Ca' Foscari University of Venice

Master's Degree programme

in Computer Science

"Software Dependability and Cyber Security"

Final Thesis

Secure 2D barcodes based on visual cryptography

Supervisor
Ch. Prof. Riccardo Focardi
Supervisor
Ch. Prof. Flaminia Luccio
Graduand
Tommaso Moretto
Matriculation Number 844996
Academic Year
2017 / 2018

Acknowledgement

Firstly, I would like to express my deepest thanks to my family for their support throughout my entire academic life.
My research would have been impossible without the aid and support of my beautiful girlfriend Giorgia and my colleague and friend Nicolò Ghiotto.
My sincere thanks to my supervisors Riccardo Focardi and Flaminia Luccio for their help.
Finally, I would like to thank all my friends Fabio Salvini, Matteo Sottana, Giorgio Magnan, Andrea Sina, Dario Burato, Fabrizio Romano, Libraesso Giovanni, Casarella Cecilia for always being there for me.

Abstract

In this thesis we investigate the application of visual cryptography to enforce 2D barcode confidentiality. The approach of visual cryptography is appealing because 2D barcodes are indeed images, but it presents non-trivial challenges. First of all, visual cryptography aims at sharing a unique secret between two or more participants while here we would like to provide each participant with a visual decryption key that can be used to recover the participant's distinguished secret; moreover, visual cryptography assumes that images are pre-shared among participants which is not practical in many situations. We contribute to the state-of-the-art by implementing and experimenting with two extensions to visual cryptography proposed together with the colleague Nicolò Ghiotto. The first scheme allows for generating a single visual ciphertext that can be decrypted by each participants visual key, giving a different (plaintext) 2D barcode. This scheme presents a practical limitation: the visual key is a random image that is very hard to scan and needs to be pre-shared. The second scheme extends the first one by permitting to derive the visual key from a password using what we call a visual key derivation function. This removes the limitation of the first scheme but introduces a scalability issue for what concerns the number of participants. We present experiments and discuss limitations with respect to real use cases.

Contents

1 Introduction **5**

2 Barcode Security **7**

2.1 QR codes 11

2.1.1 Storage capacity 11

2.1.2 Structure of a QR code 12

2.1.3 Error correction code 15

2.1.4 Encoding and decoding phases 15

2.2 Possible attacks to 2D barcodes 17

2.2.1 Phishing 18

2.2.2 Malware propagation 19

2.2.3 Barcode tampering 20

2.2.4 SQL and command injections 22

2.2.5 Cross-site script attacks (XSS) 24

2.3 Authenticity of QR codes 26

2.4 Confidentiality of QR codes 27

2.4.1 Encryption 28

2.4.2 Visual encryption 32

3 Visual Cryptography **34**

3.1 A brief overview of Visual Cryptography 34

3.1.1 Example 35

3.2 General access structures for Visual Cryptography 37

3.2.1 Example 37

3.3 Extensions for Visual Cryptography 42

3.3.1 Example 42
3.4 Visual key derivation function 45

4 Shared key extensions for VCS 49

4.1 A pre-computed shared key model 49
4.1.1 Example 50
4.1.2 Implementation 54
4.1.3 Scalability and performance 59
4.2 A pre-computed cyphertext model 62
4.2.1 Example 63
4.2.2 Implementation 66
4.2.3 Scalability and performance 68

5 Use cases 73

5.1 Role-based access control 73
5.2 User authentication 76
5.2.1 Ticket validity 79

6 Conclusion 81

1 Introduction

The term cryptography derives from the two Greek words “$\kappa\rho\upsilon\pi\tau\acute{o}\zeta$ ” and “$\gamma\rho\alpha\theta\acute{\iota}\alpha$”, where the first is literally translated as “secret” while the meaning of the second is “handwriting”. Cryptography is the art of hiding information in a way such that only the writer and the receiver are able to understand the content of the message sent. The information to protect may vary from e-mail messages to electronic signals. Cryptography is based on two operations. The first is called encryption and has the goal of transforming the information in a way such that it becomes unreadable producing what is called ciphertext. The second one is called decryption and, taking as input the ciphertext generated by the encryption process, it retrieves the original message which is called plaintext. In standard cryptography these two processes are based on mathematical transformations which allow to encrypt and decrypt the secret information. On the contrary, visual cryptography stores the secret information into an image (usually black and white) and then it encrypts the latter by splitting it into several sub-images called transparencies. The obtained transparencies do not carry any information about the original secret information, even if basically they represent a part of the original image. In order to retrieve the secret, the n transparencies must be overlapped generating the original image. As consequence the decryption process does not require any cryptographic computation making it understandable even to users which do not have any knowledge of cryptography.

As mentioned before in visual cryptography the secrets are represented as images, consequently one possible scenario is the one in which this technique is used in order to enforce the confidentiality of barcodes. As a matter of fact, barcodes are a graphical machine-readable representation of data which can be easily decoded through the use of a mobile device. Consequently, this feature introduces a lack of confidentiality for barcodes since any user possessing a mobile device can extract the information stored inside them. It is therefore clear that to improve the confidentiality of barcodes the approach of visual cryptography is appealing (it is important to stress the fact that they are fully fledged images).

The thesis is divided into three parts. In Chapter 2 we describe what a barcodes is and we focus our attention on QR codes, the most commonly used ones. Then, we present known attacks on barcodes and we discuss about their authenticating and confidentiality.

Chapter 3 introduces the definition of visual cryptography, describes known schemes and identifies their limits. In the last part of the chapter we present the two models that we have proposed in order to overcome these limitations.

Chapter 4 implements two real world applications based on the two models defined in the previous section that improve the confidentiality of QR codes when they are used in real world scenarios.

2 Barcode Security

Barcodes are a machine-readable representation of data. Usually, this information describes some characteristics of the object which carries the barcode. Nowadays, it is possible to see barcodes on everything from food items at the supermarket to patient wristbands at hospitals. They come on different forms and versions. First, we can distinguish them in two main families: the 1D (one-dimensional) and the 2D (two-dimensional) barcodes.
A 1D barcode consists of vertical lines arranged inside a rectangle. They can encode different data varying the spaces between the vertical lines that compose the barcode and their widths (see Figure 1).

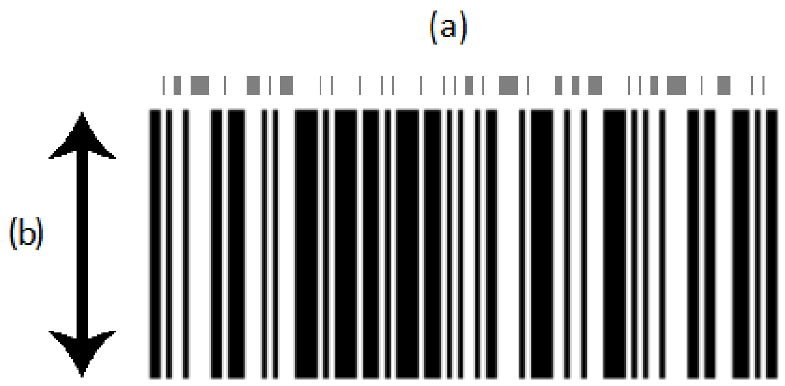


Figure 1: Structure of one-dimensional barcode. (a) shows the variation of the space between the vertical lines. (b) shows how the widths of the vertical lines can change.

On the other hand a 2D barcode, instead of using vertical lines, stores the information into a grid with special patterns of squares, called modules, arranged in both the vertical and the horizontal direction (see Figure 2).

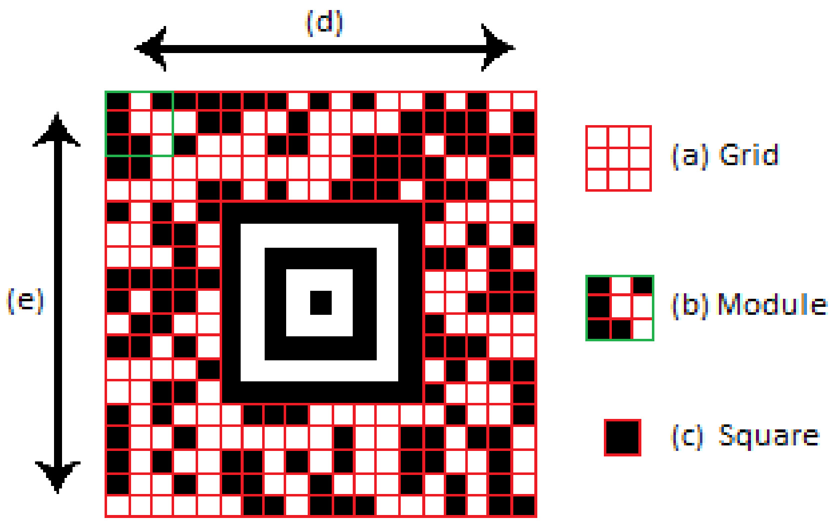


Figure 2: Structure of two-dimensional barcode. The information are encoded into modules (b) which are composed of black or white squares (c). Each of these modules are then arranged into a grid (a) using both the vertical (d) and the horizontal (e) direction.

Nowadays 2D barcodes are largely used since they have a capacity higher than the one of 1D barcodes, and they allow to store different types of information such as: text, Vcard (people's contact), Uniform Resource Locators (URLs), etc.
Among the 2D barcodes the Quick Response (QR) code is the most commonly used one. It was developed in Japan and it has the highest storage

capacity, since it was created to store the Kanji characters. Besides QR codes, it is also possible to find Data Matrix, PDF417 and Aztec Codes. Tables 1, 2, 3 and 4 show a comparison of the previous 2D barcodes and their applications.

QR code	Features	Utilization
	Max capacity (numeric): 7089 Max capacity (alphanumeric): 4296 Check digit: Error correction code	Advertising, government and public service, access control and mobile payment.

Table 1: QR code description [12].

Aztec code	Features	Utilization
	Max capacity (numeric): 3832 Max capacity (alphanumeric): 3067 Check digit: Error correction code	Patient-safety applications, including patient identification, wristbands and labels for unit-of-use medications, blood products and specimens.

Table 2: Aztec code description [12].

DataMatrix code	Features	Utilization
	Max capacity (numeric): 3116 Max capacity (alphanumeric): 2335 Check digit: Error correction code	Encodes large amount of data information. Ideal for marking very small containers. Used by the pharmaceutical industry for unit dose and product marking.

Table 3: DataMatrix code description [12].

PDF417 code	Features	Utilization
	Max capacity (numeric): 2710 Max capacity (alphanumeric): 1850 Check digit: Error correction code	Used in logistic and in governmental applications.

Table 4: PDF417 code description. [12]

2.1 QR codes

We are interested on applying our experiments to a specific family of 2D barcodes, i.e, the QR codes. In 1994 QR codes were introduced for the first time by Denso Wave. They were initially used in automobile industry in order to track vehicles during manufacturing.
Over the last years their use has changed drastically and now they are used in a much broader context. This result is a consequence of the fact that QR codes, among 2D barcodes, are the ones with the fastest readability and the highest storage capacity. As a consequence, nowadays it is possible to run across QR codes used for different purposes such as to store: text, Uniform Resource Identifiers (URI), vCard contacts, etc.

2.1.1 Storage capacity

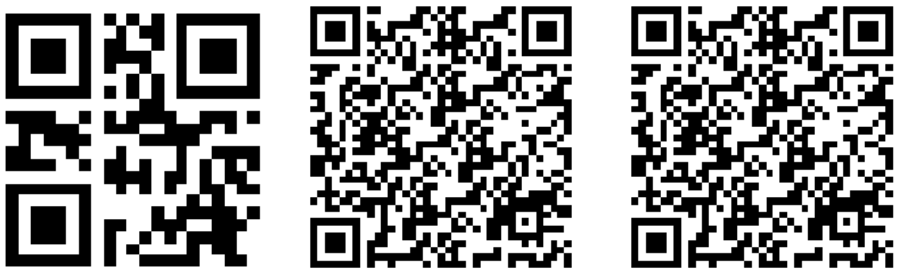
The information that a QR code can store depends on the following parameters:

- Data types: define if the data are numeric, alphanumeric, byte or kanji.
- Version types: from 1 (21x21 modules) to 40 (177x177 modules), identify the dimension of the final QR code.
- Error-correction level: identifies the percentage of codewords that can be restored. It can assume the following four values: Low, Medium, Quartile, High.

Table 5 presents the storage capacity of a version 40 of a QR code with error correction level L, while Figure 3 shows the same information encoded into different versions of QR codes.

Data type	Max. characters	possible characters, default encoding
Numeric	7089	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Alphanumeric	4296	AZ (upper-lower case), 09,space, \$, %, *, +,-, ., /, :
Binary/byte	2953	ISO 8859-1
Kanji	1,817	kanji alphabet

Table 5: Maximum storage capacity of a QR code (40-L)



(a) V1 QR code (b) V5 QR code (c) V10 QR code

Figure 3: Different versions of QR codes containing the same alphanumeric string.

2.1.2 Structure of a QR code

QR codes are represented as a square grid within which square dots are arranged in order to give a graphical representation of textual information. The square grid is divided in different areas which have different roles. Figure

4 shows the distribution of these areas (for this image a version 10 QR code was chosen so as to highlight all of the patterns).

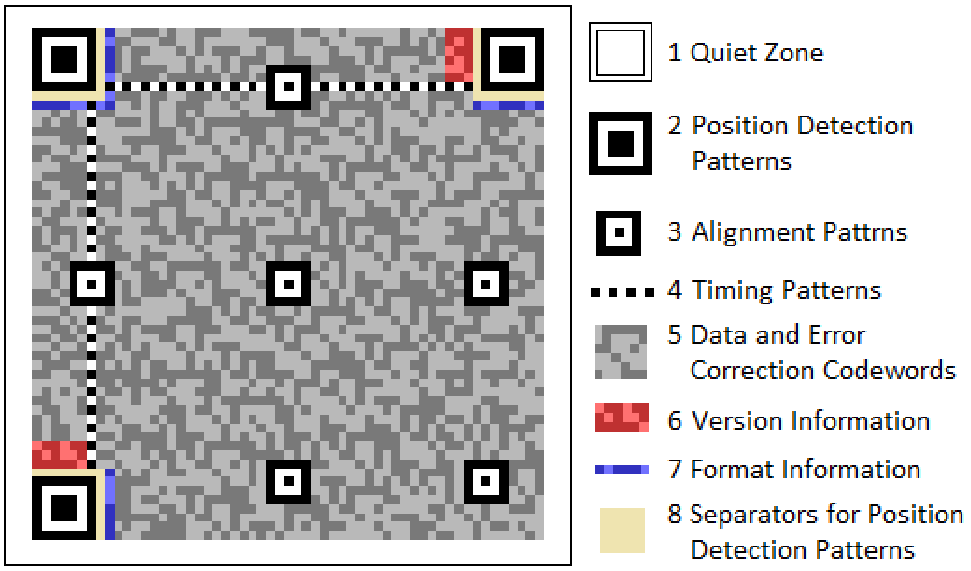

1 Quiet Zone
2 Position Detection Patterns
3 Alignment Pattrns
4 Timing Patterns
5 Data and Error Correction Codewords
6 Version Information
7 Format Information
8 Separators for Position Detection Patterns

Figure 4: Structure of a QR code.

Quiet zone: consists of a white empty square which surrounds the QR code (see Figure 4, point 1).

Position detection patterns: three identical patterns are positioned at the upper left, upper right and lower left corners of the QR code (see Figure 4, point 2). These patterns are always identical for every version of QR code and consist of a 7x7 (modules) black empty square which contains a 3x3 (modules) black filled square. Using these patterns the scanner can detect the location and the orientation of the QR code in the field of view.

Alignment Patterns: if the QR code is scanned in a wrong way, producing a distorted image, these alignment patterns help to realign the module allowing a correct decodification of the content (see Figure 4, point 3).

Timing Pattern: consists of a row or a column of alternating black and white modules (starting and ending with a black module). In QR codes standards there are two timing patterns, a vertical and an horizontal one (see Figure 4, point 4). The vertical timing pattern is located along column 7, in every QR code version, starting after the upper left separator and ending before the lower left one. On the same way the horizontal timing pattern is located along row 7, in every QR code version, starting after the upper left separator and ending before the upper right one. These two patterns allow to identify the symbols density and simplify the research of the format and version information pattern.

Data and error correction codewords: this region contains the information and the associated error correction code represented through the use of graphical patterns (see Figure 4, point 5).

Version information: identifies the data type of the information stored in the QR code (see Figure 4, point 6).

Format Information: stores the information about the error correction code and the visual pattern used to encode (see Figure 4, point 7).

Separators for position detection patterns: these separators are standard for every QR code version and consist of an 8x8 (modules) white empty square which distinguishes the position detection patterns from the encoding region (see Figure 4, point 8).

2.1.3 Error correction code

The information stored inside a QR code is split into codewords of length 8 bits (1 Byte) and an error correction code is applied to each of them. As stated above the data are encoded into a QR code through the use of graphical patterns. It is thus clear that using an error correction code, it will be possible to extract the content without error even if the QR code is damaged. The different levels of error correction are shown in Table 6.

Error Correction Level	Recovery Capacity
Level L (Low)	7% of codewords can be restored
Level M (Medium)	15% of codewords can be restored
Level Q (Quartile)	25% of codewords can be restored
Level H (High)	30% of codewords can be restored

Table 6: Error correction level [12].

2.1.4 Encoding and decoding phases

The writer generates the message that he wants to store (Step 1 in Figure 5) and passes it as an input to the application which is responsible for the generation of the QR code. First of all, the message is converted into a bytestring (Step 2) and put together with the version and format information (Step 3). At this point starting from the bytestring and the additional information an error correction code (Step 4) is built. Finally, the new bytestring is split into codewords of 8 bits length and each of them is represented through a graphical pattern (Step 5). During the decodification phase the reader has to scan the QR code with its mobile device (Step 6). Thereupon the scanner application, using the error correction code, reconstructs the original bytestring (Step 7). Finally, using the version and format information the scanner application decodes each codeword extracting the original message.

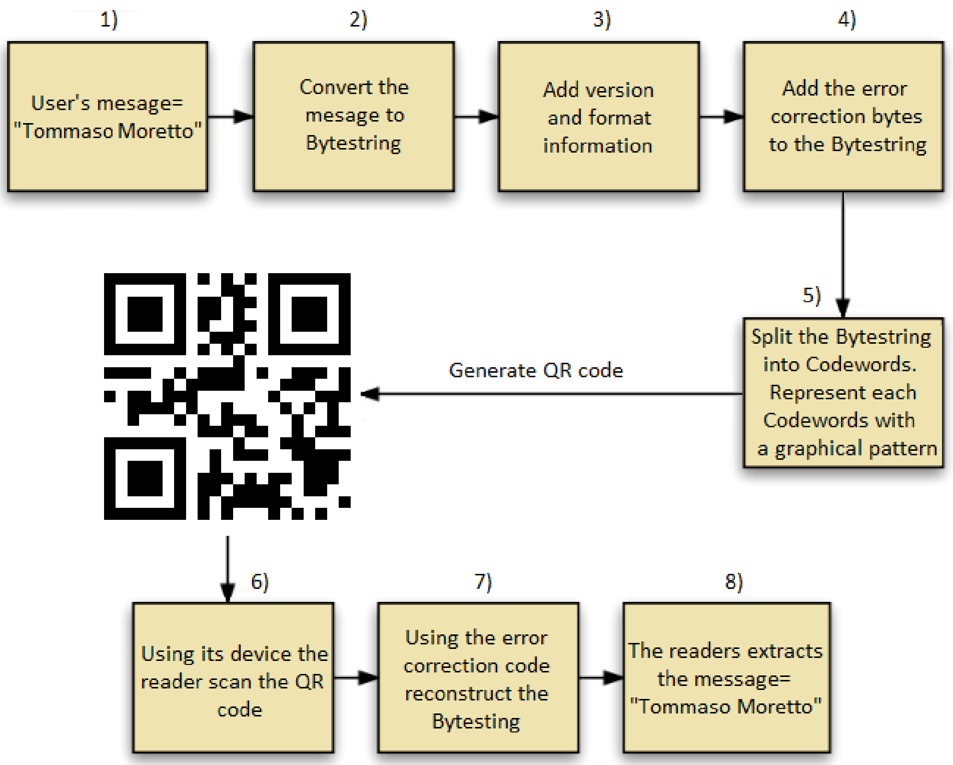

1)
User's mesage= "Tommaso Moretto"
2)
Convert the mesage to Bytestring
3)
Add version and format information
4)
Add the error correction bytes to the Bytestring
5)
Split the Bytestring into Codewords. Represent each Codewords with a graphical pattern
Generate QR code
6)
Using its device the reader scan the QR code
7)
Using the error correction code reconstruct the Bytesting
8)
The readers extracts the message= "Tommaso Moretto"

Figure 5: Encoding and decoding process of a QR code.

2.2 Possible attacks to 2D barcodes

One of the common uses of 2D barcodes is to store a link to a Web page which shows information about a certain service or product. If the users scan the barcode the URL is shown and then they can decide to either access or not the Web page. Furthermore, barcodes can be used for modeling physical access control, identification and warehouse logistics. In these situations, the information stored inside the 2D barcodes are used as input to applications that act consequently.

Since barcodes are used to provide information to users or input to their applications, a crucial problem is that these kind of stored information are untrusted. As a matter of fact, an attacker can encode malicious URLs (phishing) or payloads that take advantage of some vulnerabilities in the back-end applications.

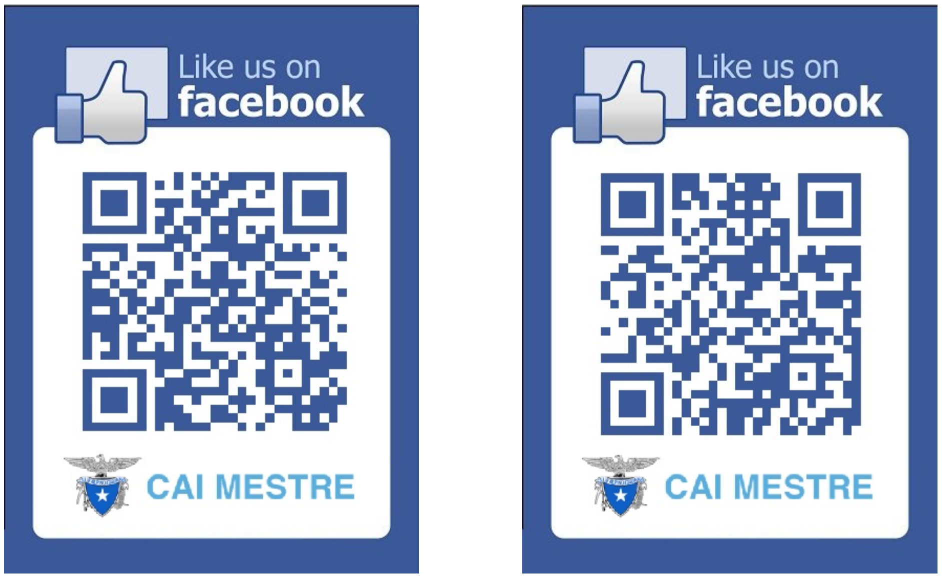
The massive diffusion of 2D barcodes increased the interest of cyber attackers who try to invade users' privacy by compromising their devices. The first attack scenario was discovered in 2011 in Kaspersky Lab and consists in a combination of phishing and malware propagation, used in order to get users' personal information. This weakness is due to the fact that the users do not need any kind of authentication to access the content of a barcode. As consequence this attack can be performed on the most common barcodes. In the following subsection different kind of attacks for 2D barcodes, previously discussed in [4], will be presented.

2.2.1 Phishing

In this kind of attack, the attacker stores inside the barcodes an URL that redirects the user to a fake Web page, normally a login page. This Web page is identical to the legitimate one and misleads the user to insert its private data, such as login details or credit card number.
[15] presents an in-depth analysis of the vulnerabilities of QR code to phishing attacks. In this paper the authors analyse the portion of the users that after scanning a QR code chose to open the related Web page and the interactions that they performed on it (it is important to remark the fact that this analysis holds for all the 2D barcodes previously described).
Analysing the results of the study turns out that the majority of the users visit the Web page for curiosity or for fun.

Example

A practical example of 2D barcodes phishing attack is presented in Figure 6. In this scenario the Alpine Italian Club of Mestre wants to promote its facebook page. One of the chosen techniques is to distribute leaflets that contain some information about the club and a QR code which redirects the device of the reader to the Facebook page of the group, asking for a like. An attacker can substitute the original QR code (see Figure 6a) with one (see Figure 6b) that redirects the reader's device to the following link: `https://www.facebouk.com/scuolacapuismestre/` (instead of: `https://www.facebook.com/scuolacapuismestre/`) which contains a fake Facebook login page. If the victim logs in, the attacker is able to steal the user's credentials.



(a) Original leaflet (b) Counterfeit leaflet

Figure 6: Example of QR code phishing. At a first sight the two leaflets seem equal, but the second stores a malicious URL.

2.2.2 Malware propagation

As seen before, an attacker can store inside a QR code a malicious URL. In this kind of attack once read the QR code, the latter redirects the user to a Web page that, exploiting the vulnerabilities of the device, installs a malware.

In this scenario an attacker may infect the devices of the users with viruses, spywares, Trojan horses or worms compromising the system.

In paper [8] the authors present the result of a really interesting analysis. They implement a tool that sifts the Web (for the most Web pages that use

QR codes to promote products or services) searching for image files of QR codes, selecting only the one that stores an URL. The aim of the analysis was to discover the portion of selected QR codes which store a malicious link. At the end the tool selects 94770 QR codes, in which 145 had a malicious behaviour. Furthermore, 94 out of 145 QR codes were used to redirect users to malicious Web pages, in order to distribute malwares.

Example

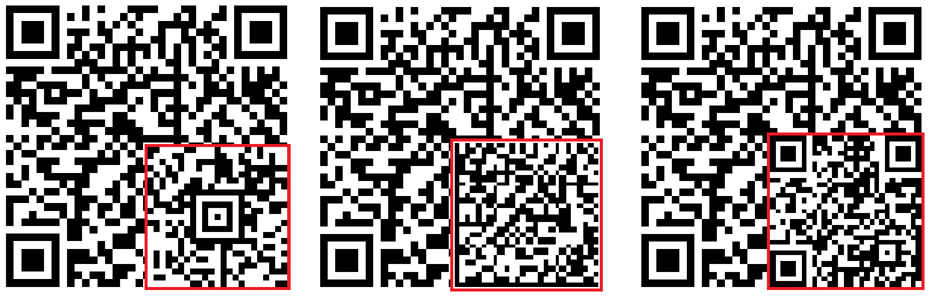
The attack works in the same way of the one described in section 2.2.1. The attacker substitutes the original barcode with one that redirects the device of the reader to a malicious Web page.

2.2.3 Barcode tampering

Nowadays companies use 2D barcodes to provide products' information or to carry out a purchase process. This system allows attackers to derive advantages from the companies' popularity by tampering the original barcode tricking the users.
An interesting study of tampered barcodes is presented in [3]. In this paper the authors show that it is possible to model a 2D barcode that fits to different standards. For example this is feasible in QR codes since segments with unknown encoding are ignored, which is what makes it possible to insert a barcode inside a QR code. In this situation the decoding of the content of the QR code is non-deterministic and it depends on which of the two areas is detected by the reader.

Example

In this thesis research, in order to have a practical example of barcode tampering, the tests performed in [3] have been repeated. First of all, three identical QR codes were built overwriting the left bottom side (the one that stores the error-correction code) with a malicious barcode.
In Figure 7 the three tampered QR codes are shown.



(a) Aztec in QR (b) Data matrix in QR (c) QR in QR

Figure 7: Example of tampered QR codes.

In order to show that the decodification of these tampered QR codes is non-deterministic, different scanner applications were downloaded and used to decode the content of the three barcodes. Every scanner is used many times, varying the distance and the rotation of the device, in order to extract the content of each QR code.
The result of these tests are presented in table 7, which shows how scanners react to tampered QR code.

	Aztec in QR		DM in QR		QR in QR	
	Outer	Inner	Outer	Inner	Outer	Inner
NeoReader	✓	✓	✓	✗	✓	✓
Qrafter	✓	✗	✓	✓	✓	✗
i-nigma	✓	✗	✓	✗	✓	✓
ScanLife	✓	✓	✓	✗	✓	✓
UberScanner	✓	✓	✓	✗	✓	✗

Table 7: Scanners reaction to tampered QR codes.

2.2.4 SQL and command injections

Another interesting attack scenario is presented in [9], in this situation the attacker stores, inside the 2D barcode, SQL statements. Since the content of the barcode is used by automated systems, to access a relational database a crucial point is the sanitization of the input. If this procedure is not done properly an attacker can encode SQL commands that, once decoded, are executed. If for example an attacker encodes the following SQL statement $DROP\ TABLE < tablename >$ with the normal input, it could cause the deletion of the table.

Furthermore, papers [9, 10] show that 2D barcodes are also vulnerable to command injection attacks. As a matter of fact an attacker can encode command-line parameters into a barcode and take control of the server host. The authors of paper [10] discuss about how it is possible to attack Samsung phones by storing into a barcode Man-Machine Interface (MMI) instructions. Once the barcode is scanned the malicious instructions are executed causing

for example the deletion of all the devices data.
This kind of situation is due to the fact that developers assume attackers can not manipulate the content of a 2D barcode considering it safe.

Example
Suppose that a pharmaceutical industry handles the inventory of their warehouses using an app which works as in Figure 8.

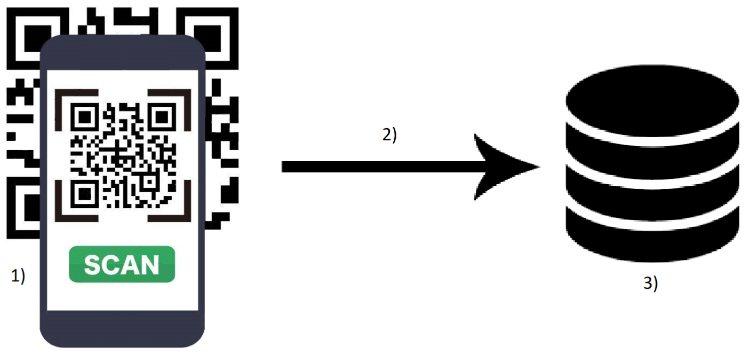


Figure 8: Application based on QR codes scanning.

First of all the app scans the barcode (1), then the latter uses the decoded information as input for the application (2) and finally stores it into the database (3). It is quite obvious that (2) is a crucial point in the application. In fact, if the developers consider the information stored inside the 2D barcode reliable, not performing any sanitization, an attacker can substitute the original barcode of a product with a malicious one containing SQL statement such as *DROP TABLE inventory*. In this situation once the malicious barcode is decoded, the content will be used as input for the application deleting all the data stored in table inventory.

2.2.5 Cross-site script attacks (XSS)

Over the last decade, smartphones became one of the most used technological devices. For this reason many developers have started working in the field of Web apps development. Since applications have to support different platforms, developers must implement distinct versions of them basing on various languages such as Java in Android and Object C in iOS.

Developers that want to implement universally supported applications, start using standard Web techniques based on technologies such as HTML5, CSS and JavaScript. In this situation an attacker can include JavaScript code into a Web application (for example using input forms where the inserted text is not sanitized) in order to take advantage of some vulnerabilities of the system.

This kind of attack is called Cross-site scripting and in paper [6] the authors show that is possible to perform it when the content of a barcode is used as input for the system. In the same paper they provide an example of XSS attack for a HTML5-based application; they chose a quite popular barcode reader called *pic2shop*. This application runs in Windows Phone, Android and iOS platforms and it is vulnerable to XSS attacks using in each of them a barcode as host for the script.

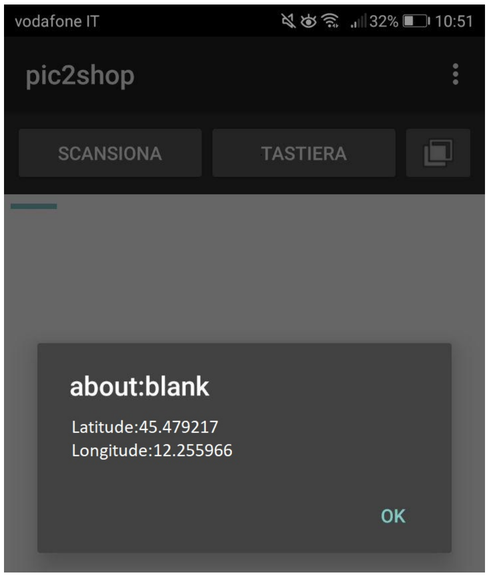
The authors created a QR code which contains a HTML tag and JavaScript code. Once the scanner reads the barcode, the malicious JavaScript code is executed stealing geolocated information of the user. Of course the attacker can store inside the barcode any attack it wants, taking advantage of whatever vulnerability it discovers. This particular vulnerability is a consequence to the fact that barcode reader applications often ask to have access to users' device resources such as in this case geolocation.

Example

```
<script>
    lat="Latitude:"+loc.coords.latitude;
    lon="Longitude:"+loc.coords.longitude;
    alert(lat + lon);
</script>
```

Figure 9: QR code which contains malicious javascript code.

In order to have a proof of concept for this attack a malicious QR code was built, as it can be seen in Figure 9, and scanned with the application *pic2shop*. Once the scanner decodes the content of the barcode, the malicious javascript code is executed producing the output shown in Figure 10.



about:blank

Latitude:45.479217
Longitude:12.255966

Figure 10: Result of QR code xss for the application *pic2shop*.

It is quite obvious that an attacker can encode inside the QR code more dangerous scripts, taking advantage of the vulnerability of the application.

2.3 Authenticity of QR codes

In the previous section some possible attacks on 2D barcodes were presented, focusing the attention on QR codes. Now, with this knowledge, it is clear that QR codes, and more in general 2D barcodes, are easily vulnerable to attacks when malicious content is encoded. The most dangerous scenario is the one in which an attacker takes advantage of this lack of authentication in order to invade users privacy, steal login information or redirect the reader to malicious websites. This is a consequence of the fact that the main use of 2D barcodes is to provide input for the readers' device.

In [5] it is shown how it is possible to achieve authentication for QR codes through the use of digital signature. Firstly, the authors prove that digital signatures are not subjected to performance issues when it is verified through the use of a smartphone. Secondly, they focus their attention on size issues since QR codes can store at most 4296 alphanumeric characters. This size limitation has been overcame signing QR codes using ECDSA and RSA with small keys. Always in [5] a detailed analysis of the usability of this signed QR code was performed in order to find the ideal key length. Despite these limitations, the results obtained applying digital signature to QR codes are good and the authors were able to implement a proof-of-concept Android app that performs the scan of sign QR code.
In conclusion it was proved that digital signature is applicable in order to achieve authentication for barcodes, preventing most of the attacks presented before, when adopted in close environments. This last constraint is due to the fact that in an open environment there is no possibility to implement an application able to scan a 2D barcode and certificate the validity of the incorporated digital signature.

2.4 Confidentiality of QR codes

In this section we will discuss how to improve the confidentiality of QR codes. This feature is completely different from the one regarding authentication. As a matter of fact, gaining confidentiality means to restrict the access to the content of the QR codes only for authorized users.

2.4.1 Encryption

In paper [7] two interesting standards for secure 2D barcodes are presented; in particular the analysis focuses on how to improve the confidentiality of QR codes. The first technique allows to create QR codes which contain encrypted messages that only authorized readers can recover. The second technique has been developed so that a reader has the possibility to verify the reliability of the QR code.

Symmetric encryption

This technique secures the barcode by encrypting the content and is called EQR(Encrypted QR codes). The authors define two possible schemes, one based on symmetric encryption and the other based on public key encryption. It is important to remind that in both the two methods the error correction code is used to correct errors on the ciphertext, so that an attacker cannot obtain any information from it.
In Symmetric EQRs (SEQRs) the shared secret key is used both by the writer, who uses it in order to encrypt the message and generate the 2D barcode, and by the reader, who uses it to extract and decrypt the content. Figure 11 and 12 show the encoding and decoding processes for a SEQR code.

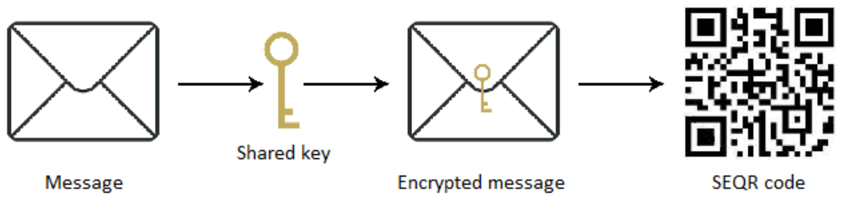


Figure 11: Encryption process for the generation of a SEQR code.

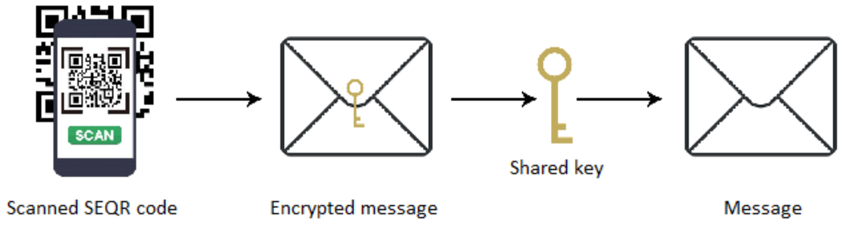


Figure 12: Decryption process for read the content of a SEQR code.

Therefore, using this scheme an user is able to authenticate an untrusted device through the use of a trusted one. Suppose for example that an user is logged in on a certain application on its trusted device and that it wants to log in on a new one that it never uses. Typing its password on an untrusted device, for example a public computer, the user could put it at risk to keyloggers. In order to avoid this problem, the untrusted device displays a SEQR which can be read only knowing the secret key. At this point the user, using its trusted mobile, can decrypt the content of the SEQR, which contains an URL that once opened will log in the user on the new device.

Asymmetric encryption

In Public Key EQRs (PKEQRs) the RSA public key encryption scheme is combined with AES, where the RSA public key is used to encrypt the AES key so that it can be included in the encrypted message. Figure 13 and 14 show the encoding and decoding processes for a SEQR code.

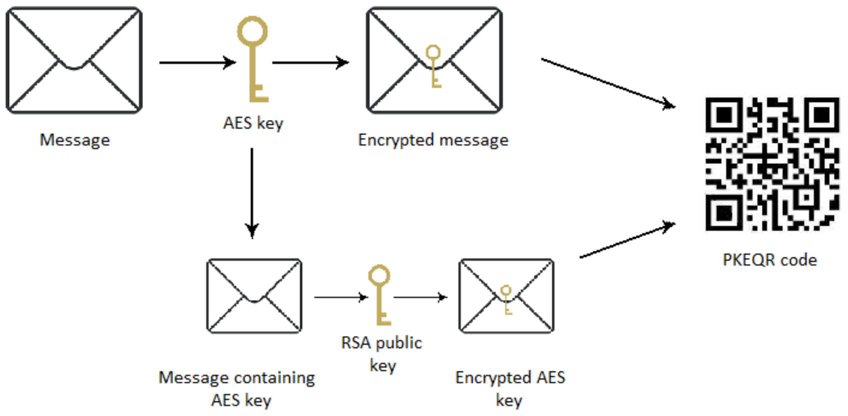


Figure 13: Encryption process for the generation of a PKEQR code.

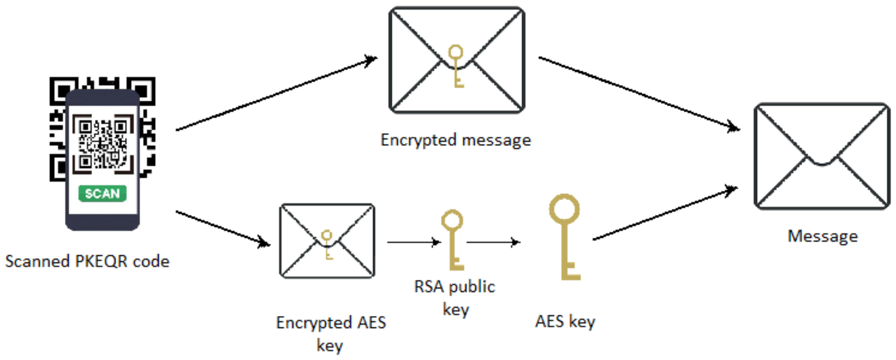


Figure 14: Decryption process for read the content of a PKEQR code.

This way of authenticating readers is not commonly used, since in commercial applications companies are interested in providing data to a large amount of people in the easiest way possible. As a matter of fact, with this scheme the encrypted message is only accessible from an user who knows the RSA secret key needed to decrypt the AES key previously encrypted with the RSA public key.

Both the two methods described introduce additional cost since they modify the QR code standards. With the first method, the one based on SEQRs, the message is encrypted and maintains the same length so the only additional cost is about time (except for the case in which the message needs to be padded). Indeed, this method requires additional time in two situations: the first is the one in which it creates the SEQR containing the encrypted message, the second is the one in which it decrypts the content of the SEQR when this is scanned from the reader. This additional cost can be ignored considering that usually the messages contain only URLs, which take up only a few bytes.

Data encrypted with PKEQRs require additional cost both in terms of time and space. As for the previous model, additional time is required in the processes of encryption and decryption. Furthermore, PKEQRs model does not preserve the length of the original message, since it has to add the encrypted secret key, requiring supplementary space. The maximum capacity of a QR code is presented in Table 1, therefore the length of the message to memorize is extremely important. Hence larger is the message, denser will have to be the QR code which consequently, at a certain point, will not be read anymore.

In [7] the authors implement the PKEQRs scheme and find its fundamental limit, which is a consequence of the fact that the smallest AES key has length 128 bits. Suppose that a certain message can be stored in a version 1 QR code (22x22 module). If the same message is encrypted in a PKEQR (using a key of length n=128 bits) it will require a version 3 QR code (29x29 module) which needs almost twice the area of the unencrypted QR code. As a consequence, if the length of the message is too long it will not be possible to use a PKEQR.

2.4.2 Visual encryption

In [14] a new approach for improving confidentiality of QR codes is proposed. The authors define a method which allows to restrict the access to the content of a QR code only to authorized users. This technique encrypts the data using a symmetric-key algorithm. As a matter of fact the two or more participants use the same private key in both the encryption and decryption phases. In this scheme, the secret key and the plaintext are encoded into two different QR codes of the same version. After this initial phase the plaintext QR code is XORed only in a specific area, which is contained in the one that stores the data and the error correction code, with the same specific area of the secret key QR code.

An example of the encryption phase is presented in Figure 15, where the plaintext and the secret key QR codes contain respectively the message “A first approach to visual encryption” and the key “Secret key” (see Figure 15.1 and 15.2). During the encryption the red bordered area of the Plaintext QR code is XORed with the one of the Secret Key QR code obtaining as result the Ciphertext QR code (the process is presented in Figure 15). With this kind of encryption the resulting Ciphertext is unreadable and as consequence it is a meaningless image which gives no information to a possible attacker.

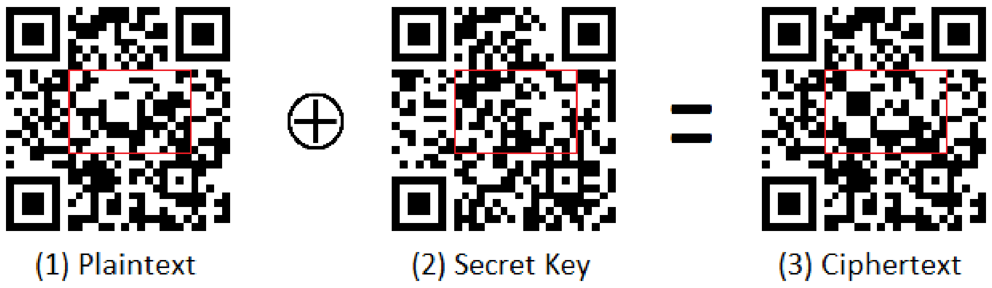


Figure 15: Visual encryption.

An example of the decyrption phase is presented in Figure 16. The original Plaintext QR code is obtained XORing the red bordered area of the Ciphertext QR code with the one of the Secret Key QR code.

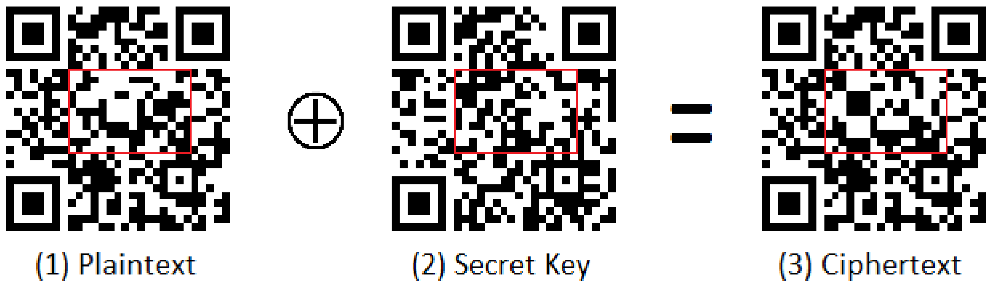


Figure 16: Visual decryption.

3 Visual Cryptography

In this chapter we are going to present visual cryptography and how its most known schemes work. Then the section analyses the fundamental limits of visual cryptography when the aim is to apply these schemes in a real word scenario. Finally the section provides a description of the two models, defined during this research thesis, which allow to overcome these limitations.

3.1 A brief overview of Visual Cryptography

Visual Cryptography was first introduced by Naor and Shamir in [13] to solve the problem of encrypting texts, notes and pictures in a secret way which can be decoded directly by human beings without needing any computation. In the basic model, two indistinguishable random noise images, called *transparencies*, can be overlapped to generate the plaintext. The previous model can be extended into a k out of n scheme, with $k \leq n$. It consists of: one original image and n generated transparencies; the image is visible if any k or more transparencies are stacked together, but if they are fewer than k the image is invisible.
In this model, the secret image consists of black and white pixels and each of them is handled separately to generate the transparencies. In each transparency every pixel of the secret image is mapped into a set of subpixels called *share*. A share is a collection of m white and black subpixels put together to contribute to the color of the original pixel. It is clear that the secret image reconstructed through the trasparencies is an expanded version of the original one. The combined share obtained by overlapping trasparen-

cies $i_1, .., i_k$, has as many black subpixel as the binary OR operation of rows $i_1, ..., i_k$, and its grey level is defined by its Hamming weight H, in which the latter is the number of symbols that are different from the zero-symbol of the alphabet used. A combined share is considered black if H is greater then a threshold d_{upper} and white if it is lower then another threshold d_{lower}.
An example of this process is presented in Figure 17.

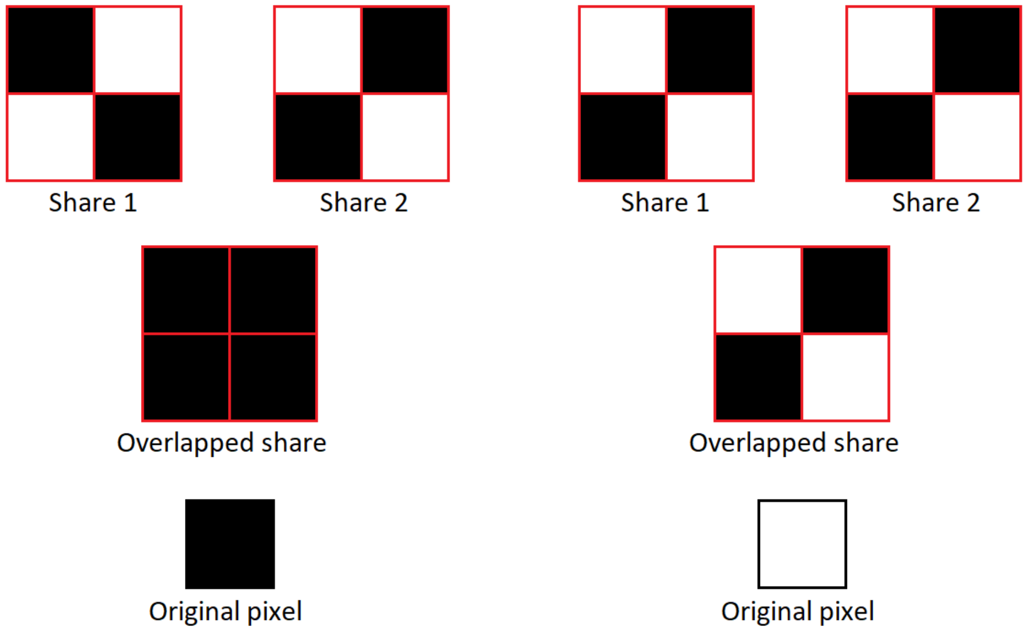


Figure 17: example of a black and a white pixel with their respective shares.

3.1.1 Example

In this paragraph an example of the described model, in which the secret image is a QR code, will be presented (see Figure 19a). As we have previously mentioned using this model it is possible to split the secret image into two transparencies (see Figures 18a and 18b). It is important to stress the fact that the two generated transparencies consist of random black and white pixels and as consequence they do not add any information about the secret. Finally, if overlapped these two transparencies generate the secret image (see Figure 19b).

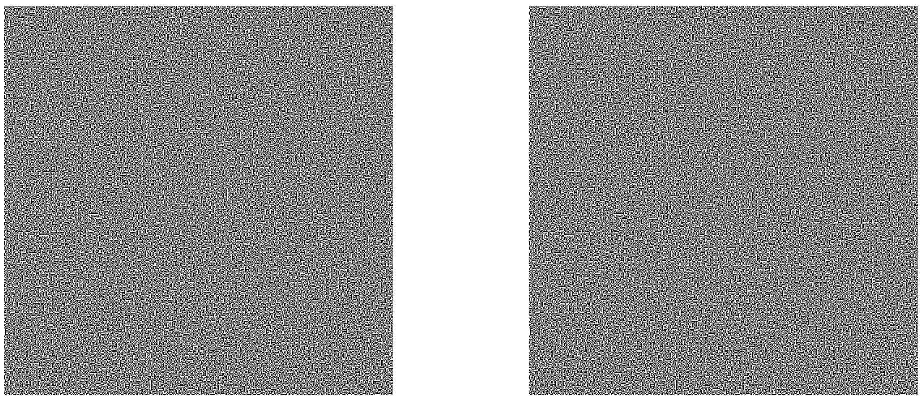

(a) Transparency 1 (b) Transparency 2

Figure 18: Transparencies.

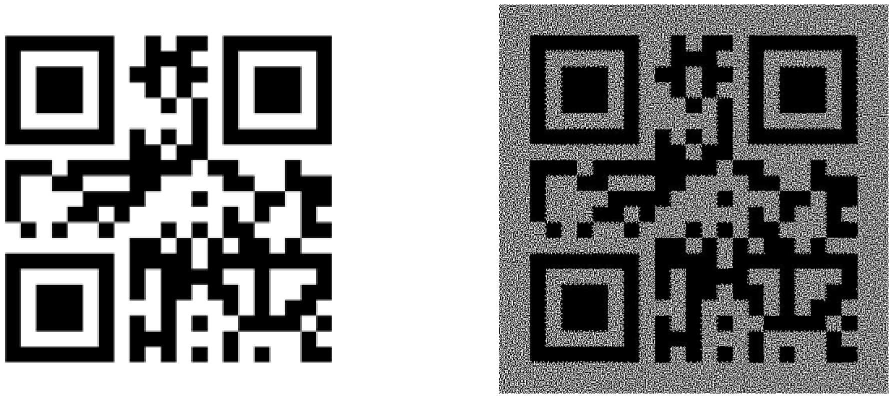

(a) Secret image (b) Overlapped transparencies

Figure 19: Original secret and overlapped transparencies.

3.2 General access structures for Visual Cryptography

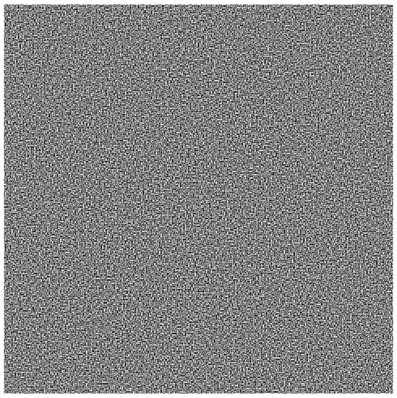
In [1] visual cryptography is extended to a set $\mathscr{P}$ of n participants in which each member receives a transparency for the secret image. Some qualified subsets of $\mathscr{P}$ are able to recover the secret image by overlapping their transparencies (Γ_{qual}), while some others are forbidden subsets (Γ_{forb}), having no way to recover the secret image.

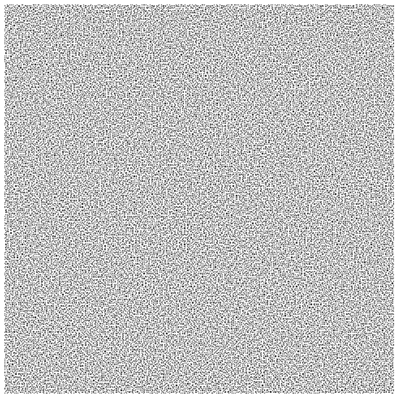
3.2.1 Example

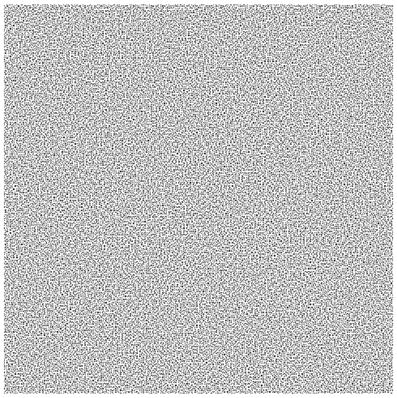
The following figures present a practical example of this extended scheme. In this case the set of participants and the one that defines the qualified accesses are the following:

- $\mathscr{P} = \{1, 2, 3, 4\}$
- $\Gamma_{qual} = \{(1, 2), (2, 3), (3, 4), (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4), (1, 2, 3, 4)\}$

As for the previous scheme, a QR code have been chosen as secret image (see Figure 24). In this case, since the cardinality of the participant set $\mathscr{P}$ is four, the secret image has been split into four transparencies one for each participant (see Figures 20a, 20b, 20c and 20d). At this point, overlapping their transparency the participants can reconstruct the original secret image if the resulting tuple (composed by the participants' id) belongs to the set Γ_{qual} of qualified accesses.
For example if participants 1 and 2 overlap their transparencies, they generate the secret since $(1, 2) \in \Gamma_{qual}$ (see Figure 21a). On the other hand if participants 1 and 4 overlap their transparencies they do not generate the secret since $(1, 4) \in \Gamma_{forb}$ (see Figure 22a).


(a) Transparency 1


(b) Transparency 2


(c) Transparency 3

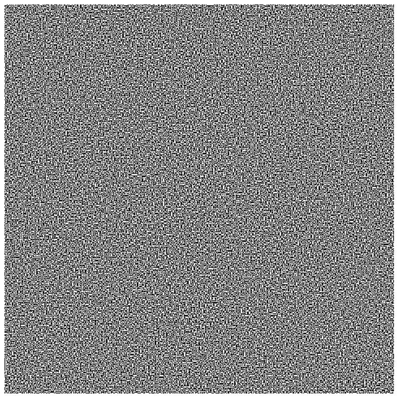
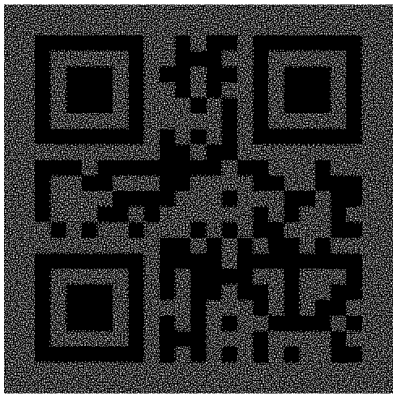
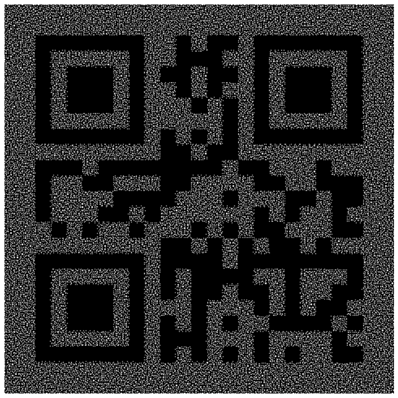
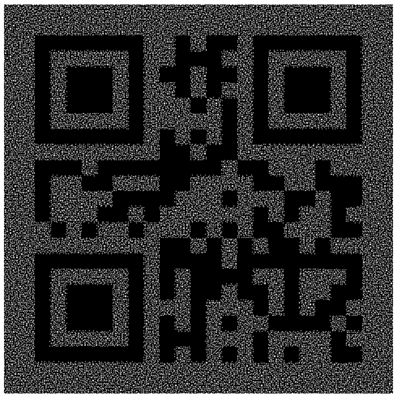

(d) Transparency 4

Figure 20: Participants in the access structure.


(a) 1 and 2 overlapped


(b) 2 and 3 overlapped


(c) 3 and 4 overlapped

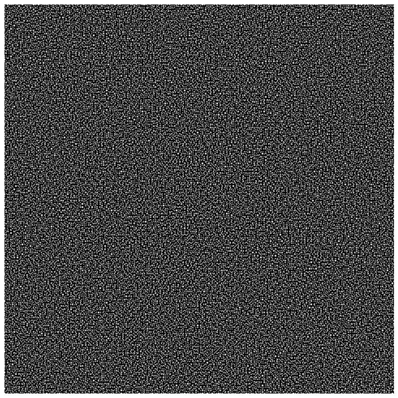
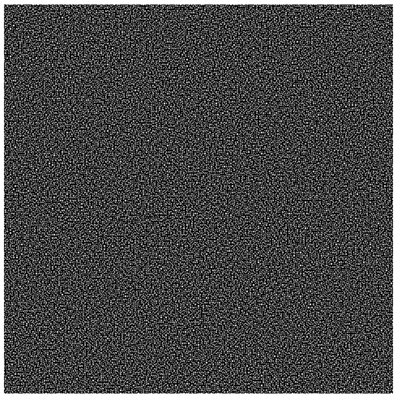
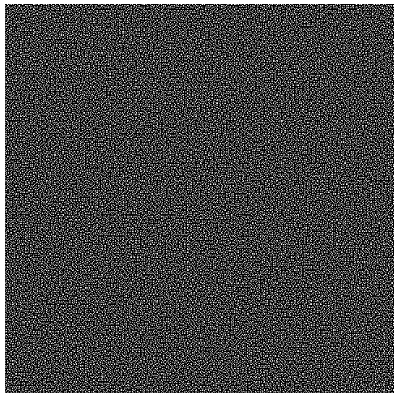
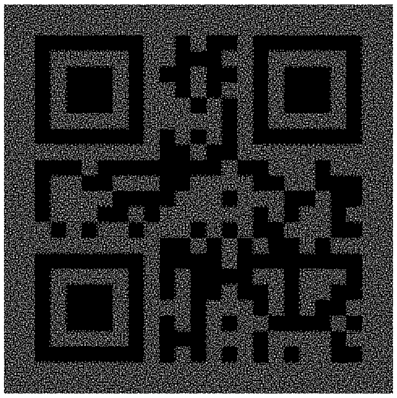

(d) 1 and 3 overlapped

Figure 21: Part 1 of the access structure.


(a) 1 and 4 overlapped


(b) 2 and 4 overlapped


(c) 1,2 and 3 overlapped

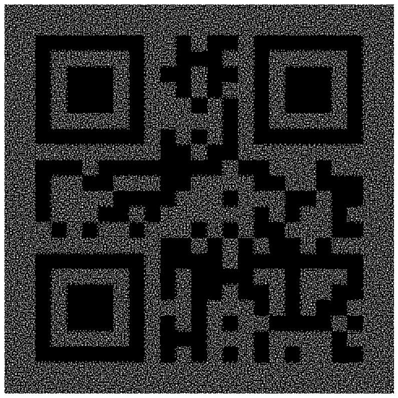
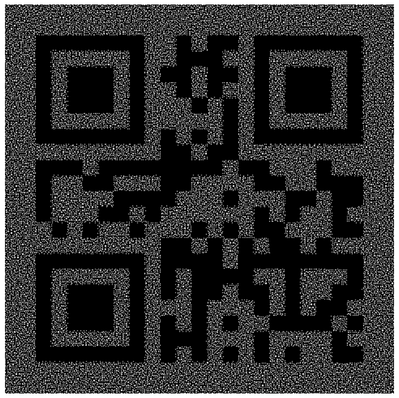
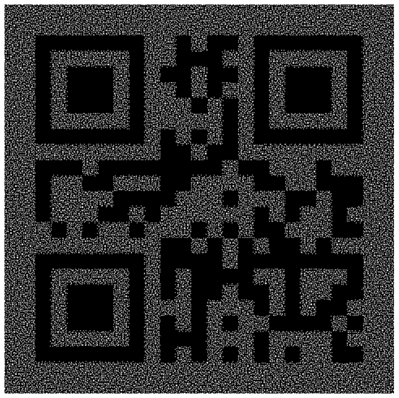

(d) 1, 2 and 4 overlapped

Figure 22: Part 2 of the access structure


(a) 1, 3 and 4 overlapped


(b) 2, 3 and 4 overlapped

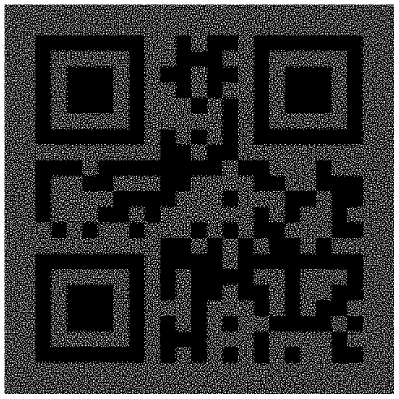

(c) 1, 2, 3 and 4 overlapped

Figure 23: Part 4 of the access structure.

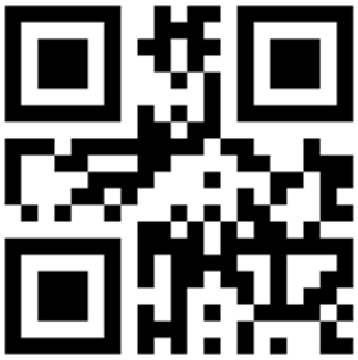


Figure 24: Secret image.

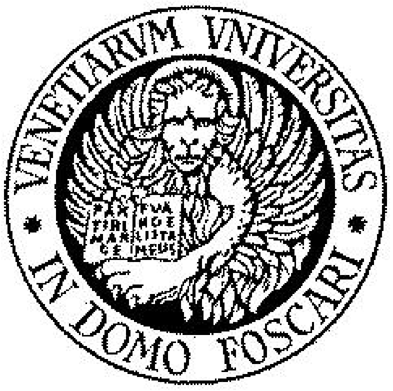
3.3 Extensions for Visual Cryptography

In [2] the visual cryptography scheme with access structure $(\Gamma_{qual}, \Gamma_{forb})$ on a set of participants $\mathscr{P}$ is extended with n images, one for each participant, that are encoded into transparencies. Differently from the previous model each pixel of the original participant's image is mapped into a share in order to fulfill the constrain defined by the set Γ_{qual} of qualified access. The transparencies generated are still meaningful, which means the participants can still recognize the original images.

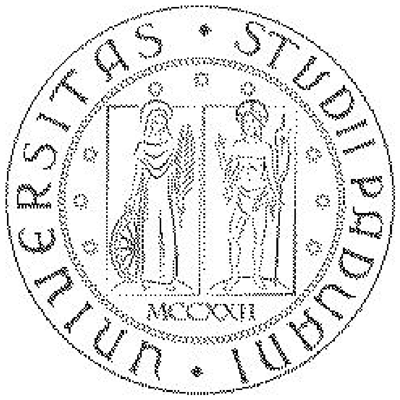
3.3.1 Example

In this example the set of participants and the one that defines the qualified accesses are the following:

- $\mathscr{P} = \{1, 2\}$
- $\Gamma_{qual} = \{(1, 2)\}$



(a) Image 1

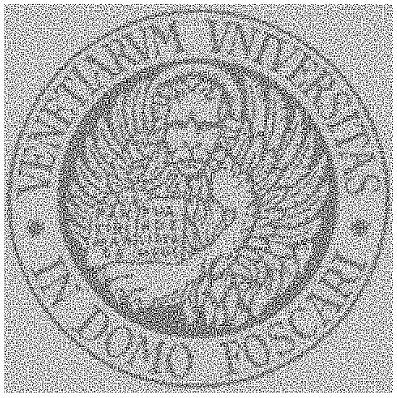


(b) Image 2

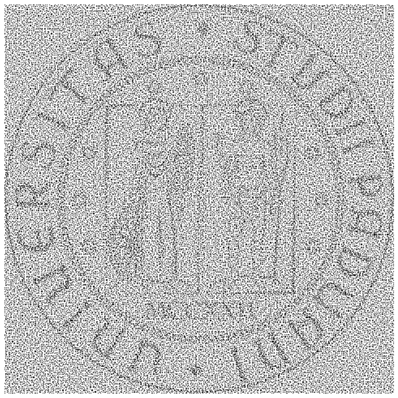
Figure 25: Participants' image.

As for the previous scheme a QR code have been chosen as secret image (see Figure 27a). Differently from the previous model the two transparencies are generated starting from the secret image and the participants' image which are transformed in order to respect the constrain defined by the set Γ_{qual} (Figures 25a and 25b show the participants' original image). Figures 26a and 26b show the generated transparencies and as can be seen they preserve the original form of the participants' image.

At this point, participants 1 and 2 can overlap their transparencies obtaining the secret since $(1, 2) \in \Gamma_{qual}$ (see Figure 27b).

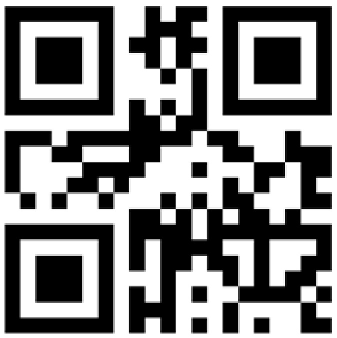


(a) Transparency 1

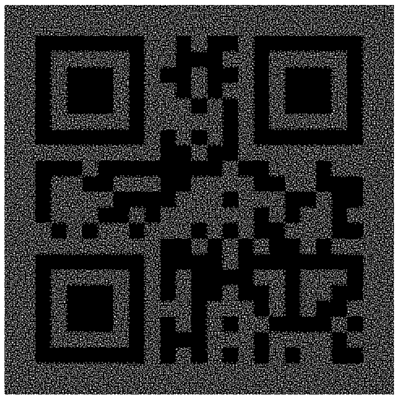


(b) Transparency 2

Figure 26: Transparencies.



(a) Secret image



(b) Overlapped transparencies

Figure 27: Original secret and overlapped transparencies

3.4 Visual key derivation function

In the previous sections two visual cryptography models were presented and in both of them the encrypted secrets were QR codes. It is thus clear that visual cryptography can be a powerful instrument to improve confidentiality of 2D barcodes since it allows to decrypt the content of the 2D barcode only by overlapping two or more transparencies, without needing any additional computation. On the other hand in reality it is impossible to scan a transparency as it is indistinguishable from a random noise and so it has no structure. As consequence, so far it is not possible to make this model portable. The solution to this problem would be to have transparencies that maintain the structure of a barcode so that they can be easily scanned by any mobile device. Below some of the first possible solutions to this problem, and their limits are presented.

Passage of transparency via URL: one solution to this problem could be the creation of a barcode containing an URL pointing to the transparency. The problem is that the transparency is stored and accessible online, as consequence an attacker could try to steal it.

Use of bluetooth to send transparencies: another solution that avoids this problematic is the exchange of the transparency between the devices, via bluetooth. Unfortunately this solution requires the use of a physical device causing for example problems of compatibility.

Compression algorithms for transparencies: in this solution the idea was to compress the image so that it could have been stored into a barcode. Unfortunately, for big images, the existing algorithms do not permit a sufficient compression to memorize the resulting image into the barcode.

Insertion of transparency into barcodes: splitting a transparency and storing each piece into different barcodes in order to let their consecutive scans retrieve the full original transparency. The problem with this implementation is that the barcodes required to generate the full image are too many.

When the aim is to improve confidentiality of 2D barcodes, all of these solutions, for different reasons, give no advantage with respect to standard cryptography. As a consequence a new way to generate transparencies was defined during this research. The idea was to generate random transparencies, without starting from the secret image, through the use of what we have called visual key derivation function. This technique uses a PBKDF (Password-Based Key Derivation Function 2) [11] which permits to produce a derived key starting from a password. In particular this function is defined as $DK = PBKDF2(PRF, Password, Salt, c, dkLen)$ where:

- PRF is a pseudorandom function of two parameters with output of length hLen (e.g. a keyed HMAC)
- Password is the master password from which a derived key is generated
- Salt is a sequence of bits, known as a cryptographic salt
- c is the desired number of iterations
- dkLen is the desired length of the derived key

- DK is the generated derived key

Every hLen-bit block T_i of derived key DK, is computed in the following way (|| identifies the concatenation operator):

- $DK = T_1 \quad || \quad T_2 \quad || \quad T_3 \quad || \quad ... \quad || \quad T_{dkLen/hLen}$

Where:

- $T_i = F(Password, Salt, c, i)$

The function F is the xor of c iterations of chained PRFs. The first iteration of PRF uses Password as the PRF key and Salt concatenated with i encoded as a big-endian 32-bit integer. The following iterations of PRF use Password as the PRF key and the output of the previous PRF computation as the salt:

- $F(Password, Salt, c, i) = U_1 \quad xor \quad U_2 \quad xor \quad U_3 \quad xor \quad ... \quad xor \quad U_c$

Where:

- $U1 = PRF(Password, Salt \quad || \quad INT\_32\_BE(i))$
- $U2 = PRF(Password, U1)$
  ...
- $Uc = PRF(Password, Uc - 1)$

At this point the generated derived key is transformed into a transparency. This process is done by mapping each character of the derived key in a set (of cardinality n) of pixels, called share, which will be part of the transparency. As a consequence it is important to correctly choose the length dkLen and the cardinality n of the share in order to obtain a transparency of the desired dimension. An example of this visual key derivation function is presented in Figure 28.

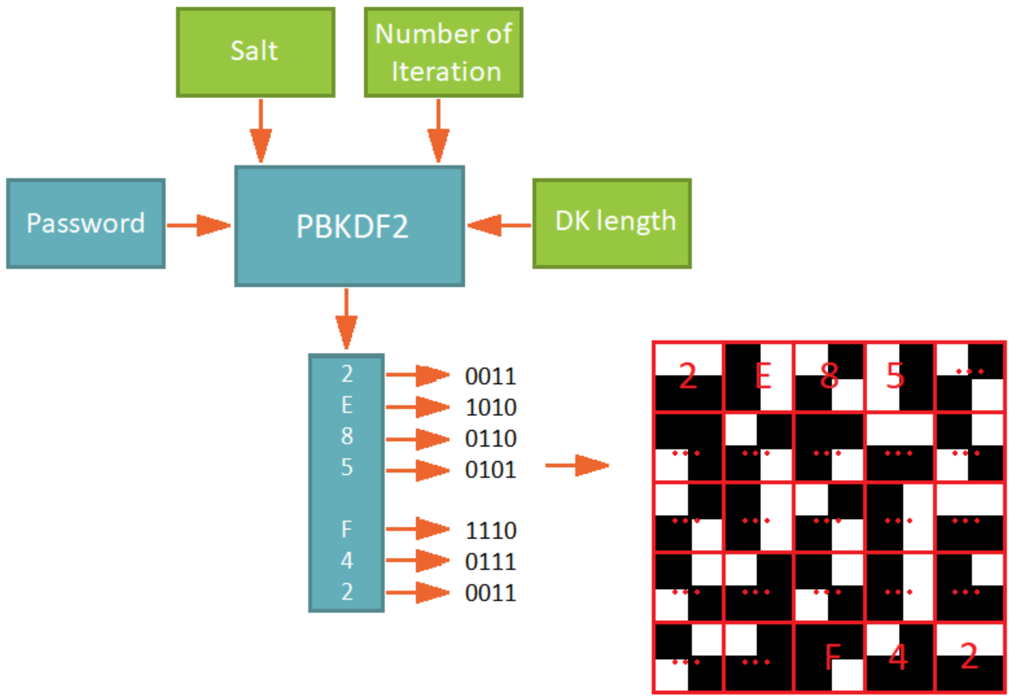


Figure 28: Visual key derivation function

In all the visual cryptography model described before the transparencies were generated starting from the secret image in order to respect the constrain defined by the set (Γ_{forb}) of qualified access. It is clear that, using a visual key derivation function to generate the transparencies, it is impossible to apply the previous model. For this reason in the next section will be presented two new models which use a visual key derivation function.

4 Shared key extensions for VCS

The analysed schemes so far, rely on at least two participants to recover a specific secret image. Our aim, for an access structure $(\Gamma_{qual}; \Gamma_{forb})$, is to let each subset Q of Γ_{qual} to have its specific secret, starting from a unique shared key k of $\mathscr{P}$ coupled with a transparency t of $\mathscr{P}$. Moreover, in a real world scenario, the transparency overlapping is not an easy task. To take advantage of visual cryptography an user should be able to recover the secret easily, for example by using a smartphone application to scan the transparencies.
As it was said before the proposed solution is the use of a visual key derivation function in order to generate the transparency starting from a password, which can be stored inside a 2D barcode making the model applicable in a real word scenario. In this section two specific solutions, based on the use of a shared key and a visual key derivation function, will be presented.

4.1 A pre-computed shared key model

Let $\mathscr{P}$ be the set of n participants in the access structure $(\Gamma_{qual}, \Gamma_{forb})$, where $\Gamma_{forb} = P(\mathscr{P}) \setminus \Gamma_{qual}$. Suppose that instead of one secret image there are n and that the n transparencies have to be meaningful with respect to the original participant's image. Therefore, $2n$ images are needed, n for the transparencies and n for the secret images. Moreover, in the model we want to achieve, a pre-computed shared key k is needed. As consequence $2n + 1$ are needed. The visual key derivation function, described in the previous subsection, is used to generate the shared key. Fixing the salt, the number of iteration, the length of the derived key and the starting password it is

possible to store the shared transparency into a 2D barcode.
In this way every participant, for example by using a smartphone application (which knows the salt, the number of iteration and the length of the derived key) to scan the 2D barcode, should be able to recover the shared key easily. In this scenario it is possible to generate different plaintexts starting from the same cyphertext, which is the shared key, using different keys, which are the participants' transparency. An interesting real word application of this model, which will be discussed better in the next chapter, is the one that allows to implement an access control structure.

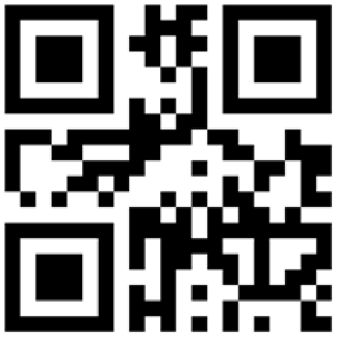
4.1.1 Example

This scheme differs from the one described above since it is necessary to generate a shared transparency, which works as a shared key. As consequence if the number of participants is n the cardinality of the set $\mathscr{P}$ will be n+1 (the id of this shared transparency will be n+1).
This additional transparency is generated totally randomly and belongs to all the tuples of the qualified accesses set. In this example the participants and qualified accesses sets are defined as follows:

- $\mathscr{P} = \{1, 2, 3\}$
- $\Gamma_{qual} = \{(1, 3), (2, 3)\}$

Where the participant's id 3 identifies the shared key transparency generated through the use of a visual key derivation function (see Figure 31c). As for the extended visual cryptography model described before each participant, with the exception of the one which is associated to the shared key, has its own original image (see Figures 30a and 30b). On the other hand instead of having only one secret image in this scheme there is one for each participant except for the one associated to the shared key (see Figures 29a and 29b). For each participant the transparency is generated starting from the associated secret image, its original image and the shared key transparency in order to fulfill the constraint defined by the set Γ_{qual} of qualified access. Figures 31a and 31b show the generated transparencies for participants one and two. As in the previous model the original image is recognizable if compared with the participant's transparency newly generated.

It is important to stress the fact that each tuple of Γ_{qual} identifies a different secret image. As a matter of fact if participant 1 overlaps its transparency with the shared key it obtains a secret image which is different from the one obtained overlapping the transparency of participant 2 (see Figures 31d and 31e).


(a) Secret image for participant 1

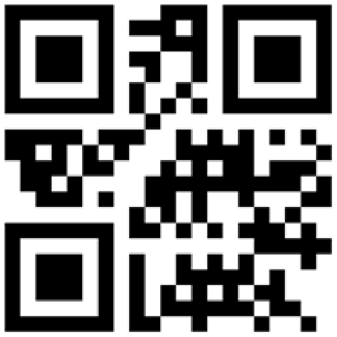
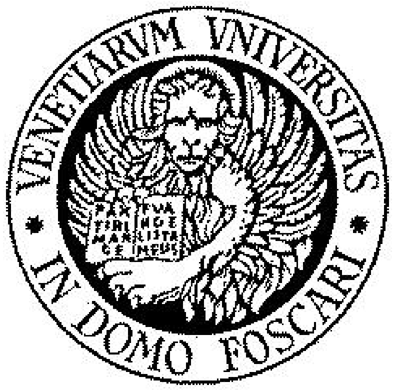
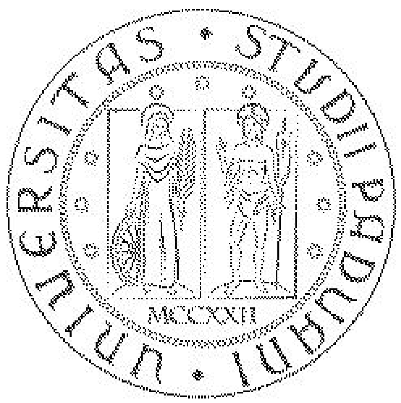

(b) Secret image for participant 2

Figure 29: Secret image.

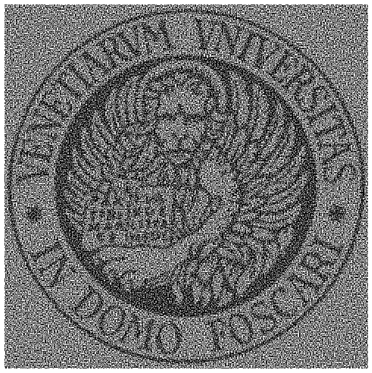


(a) Image for participant 1

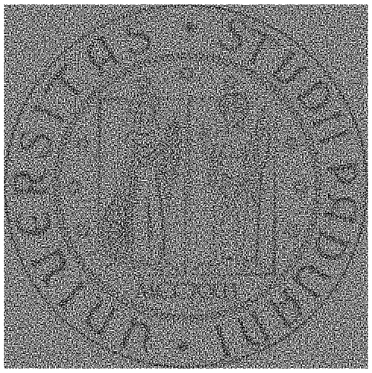


(b) Image for participant 2

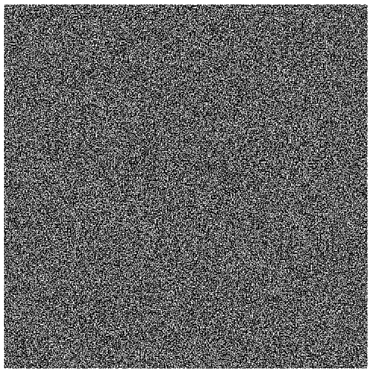
Figure 30: Starting image of the participant.



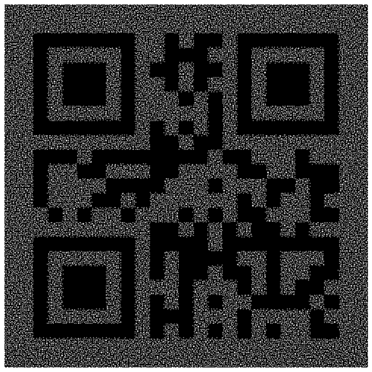
(a) Transparency for participant 1



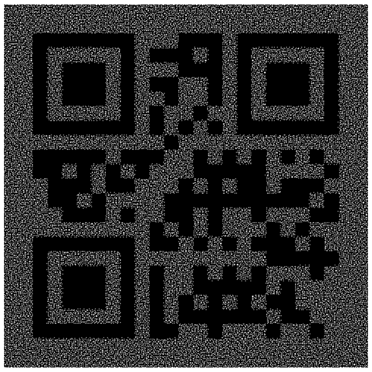
(b) Transparency for participant 2



(c) Shared key k



(d) Transparency 1 and k overlapped



(e) Transparency 2 and k overlapped

Figure 31: Example of a priori Shared key extended visual cryptography with two participants.

4.1.2 Implementation

This paragraph discusses the implementation choice that we have done during the development of this cryptographic scheme. In particular it focuses its attention on the description of the implemented cryptographic algorithms and the software design decisions that we have taken. The first consideration that has to be done regards the fact that the model just described is nothng more then a variation of the EVCS scheme. As a matter of fact in both the two models any user has to be able to recognize its starting image from its generated transparency so it is necessary to define two different classes of shares. This two sets allow to distinguish shares that can be interpreted as a white pixel from shares that identify a black pixel, allowing to generate transparencies which are still meaningfully with respect to the original participants' image. The examples provided in this chapter, for the pre-computed shared key model, work with shares composed by four subpixels but it is possible to extend them to a generic number of subpixels m. In order to distinguish them we built the following two classes of shares:

- White pixel: all the shares composed by $m = 4$ subpixels with Hamming weight equal to two.
- Black pixel: all the shares composed by $m = 4$ subpixels with Hamming weight equal to three.

Clearly, shares with Hamming weight equal to one or four are not used in the model. This is not a visual cryptography constrain, which has to be fulfilled, but a personal implementation choice. As for the EVCS model in the pre-computed shared key scheme there is the necessity to define the so called *basis matrices* [2]. The notation for these matrices is ${S_c}^{p_1,p_2}$, in which $c, p_1, p_2 \in \{0, 1\}$. The latter identify that in order to generate the i-th c pixel of the secret image (in which c identifies the color) two shares p_1 and p_2 have to be considered, which represent the i-th ones of the shared key and the associated transparency. The *basis matrices* used to implement the scheme are the following:

$$ {S_0}^{0,0} = \begin{bmatrix} 0011 \\ 1010 \end{bmatrix} \quad {S_0}^{0,1} = \begin{bmatrix} 0011 \\ 1011 \end{bmatrix} \quad {S_0}^{1,0} = \begin{bmatrix} 1011 \\ 1010 \end{bmatrix} \quad {S_0}^{1,1} = \begin{bmatrix} 1011 \\ 1011 \end{bmatrix} $$

$$ {S_1}^{0,0} = \begin{bmatrix} 0011 \\ 1100 \end{bmatrix} \quad {S_1}^{0,1} = \begin{bmatrix} 0011 \\ 1101 \end{bmatrix} \quad {S_1}^{1,0} = \begin{bmatrix} 1011 \\ 1100 \end{bmatrix} \quad {S_1}^{1,1} = \begin{bmatrix} 1011 \\ 1101 \end{bmatrix} $$

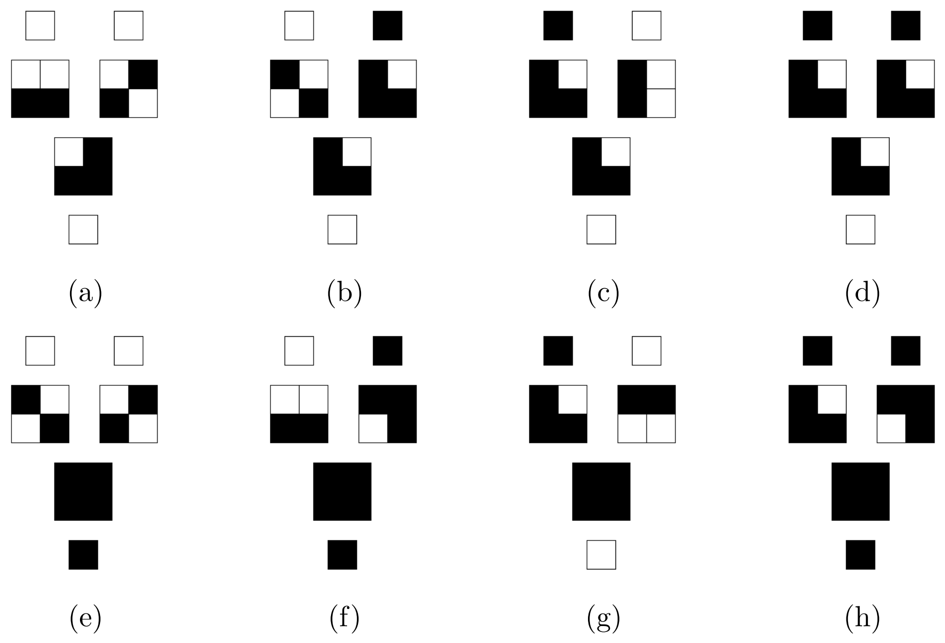


Figure 32: *Basis matrices.*

Starting from each *basis matrix* ${S_c}^{p_1,p_2}$ it is possible to define the associated *collection* ${C_c}^{p_1,p_2}$. The generic *collection* ${C_c}^{p_1,p_2}$ contains all the matrices obtained by the permutation of the columns of the associated *basis matrix* ${S_c}^{p_1,p_2}$. In this way all the possible couples of shares, which produce a black or white secret, are generated.
At this point using the previous definitions it is possible to define the two algorithms required to generate respectively the shared key and a generic transparency of the model. The first procedure, the one used to generate the shared key, is listed below and simply consists in the pseudocode for the implementation of the visual key derivation function described in the previous paragraphs.

Algorithm 1 SKGenerator

```
1: procedure SKGenerator($skLength, validShares$)
2:     $password \leftarrow randomPwd()$
3:     $salt \leftarrow randomSalt()$
4:     $iteration \leftarrow randomItr()$
5:     $pbkdf2 \leftarrow PBKDF2(password, salt, iteration, skLength)$
6:     $dictionary \leftarrow map\ between\ characters\ and\ validShares$
7:     $SK \leftarrow$ ""
8:     for all $ch \in pbkdf2$ do
9:         $SK \leftarrow SKt + dictionary.GET(ch)$
10:    end for
11:    return $SK$
```

The second procedure, the one used to generate the i-th participant's transparency, is listed below. The algorithm takes as input four parameters: the secret image, the original participant's image, the previously generated shared key and the collections ${C_c}^{p_1,p_2}$ $\forall c, p_1, p_2 \in 0, 1$. The function for each tuple (c, sk, o), in which c and o identify the color of the i-th pixel of the secret and original images and sk the color of the i-th share of the shared key, performs the following operations. First of all the pair which contains the share sk is searched in the corresponding collection ${C_c}^{sk,o}$. Secondly the share which is coupled with sk is extracted and assigned to the i-th position of the participant's transparency. The following listing shows the pseudocode instructions required to generate a generic transparency.

Algorithm 2 TransparencyGenerator

```
 1: procedure TransparencyGenerator($secretImage, originalImage,$
    $sharedKey, {C_0}^{p_1,p_2}, {C_1}^{p_1,p_2}$)
 2:     $transparency \leftarrow$ ""
 3:     for $i \leftarrow 1, LEN(secretImage)$ do
 4:         $skShare \leftarrow extract\ i-th\ share\ of\ the\ sharedKey$
 5:         if $secretImage[i]\ is\ WHITE$ then
 6:             if $originalImage[i]\ is\ WHITE$ then
 7:                 if $skShare\ identify\ a\ WHITE\ share$ then
 8:                     $Search\ in\ {C_0}^{0,0}\ for\ a\ pair\ with\ shared\ key\ skShare$
 9:                 else
10:                     $Search\ in\ {C_0}^{0,1}\ for\ a\ pair\ with\ shared\ key\ skShare$
11:                 end if
12:             else
```

```
13:             if $skShare\ identifies\ a\ WHITE\ share$ then
14:                 $Search\ in\ {C_0}^{1,0}\ for\ a\ pair\ with\ shared\ key\ skShare$
15:             else
16:                 $Search\ in\ {C_0}^{1,1}\ for\ a\ pair\ with\ shared\ key\ skShare$
17:             end if
18:         end if
19:     else
20:         if $originalImage[i]\ is\ WHITE$ then
21:             if $skShare\ identify\ a\ WHITE\ share$ then
22:                 $Search\ in\ {C_1}^{0,0}\ for\ a\ pair\ with\ shared\ key\ skShare$
23:             else
24:                 $Search\ in\ {C_1}^{0,1}\ for\ a\ pair\ with\ shared\ key\ skShare$
25:             end if
26:         else
27:             if $skShare\ identifies\ a\ WHITE\ share$ then
28:                 $Search\ in\ {C_1}^{1,0}\ for\ a\ pair\ with\ shared\ key\ skShare$
29:             else
30:                 $Search\ in\ {C_1}^{1,1}\ for\ a\ pair\ with\ shared\ key\ skShare$
31:             end if
32:         end if
33:     end if
34:     $transparency \leftarrow transparency + share\ founded\ in\ {C_c}^{p_1,p_2}$
35: end for
36: return $transparency$
```

The implementation of this two procedures has been done in *python* since it is a really versatile language. In particular *python* has been chosen since it provides the module PIL and *hashlib*. The first is a library that allows to handle and build images in a very easy way. The second one defines a common interface to different secure hash and message digest algorithms, allowing to easily implement the visual key derivation function.

4.1.3 Scalability and performance

In Computer Science the concept of scalability is defined as the capability of a system to adapt to a growing workload. In particular, with respect to the model previously described, this feature holds properly since the generation of each transparency is independent from the other. Consequently the addition of a new transparency does not require the recomputation of the ones previously evaluated. As a matter of fact the i-th transparency is generated starting from the precomputed shared key and the i-th participant's secret, therefore each transparency requires more or less the same amount of time to be created.

Let us consider a scenario in which the cardinality of the set $\mathscr{P}$ of participant is n and a new participant has to be added. The time required to generate the new participant's transparency is equal to the one required to generate the n+1-th transparency in a model in which the cardinality of the set $\mathscr{P}$ is n+1 and the new participant already belongs to $\mathscr{P}$.

In Figure 33 the blue line presents the relation between the computational cost (computed as the average of more analysis measured in seconds) taken by the model to generate the transparencies and the number of participants.

It is thus clear that the relation is directly proportional and this is a consequence of the fact that each transparency is computed independently.

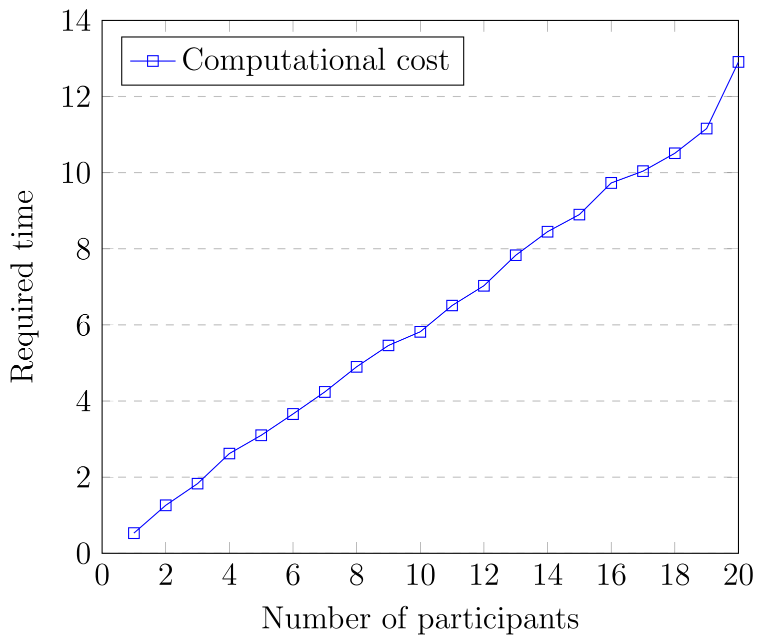


Figure 33: Computational cost of the model with respect to the number of participants

In Figure 34 an histogram presents the comparison of the computational cost for the generation of a generic i-th transparency and the addition of a new one in a model with n participants. It is interesting to stress the fact that this two operations have more or less the same computational cost, implying that the model is salable in time for what concern the addition of new users.

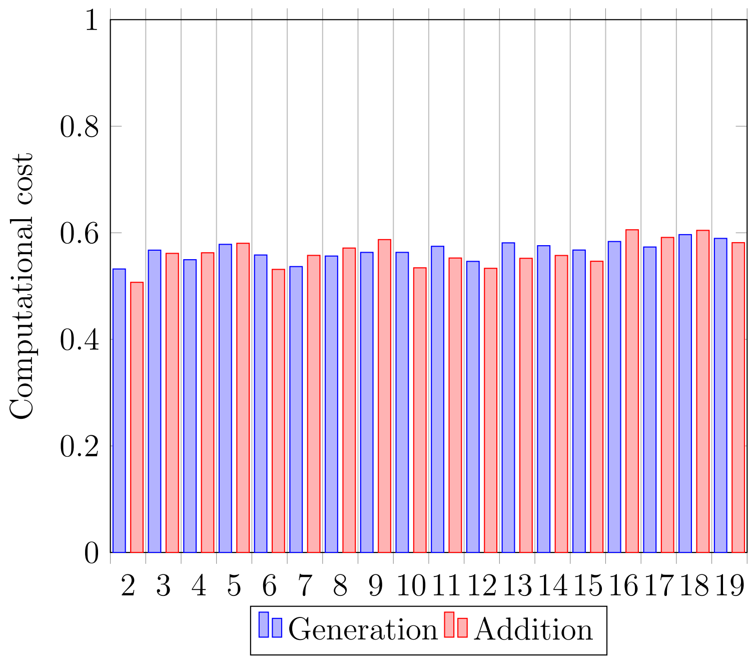


Figure 34: Comparison between computational cost for the generation of a generic transparency and the addition of a new one in a model with n participants.

4.2 A pre-computed cyphertext model

Let $\mathscr{P}$ be the set of n participants, each one with its respective secret image. We want to generate the shared key k starting from n random generated transparencies, such that each participant's transparency, overlapped with it, generates the secret image for that participant.

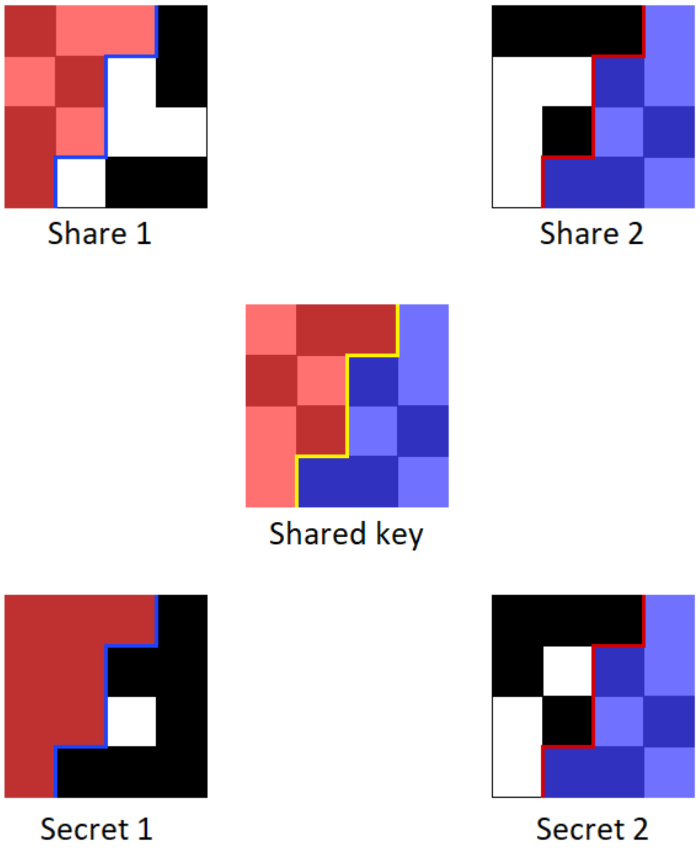


Figure 35: Importance of the subpixels of a share with respect to the shared key, in a scheme with 2 participants

Differently from the previous model, the visual key derivation function is used in order to generate randomly the n transparencies starting from a password and fixing the other parameter. Each pixel in the secret image becomes a set of $n \times m$ subpixels (share) in the transparencies. In this way, we can generate k by giving importance to m specific subpixels for each participant, such that by overlapping them we get information of the ith secret image (Figure 35).

In this way every participant, for example by using a smartphone application (which knows the salt, the number of iteration and the length of the derived key) to scan the 2D barcode, should be able to recover the shared key easily. In this scenario it is possible to generate different plaintexts starting from the same cyphertext, which is the shared key, using different key, which are the participants' transparency. An interesting real word application of this model, which will be discussed better in the next chapter, is the one that allows to implement an access control structure.

4.2.1 Example

In this example the participants and qualified accesses sets are defined as follows (as for the previous model there is an additional transparency which works as shared key):

- $\mathscr{P} = \{1, 2, 3\}$
- $\Gamma_{qual} = \{(1, 3), (2, 3)\}$

Where the participant's id 3 identifies the shared key transparency. In this use case the participants' transparency, except for the transparency which

works as shared key, are generated totally randomly through the use of a visual key deviation (see Figures 37a and 37b). At this point the shared key is computed in order to fulfill the constrain defined by Γ_{qual} starting from the two transparencies (see Figure 37c).
Finally, if participant 1 overlaps its transparency with the shared key it obtains a secret image which is different from the one obtained overlapping the transparency of participant 2 (see Figure 37a and 37b).

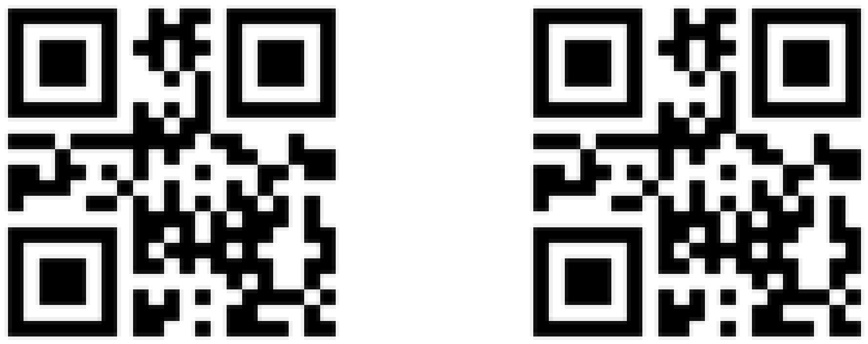
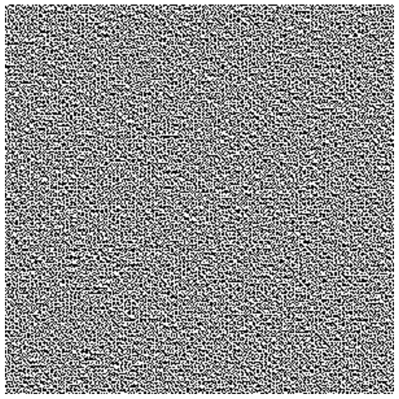
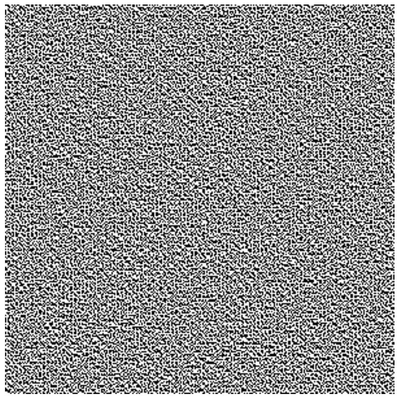
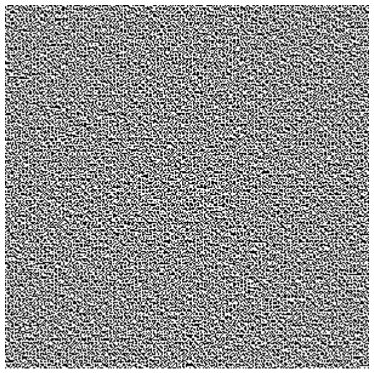
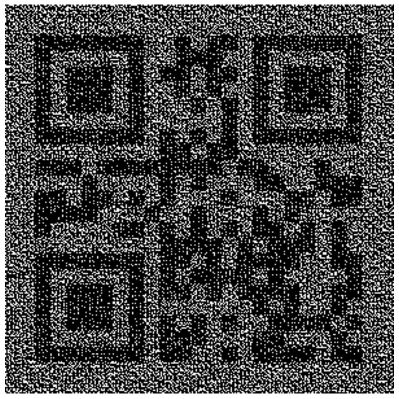

(a) Secret image 1 (b) Secret image 2

Figure 36: Secret images


(a) Trasparency 1


(b) Trasparency 2


(c) Shared key


(d) Secret image 1 overlapped

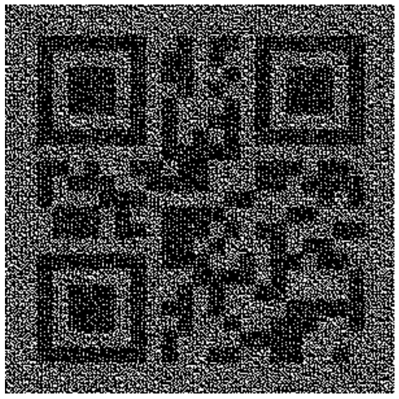

(e) Secret image 2 overlapepd

Figure 37: Example of a pre-computed cyphertext model with two participants.

4.2.2 Implementation

As for the pre-computed shared key model, this paragraph will discuss the implementation choice that we have done during the development of this cryptographic scheme. In particular it focuses its attention on the description of the implemented cryptographic algorithms and the software design decisions that we have taken. In this model the n transparencies are generated totally randomly through the visual key derivation function, while the shared key is built starting from them. As a consequence for what concerns the generation of the transparencies it is sufficient to apply the algorithm defined for the generation of the shared key in the pre-computed shared key model. On the other hand in this scheme the generation of the shared key is a bit more complicated. As a matter of fact, in order to implement the algorithm, which generates the shared key, it is necessary to consider n standard visual cryptography schemes in which the set of participants $\mathscr{P}$ and the qualified access structure Γ_{qual} are defined as following:

- $\mathscr{P}_i = \{s_i, t_i\} \ \forall i \in [1, n]$
- $\Gamma_{qual\ i} = \{(s_i, t_i)\} \ \forall i \in [1, n]$

Where $t_i \ \forall i \in [1, n]$ is equal to the i-th participant's transparency, except for the fact that for each share only the subset of m important subpixels is considered (see Figure 35). Starting from the i-th participant's secret and the transparency t_i, $\forall i \in [1, n]$, it is possible to define n different s_i transparencies, one for each standard visual cryptography schemes. The shares of these n resulting transparencies identify the different subset of m important subpixels of the shares of the shared key that we want to generate for the pre-computed cyphertext model. As for the previous scheme it is necessary to define the *basis matrices* and the *collections* which result to be simpler since there is no constrain on the shared key and the transparencies except for the one defined by the set of qualified access structure Γ_{qual}. The basis *basis matrices* used to implement the scheme are the following:

$$S_0 = \begin{bmatrix} 0011 \\ 0011 \end{bmatrix} S_1 = \begin{bmatrix} 0011 \\ 1100 \end{bmatrix}$$

The collections C_0 and C_1 contain all the matrices obtained by the permutation of the columns of the associated *basis matrices* S_0 and S_1. The algorithm for the generation of the shared key is listed bellow.

Algorithm 3 SKGenerator

```
1: procedure SKGenerator($secretImages, transparencies, C_0, C_1$)
2:     $sharedKey$
3:     for $i \leftarrow 1, LEN(secretImages)$ do
4:         for $j \leftarrow 1, LEN(secretImages[i])$ do
5:             $tShare \leftarrow extract\ j-th\ share\ of\ transparencies[i]$
6:             if $secretImages[i][j]\ is\ WHITE$ then
7:                 $Search\ in\ C_0\ for\ a\ share\ tShare$
8:             else
9:                 $Search\ in\ C_1\ for\ a\ share\ tShare$
10:            end if
11:            $add\ the\ share\ founded\ in\ C_c\ to\ the\ sharedKey\ in\ the\ correct$
12:              $position$
13:        end for
```

```
14:     end for
15:     return $sharedKey$
```

As for the previous model, the programming language used to implement the sheme is *python*.

4.2.3 Scalability and performance

In this paragraph the scalability of this second model will be analysed. Unfortunately, the results for what concerns the scalability of the scheme are not satisfactory as the one of the previous model. This is a consequence of the fact that in this model all the participant's transparency are generated randomly through the use of a visual key derivation function starting from a master password. Moreover, the shared key is defined starting from the n precomputed transparencies and the n secret images. Consequently, the addition of a new transparency requires the recomputation of the shared key. This lack of scalability is a consequence of the fact that in this model the participant's transparencies are precomputed. If we consider the example of Figure 35 the addition of a new participant's share results to be a problem. In the example the two transparencies' shares and the shared key one have dimension $n \times n$ (4×4), but only a subset of the pixels (the $m = \frac{n^2}{2} = \frac{16}{2} = 8$ red one) is involved in the generation of the first secret share, the remaining one (the $m = \frac{n^2}{2} = \frac{16}{2} = 8$ blue one) instead is used for the generation of the second secret share. Obviously, the addition of a new participant's share requires the modification on the scheme presented in Figure 35. As a matter of fact if the wish is to give again importance to $m = 8$ specific subpixels for both the two old transparencies and the new one, the dimension of the shares have to be increased. Consequently the two old transparencies have to be thrown away and recomputed together with the added one in order to have the required dimension. Moreover, since the dimension of the shares is increased, the organization of the important subpixel has to be redefined. Figure 38 shows how the example of Figure 35 could be extended for a new participant.

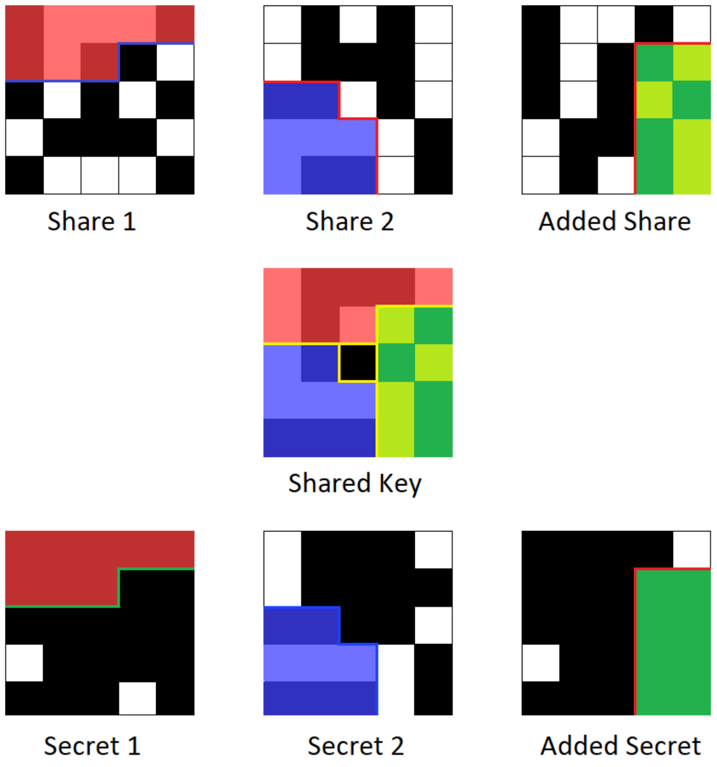


Figure 38: Importance of the subpixels of a share with respect to the shared key, for the scheme presented in Figure 35 extended to a new participant. The set Γ_{qual} of qualified access is extended with the pair $(AddedShare, SharedKey)$.

The two old shares are recomputed in order to have greater size, while the shares for the new participant is generated through the visual key derivation function. The modification of the structure, as a consequence, requires the recomputation of the shared key share introducing additional computation time. Therefore, if the model has already been computed for n-1 participants and a new one is added, the computational cost of this operation will be equal to the cost of recomputing the scheme for n participants.

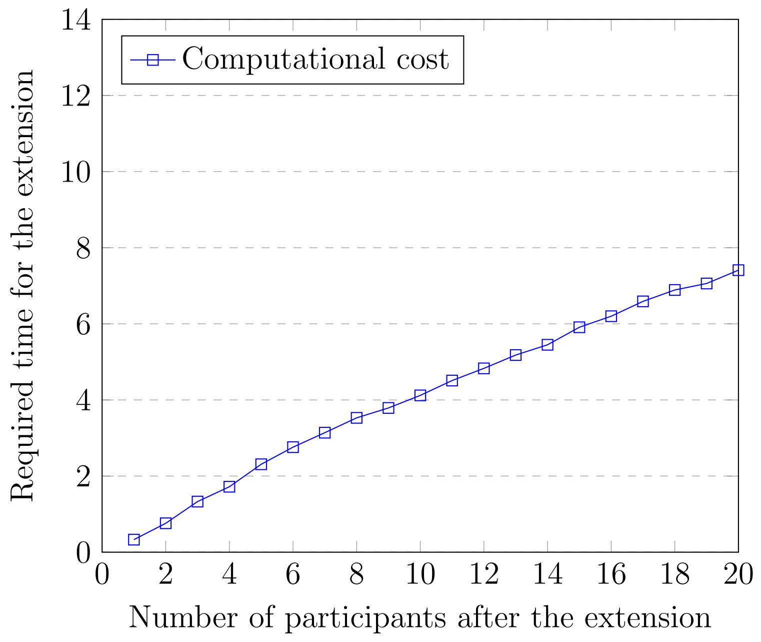


Figure 39: Computational cost of the model with respect to the number of participants

Figure 40 presents the relation between the time required to generate the shared key and the number of participants. Clearly, higher will be the number of participant higher will be the dimension of the transparencies and the shared key. As a consequence the time required to generate the shared key will increase directly proportional with respect to the number of participants. Finally, considering the fact that only a specific subsection of the shared key pixel is involved in the generation of the i-th secret, the readability of the resulting secret QR code depends on the number of participants. In Figure 40 the histogram represents the relation between the percentage of correct decodifications of the secret QR code obtained using the model and the number of participants.

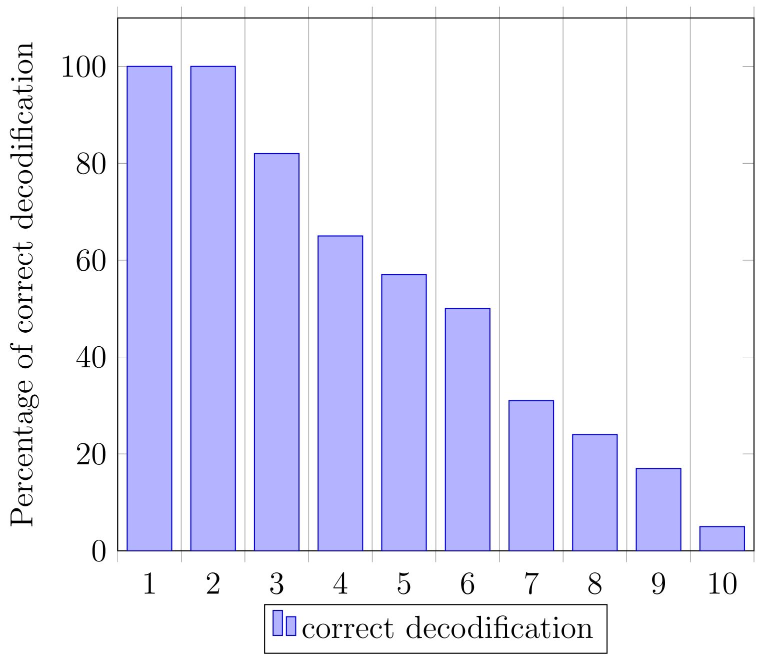


Figure 40: Readability of the QR codes secrets with respect to the number of participants.

The percentages presented in the chart have been obtained by scanning the secret QR codes (with different illumination, angle and distance) and counting the number of the exact decodifications.

5 Use cases

In this chapter two real use cases, for the models described before, will be presented and analysed. In particular one of them will use the pre-computed shared key model while the other will provide a practical application of the pre-computed cyphertext model.

5.1 Role-based access control

This first practical application is based on the scheme that we have called a pre-computed shared key model. In computer systems, a role-based access control is a technique which allows to restrict the access to a certain resource only to authorized users. When the purpose is to obtain restricted access to the resources (which in this scheme are QR codes), the pre-computed shared key model suits perfectly. As a matter of fact using this scheme it is possible to define a shared key transparency, which works as cyphertext, that can be turned into different QR codes depending on which participants' transparency is used, which works as the secret key needed.
Figure 41 shows how to define a role-based access control structure. In this example there are three classes of users: administrators, public users and guest users. Each class of user has its own secret transparency which, if overlapped with the shared key, generates different secrets depending on the user's authorizations. So far this role-based access control structure can be implemented through the use of the classical visual cryptography schemes.

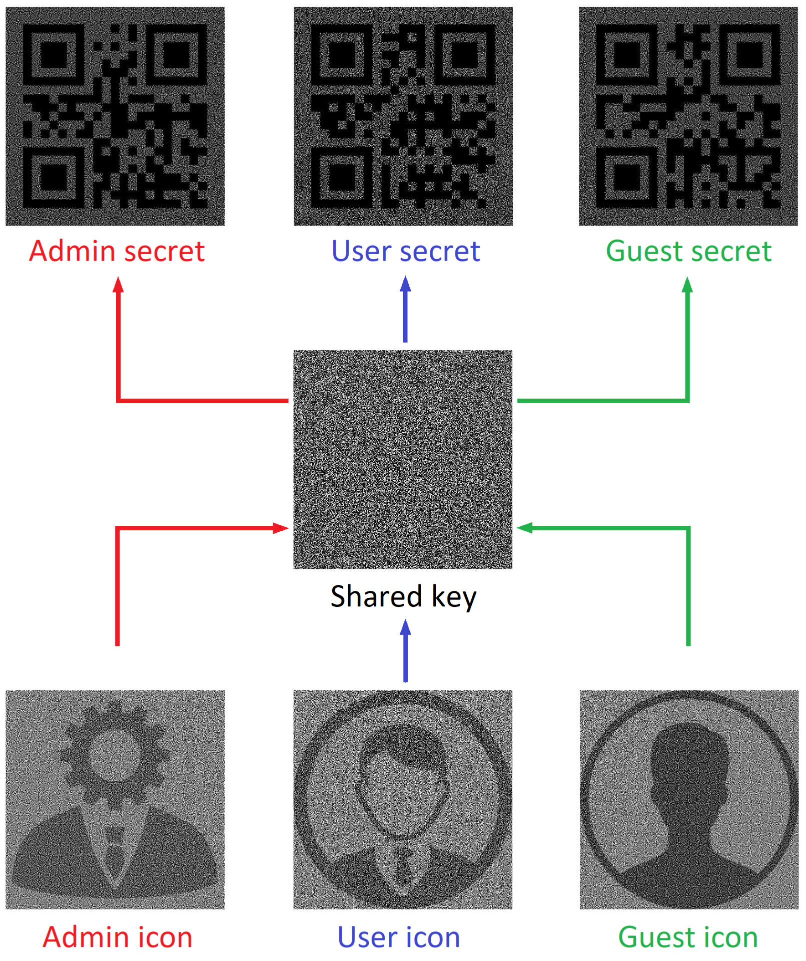


Figure 41: Role-based access control based on the pre-computed shared key model.

The innovation of the proposed scheme is the fact that the shared key is generated through the visual key derivation function, defined in the previous section. This new feature allows to store the shared key into a QR code as the password, the salt and the length of the derived key and the number of iterations needed to generate it with the visual key derivation function.

In this way the presented model is applicable in scenarios such as: supermarkets, warehouses, pharmacies, etc. The possibility to store the entire shared key inside a QR code allows to make the model portable. Indeed it is possible to attach the QR code, which contains the shared key, on a physical item such as products of a supermarket, a warehouse or a pharmacy. At this point the users can scan the QR code and, using the stored information, generate the shared key. The crucial point is that an user, by overlapping its own transparency, obtains a secret which depends on its role (e.g. customers, vendors, supervisors, etc).

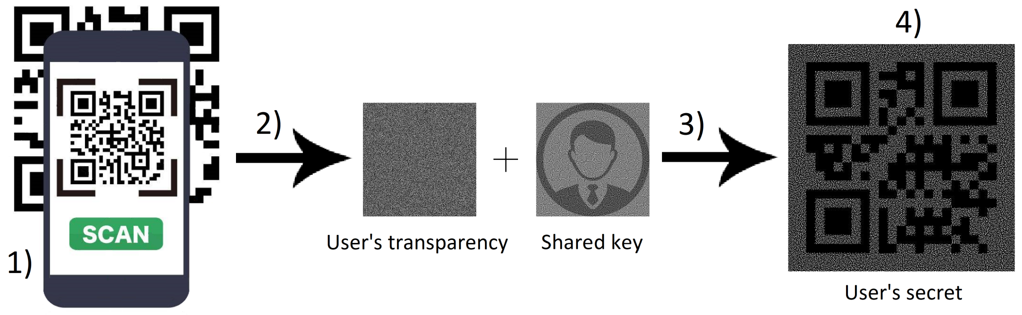


Figure 42: Real use case scenario.

An example of this real use case is presented in Figure 44. In particular the user has to perform the following steps:

1. Using a trusted device the user scans the QR code extracting the password, the salt and the number of iteration.

2. Through the visual key derivation function and the information generated at point 1), the shared key is generated.

3. The user's transparency and the just generated shared key are overlapped obtaining the secret image.

4. The obtained overlapped image is scanned decoding the content of the QR code.

In this scheme the fact that everyone can generate the shared key, since the password, the salt and the number of iteration of the visual key derivation function are public, is not an issue. This is a consequence of the fact that the shared key in this schemes works as cyphertext and does not carry any information about the different participants' secret, which are the different encoded plaintext.

5.2 User authentication

In this second subsection a real application, based on the scheme that we have called a pre-computed cyphertext model, will be presented. The common purpose of this two examples is to provide a validation to the users of a certain environment. As a consequence the pre-computed cyphertext mode suits perfectly for this necessity. The general idea behind this use case is to collect all the participants' information and than to store them into QR codes. Once performed this operation the QR codes will be used as secrets for the pre-computed cyphertext model.
The participants' transparency are generate through the visual key derivation function and are used together with the participants' QR code in order to generate the shared key. Figure 43 shows how to define a scheme for user authentication. In this example there are three different users and each of

them has its own secret transparency which, if overlapped with the shared key, generates the user access credentials. So far this user authentication model can be implemented through the use of the classical visual cryptography schemes.

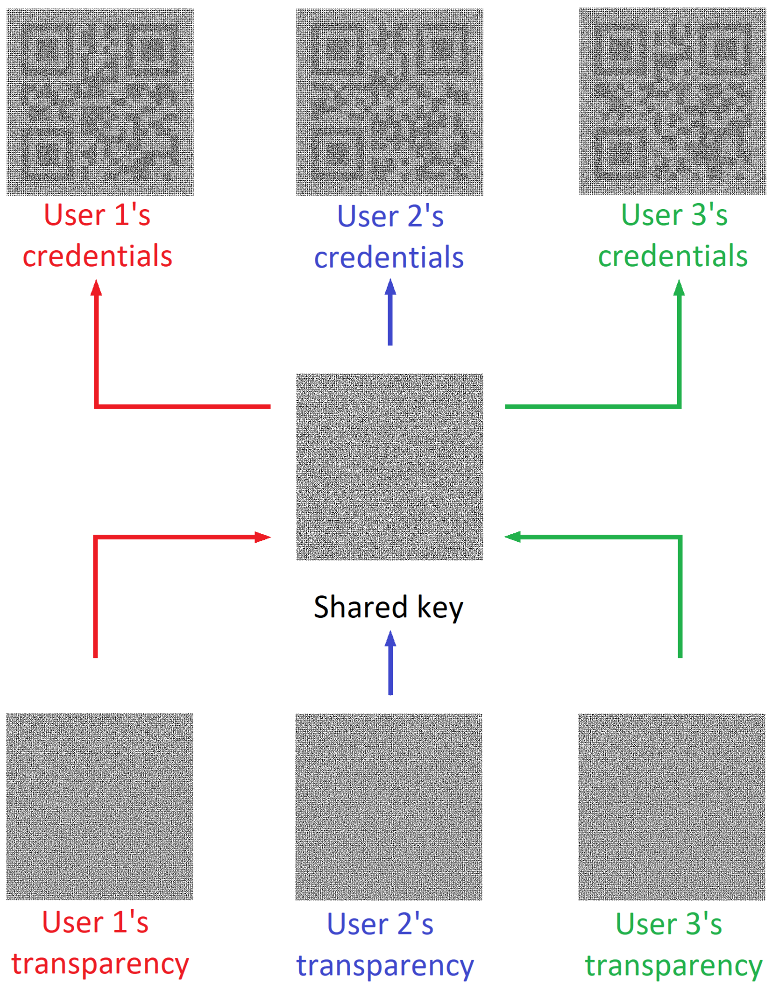


Figure 43: User authentication sheme based on the pre-computed cyphertext model.

The innovation of the proposed scheme is the fact that the participants' transparency are generated through the visual key derivation function, defined in the previous section. In this way the transparencies can be stored into QR codes as the password, the salt, the length of the derived key and the number of iterations needed to generate them with the visual key derivation function. In this way the transparency of the users can be inserted into a badge or a similar device which will allow the users to provide their credentials in order to obtain access to a certain environment.
It is really important to keep secret the mapping table which turns the character of the derived key into the subpixel of the transparency. This requirement is needed so that a malicious user is not able to generate the transparency of other participants.

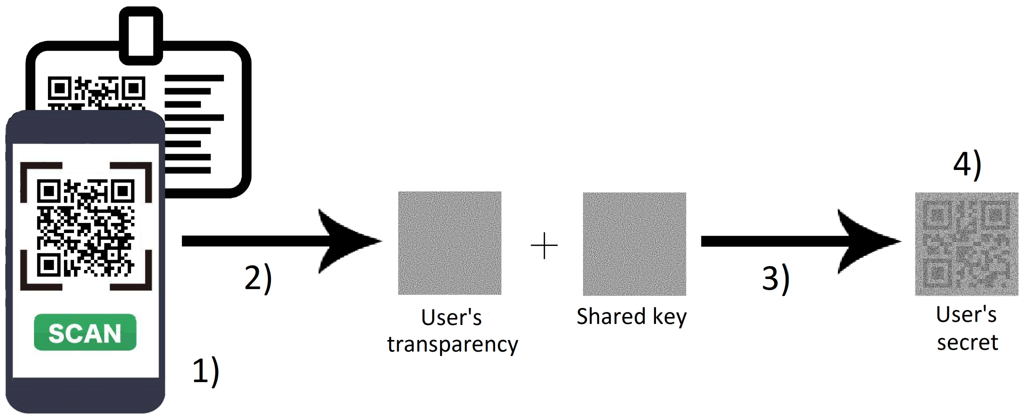


Figure 44: Real use case scenario.

An example of this real use case is presented in Figure 44. In particular the user has to perform the following steps:

1. The user's badge is scanned through a device which extracts the infor-

mation stored inside the QR code.

2. The application used by the device to scan the QR code, using the password, the salt, the number of iteration and the length of the derived key, generates the user's transparency through the visual key derivation function. It is important to stress the fact that this application has to know the mapping table which allows to transform each character of the derived key into a subset of pixels of the transparency.

3. The user's transparency just generated and the shared key, which is stored inside the device application, are overlapped obtaining the secret image.

4. The obtained overlapped image is scanned decoding the content of the QR code extracting the user's credentials. This information will be used to provide authorization for the user for a certain resource.

5.2.1 Ticket validity

An interesting real use case for the model just defined is the one in which this visual cryptography scheme is used in order to certificate the validity of tickets improving the confidentiality of the information stored in them.
Anyone who has flown in the last decade have seen that by now every boarding pass contains a barcode which is scanned upon boarding the airplane. The barcode printed on the boarding passes may allow a malicious attacker to discover personal information and the future travel plans of the passengers. In this scenario the model just defined results to be very useful since it allows to encrypt the content of the barcodes printed on the boarding passes

improving the confidentially of the information stored in them.

The only limitation in the application of this scheme is the fact that as shown in Figure 40 the readability of the secret decreases directly proportionally with respect to the number of participants. As a consequence, for reason of usability, the number of participants should be less than or equal to three. Clearly if the wish is to apply this scheme in order to improve the confidentiality of the information stored in the barcodes printed on the boarding passes, a fundamental modification has to be introduced. To avoid this limitation it is possible to store an additional parameter inside the barcodes that identifies which are the subpixel of each share of the shared key that are related to the participant's secret (see Figures 35 and 38). In this way all the noise introduced by the subpixel of the shares which are not related to the secret image is removed. With the introduction of this additional parameter, the model overcomes the limitation for what concerns the number of users and it allows to improve the confidentiality of the boarding passes barcode.

6 Conclusion

As we have already pointed out in the previous chapters, 2D barcodes have been in used for many decades, but so far very little work has been done on barcode confidentiality.

As a matter of fact in the scientific literature it is possible to find only a few solutions which provide the feature of confidentiality for barcodes. Furthermore all the proposed solutions are based on classical cryptography (e.g., the application of RSA or AES encryption) which merely encrypts the content of the barcode.

The aim of this thesis was to study the state of the art for what concerns the techniques of visual cryptography inspecting their advantages and disadvantages and trying to define new schemes which allow to provide confidentiality for barcodes. The approach of visual cryptography was appealing for two main reasons. The first was the fact that it is an encryption technique in which the decryption process does not require any cryptographic computation, the second one was a consequence of the fact that barcodes are images and visual cryptography is applicable to these.

The core of our research was the definition of two new models which have the same advantages of the known visual cryptography techniques but that try to overcome the well-known limitations of portability. As shown in the previous section, visual cryptography schemes split the secret image into n transparencies which are indistinguishable from a random noise making them not possible to be scanned.

The portability of the visual cryptography schemes was possible thanks to the definition of what we have called a visual key derivation function. This method is based on the use of a key derivation function (e.g PBKDF2) which allows to generate the n transparencies starting form some fixed parameter and a n master passwords, one for each participant. The capability to generate a transparency, starting from a master password, allows to store it inside a barcode. In this way, it is possible to encrypt the secret inside the barcode as one of the two transparencies needed to generate it. Using the two defined models it was possible to define two real use cases. The first one generates a scheme which allows to handle a role-based access control structure in which it is possible to extract different secrets starting from the same barcode. The second one was used in order to generate a scheme which allows to authenticate users just using a device and a barcode-reader software.

Future work should focus on the study of the use of these new cryptography techniques, not just to provide confidentiality but also to build a scheme which allows to obtain authenticity for barcodes when applied to closed environment. Furthermore, for what concern the theory behind the two models that we have implemented it should be interesting to search for a model that combines the scalability of the first and the usability of the second one.

References

[1] G. Ateniese, C. Blundo, A. D. Santis, and D. R. Stinson. Visual cryptography for general access structures. *Information and Computation*, 129(2):86 – 106, 1996.

[2] G. Ateniese, C. Blundo, A. D. Santis, and D. R. Stinson. Extended capabilities for visual cryptography. *Theoretical Computer Science*, 250(1):143 – 161, 2001.

[3] A. Dabrowski, K. Krombholz, J. Ullrich, and E. R. Weippl. QR inception: Barcode-in-barcode attacks. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones &#38; Mobile Devices*, SPSM '14, pages 3–10, New York, NY, USA, 2014. ACM.

[4] R. Focardi, F. L. Luccio, and H. A. M. Wahsheh. *Security Threats and Solutions for Two-Dimensional Barcodes: A Comparative Study*, pages 207–219. Springer International Publishing, Cham, 2018.

[5] R. Focardi, F. L. Luccio, and H. A. M. Wahsheh. Usable cryptographic QR codes. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pages 1664–1669, Feb 2018.

[6] X. Jin, X. Hu, K. Ying, W. Du, H. Yin, and G. N. Peri. Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 66–77, New York, NY, USA, 2014. ACM.

[7] D. W. C. Z. Kevin Peng, Harry Sanabria. Security overview of QR codes. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 2157–2164, New York, NY, USA, 2015. ACM.

[8] A. Kharraz, E. Kirda, W. Robertson, D. Balzarotti, and A. Francillon. Optical delusions: A study of malicious QR codes in the wild. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '14, pages 192–203, Washington, DC, USA, 2014. IEEE Computer Society.

[9] P. Kieseberg, S. Schrittwieser, M. Leithner, M. Mulazzani, E. Weippl, L. Munroe, and M. Sinha. Malicious pixels using QR codes as attack vector, 01 2012.

[10] K. Krombholz, P. Frühwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl. QR code security: A survey of attacks and challenges for usable security. In T. Tryfonas and I. Askoxylakis, editors, *Human Aspects of Information Security, Privacy, and Trust*, pages 79–90, Cham, 2014. Springer International Publishing.

[11] R. laboratories. *PKCS 5: Password-Based Cryptography Specification Version 2.1.* 2012.

[12] A. Mishra and M. Mathuria. A review on qr code. *International Journal of Computer Applications*, 164(9):17–19, Apr 2017.

[13] M. Naor and A. Shamir. Visual cryptography. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, pages 1–12, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[14] B. N. A. Sawsan K. Thamer. *A New Method for Ciphering a Message Using QR Code*, pages 19–24. Computer Science and Engineering, Vol. 6 No. 2, 2016.

[15] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. F. Cranor, and N. Christin. QRishing: The susceptibility of smartphone users to QR code phishing attacks. In A. A. Adams, M. Brenner, and M. Smith, editors, *Financial Cryptography and Data Security*, pages 52–69, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.