Usable Cryptographic QR Codes

Riccardo Focardi
Università Ca' Foscari Venezia, Italy
focardi@unive.it

Flaminia L. Luccio
Università Ca' Foscari Venezia, Italy
luccio@unive.it

Heider A. M. Wahsheh
Università Ca' Foscari Venezia, Italy
heider.wahsheh@unive.it

***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 signature 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 signature schemes. Interestingly, we find out that in some cases security breaks usability and we provide recommendations for the choice of secure and usable signature schemes.**

I. 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 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 [20]. They are also the barcodes with higher data capacity [4], and may store different types of data, such as numeric, alphanumeric, binary and Kanji characters [10].

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 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 [12]. 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. [15].

In the last years there has been a large increase in the use of barcodes in every day 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 [6].

In this paper, we present the first systematic study of *usable* state-of-the-art cryptographic primitives inside QR codes. We select standard, popular signature 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 signature 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 signature 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.

Contributions: Our contributions can be summarized as follows: (i) we survey attacks on QR codes and discuss the potential benefits of enhancing them with digital signature; (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 digital signature schemes, with various key sizes and formats; (iv) we evaluate the digital signature schemes with respect to the usability analysis.

Related work: In [9], 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 and the embedding of actual signatures is left as a future work. In [19], 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 [17], 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 [23] 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 [8] and the Phishtank API [18]. There exists some commercial solutions for secure QR codes, e.g. [3], [22], but there is no publicly available description of the proprietary technology which prevents us to perform any security analysis.

Paper Structure: The rest of the paper is organized as follows: in Section II we present well known attacking scenarios and discuss which are prevented by adopting digitally signed QR codes. Section III analyzes the performance of QR code scanning with respect to different usability factors, and discusses the results of different usability tests. In Section IV, 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 primitive with respect to usability. Section V draws some concluding remarks.

II. Security of digitally signed QR codes

In the following, we consider the most prominent attack scenarios for QR codes and discuss in which extent digitally signed 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 [15], [16], [20].

Malware propagation: In [13] 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 [4], [15].

SQL and command injections: The study of [15] 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 [11].

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 [14].

Discussion: Enhancing QR codes with digital signature prevents the above attacks only when it is not possible for an attacker to perform a legitimate signature. 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 [21]. However, in a closed/controlled environment, the reader might be configured to only recognize internal certificates and verifying the signature would prove the trustworthiness of the QR code content. For example, a supermarket with its own app might be configured to only user the supermarket's public key for signature verification.

III. Usablility of QR code scanning

In this section we study the degree of usability of QR code scanning with respect to the size of the payload. This is of ultimate importance in order to establish the maximum space overhead available for cryptographic material. Interestingly, we will see (cf. Section IV) that, for some cryptographic schemes, using strong key lengths might compromise usability.

We focus on phone-based readers, as we believe this is the most common use case. Below, we describe the experiments that we performed to measure the average time for scanning barcodes of various sizes which, in turns, can be interpreted in terms of usability.

A. Usability parameters

We define QR code usability based on the success and performance of scanning. Along ISO 9241, we consider: *satisfaction*, the user comfort in terms of simplicity to perform the scanning; *efficiency*, the time required to perform the scanning; *effectiveness*, the possibility of successfully scanning a barcode. We have conducted a usability survey among 351 users from 8 universities in 5 different countries. Participants were about 50% males and females, 65% were between 18 and 24 years old and 35% between 25 and 35 or over. The survey (that we do not include for lack of space) proposed some barcodes to users and asked questions about their scanning experience. Based on the answers we have distilled the following usability parameters.

We first introduce the Readability Range, i.e., the range of distances inside which a barcode is readable.

Definition 1: The *Readability Range (RR)* is the difference between the maximum (D_{Max}) and minimum (D_{Min}) distances in centimeters between the scanning device and the barcode, inside which the barcode is readable, i.e., $RR = D_{Max} - D_{Min}$.

Intuitively, the larger is *RR* the bigger is the tolerance over the scanning distance, which naturally makes scanning more user-friendly, improving *satisfaction*. Our users considered usable an *RR* of at least 20 centimeters.

For what concerns *efficiency*, we consider the *Scanning Time (ST)*, i.e., the time, expressed in seconds, required to scan a barcode and extract its content. *Barcode Readability (BR)* classifies the user scanning experience in term of ST. Users of our survey gave the following classification:

Definition 2: Barcode Readability (BR) is defined as:

Normal if $ST < 5$;
Reasonable if $5 \leq ST < 10$;
Hard if $10 \leq ST < 15$;
Unreadable if $ST \geq 15$, or failure.

We will call Normally Readable barcodes (*NR*) the barcodes with Normal *BR*. We will consider usable a barcode that is *NR* at least 75% of the times.

We now consider *effectiveness* in terms of percentage of barcodes that are correctly read. In fact, it might happen that some patterns in a barcode image correspond to another valid barcode. In this case, the application might return the payload from the spare barcode with unexpected consequences [4].

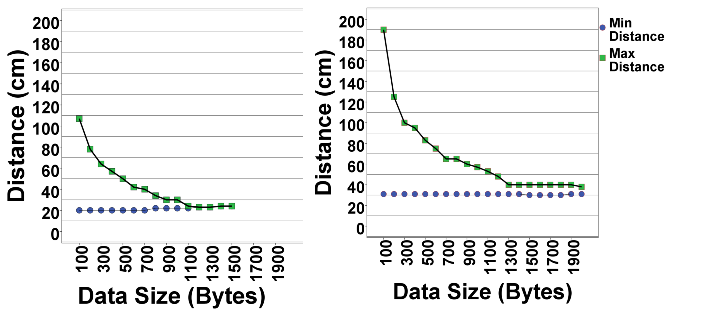
Definition 3: Given a set of barcodes B, let B_M be the set of barcodes that are incorrectly decoded. Then, the *Misleading Percentage (MP)* is defined as $MP = \frac{|B_M|}{|B|}$.

Misleading confuses users and is source of attacks. Our users considered tolerable a Misleading Percentage of at most 5%.

B. Estimating usability

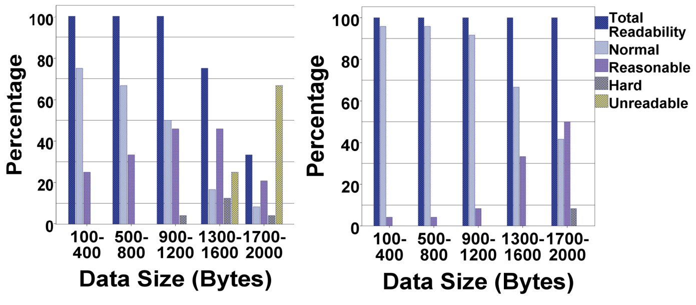
In order to evaluate how the various parameters discussed in Section III-A may affect the usability of barcode scanning, extensive experiments where conducted with different image and data sizes. For the experiments, we have used an Android smartphone with a 1.2 GHz dual-core CPU, 1 GB of RAM and a 8 MP Camera. An application running on a laptop computer generated and displayed 480 different barcodes containing random data that were automatically scanned by the smartphone. In order to simulate human hand shaking, the barcode images were moved and zoomed. To confirm results, more experiments were conducted by humans, so to have real hand shaking in place.

The generated barcodes were 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



(a) *RR* for 300×300. (b) *RR* for 500×500.

Fig. 1: Measuring the Readability Range (*RR*).



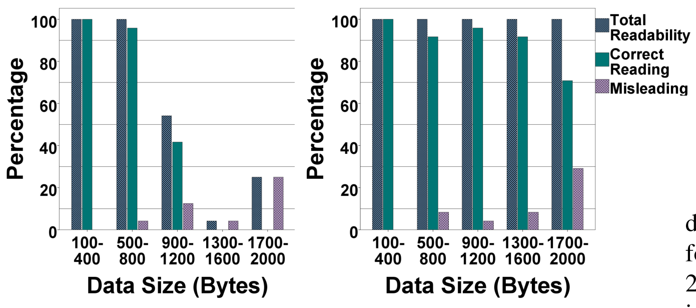
(a) *BR* for 300×300. (b) *BR* for 500×500.

Fig. 2: Measuring the Barcode Readability (*BR*).

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 from 100 to 2000 bytes. For lack of space we show only parts of the results for the different sizes.

The first set of experiments measures the Readability Range (*RR*). Figure 1a and Figure 1b show the measured *RR* for (300 and 500 pixels) barcodes, where the X-axis represents the data size in bytes and the Y-axis represents the distance between the scanning device and the barcode image. We observe that when the data size increases, the *RR* becomes narrower. For small data size, barcodes can be read with wide *RR*, for example the barcode with 100 bytes data size is readable from a distance of 31 to 190 cm ($RR = 159$ cm) for 500×500 barcodes, while the same barcode size with 900 bytes data size is readable from 31 to 60 cm ($RR = 29$ cm), reducing usability. Results also show that the image size plays important role. When the image size is larger, the *RR* becomes wider for the same data size. For example, if we compare the *RR* for two barcode images with the same data size (500 bytes) and different image sizes of 300 and 500 pixels, we have that *RR* is 30 cm and 52 cm, respectively.

The second set of experiments evaluates the relation be-



(a) *MP* for 200×200. (b) *MP* for 400×400.

Fig. 3: Measuring the Misleading Percentage (*MP*).

TABLE I: Usability summary.

Size	*RR* $\geq$ 20 (bytes)	*NR* $\geq$ 75% (bytes)	*MP* < 5% (bytes)	Max size (bytes)
200×200	400	400	800	400
300×300	700	400	800	400
400×400	1100	1200	1200	1100
500×500	1100	1200	1200	1100

tween barcode data size, image size and the Scanning Time (*ST*) in terms of Barcode Readability (*BR*). Figure 2a and Figure 2b show the *BR* measurements with 300×300 and 500×500 pixel barcodes. The data sizes were grouped into five ranges of 100-400, 500-800, 900-1200, 1300-1600 and 1700-2000 bytes. The X-axis represents the data size groups, the Y-axis represents the measured *BR*. E.g., in the first group of 100-400 bytes, the first blue bar represents the total Barcode Readability with a value of 100%. Note that, the total Barcode Readability is the summation of normal (75%), reasonable (25%) and hard (0%), in other words, we include the barcodes with Scanning Time less than 15 seconds (cf. Definition 2). The yellow bar represents the percentage of unreadable barcodes. The summation of the percentage of total Barcode Readability and of unreadable barcodes represents the whole set of barcodes (100%).

We observe that, when the data size increases, readability becomes harder and requires more time, i.e., *ST* increases. For example, barcodes of size 500×500 with 100-400 bytes data size are all readable, and 95.8% of them with normal Barcode Readability (i.e., *ST* less than 5 seconds, cf. Definition 2). When data size increases to 1700-2000 bytes, barcodes are all readable: 41.7% with normal readability, 50% with reasonable readability (5-10 seconds) and 8.3% with hard readability (10-15 seconds). We also notice that larger printing image size gives smaller *ST*'s. For example, comparing the groups 900-1200 bytes in Figures 2a and 2b, we notice that the *ST* has lower values in larger images: 91.6% normal, 8.4% reasonable and 0% hard for 500×500 images with respect to 50% normal, 45.8% reasonable and 4.2% hard for 300×300 images.

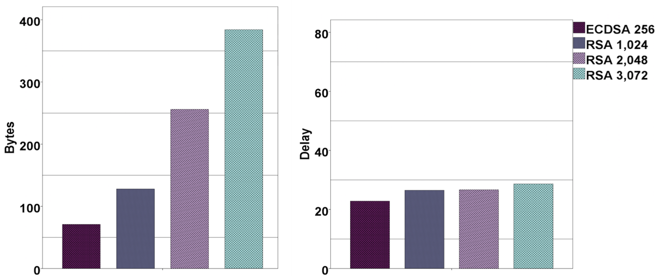
The third set of experiments evaluates the relation between barcode data size, image size and the correct barcode decoding. Figure 3a and Figure 3b show the percentage of correct reading versus the Misleading Percentage (*MP*) for 200×200 and 400×400 pixels barcodes, respectively. The X-axis represents data size groups while the Y-axis represents the fraction of barcodes that are correctly and incorrectly decoded, among the ones that are readable. We notice that *MP* increases when the data size increases. In fact, barcodes become denser and the probability of misleading barcode decoding increases. E.g., for 400×400 barcodes *MP* is 0% for the group of 100-400 bytes and becomes 29.2% for 1700-2000 bytes. Interestingly, *MP* generally decreases when the image size increases, since larger image sizes allow for a more accurate scanning of barcodes with large amount of data. For example, comparing *MP* values for two barcode images of 400×400 and 200×200 pixels with the same data size of 900-1200 bytes, we observe that *MP* is 12.5% for 200×200 barcodes and 4.2% for 400×400 barcodes.

Table I summarizes the results of our experiments on the usability of QR code scanning. For each of the tested barcode sizes we report the maximum data size, in bytes, that can be included in the code providing a good level of usability. As we have previously indicated, we require that the Readability Range (*RR*) is at least 20 cm, that the percentage of Normally Readable barcodes (*NR*) is more than 75% and the Misleading Percentage (*MP*) is less than 5%. Interestingly, we get similar values for the various parameters. In the last column we pick the lowest one, i.e., the maximum size that is compatible with all the selected parameters.

IV. Digitally Signed QR codes

In the following we discuss the time and space overhead of selected cryptographic primitives implemented in standard Android smartphone libraries. In particular, in Section IV-A we study time and space overhead of digital signature standard mechanisms; in Section IV-B we discuss time overhead of data formats, which are necessary to embed the cryptographic data together with the QR code payload; finally, in Section IV-C we summarize the usability of QR codes with the various digital signature algorithms and key sizes.

In order to compute the size of digital signatures and the average needed time to verify a signature, we have considered 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, ENISA [5] 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 to adopt 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



(a) Space overhead for RSA and ECDSA signatures.

(b) Time overhead for RSA and ECDSA signature verification.

Fig. 4: Space overhead, in bytes, and time overhead, in milliseconds, for digital signatures in Android.

TABLE II: Overall size example.

Algorithm	Key length (bits)	with certificates		without certificates	
		JSON (bytes)	ASN.1 (bytes)	JSON (bytes)	ASN.1 (bytes)
ECDSA	256	398	377	151	136
RSA	1,024	588	567	213	198
RSA	2,048	976	955	341	326
RSA	3,072	1,360	1,339	469	454

TABLE III: Overall size with 200 bytes of data using JSON.

Algorithm	Key length (bits)	Signature (bytes)	JSON (bytes)	JSON 1 cert. (bytes)	JSON 2 cert. (bytes)
ECDSA	256	64	218	464	710
RSA	1,024	128	282	688	1094
RSA	2,048	256	410	1072	1734
RSA	3,072	384	538	1456	2374

leave as future work a comparison between the different variants of the signature algorithms. The interested reader can refer to [5] for more detail.

A. *Time and space signature overhead*

Figure 4a presents the signature lengths for different key lengths of RSA and ECDSA. Notice that, ECDSA signature length is twice the size of key length, i.e., 512 bits = 64 bytes, while RSA signature length is equal to the key length. However, adding a digital signature will require more control data than just the digital signature itself as we will discuss in Section IV-B.

We have developed an Android mobile application to test signature's verification overhead. The average signature verification delay is shown in Figure 4b. The tests has been performed on an Android smartphone with 1.2 GHz dual-core CPU, 1 GB of RAM. We notice that, digital signature verification consumes only a small time interval in milliseconds for the various key lengths and algorithms. Thus, the considered digital signature algorithms and key lengths can in principle be used in QR codes in terms of signature size and verification delay. However, size can be critical with respect to usability in some cases, as we will discuss in Section IV-C.

B. *Format overhead*

Since we have seen that size influences usability, we will aim at the most concise possible format providing reliable encoding and decoding. We aim at embedding the following:

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;

Generator The identity of the QR code generator;

Algorithm The cryptographic mechanisms adopted;

Signature The digital signature;

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

We consider two possible standard formats: JavaScript Object Notation (JSON) [1], and Abstract Syntax Notation One (ASN.1) [2]. JSON is more verbose than ASN.1. For the same data structure we have observed that ASN.1 requires about 75% of the space required by JSON. However, JSON has the advantage of being human readable which might be useful when the QR code is scanned using standard readers, as it would provide a meaningful result anyway. Our best encoding of all the required fields (including the certificate) requires 83 and 104 bytes for ASN.1 and JSON, respectively. Without certificate the overhead reduces to 39 and 54 bytes, respectively. We have used very short but human readable tags. 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 standard formats in our study.

Table II presents an example of the overall size of a signed QR code with and without the certificate. The payload is a 33 byte long URL, and the certificate contains test data. We notice that including the certificate significantly increases the data size which, in turns, reduces usability.

C. *Usability evaluation*

Table III summarizes an estimation of the data size for the various algorithms and key size, up to two certificates, using the most verbose format (JSON), and assuming 100 bytes of data for the payload, ID's, etc. and 100 bytes for meta-data of each extra certificate. Without certificates, we just sum up the size of signature with the overhead for JSON structure plus the 100 bytes of data. For certificates, we have to add one more signature, one public key, JSON overhead and 100 more bytes, and so on. We consider more than one certificate in order to evaluate usability with certificate chains. Crossing Table I with Table III, we obtain Table IV.

ECDSA and RSA 1,024 without certificates are the only ones that fit small and big QR codes. ECDSA with one certificate and RSA 2,048 without certificates are borderline (slightly bigger than 400 bytes), so for small payloads they might provide usable QR codes. All other algorithms except RSA 3,072 with one and two certificates and RSA 2,048 with two certificates are fine with big QR codes (at least 400×400). RSA 3,072 with one certificate and 2,048 with two certificates are too big and might result in poor usability. Notice that with

TABLE IV: Usability of the cryptographic solutions.

Algorithm	Key length (bits)	Signature (bytes)	JSON (bytes)	JSON 1 cert. (bytes)	JSON 2 cert. (bytes)
ECDSA	256	64	✓	✓[a]	✓[b]
RSA	1,024	128	✓	✓[b]	✓[b]
RSA	2,048	256	✓[a]	✓[b]	✗
RSA	3,072	384	✓[b]	✗	✗

[a] Requires at least 400×400 size; fits smaller sizes with small payloads;
[b] Requires at least 400×400 size;

two certificates ECDSA and RSA 1,024 require bit QR codes. The only algorithm that scales up to 3 certificates is ECDSA.

We have implemented a proof-of-concept secure QR code reader for Android based in the Zebra crossing (*ZXing*) library [7]. The reader supports the various digital signature schemes and key lengths discussed in Section IV-A. We have tested our implementation on 480 different barcodes confirming our usability evaluation on the various algorithms and key sizes: the app is usable in all of the cases pointed out in Table IV.

V. 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. Digital signature is a standard effective way to authenticate the barcode content and prevent most of the attacks on QR codes when adopted in closed environments, i.e., when the public keys of trustworthy entities are clearly established. However, it 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* digitally signed QR codes.

First of all we have tested that modern smartphones do not have performance issues for what concerns signature verification. Notice that, this was not the case a few years ago [19]. 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). Bigger RSA keys requires bigger print sizes, and RSA with 3,072 bit keys give usability problems when one certificate is included in the QR code. In fact, when certificates are included, we have pointed out potential usability issues for all of the experimented signature schemes. Despite these limitations, our results are promising and 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.

As a future work, we intend to study less popular signature schemes to look for potential secure-and-usable alternatives to the popular ones, and we want to extend our solution to also provide confidentiality through encryption.

Acknowledgment: Work partially supported by CINI Cybersecurity National Laboratory within the project FilieraSicura funded by CISCO Systems Inc. and Leonardo SpA.

References

[1] JSON, 2016. http://www.json.org.
[2] ASN.1, 2017. http://www.itu.int/en/ITU-T/asn1.
[3] 2D Technology Group Inc. Barcode Security Suite, 2016. http://www.2dtg.com/node/74.
[4] A. Dabrowski, K. Krombholz, J. Ullrich, and E. Weippl. QR Inception: Barcode-in-Barcode Attacks. In *Proc. of SPSM'14*, pages 3–10, 2014.
[5] European Union Agency for Network and Information Security (ENISA). Algorithms, Key Size and Parameters Report, 2014.
[6] R. Focardi, F.L. Luccio, and H.A.M. Wahsheh. Security Threats and Solutions for Two Dimensional Barcodes: A Comparative Study. In Daimi K., editor, *Computer and Network Security Essentials*, pages 207–219. Springer, 2018.
[7] GitHub. ZXing project home. https://github.com/zxing/zxing/.
[8] Google. Google Safe Browsing API. https://developers.google.com/safe-browsing/.
[9] T. Ishihara and M. Niimi. Compatible 2D-code Having Tamper Detection System with QR-code. In *Proc. of the IIHMSP'14*, pages 493–496. IEEE, 2014.
[10] ISO/IEC Standard. ISO/IEC 18004:2015, Information technology – Automatic identification and data capture techniques – QR code 2005 Bar code Symbology Specification, 2015.
[11] X. Jin, X. Hu, K. Ying, W. Du, H. Yin, and G. Peri. Code Injection Attacks on HTML5-based Mobile for Apps: Characterization, Detection and Mitigation. In *Proc. of ACM CCS'14*, pages 66–77, 2014.
[12] Kaspersky Lab. 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, 2011.
[13] A. Kharraz, E. Kirda, W. Robertson, D. Balzarotti, and A. Francillon. Optical Delusions: A Study of Malicious QR Codes in the Wild. In *Proc. of IEEE/IFIP DSN'14*, pages 192–203, 2014.
[14] P. Kieseberg, M. Leithner, M. Mulazzani, L. Munroe, S. Schrittwieser, M. Sinha, and E. Weippl. QR Code Security. In *Proc. of MoMM'10*, pages 430–435, 2010.
[15] P. Kieseberg, S. Schrittwieser, M. Leithner, M. Mulazzani, E. Weippl, L. Munroe, and M. Sinha. Malicious Pixels Using QR Codes as Attack Vector. *Trustworthy Ubiquitous Computing*, 6:21–38, 2012.
[16] K. Krombholz, P. Fruhwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl. QR Code Security: A Survey of Attacks and Challenges for Usable Security. In *Proc. of HAS'14, 8533*, pages 79–90, 2014.
[17] K. Peng, H. Sanabria, D. Wu, and C. Zhu. Security Overview of QR Codes, 2014. MIT Student Project.
[18] Phishtank. https://www.phishtank.com/.
[19] F Razzak. Spamming the Internet of Things: A Possibility and its Probable Solution. In *Proc. of MobiWIS'12*, pages 658–665, 2012.
[20] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. Cranor, and N. Christin. QRishing : The Susceptibility of Smartphone Users to QR Code Phishing Attacks. In *Proc. of FC'13, LNCS, Springer, 7862*, pages 52–69, 2013.
[21] Wired. Sneaky exploit allows phishing attacks from sites that look secure, 2017. https://www.wired.com/2017/04/sneaky-exploit-allows-phishing-attacks-sites-look-secure/.
[22] V. Yakshtes and A. Shishkin. Mathematical Method of 2-D Barcode Authentication and Protection for Embedded Processing, 2012. https://www.google. com/patents/US8297510.
[23] H. Yao and D. Shin. Towards Preventing QR Code Based for Detecting QR Code Based Attacks on Android Phone Using Security Warnings. In *Proc. of ACM ASIA CCS'13*, pages 341–346, 2013.