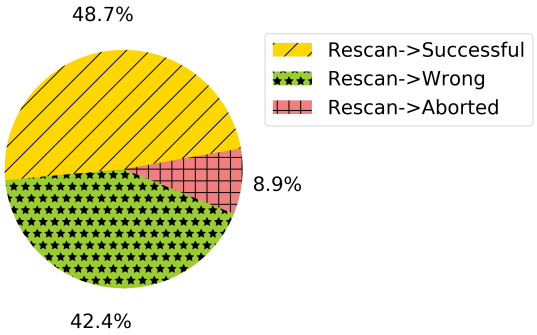


Figure 9: Rescan Impact.

Inspired again from (Alturki and Gay, 2017) we give the following definition of Efficiency:

Definition 2. *Efficiency is the relative time for accomplishing successful tasks, defined formally as:*

$$Efficiency = \frac{time\ for\ successful\ tasks}{time\ for\ all\ tasks}$$

Definition 3. *Satisfaction (A. Sergeev, 2010) is the user comfort in terms of simplicity to perform the scanning, determined by users' answers.*

$$Satisfaction = \left(\frac{\sum_{j=1}^{U}\sum_{i=1}^{Q_+} P_{ij}^{+} + \sum_{j=1}^{U}\sum_{i=1}^{Q_-} P_{ij}^{-}}{(Q_+ \times Q_-) \times U}\right)$$

Where:

- U*: represents number of users;*
- Q_+*: the number of positive answers;*
- Q_-*: the number of negative answers;*
- P_{ij}^{+}*: positive weight for the answer to a positive question;*
- P_{ij}^{-}*: negative weight for the answer to a negative question.*

In our setting, positive answers Q_+ are the ones of Questionnaire 1 while the negative answers Q_- are the ones of Questionnaire 2. We assigned the following weights: *Excellent* 3/3, *Good* 2/3 and *Bad* 1/3, for Q_+ and $neg/4$, where neg is the number of negative replies, for Q_-.

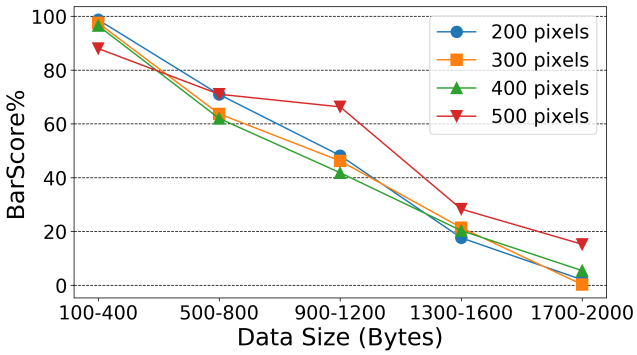


Figure 10: Barcode Usability Score (*BarScore*).

Definition 4. *The usability score for barcodes (BarScore) is defined as:*

$$BarScore = (Effectiveness + Efficiency + Satisfaction)/3$$

Fig. 10 plots the *BarScore* values for the tested image and data sizes. The X-axis represents the data size groups, the Y-axis represents *BarScore* and each line represents a specific image size.

We consider three usability levels:

- High Usable level (H): $BarScore \geq 80\%$.
- Low Usable level (L): $60\% \leq BarScore < 80\%$.
- Unusable level (U): $BarScore < 60\%$.

Figure 11 summarizes the resulting usability levels for barcodes.

4. Usable cryptographic QR codes

We consider selected symmetric and asymmetric key cryptographic primitives provided by standard Android smartphone libraries, that might be used to provide data confidentiality, integrity and authenticity. Augmenting the QR code payload with cryptographic material makes the barcode denser and harder to read. We study in detail the usability of adopting such cryptographic primitives based on the barcode usability result of previous section.

In particular, in Section 4.1 we describe the primitives and in Section 4.2 we study their time overhead; in Section 4.3 we discuss space overhead of embedding the cryptographic data together with the QR code payload; finally, in Section 4.4 we summarize the usability of QR codes with the various cryptographic algorithms and key sizes.

Data Size (Bytes)	200 Pixel	300 Pixel	400 Pixel	500 Pixel
100-400	H	H	H	H
500-800	L	L	L	L
900-1200	U	U	U	L
1300-1600	U	U	U	U
1700-2000	U	U	U	U

Figure 11: Summary of usability levels.

4.1. Cryptographic primitives

Digital signature. We consider the two most commonly used digital signature algorithms: RSA with key lengths 1,024 bits, 2,048 bits and 3,072 bits and Elliptic Curve Digital Signature Algorithm (ECDSA) with key length of 256 bits. We use SHA-256 as hash function. For new applications, European Union Agency for Network and Information Security (ENISA) (2014) recommends a key length of 3,072 bits for RSA and of 256 bits for Elliptic Curve, so we will consider RSA 1,024 as low-secure, 2,048 as medium-secure, and RSA 3,072 together with ECDSA 256 as high-secure. However, it is worth noticing that ENISA recommends adopting only certain variants of RSA and ECDSA for new applications, i.e., the ones provided with a security proof in a strong computational model. We believe that, size and performance are not significantly affected by picking a specific variant. We thus report on results achieved using the default implementations offered by the cryptographic library, and we leave as future work a comparison between the different variants of the signature algorithms. The interested reader can refer to (European Union Agency for Network and Information Security (ENISA), 2014) for more detail.

Message Authentication Codes (MACs). We consider Hash-based Message Authentication Code (HMAC), a standard mechanism for achieving authentication and integrity under a symmetric key. HMAC is based on a secure

hash function and can be proved secure if the hash function is a pseudo random function. In this study we considered standard implementations based on SHA-256, SHA-384 and SHA-512, which provide strong security (European Union Agency for Network and Information Security (ENISA), 2014).

Symmetric key cryptography. We consider Advanced Encryption Standard (AES) under four standard modes: Cipher Block Chaining (CBC), Output Feedback (OFB), Cipher Feedback (CFB) and Galois/Counter Mode (GCM). AES supports three key lengths: 128, 192 and 256 bits. AES-128 is considered secure for future applications (European Union Agency for Network and Information Security (ENISA), 2014), but it is worth noticing that time overhead for symmetric cryptography is negligible and, since the block size is 128 bits independently of the key length, there is no practical difference in adopting AES-192 or AES-256.

4.2. Time overhead

We have developed an Android mobile application to test time overhead for the slowest cryptographic primitive considered: signature verification. The tests have been performed on an Android smartphone with 1.2 GHz dual-core CPU, 1 GB of RAM. We found that digital signature verification took, on average, between 20 and 30 milliseconds for the various key lengths and algorithms, posing no usability problem. The time overhead for HMAC and symmetric cryptography is orders of magnitude smaller than the one for digital signature, so we can safely conclude that time is not problematic for any of the considered cryptographic primitive. However, size can be critical with respect to usability in some cases, as we will discuss in Section 4.4.

4.3. Space overhead

We need to embed at least the following information:

Payload The actual data that we want to load in the QR code. It can be offline information requiring no Internet connection, or an URL referencing to an external resource. Payload is encrypted when using symmetric cryptography;

Generator The identity of the QR code generator;

Algorithm The cryptographic mechanisms adopted;

Digital signature	Signature size	without certificate	with certificate
ECDSA 256	64	118	365
RSA 3,072	384	436	1327
RSA 2,048	256	308	943
RSA 1,024	128	180	555

Table 1: Control data overhead (bytes) for digital signatures.

Auth. & integrity	HMAC size	Overall size
HMAC 256	32	109
HMAC 384	48	125
HMAC 512	64	141

Table 2: Control data overhead for HMAC (bytes).

Signature/HMAC The digital signature or HMAC;

Certificate The certificate of the QR code generator. This applies to digital signatures and can be included in the barcode or referenced through an URL.

We use JavaScript Object Notation (JSON) JSON (2016). In principle it would be possible to adopt ad-hoc, less verbose, formats but at the price of a less reliable encoding and decoding. So, even if there is margin for improvement, we preferred to adopt a standard format in our study.

Table 1 reports the overhead with or without certificates for digital signatures. The first column reports the size in byte of the signature without control data. Notice that, ECDSA signature length is twice the size of key length, i.e., $256 \times 2 = 512$ bits $= 64$ bytes, while RSA signature length is equal to the key length. Table 2 shows authentication and integrity control data overhead for HMAC with three different block lengths: 256, 384 and 512 bits. Table 3 shows the overhead for AES in various modes. Notice that, CBC requires padding to reach a multiple of the 16 bytes block size and GCM contains extra data to achieve authenticated encryption. Intuitively, GCM provides, at the same time, confidentiality and data authentication/integrity.

AES	Overall size
CBC	69[a]
OFB	69
CFB	69
GCM[b]	85

[a] requires padding to reach a multiple of 16
[b] guarantees both authentication and data integrity

Table 3: Control data overhead for AES different modes (bytes).

4.4. Usability evaluation

Crossing Figure 11, Table 1, Table 2 and Table 3, we obtain Table 4 which summarizes the usability of cryptographic barcodes, assuming up to 200 bytes for the payload.

(European Union Agency for Network and Information Security (ENISA), 2014) recommends a key length of 3,072 bits for RSA and of 256 bits for Elliptic Curve, so we will consider RSA 1,024 as low-secure, 2,048 as medium secure, and RSA 3,072 together with ECDSA 256 as high secure. According to Table 4, we recommend ECDSA for high usable/secure digital signature scheme. Including a certificate in the QR code gives usability problems, suggesting that using on-line certificates, downloadable via HTTPS and cached by the application might offer an appealing alternative. In fact, when certificates are included, we observe usability issues for all of the experimented signature schemes.

HMAC is highly usable and secure and might provide a suitable alternative to digital signatures for authentication and integrity. Of course, since HMAC uses symmetric keys, its adoption requires suitable key management mechanisms that allows for distributing such shared secrets in a secure way. HMAC is typically used in closed, corporate applications where all readers share the symmetric key.

AES is highly usable and secure (any key size can be adopted) in all of the considered modes. However, we recommends GCM since it provides both confidentiality and authentication/integrity.

We have implemented a proof-of-concept usable barcode security tool Barcode Security Studio (*BarSec*) (Wahsheh and Luccio, 2019). *BarSec* adopts symmetric and asymmetric cryptographic mechanisms in order to generate safe and usable QR codes. It abstracts away the underlying security mechanisms by proposing high level security objectives that include: barcodes

Solution	Key/hash size (bits)	H	L	U
ECDSA	256	✓		
RSA	1,024	✓		
	2,048		✓	
	3,072		✓	
ECDSA (cert.)	256		✓	
RSA (cert.)	1,024		✓[a]	
	2,048			✓
	3,072			✓
HMAC	128	✓		
	256	✓		
	384	✓		
AES (CBC)	128-256	✓		
AES (OFB)		✓		
AES (CFB)		✓		
AES (GCM)		✓		

[a] preferred 500 pixels

Table 4: Usability of cryptographic barcodes: High (H), Low (L), Unusable (U).

authentication, data integrity, access control and confidentiality. It provides usability warning messages based on the present usability study, and can be used for both generating and reading QR codes. In addition, it provides detailed information about the used algorithms, usability level, scanning time and size overhead. Two of the authors have implemented an Android reader application that supports the above mentioned functionalities (Wahsheh and Luccio, 2019).

5. Conclusion

QR codes may be subject to attacks in which malicious content is embedded in the barcodes in order to break user's privacy, steal credentials, redirect to malicious websites or install malware. In fact, a QR code is just a medium that provides input and, as such, might easily become source of attacks. Moreover, QR codes might contain confidential data that should only be read by authorized users. Cryptography offers standard primitives that provide data confidentiality, integrity and authenticity, preventing most of the attacks on QR codes, especially when they are adopted in closed environments, i.e., when the public keys of trustworthy entities are clearly established, and symmetric key can be securely exchanged. However, cryptography is rarely adopted in this setting since QR codes have limited space and are usually scanned by smartphones that do not generally offer the same performance as personal computers or laptops. This motivated us to perform a systematic study of *usable* cryptographic QR codes.

First of all, we have tested that modern smartphones do not have performance issues when performing typical cryptographic operations, including digital signature which is usually considered one of the heaviest. Notice that, this was not the case a few years ago Razzak (2012). Then, we have considered size issues. QR codes can potentially embed up to about 3Kbytes which would allow for easily embedding digital signature and certificates. However, we have performed a series of experiments to check in which extent such "big" QR codes can be efficiently scanned, with a reasonable user experience. We have considered the scanning time, the distance range tolerated while scanning, and the possibility of spuriously scanning other (simpler) barcodes that appear, by chance, in the QR code.

Our results show that ECDSA and RSA with small keys are usable on QR codes, even when printed in small sizes (for example on supermarket products). Including a certificate in the QR code gives usability problems,

suggesting that using on-line certificates, downloadable via HTTPS might offer an appealing alternative. In fact, when certificates are included, we have pointed out potential usability issues for all of the experimented signature schemes. Of course, downloading a certificate would not come for free, but the application might transparently download it once and cache it internally so to require connectivity only once. HMACs and AES-based symmetric encryption are both highly secure and usable.

We have implemented a proof-of-concept Android app that performs the scan of cryptography enhanced barcodes, confirming our findings. We have used standard algorithms and formats so we are confident that our solution might be employed in practice (Wahsheh and Luccio, 2019).

In this study we have only focused on whether QR codes would remain usable when cryptographic information is incorporated in the payload, in terms of the scanning experience. However, notice that other orthogonal factors might affect the usability of cryptographic barcodes. For example key management is a crucial issue that makes cryptographic solutions complex to deal with, and involves usability, e.g., when users are asked questions about certificate validity. Similarly, it would be important to study appropriate ways to warn the user about possibly insecure barcodes, so that the user understands the consequent security risks. We leave this as future work. We also plan to study less popular signature schemes to look for potential secure-and-usable alternatives to the popular ones.

Acknowledgment. Work partially supported by CINI Cybersecurity National Laboratory within the project FilieraSicura: Securing the Supply Chain of Domestic Critical Infrastructures from Cyber Attacks (www.filierasicura.it) funded by CISCO Systems Inc. and Leonardo SpA.

References

2D Technology Group Inc., 2016. Barcode Security Suite. http://www.2dtg.com/node/74.

A. Sergeev, 2010. UI Designer - ISO-9241 Satisfaction Metrics - Theory of Usability. http://ui-designer.net/usability/satisfaction.htm.

Albertini, A., 2014. Funky File Formats, in: 31st Chaos Communication Congress (31C3). https://fahrplan.events.ccc.de/congress/2014/Fahrplan/events/5930.html.

Alturki, R., Gay, V., 2017. Usability Testing of Fitness Mobile Application: Case Study Aded Surat App. International Journal of Computer Science and Information Technology (IJCSIT) 9(5), 107–127.

Dabrowski, A., Krombholz, K., Ullrich, J., Weippl, E., 2014. QR Inception: Barcode-in-Barcode Attacks, in: Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM'14), pp. 3–10.

European Union Agency for Network and Information Security (ENISA), 2014. Algorithms, Key Size and Parameters Report.

Focardi, R., Luccio, F.L., Wahsheh, H.A.M., 2018a. Security Threats and Solutions for Two Dimensional Barcodes: A Comparative Study, in: Daimi, K. (Ed.), Computer and Network Security Essentials. Springer, pp. 207–219.

Focardi, R., Luccio, F.L., Wahsheh, H.A.M., 2018b. Usable cryptographic QR codes, in: Proceedings of the 2018 IEEE International Conference on Industrial Technology (ICIT'18), Lyon, France, pp. 1664–1669.

Google, 2019. Google Safe Browsing API. https://developers.google.com/safe-browsing/.

Grover, A., Braeckel, P., Lindgren, K., Berghel, H., Cobb, D., 2010. Parameters Effecting 2D Barcode Scanning Reliability. Advances in Computers 80, 209–235.

Ishihara, T., Niimi, M., 2014. Compatible 2D-code Having Tamper Detection System with QR-code, in: Proceedings of the 10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP'14), IEEE. pp. 493–496.

ISO, 2018. ISO 9241-11:2018, Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:en.

ISO/IEC Standard, 2015. ISO/IEC 18004:2015, Information technology – Automatic identification and data capture techniques – QR code 2005 Bar code Symbology Specification.

Jin, X., Hu, X., Ying, K., Du, W., Yin, H., Peri, G., 2014. Code Injection Attacks on HTML5-based Mobile for Apps: Characterization, Detection and Mitigation, in: Proceedings of the 21st ACM Conference on Computer and Communications Security (ACMCCS'14), pp. 66–77.

JSON, 2016. Introducing JSON. http://www.json.org.

Kaspersky Lab, 2011. Malicious QR Codes: Attack Methods & Techniques Infographic. http://usa.kaspersky.com/about-us/press-center/press-blog/2011/malicious-qr-codes-attack-methods-techniques-infographic.

Kharraz, A., Kirda, E., Robertson, W., Balzarotti, D., Francillon, A., 2014. Optical Delusions: A Study of Malicious QR Codes in the Wild, in: Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'14), pp. 192–203.

Kieseberg, P., Leithner, M., Mulazzani, M., Munroe, L., Schrittwieser, S., Sinha, M., Weippl, E., 2010. QR Code Security, in: Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM'10), pp. 430–435.

Kieseberg, P., Schrittwieser, S., Leithner, M., Mulazzani, M., Weippl, E., Munroe, L., Sinha, M., 2012. Malicious Pixels Using QR Codes as Attack Vector. Trustworthy Ubiquitous Computing 6, 21–38.

Krombholz, K., Fruhwirt, P., Kieseberg, P., Kapsalis, I., Huber, M., Weippl, E., 2014. QR Code Security: A Survey of Attacks and Challenges for Usable Security, in: Proceedings of Human Aspects of Information Security, Privacy, and Trust (HAS'14), pp. 79–90.

Krombholz, K., Hobel, H., Huber, M., Weippl, E., 2013. Social Engineering Attacks on the Knowledge Worker, in: Proceedings of the 6th International Conference on Security of Information and Networks (SIN 2013), pp. 28–35.

Peng, K., Sanabria, H., Wu, D., Zhu, C., 2014. Security Overview of QR Codes. MIT Student Project.

Phishtank, 2019. https://www.phishtank.com/.

Razzak, F., 2012. Spamming the Internet of Things: A Possibility and its Probable Solution, in: Proceedings of the 9th International Conference on Mobile Web Information Systems (MobiWIS'12), pp. 658–665.

Vidas, T., Owusu, E., Wang, S., Zeng, C., Cranor, L., Christin, N., 2013. QRishing : The Susceptibility of Smartphone Users to QR Code Phishing Attacks, in: Proceedings of Financial Cryptography and Data Security (FC'13), LNCS, Springer, 7862, pp. 52–69.

Wahsheh, H.A.M., Luccio, F.L., 2019. Evaluating Security, Privacy and Usability Features of QR Code Readers, in: Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019), Prague, Czech Republic, pp. 266–273.

Wired, 2017. Sneaky exploit allows phishing attacks from sites that look secure. https://www.wired.com/2017/04/sneaky-exploit-allows-phishing-attacks-sites-look-secure/.

Yakshtes, V., Shishkin, A., 2012. Mathematical Method of 2-D Barcode Authentication and Protection for Embedded Processing. https://www.google.com/patents/US8297510.

Yao, H., Shin, D., 2013. Towards Preventing QR Code Based for Detecting QR Code Based Attacks on Android Phone Using Security Warnings, in: Proceedings of the ACM ASIA Conference on Computer and Communications Security (ACM ASIACCS'13), pp. 341–346.