Bachelor's Thesis

Spring 2015

School of Health and Society
Department Design and Computer Science

Enhancing QR Code Security

Writer
Shuang Zheng
Linfan Zhang

Supervisor
Fredrik Jönsson

Examiner
Qinghua Wang

School of Health and Society
Department Design and Computer Science
Kristianstad University
SE-291 88 Kristianstad
Sweden

Author, Program and Year:
Shuang Zheng, Bachelor's Programme in Software Development and Engineering 2012
Linfan Zhang, Bachelor's Programme in Software Development and Engineering 2012

Instructor:
Fredrik Jönsson, Ph.D., Senior Lecturer, Kristianstad University (HKr)

Examination:
This graduation work on 15 higher education credits is a part of the requirements for a Degree of *Bachelor of Science (180 credits), Main field of study: Computer Science* (as specified in the English translation).

Title:
Enhancing QR Code Security

Abstract:
Quick Response code opens possibility to convey data in a unique way yet insufficient prevention and protection might lead into QR code being exploited on behalf of attackers. This thesis starts by presenting a general introduction of background and stating two problems regarding QR code security, which followed by a comprehensive research on both QR code itself and related issues. From the research a solution taking advantages of cloud and cryptography together with an implementation come after. We also give an objective evaluation on the outcome in comparison to existing QR products. They are based on the purpose of enhancing QR code security and aim to interpret how we have tackle the specified problems meanwhile to suggest possible further work for bringing security of QR code to a higher level.

Language:
English

Approved by:

_____

Ph.D. Qinghua Wang Date
Examiner

Table of Contents

List of Abbreviations

Admin	Administrator
DoS	Denial of Service
EC level	Error Correction level
GUID	Globally Unique IDentifier
HTTPS	Hypertext Transfer Protocol Secure
IIS	Internet Information Server
MAC	Message Authentication Code
MD5	Message Digest 5
MITM attack	Man-In-The-Middle attack
MMI code	Man-Machine-Interface code
PKEQR	Public Key Encrypted QR code
QR code	Quick Response code
RX digest	Received traffic digest
SEQR	Symmetric Encrypted QR code
SMS	Short Message Service
SQL	Structured Query Language
SPayD	Short Payment Descriptor
TX digest	Transmitted traffic digest
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
ZXing	Zebra Crossing

List of Figures and Tables

1 Introduction

1.1 Background

QR code, abbreviated from Quick Response code, is a two-dimensional barcode. A QR code is able to store and convey data including web link URLs (Uniform Resource Locators), plain text, email addresses, contact information and so on. It was initially designed for purpose of tracking vehicle parts during manufacture in industry procedure [1]. However later QR code aroused the public's attention and became personal- and advertising vector thanks to its versatility and ease of use. Any person is able to gain his/her own QR code through providing data which is going to be encoded into the code, either via some sorts of software or web sites. Once upon the finish of code generation, data stored in this code can be extracted via so-called decoder or scanner - an application or a device to be used for decoding the QR code and obtaining the stored data.

Figure 1. An example of QR code

As can be seen from the example above (Figure 1), one property of QR code is that the code is only machine-readable. As a consequence human beings are unable to directly get the real data in code until they scan and decode it with devices or applications. This opens possibilities of both positive- and negative usages. On one hand, need of specific scanning or decoding terminal conceals data instead of exposing everything in plaintext. While on the other hand, attackers aim to take the advantage of human-unreadability and thus manipulate the data behind at any point which can be exploited, for example, to encode malicious information into a code and users are tricked when they decode the code.

1.2 Research Questions

In order to focus in specific tasks and determine the purpose of research, two research questions are to be answered in this thesis.

Research questions 1: How can private data stored permanently in QR code be protected?

Once a QR code is generated, the data stored in it can be easily extracted via scanning software at any time and by anyone, as long as it is not seriously corrupted. Some QR codes storing personal information like name and ID number hence risk of being used for illegal purposes including forging bank card, fraud and blackmail.

Research question 2: How to make QR code more resistible against malware?

Users do not exactly know the content behind a QR code until they scan it then enter the site or view the data behind. It might force you to imperceptibly download malware or application containing virus. Besides, QR code itself is not well equipped with meticulous protection which means there exist vulnerabilities of being exploited for attackers' purposes. Similarly in QR code scan, software defence is far from satisfaction and sufficiency to effectively prevent incidents.

1.3 Motivation

The two aforementioned research questions can be regarded to have covered many scenarios in daily usage of QR codes.

For the first research question, there are plenty of web sites which offer service for people to generate their own codes or codes for certain purposes such as loyalty card in a club and name card in a company. By simply taking a website as an example, the site *qrcode-monkey* [2] provides visitors functionalities of

generating various types of QR codes including URL, text, email, phone, SMS (Short Message Service), vCard, meCard, location, social network links (Facebook, Twitter, YouTube in this case), as well as Wi-Fi (Wi-Fi Id and password). Because of ease-of-use on these web sites, many users tend to have their own QR code as a novel way of storing information. But when the QR codes with text, email, SMS, Wi-Fi credentials are created, such categories of offline data are permanently in the generated code. Though a simple scan/decoding replaces manual typing text or password, these pieces of data are not expected to be exposed to those who are with malicious intents whereas common QR code generation providers offer no means against that.

What is worse, the problem can also be caused in more serious cases. Speed et al. [3] argued that QR codes being adopted in some official uses is worrying, for instance, the ticket system for bullet trains in China where a QR code with passenger's sensitive data is placed onto the paper ticket. Frauds and impersonations have been detected due to the fact that passenger's name, identification etc. are stored in this ticket and attackers have successfully exploited vulnerabilities to play tricks.

For the second research question, although some related works have been conducted aiming to study the security of QR code, they are not sufficient and efficient to achieve the goal of relieving concerns over QR code security issues.

Kieseberg et al. [4] examined QR code and how they can be used to attack both human interaction and automated systems. This indicated and raises topics for further study in scope of QR code security. Dabrowski et al. [5] analysed a case named "Barcode-in-Barcode attack". Nevertheless in the article, they promoted neither security enhancements nor countermeasures in depth. In the article written by Kapsalis [6], the author discussed about security of QR code even put forward countermeasures. However the two countermeasures proposed were very indirect and implicit, which were URL inspection before redirecting to web sites and education on users' awareness to potential hazards.

There is a trend towards wide and multi-functional utilization of QR code in the future as such it makes us convinced that to investigate approaches to enhance QR code is imperative.

1.4 Structure of the Thesis

Followed by the introduction comes the next chapter presenting methodologies which have been applied. The method is mainly literature review, including research on modules of QR code, QR code encoding and decoding, attacking methods, levels of error correction and software. The research is presented after the method chapter.

Immediately after research is the result we finally came out with. The result is illustrated step by step and part by part, for which readers are able to clearly understand every developing process. Besides, critical parts are presented in details and interpreted, to explain the reasons of conducting certain specific tasks.

Furthermore discussion on the result is held in the following chapter. It contains evaluation of the result in relation to the goals. In the same chapter, an objective assessment is given, on what the pros and cons are with the result, as well as, how the result excels earlier existing work in comparisons.

Finally we draw the main conclusions that are reached from our research meanwhile provide some recommendation with respect to how the result can be applied and how it can be improved in further work.

2 Method

For the study in this thesis, the method applied was literature review. It mainly consisted of articles from digital libraries, book chapters and online resources. During the procedure, *ACM Digital Library* and *DiVA portal* were two dominating destinations which covered plenty of professional resources to aid the study on QR code security.

The scope of literature review ranged not only within acknowledgement of QR code but also related work regarding information security. This was because of that QR code can be affected when data of QR code is transmitted or exchanged, for example when a QR code is decoded by software or is redirecting users to a resource location. Study field about QR codes arouses public awareness recently within circa 5 years thus the literature references are quite new.

In order to answer the two raised research questions, we focused on following five parts of research in relation with QR code which come in the next chapter.

- Modules of QR code
- QR code encoding and decoding
- Attacking methods
- Levels of Error Correction
- Software

3. Research

3.1 Modules of QR Code

Module refers to the dot that makes up QR Code. Number of modules which is also called module configuration, varies from version to version and QR code has totally 40 versions defined by the inventor of QR code – Denso Wave [7]. Every version has specific QR code dimension and pattern. This means that any version of QR code has a particular number of modules as such has different data capacity limit according to the amount of data, character type and error correction level (i.e. EC level, error correction refers to code words for correcting wrong data, higher EC level has better tolerance to QR code corruption but requires more code words) [8] [9]. As the amount of data increases, more modules are required to comprise QR Code and it results in larger QR code symbols. For example, a QR code of version 2 has 25x25 modules, stores 272 data bits without version information. For a QR code whose version is higher than 6, say version 7, has 45x45 modules, stores 1248 data bits however you must include an 18-bit version information in order to avoid making detectors confused [10]. In a nutshell, the module configuration is fixed according to the version and the version determines other factors.

Moreover, by taking a deeper look into layout of QR code, modules in the data area are grouped into section of 8 modules and the data is read sequentially [11]. This is shown in Figure2, of which the grey areas stand for data bits storing data information to carry. In our research we did not bother the other parts. What is worth paying attention to is that: Since every 8 modules are grouped into a section and the data is read section by section, if any change or reordering of single module happens, the entire 8-bit block section is rendered or destroyed and causes the data to be unreadable by detectors. Although with the help of error correction in QR code, it is possible to remain reading the original data. Yet when the QR code is of low level of error correction and it happens that the QR code is seriously damaged, even alter of a single bit weighs.

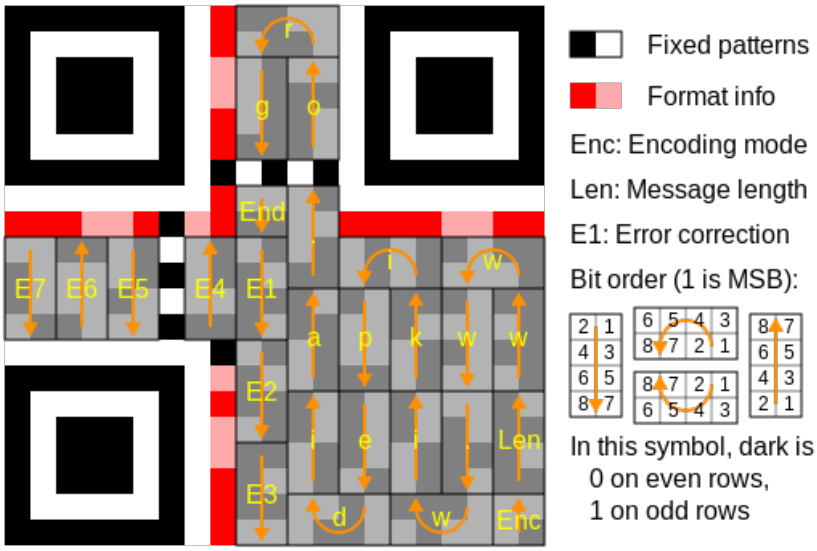

Figure 2. Layout of QR code [12]

To summarize the two aforementioned findings regarding modules of QR code, they are:

1) Once upon a QR code is generated, due to version and related factors, module configuration is fixed;

2) Change or reordering of bit(s) should be avoided, otherwise the expected data risks of becoming no longer detectable or readable. Even a single bit can have effect on other parts of information.

3.2 QR Code Encoding and Decoding

Latest technologies brings QR code encoding and decoding to a higher stage. The first thing comes into mind to make QR code safer might be encryption during encoding and decoding process of a QR code. In most cases, intents of using QR code is to gain maximum accessibility, such as guiding potential customers to advertising web sites or present information of products. Though it is not very common to have encrypted QR code, there can be a need to have such type of QR code meeting different security specifications.

Encryption is a process of encoding the original information in a way that only the expected authorized recipients can attain and read it. There are two popular approaches in encryption which are symmetric-key encryption and public-key encryption. The former applies the technique that both the sender and recipient share a secret key for security purposes. While the latter requires a so-called public key which is used for encrypting data to be transmitted, and on the recipient side a private key matching the public key is used to decrypt the data received. The two encryption methods are able to be employed into QR code.

In the article written by Kevin Peng and his colleagues [13], two QR code security standards using encryption were introduced - Symmetric Encrypted QR code (SEQR) and Public Key Encrypted QR code (PKEQR). The SEQR made use of symmetric-key encryption and the PKEQR adopted the public-key encryption as described above respectively.

Although it is feasible to apply the two encrypting methods into QR code, there are certain drawbacks which should not be ignored. Firstly, since keys might be included into the QR code during encoding and decoding steps, the accuracy of key data must be guaranteed to a greater extend. This indicates a demand on higher error correction level to ensure that the keys on both of sender and receiver sides are exactly identical or valid. Secondly, due to the existence of keys, part of data space is obviously occupied for storing such extra information. Thirdly, which may not play so important role as the formers, is that the computation time to gain and read a QR code now becomes longer. It is not so essential for that most of users do not have a strict requirement on encoding- or decoding time of QR code and the increase of computation time in these cases is not significant.

Besides encryption technologies, two of literature reviews provided us a new vision. The first one discussed about QR code and watermarking [14] which was initially used for copyright. However if judging from another aspect, protection of copyright can be utilized for protecting the integrity of data information, that is, to assure the message you get is the same as the one sent. As stated in the article, watermarking was done through encoding and decoding extra data bits into a QR code. The other literature presented a general idea of a technique for data hiding

to keep the hidden data secretive [15]. The solution given was to make use of both steganography and cryptography. In details, they put the image with hidden data into a QR code and used a powerful encryption algorithm.

These articles discussing about QR code encoding and decoding brought forward sound grounds for further improvement of QR code security however considering the complexity of some of the algorithms the authors presented and time limit for us, they are to be suggested in recommended further work. In our work, we would adopt MD5 (Message Digest algorithm 5, a widely used cryptographic hash function producing a 128-bit hash value to verify data integrity [16]) which would be described in chapter 4.

3.3 Attacking Methods

Diverse attacking methods are available and have been detected in usage of QR code. Hereby two categories of attacking methods are analyzed:

- QR code as attack vector

A QR code acting as attack vector does harm indirectly, which typically refers to that the QR code conveys data which would lead users who decode the code into traps on behalf of attacker, for instance leak of information. In such case a QR code can carry malicious/phishing URL and redirect users to a site which is well-conceived by the attacker and pretends to be authorized or benign ones.

- QR code itself

At the same time, there are risks that a QR code itself is an attack without being a vector to other external location. By this some commands calling decoding terminal to do certain works are encoded into the QR code. Once the code is decoded the commands are triggered and executed, causing damage to information, operating system and other possible sequence the attacker expects to gain.

3.3.1 QR code as attack vector

The security issues usually arise together with the broader and more frequent use of technology, so does the QR code. Since QR code has the capability of carrying URL web links and other type of data, potential attacks emerge by taking the advantage of this QR-code feature. It is not hard to imagine that once a user scans or decodes a QR code, he/she may be redirected to a website which is already manipulated by a person with malicious purposes. In such case the QR code is used as attack vector, trying to fetch private information and do the works fitting attacker's wishes. Introduced by Kapsalis [6], a fairly easy way to achieve this was a replacement sticker. Simply covering a fake QR code containing malicious data over the genuine harmless QR code did not require much effort but indeed succeeds. A significant ground was the users' inadequate awareness to distinguish legitimate web site address and those which were not.

MITM, abbreviated from Man-In-The-Middle is an attack method where an attacker secretly relays or changes communication between two parties who believe they are directly communicating with each other [17]. To be more professional, instead of replacing original QR code with sticker, hackers who play tricks like Man-In-The-Middle attack, is still able to bring victims to their intended web sites even though the victims have scanned/decoded the real and original QR code [18].

According to the facts described above, some countermeasures have been announced. Narayanan [19] suggested that printing the URL besides the QR code, making sure you had good control of your QR code in public area and using a third party to do certain credential works. In addition, from the article "Security of QR code"[6], the author promoted two measures which were education on users to equip them with better security conscious and manual checking of URLs before visiting the links. However from our opinion, they tend to be indirect and implicit. As a consequence, we have found a strategy to enhance security of QR code, taking aforementioned problems into consideration. Further details are in the incoming chapter of Result.

3.3.2 QR code itself

Another issue related to QR code is the hazard which may be caused by QR code itself. Typically they are SQL injection and Command injection. Kieseberg et al. [4] argued that appending a semicolon followed by a SQL query to the encoded data could make manipulation to backend dataset possible. Additionally, using the encoded information as a command line parameter without sanitization could open possibility of running arbitrary commands on the behalf of attackers.

Despite that modern systems are armed with security configurations to escape being failed by SQL injections and Command injections, examples of fails in the real world still appear. In 2012, it was confirmed that Man-Machine-Interface codes (MMI code, a kind of software/hardware interface code) could be used to operate different attacks against Samsung devices [20]. A QR code with encoded MMI codes *2767*3855# would be triggered and run execution which erased all data stored in the mobile device. In the same article, the writers gave brief introduction to a list of instances, showing that there did exist vulnerabilities to be exploited by use of QR code to inject SQL queries or commands.

Similarly in Peng et al.'s article, they attempted to find out QR code reader bugs including popular scanner applications SPayD (Short Payment Descriptor) and ZXing (Zebra Crossing) [13] [21] [22]. They discovered that both SPayD and ZXing lacked ability to prevent code injections and unauthorized actions, meaning that system could be vulnerable to manipulation of internal system data. While for ZXing, information leaking could also happen for the reason that it used implicit intents which means any application had right to acquire data that ZXing broadcasted on the Internet.

Security specialists have been committed to seeking preferable actions against SQL- and Command injections yet no one can promise an absolute safe scheme. The anti-injection combat is regarded as long-term due to the ever-changing technologies make everything possible. What is more complicated is that, the injections vary by programming languages and platforms which have close coupling to backend datasets or databases.

To be concentrated in a focused field and get rid of too much worries over QR code injection problems, we selected a specific programming language and adopted techniques related to it for better outcomes. The chosen programming language was C# on .NET [23], together with the help of Windows Azure, a cloud platform [24]. This will be illustrated in the Result chapter and motivated in the Discussion chapter.

3.4 Levels of Error Correction

Every QR code has error-correction capability to allow a QR code to be read even it is dirty or damaged. Also this is very powerful when the QR code is read at a high speed but keeps reliable [25]. There are four levels of Error correction which are L, M, Q and H bearing tolerance to corruption up to 7%, 15%, 25% and 30% respectively [4][6][26]. In many cases, the choice of taking the error correction level L is efficient enough for common uses in daily life. The higher error correction level is, the more code words would be used as such decreases the amount of data which can be stored inside the code.

In an attacking scenario in Kapsalis' research [6], attempts to attack the error correction part using brute-forcing method did not lead to any expected results. Instead, the produced QR code was unreadable from the reading devices and it could be decoded as junk data. This proved that the error correction was stable and not a target attackers would like to capture at present.

The primary aim of analyzing among different levels of error correction was to pick a preferable model for development of implementation meanwhile to fix in a single correction level. Initially it was expected to carry as much data as it can in one QR code. However later we decided to choose the H level of error correction, having approximately 30% amount of data correction. The reasons are follows: Above all, for the sake of integrity and confidentiality of information, certain data of potential techniques have to be encoded into the QR code, e.g. unique identifier to locate where the real information is. If the identifier tends to contain certain

error due to damage and is not eventually corrected, real data behind shall not be retrieved. It can also be public key or digital signature in the scope of cryptography to be encoded in to a QR code. Besides, possible further works to enhance QR code security including watermarking and steganography which were described in section 3.2 have the demand of precision, meaning that the security information to be extracted at the other side should be original otherwise unexpected output can be attained. Finally, it is worth mentioning that there are means to improve data capacity rather than restraining the error correction part. Examples are Color Barcodes and Multiplexing [26].

Therefore, we are to take level H (30% tolerance of damage) as error correction level in the implementation of result.

3.5 Software

The term software under this picture mainly refers to the scanner application or decoding software. During the earlier period of QR code usage, scanner applications played more important roles and people got used to retrieving information through scans on devices, basically the smart phones. Nonetheless after the QR code became known by a growing number of public, single way to extract data from a QR code seemed to be awkward. How could you get information in a QR code if this code was shown on your mobile phone? Certainly you could send it to other devices first and then scanned it but this was not so convenient and went against the aim of “Quick Response”. As a consequence, nowadays common holders of QR code have option to extract data behind a QR code directly by choosing the code symbol image and decoding it on devices like laptops or portable devices, without the must to decode by camera or scanning intermedia.

To offer user decoding an image rather than scanning via camera is our intention of implementation, not only because of user experience and mistakes caused during scanning does not influence the research, but also for more

technical concerns: The article “QR Inception: Barcode-in-Barcode Attacks” [5] made a detailed evaluation among a variety of barcode scanners in the case of “barcode-in-barcode”. Barcode-in-barcode in the sense of QR code is referred to a secondary QR code being embedded into a larger one. The authors demonstrated that it was possible for the same QR code to return different data when the QR code was scanned by different scanning applications. The causes were mainly multiple standard ambiguity and various detecting ability of cameras. Many scanners implement more than one symbology. Due to this fact, when a device is confronted with multiple QR codes in a single picture, it may fall into ambiguous situation. At the same time, it oftentimes happens for users to try more than once to finally aim exactly at the QR code. This makes it highly probable in the scenario of barcode-in-barcode that an inner QR code is read first and the scan of outer one is skipped, leading to distinct data.

Another issue regarding software in the field of QR code is user-side efficiency of prompts and warnings before a QR code is going to guide users to malware-, phishing URLs or data source. One of the reviewed article promoted a potential solution named “SafeQR”, where security warning were to be given to users in case a malicious website is to be redirected [27]. The solution was built on a specific list of detected malicious- and phishing URLs and a support library which offered better inspecting efficiency. The outcome of user study conducted by them showed that “SafeQR” was superior to other existing scanning software. One of the factors to success was the careful design of security warning since users’ decisions to obey or ignore warnings had strong relationship with the user interface design.

Based on idea of solution “SafeQR”, we have done further research on effectiveness on how people react to certain warnings on browsers for that many problems are derived from URLs conveyed by QR codes. Two articles [28] [29] revealed that, as long as people are not totally interrupted by certain warnings, for example, a full-window prompt pops up and stops further visiting of website, they tend not to be alerted and perform to continue browsing even the site is warned as phishing- or malicious ones. Such findings together with “SafeQR” have

enlightened us to emphasize more on how user-side can help to prevent people from being likely directed to phishing or malicious web sites.

Fruitful outcomes are acquired through the researches. To act as a more coherent transition from research to result, hereby is a summary:

The research on modules of QR code has showed that, after a QR code is generated it is not feasible to alter or reordering data bit of modules since the module configuration is fixed. The prerequisite of keeping count of times updated at every scan thus cannot be achieved since QR code is static as just explained. There is a need to make QR code more dynamic and our result which is to be illustrated in the incoming part hits the goal.

For QR code encoding and decoding, it opens possibilities of enhancing QR code with ever-changing technologies including cryptography, watermark and steganography. However they have certain drawbacks and should be thought twice before taking actions. Taking time limitation and complexity of algorithms into consideration, in our result we are to take the advantage of an efficient cryptograph technique MD5. Though others like keyed methods, water marking and steganography are recommended in possible further works.

Through study of the two sorts of attacking methods in QR code world, it has aroused attention to prevention on both vector-based and injection-based attacks. Actually the success of deceive depends much on how “professional” the fake websites for that QR code is more a courier of the data than the attacking tool. Still it is a special issue linked with QR code and for this reason we would like to introduce a scheme containing an entire flow which holds back malicious use of QR code from the generating stage. So far, the most efficient measure against phishing- and malware- attacks are blacklisting and filtering. In our result, they are still applied however from a different aspect. Additionally, to ease concerns about other potential and possible attacks, we make the best use of existing platform provider which is

Windows Azure cloud platform. The solution is built in support of Windows Azure as well as utilizes many features offered by the cloud.

When it comes to the error correction level of QR code, we have eventually taken the option to use the H level with maximum 30% tolerance to damage of QR code. We have argued that, if the key goals in information security would like to be ensured, it is necessary to encode special data into a QR code, for instance a unique identifier to locate the real data behind, which requires precision and integrity on both the sender and receiver sides. It can also be extra data bits such as key pair in the scope of cryptography mentioned in the section of QR code encoding and decoding as recommended improvements. But the extra security-like data definitely affects the space of storage for the data a QR code can carry with and makes the QR code symbol larger. Even so, as the topic of this thesis is enhancement of QR code security, we prioritize the guarantee of security so that the decision to pick H level has been made to provide highest error correction for avoiding error of QR code data during transmission or under damages.

The last approach was regarding QR code software. We have found the capability of preciously detecting expected QR code is not well equipped in many camera and scanners. And direct-decoding software tends to be more secure in attacking scenarios like “barcode-in-barcode”. In addition, user-side prompt and warnings play critical parts in the case the QR code is used on the behalf of attackers. As a consequence, we are not to operate the decoding process via cameras or scanners. Instead, we have had the user-side software written in C# as a decoding application, implying that users are able to choose image and directly decode the QR code to retrieve the data they want meanwhile in the application the prompt stops user from further visiting when the request is invalid due to inadequate access right or malicious changes has taken place.

4 Result

Bear in mind of researches and outcomes carried out in the previous chapter, this chapter presents our result of work. At the same time, crucial parts of solution are illustrated in details, for the sake of a more comprehensive understanding both macroscopically and microscopically

4.1 General Idea

The result of work to be proposed is a scheme covering the generation of QR code, encoding and decoding of QR code, storage of data that is behind a QR code, as well as the user-side control. There are three illustrative roles which are the Requestor who requests for creation of a QR code, the User who scans/decodes the created QR code, and, the Admin who has the right to manage and edit QR codes on the cloud. Following is the schematic diagram of our solution. (Figure 3)

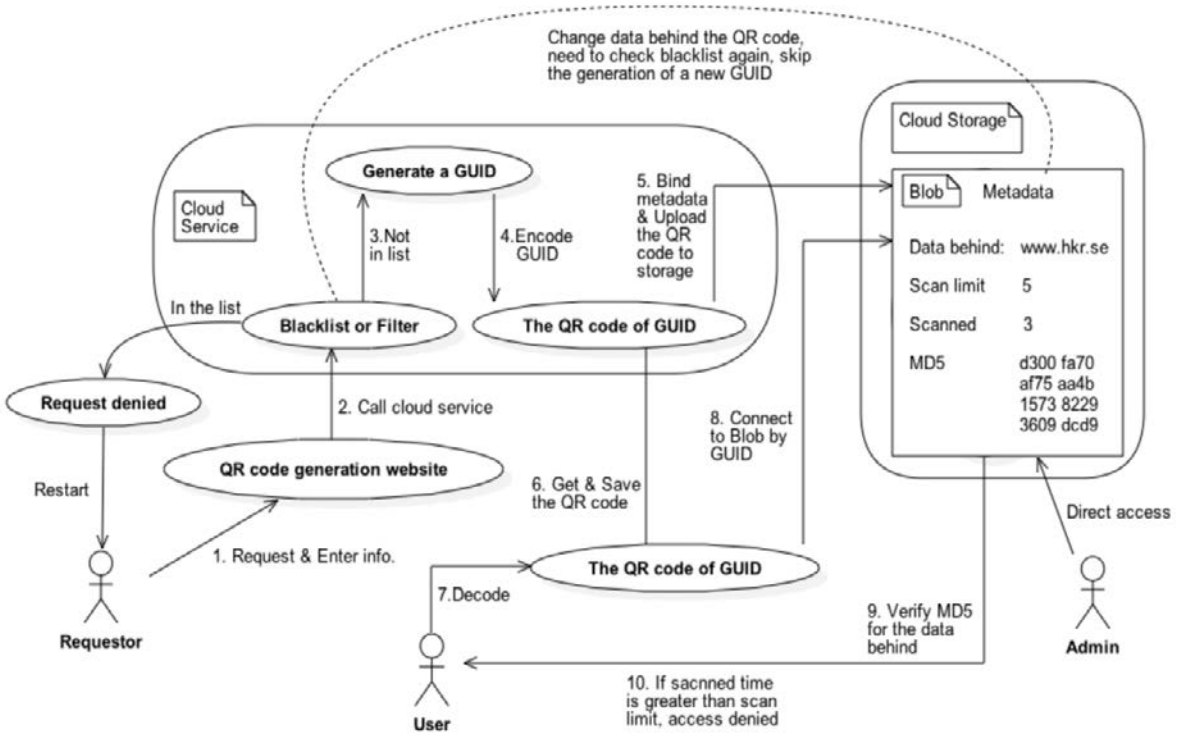

Figure 3. Schematic diagram of solution

The start points can be the Requestor requesting for a generation of QR code. The Requestor would like to make his/her own site more popular by spreading via QR code. He/She visits our web site where QR code generation service is provided. After inputting essential information including the real data (in this case, his/her website URL), scanning time limit and optionally a logo to be embedded to attract the public's attention, the website transfers those inputs given by the Requestor to the cloud side and call the cloud service which operates the internal generating works.

In the first place, the URL/data is checked by going through blacklist and/or filter. If the URL/data is already reported to be malware-, phishing websites or is detected as code injection, request is denied. Otherwise the requested data information is passed onto the cloud.

On the cloud, there are two important terminologies. The first one is GUID, abbreviated from Globally Unique IDentifier, a unique reference number used as an identifier of a unique data source in computer software [30]. The second one is blob, which is a file of any type and size which is stored on the cloud storage [31]. A blob can be bound to a set of metadata.

In our scheme, when the requested data information is not in the blacklist or being filtered as code injection, it continues to generate a GUID. Next, the GUID, instead of the real data, is encoded in the QR code and the generation of QR code is completed. Afterwards a blob is created, having the reference to be the GUID. The QR code is now stored as a blob on cloud storage and can be found by the GUID given to it. At the moment just before the QR code blob is to be deposited into the destination, metadata is bound to this blob. The metadata is a set of information which would plays essential role when the QR code is in user's hand. We hereby store the data behind, which is the original URL/data inputted by the Requestor. In addition, maximum scan times and the count of times of scans already took place are together with the real URL/data as metadata. The process above is shown in Figure 4.

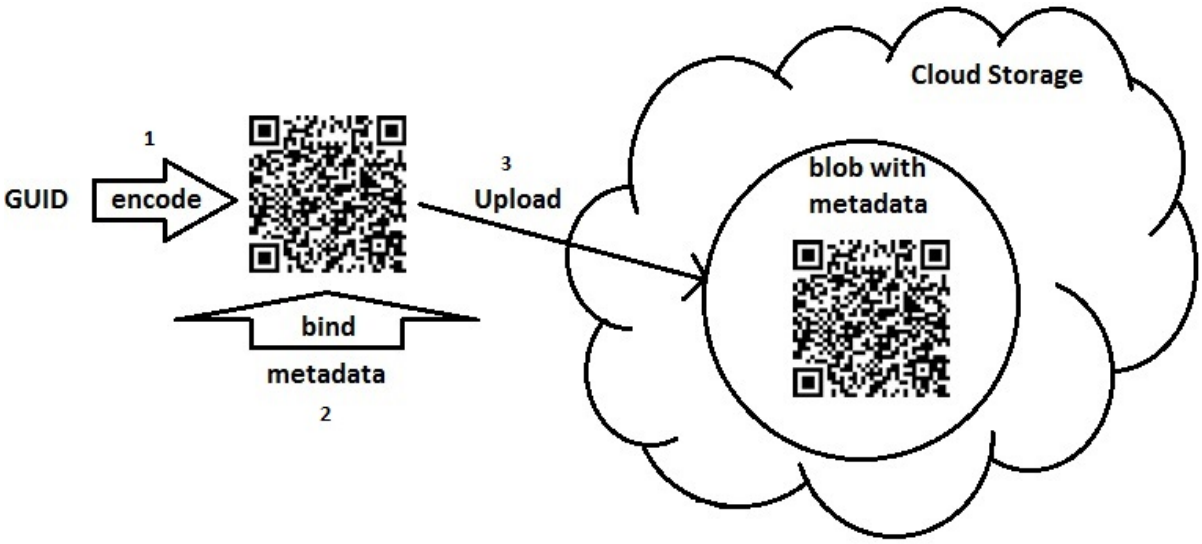

Figure 4. Encoding GUID, Binding metadata and Uploading to storage

One last thing to mention in the blob metadata is the MD5, which is attached for the sake of integrity of information. Before introduction to usage of MD5 in our solution, a brief description of MD5 is given below with the help of Figure 5:

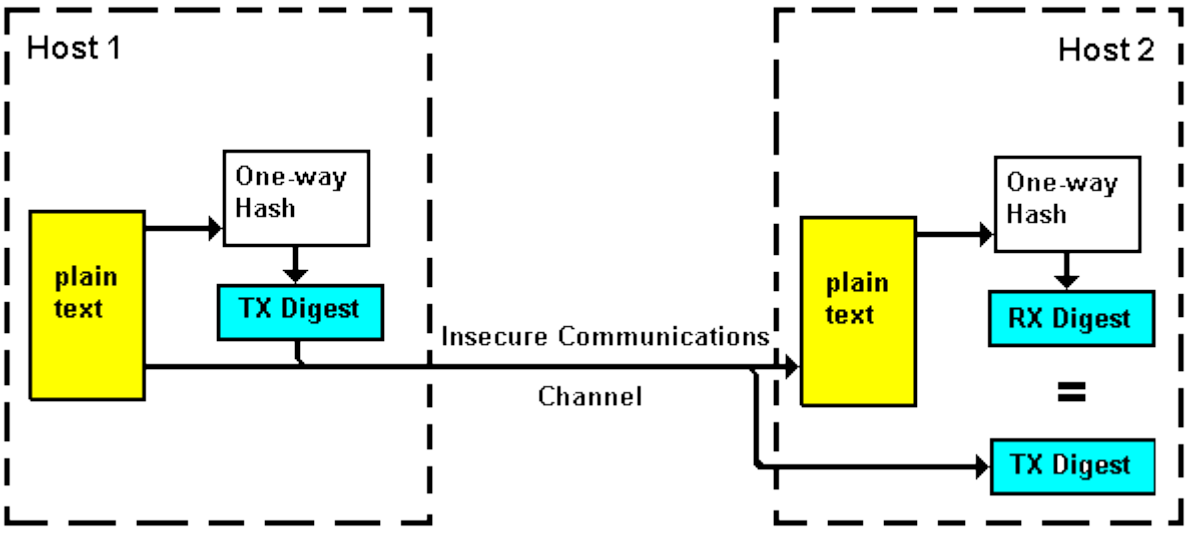

Figure 5. Message Digest algorithm 5 working principle [32]

In Figure 5, the plaintext is the message the sender initially would like to send to the receiver. The one-way hash is a cryptography-based hash function, applying some mathematical techniques to gain the goal easy-to-compute on every input but hard-to-invert given the image of a random input. [33]. The TX Digest and

RX Digest stand for Transmitted- and Received traffic respectively and in our case they refer to the hash value calculated from the plaintexts on both sending and receiving sides.

Now that host 1 (the sender) would like to transmit message to host 2 (the receiver), the plaintext from host 1 is hashed and this produces a hash value TX Digest. The TX Digest is transmitted together with plaintext via probably insecure communication channel such as a public network. Once upon a message plus message digest are arrived to the host 2, the received plaintext is hashed to obtain a RX Digest. Since the algorithm of one-way hash function will provide sole output value if the input is identical, the TX digest which was calculated before landing to host 2's hand is now compared with the lately produced RX digest. If the TX digest has no difference from the RX digest, it indicates that the plaintext from the sender side is exact the same as the plaintext got by the receiver side.

In our solution, this MD5 is not calculated from the GUID and it is not meaningful enough to ensure the GUID to be genuine. Instead, it is computed by hashing the real URL/data behind the QR code of GUID. Based on this, once the real data behind is altered by some means, the MD5 can notify user when the QR code of GUID is decoded and would like to produce URL/data which differs from the origin.

Got a general background of the generation, the storage of QR code and how MD5 works in the solution, now it is the turn on user side to explain the rest part. We assume that there is at least one User who has seen and fetched the QR code generated by the Requestor above. The User then saves the QR code as an image and applies our decoding software to decode the QR code he/she has just got. Note that, since the only data this QR code carries is the GUID, the User has no idea about what it is. However the decoding software is connected to the cloud storage and tries to acquire the blob which has this decoded GUID. If the expected blob is found, then the metadata of this blob is read. For security purpose introduced above, the first thing the decoder does it to verify the MD5, making sure that no modification has happened during the transmission of data traffic. When the verification is successful, the decoder goes on to collect metadata of

maximum scan limit and the count of how many times this QR code has been decoded or scanned. Only when the scanned time does not exceed the number of scan limitation, the real data behind the QR code is attained which hereby would direct the User to the website whose owner is the Requestor.

The metadata and corresponding purposes are listed in the following Table1.

Metadata	**Purpose**
Data behind	The real data behind the QR code of GUID. The type of the "Data behind" can be text, URL, email address and similar types of binary data.
Scan limit	The maximum scan time limit of a QR code. The scanned time should be less than this scan limit otherwise access is denied.
Scanned	The count of times that this QR code has been successfully scanned/decoded.
MD5	Message Digest for ensuring that the "Data behind" not to be tampered during the transmission.

Table 1. The metadata and purposes

Furthermore, beside the Requestor and the User, the third role is the Administrator (Admin) who has the privileges to access the cloud storage directly by logging onto the cloud. The Admin thus is able to view those generated QR code together with the metadata attached, at the same time to manage them by certain purposes like blocking a newly detected malicious website or injection code. The metadata, especially the scan limitation is thus handled by the Admin including changing the value or resetting it to zero.

For a relatively more secure operation, once a request to change the data behind is sent to the Admin, the user's data should not be simply changed as wishes. What it is ought to be done is that, before any modification, pass through the blacklist and the filter again to ensure that the upcoming URL/data is benign. Since the data behind to be changed is for the same QR code, no generation of a new GUID is carried out so that this kind of QR code becomes more dynamic and reusable, compared with traditional ones which have only one-time storing of static information and you have to totally generate a new QR code when there is a need to modify the information.

4.2 Partial Details

The solution is written on .NET platform in programming language C#, with the support of Windows Azure Cloud. In this section the entire scheme is divided into three parts and described in details.

4.2.1 The cloud service

The blacklist and filter, which is firstly gone through when the request arrives onto the cloud, is imperfect in our implementation. This is due to that professional blacklists containing universal URLs reported as malware-, phishing- and well performed code-injection filters are charged by providers like Norton [34] and Kaspersky Lab [35]. As such we implement a simple version of blacklist/filter to

simulate the functionality, that is, create a string list and add certain elements into it. When the request hits the item enumerated in the list the request is denied. For sure when the solution comes into practical usage, professional blacklisting service and code injection filters are demanding for the security level. Fortunately, nowadays programming languages have their own countermeasures to mitigate certain kind of code injection such as parameterized statement, escaping, pattern checking and so on [36] [37]. This is also one of the reasons we chose C# as the implementation language since it is modern and armed with certain protection against code injections. The blacklist and filter are the initial checkpoint of the incoming data to be stored behind the GUID of a QR code. Thanks to this, a good control of QR code abuse for malicious purposes is also accomplished.

Another crucial part is the GUID and utilization of it. GUID is randomized enough to make it negligible to generate the same number twice and it is applied in Windows Azure cloud storage, as in our solution. The idea of encoding the GUID into a QR code rather than encoding the real data attached to this GUID is for the following grounds: For that the traditional QR code is “static”, meaning that the data in a QR code cannot be changed once it is generated. With GUID, we do not directly point the QR code to the real data behind which the Requestor would like to share with others. Instead, a GUID acting as a reference makes it possible to legitimately change the data bound to this GUID as well as the QR code. By doing in this way, the QR code is safer. Imagine that even a QR code with a benign web site address is captured by criminals and the criminals would like to alter the information into malicious one. He will fail to do this because, what he gets is barely a GUID pointing to attached data, however if he would like to attack or by some means tamper the data, he have to first contact the Admin and go through the blacklist and filter. Another scenario can be that the attacker would like to generate a new QR code with similar GUID to trick users. This is neither feasible since the User uses our decoding software and only those GUIDs available on our cloud storage are valid. Again as mentioned above, theoretically it is of negligible possibility to have two identical GUID being generated. Therefore any attempt to create a new GUID pointing to the attacker’s own storage would not work and resulting in no-blob-found error. Still, the prerequisite

is that both the Requestor of a QR code and the User who decodes the QR code have indeed only use our service and software. This is regarded that we provide a trusted brand and in order to benefit from this scheme system, the Requestor and User should not have used other products including the service and the decoding software in this case. Screen capture of the blob storage is illustrated in Figure 6, as can been seen, only the GUID is visible as a reference to locate the blob.

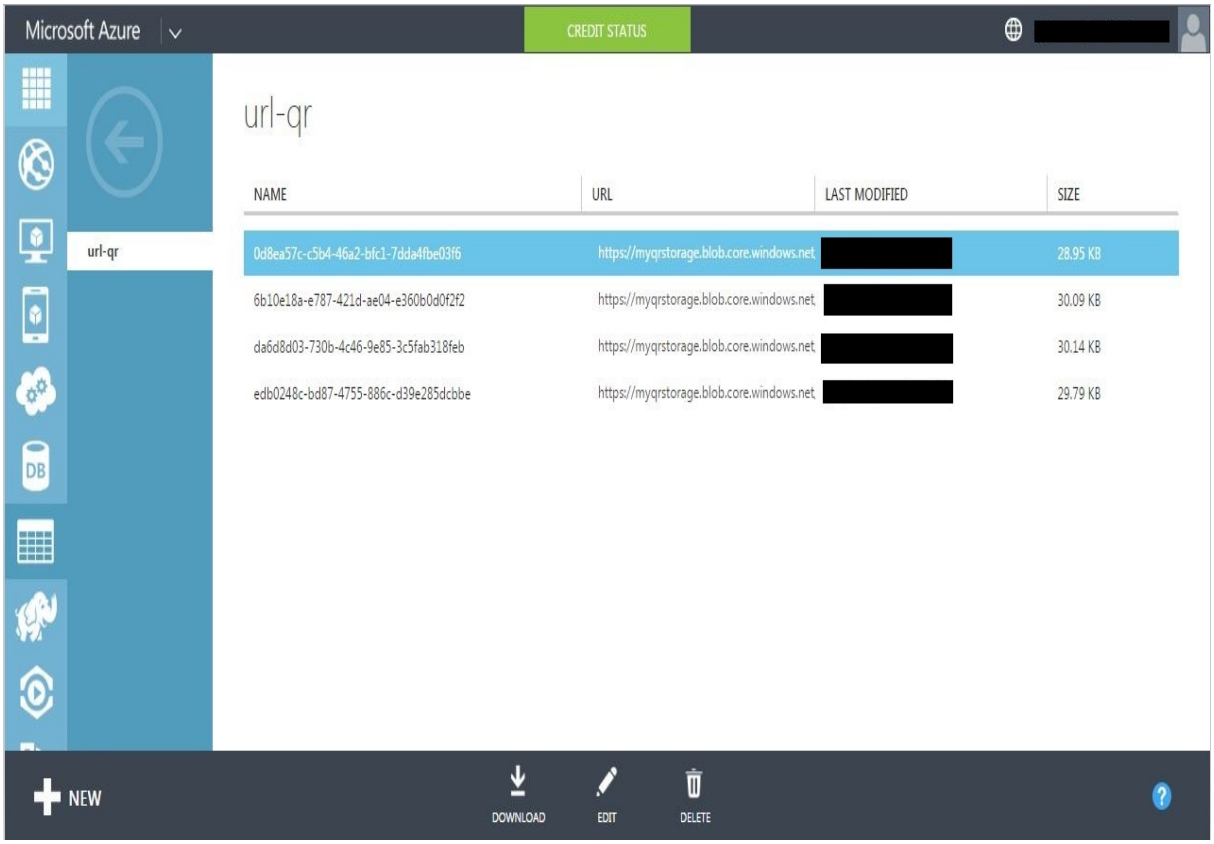

Figure 6. Cloud blob storage on Windows Azure

4.2.2 The cloud storage

The cloud storage stores blobs and every blob can have properties and metadata attached, as shown Figure 7 beneath. Properties are produced by default and we should pay some attention to the name, the absolute URI (Uniform Resource Identifier, plays similar role like URL [38]) and the content MD5.

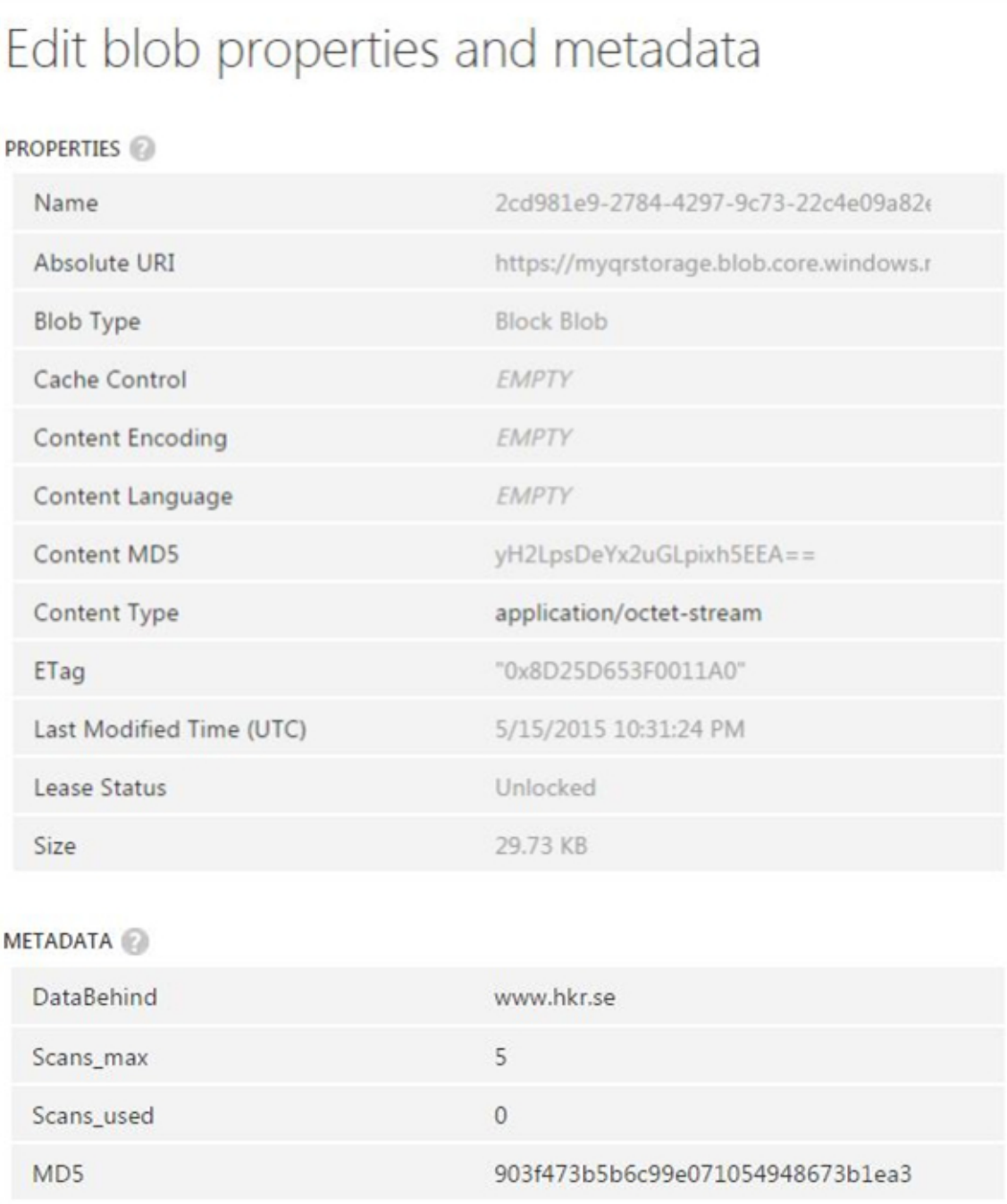

Figure 7. Blob properties and metadata

The name is the GUID as a reference. We store and retrieve the blob by this ID. The absolute URI is a link address to the blob storage so one may argue that what it would happen if the URI is exposed to others. Since that the blob container where the blobs are stocked has been made private, no much concern is necessarily to be aroused. Only those who have succeeded in logging as the administrator are able to visit the blob through this URI. The last term to be

emphasized is the content MD5. This content MD5 is calculated from the binary data stored into this blob, which in this case, is the QR code of GUID symbol. In our solution, what we would like to protect and ensure is the integrity of the real data behind this QR code, i.e. the metadata “Data Behind”. We therefore generate another metadata called MD5 which is computed from the “Data Behind” to ensure that the decoded QR code of GUID will definitely return the real data the original Requestor has inputted. To be more motivated, we again explain why it is more meaningful to verify the integrity of “Data Behind” but not the GUID: As long as the Requestor and User have applied our system, only valid GUID will point to the blob and fetch the data behind. Two possible types of attacking can happen here are replacement of the origin by a fake one and tampering to the data behind. In the former situation, it leads to no-blob-found error since our software will never find any blob linked with fabricated GUID. And in the latter scenario, MD5 prohibits the crime because of verification will fail.

We have two remaining metadata to introduce here, which are the “Scans_max” and the “Scans_used”. They are straight-forward. Once a valid QR code of GUID is successfully decoded, it queries the blob to return metadata. For the security purpose in some official usages, for instance, the train tickets storing personal information, it is usually expected that the QR code on the ticket to be disposable. Our solution resolves this problem with the support of these two pieces of metadata. When the valid blob is connected and the MD5 verification has passed, the “Scans_used” and “Scans_max” are checked. Only under the circumstance that the count of used scan is less than the maximum scan, access is finally granted. Once such steps are finished with no exception, the value of used scan times is incremented by one, indicating that one successful view of the metadata has processed.

Another point in the cloud storage is the direct access belonging to the Admin. This is designed for easier management of the QR codes and data behind, consisting of modification of metadata, viewing and deleting the blobs etc. Thanks to one of the facts in our system, that is, a QR code can be reused to bind different data behind with one GUID, Requestors can hence choose to do so. This

should be done via contacting the Administrator. The Administrator then confirms the request through blacklist or filter. Of such a process the security level is raised. This is more like a centralized control which can cause triviality but we believe it does the work before any further improvements are found.

4.2.3. The decoding software

In order to benefit from our solution system, the User is expected to apply our decoding software rather than those which have been published online.

The decoding software mainly does three tasks: decodes the QR code, verifies validity and returns data. The decoder decodes the QR code of GUID and thus builds connection to the cloud and the appropriate blob. This is done by confirming the credentials. For security reasons, the storage credentials including storage account name and storage account key, which together are used to validate access right of connection, are not directly saved in the software and not able to be viewed on the web sites. This is due to that by chance the software or website are cracked by certain means, the attacker is still unable to pretend to be the true software or website as we provided. The protection is done by storing the credentials into the web configuration file, a configuration file which is not visible when deployed onto the server or cloud. Then in the software we call configuration manager to implicitly get the credentials for further connections via HTTPS (Hypertext Transfer Protocol Secure, a secure communication protocol [39]). The code can look like the following.

In the web configuration file:

```
<appSettings>
  <add key="accountName" value="name" />
  <add key="accountKey" value="wSxtLPwUL6aFwCpjHY3AElHsfFfVi==" />
</appSettings>
```

And in the decoder application:

```
string accountName = WebConfigurationManager.AppSettings.Get("name");

string accountKey = WebConfigurationManager.AppSettings.Get("accountKey");

StorageCredentials creds = new StorageCredentials(accountName, accountKey);

CloudStorageAccount account = new CloudStorageAccount(creds, useHttps:
true);
```

By default, the directory browsing which lists documents of a deployment on server is disabled in IIS (Internet Information Server, the web server of Microsoft [40]). So the users can neither see the directory of deployed documents nor view the contents including the web configuration file.

In short, the decoder decodes the QR code, connects to cloud and fetches data, verifies the MD5, determines if the QR code is still available by checking the count of maximum- and used scans, as well as, returns expected result if everything is of no problems.

4.3 Implementation

As our implementation aims to illustrate main features of the solution and simulate the outcomes, we did not put emphasis on beautification of the interface. We applied cloud storage but did not deploy it to be cloud service due to costs on cloud. However it makes no difference for that such tests produce very close outcomes to the practical usage except the location of the service hereby is the localhost.

Both of the generator and decoder/scanner are web applications (which can be published as websites), written in programming language C#, on Visual Studio 2012 which is of the .NET platform.

In the generator website (application), a Requestor enters input data which can be normal text data or URL. The Requestor has the choice to embed a logo onto the QR code to be generated. Moreover, he/she enters the maximum scan times (decoded times) in order to have a limited number of times that the QR code is expected to be successfully decoded. For example, if he/she would like to make the QR code disposable then he/she enters 1. After clicking on the "Generate QR Code" button, the process begins and if everything is fine a short line of message pops up indicating where the generated QR code of GUID is saved and what GUID it has. See Figure 8 for how the website looks like.

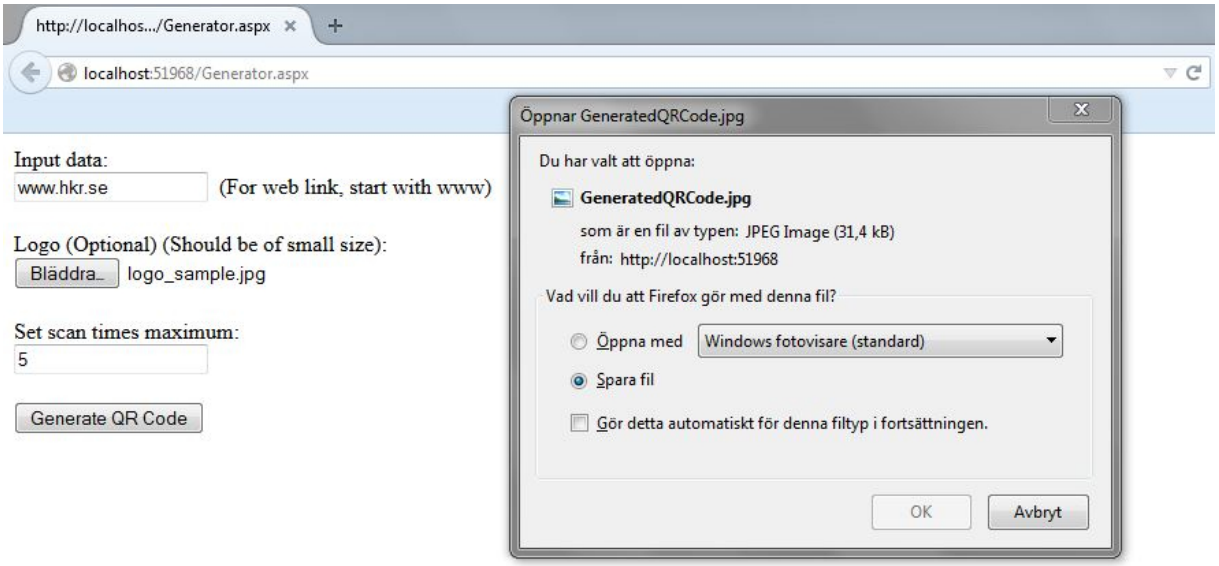

Figure 8. The generator web application/website

The decoder application (website) is shown in Figure 9 below. The User decodes a QR code of GUID by choosing the image which is on his/her computer or device and clicks on the "Decode QR Code" button. Since that two things need to be confirmed as mentioned, which are the MD5 and count of scan times. Error messages can emerge if the two items do not pass the validation. In Figure 10 the User has made choice to decode a QR code which the time limits have reached and in Figure 11 the data behind to be fetched tends to have been altered so that the MD5s are not identical.

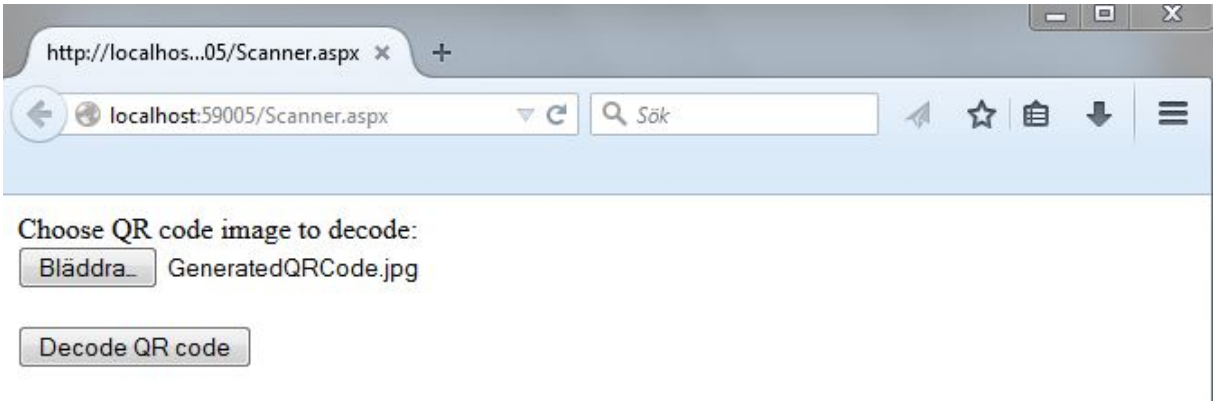

Figure 9. The default decoder application/website

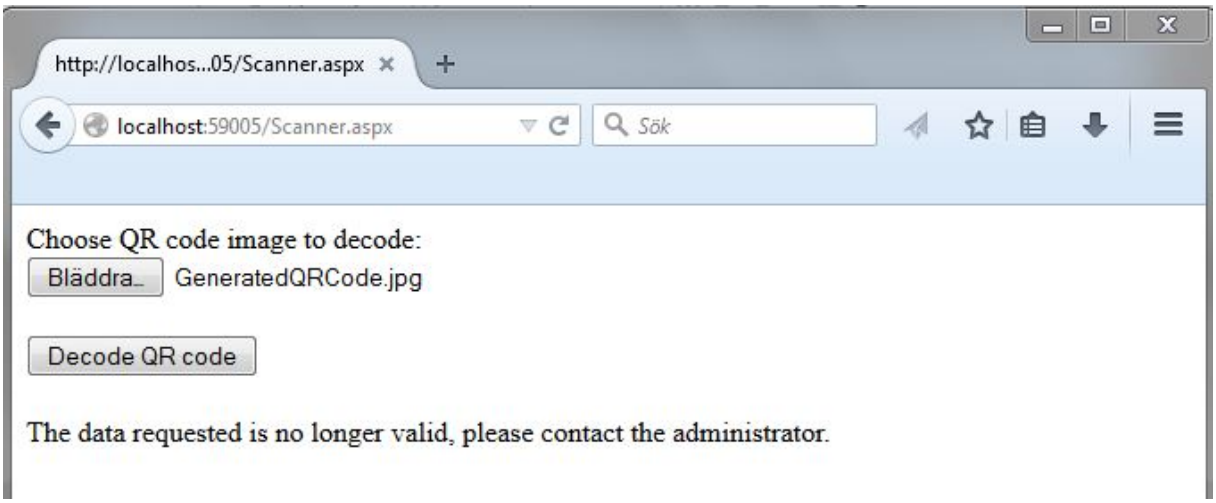

Figure 10. The decoder application/website with message of invalidity

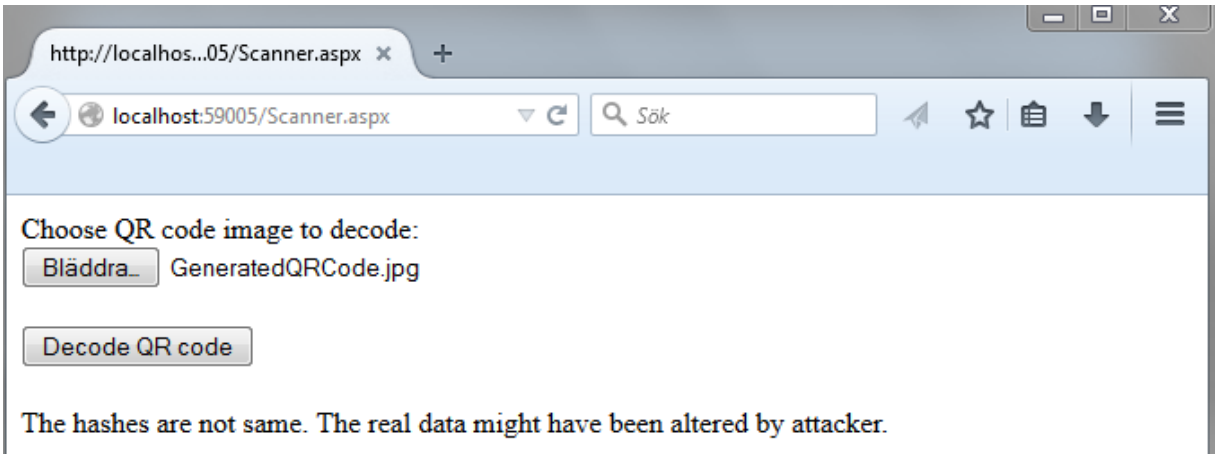

Figure 11. The decoder application/website with message of warning

In the final scenario where the MD5 verification fails, we simulated it by changing the data behind on cloud. This is not the real case to happen in daily usage since once the attacker has gained the access to the Windows Azure administrative page, meaning to be able to manage all the blobs, this attacker can thus change everything but not simply the data behind. What we meant “simulate” was that we tried to obtain the similar outcome through a simulating way, in this case, manually change on the data behind. However, the MD5 does make contribution in the attacking methods such as Man-In-The-Middle, where the real data behind might be captured and tampered in the middle way but the attacker can hardly produce a valid fake MD5. Besides, the possibility of this is relatively insignificant due to the fact that transmission using Windows Azure cloud is based on HTTPS.

From the research on software we are aware of that the majority of users do not pay enough attentions to warnings or error messages and seem to be more willing to continue their works. Due to this fact, we do not implement any prompt either of error or warning and as a replacement, absolute deny of access to further visit of the resource takes place when improper behaviour has been detected like alter of original data and attempt to obtain invalid data out of limitation. Unless the data fulfils qualifications to be non-malicious, non-phishing, non-code-injective, non-tampered and still valid, the URL is redirected or the real text data behind is shown, according to the input type. People may argue that this sort of enforcement can fail into low accessibility of data or incorrect block of resources. However we believe that solid implementation of blacklist or filter and improvements to make the system more professional will curb aforementioned failures. Now that prompt to users is inefficient, there is no sufficient reasons to continue employment.

The solution has resolved issues raised in the problem description section. On one hand, private data stored permanently in a QR code is now protected thanks to the adoption of cloud techniques, along with the metadata of count of scans. It is no longer “permanently stored” when an appropriate scan time limit is set. For instance, in a ticket service, once after the successful check-in has made the QR code is not valid anymore,

relieving the concerns that any crime picks up the abandoned QR code and steals the personal data or data which is not wished to be exposed after a number of times of usage. On the other hand, the QR code is more resistible against malware. Users do not exactly know the content behind a traditional QR code until they decode it and enter the sites. We do not try to fight against this and on the contrary we make it more "undetectable" – we take the advantage of GUID, which makes it incomprehensive even when the QR code is decoded. But this does not imply no benefit. Instead of being victim of malware- phishing attacks, to believe in the system we provided ensures that the sole GUID is pointing to a specified blob so that the metadata is trusted. Moreover, the adoption of MD5 makes users more convinced that the real data behind the QR code of GUID is exactly the same as the one the honest Requestor has already made through the systematic scheme.

5 Discussion

Followed by the result, in this chapter a discussion is held revolving the solution we have proposed. Also an evaluation of the result is given in relation of goals, leading to a comprehensive analysis on pros and cons with the solution by comparing it to existing works which have been done and discussed earlier.

5.1 Pros

- Information security

The three key factors in information security are confidentiality, integrity and availability. We aim to accomplish all of them in our solution. The confidentiality is ensured by the limitation and the count of scan times, the set of data wished to be used under a limited times will no longer be available when the maximum of scan times is reached. This plays a crucial role when the real data is of privacy and importance meanwhile is not expected to remain available permanently. For the integrity, MD5 as metadata attached to the blob does the work. Even if it happens under HTTPS that a hacker succeeds in conducting a MITM attack like the one introduced in the article written by Kim et al. [18], MD5 ceases further access of the probably tampered data behind. When it comes to the availability, the blacklist and filter acting as the initial checkpoints of a generation of QR code hit the goal. By doing so, only available resource data passing the confirmation, e.g. a benign website which is not shut down, is provided as long as the scan limits have not been reached. In a more general and essential case, availability of developed sites which are deployed on Windows Azure Cloud, are highly guaranteed with at least 99.9% availability. Also, a service level of higher availability is able to be achieved if extra fees are paid [41].

Besides, as mentioned in chapter 3, code injection is another way attackers would like to go. The code injection can be mitigated by filtering and modern programming language technique like parameterized statement, escaping, pattern

checking [36] [37]. In our solution we have filtering meanwhile the program is written in C# which offers the techniques against code injection to a greater degree.

One thing is also worthy being recognized is that, the service, storage, related sources and communication are on the cloud. This brings significant advantages for the reason that the cloud platform is technically stable, taking many security elements under their own control. Our solution is built on such a trustable platform and makes best use of it to enhance the security of QR code. Take an assumption as a possible attack case, an attacker attempts to launch a Denial-of-Service (DoS) to our system. What he has to do in the first place may be to take the blob storage service down. This is not easily achieved due to plenty of protections have been equipped to such a cloud platform run by the reliable provider Microsoft.

Compared to existing QR code decoders at present, many of them inflexibly follow the decoding algorithm and return data, for instance, ZXing decoder [21] and NeoReader [42] on many platforms, which are two popular barcode decoding software. What they do are barely decoding the QR code and completing specific work like presentation of information and redirection to websites. Only a few of productions like Norton QR scanner [32] takes security concerns into account still what their measure is to show message before redirecting to websites by URL, which are far from satisfaction to prevent users from various tricks and hazards. In addition, those decoders have a list of vulnerabilities but little prevention to code injection as described in the paper “Security Overview of QR Codes” [13]. The vulnerability causes concerns of security. Walking through the current app market and searching for QR-code apps, it might not be hard to find a variety of QR code scanners/decoders similar to ZXing and NeoReader. In Yao and Shin’s study [27], 24 of 31 tested QR code scanning/decoding apps had functionality to push a notification to user to confirm whether to continue visiting or not. This was done through displaying URL to users, which made it possible for users to easily get bored whenever a site is being redirected. The more times the users have to make a choice, the more likely the users would turn to other apps without such annoying

features as such it leads to low level of security. What is more, even that the URLs are showed the users do not have idea about if the URLs are bringing them to benign sites or malicious ones.

Likewise, one of the two countermeasures Kapsalis proposed in his discussion after an empirical study was to display the entire URL but not shortened one and ask the user to either continue or quit [6]. At the same time, he argued that the user awareness of potential risk needed to be educated to a higher level. Still the two countermeasures promoted are not of efficiency.

Our solution has successfully tackled those information security problems in existing productions including guarantee of three key factors, code injection prevention and non-disruptive user browsing experience as such information security within QR code is improved.

- Control over abuse of QR code

Nowadays QR codes are generated without any difficulty. A simple website offering QR code service produces thousands or millions of QR codes without taking control after generations. It has a tendency of QR code abuse and causes more problems of QR code concerning security. Due to the fact that it is of little effort to achieve an on-purpose replacement of original QR code by covering an existing code with a new one, it makes the situation worse to control over abuse of QR code [6], leading disordering to many services in the society including banking and advertising .

The solution offers an entire system both for generation and consumption of data. Credible generation results in reliable retrieval. A system like our product can thus mitigate diverse and unmanageable spread of malicious QR codes. Additionally at the moment there is a demand to modify the data behind which is connected to the particular QR code, it is not efficient to create a totally new one. In the solution we put forward, a QR code has the reusability since that the real data can be changed at the backend and the same QR code of GUID still works at the front side.

- A trusted brand

A trusted brand to be built and applied into practice is a long-term issue. Nevertheless to promote the security, a revolution of scheme usually takes effect. Traditional QR code Requestors do not focus on security other than typically on commercial profit that versatility and convenience of QR code can bring for them, which turning the security of QR code from bad to worse. If a sound scheme including the whole process management rises and people believe in it, the future of QR code security has the opportunity to be improved to a great extent with such a trusted brand and build a good reputation in the society for the public to be more willing to use this kind of new QR-code scheme.

- Tackling QR-code related ethical issues in society

Privacy has been put enough emphasis in society yet traditional QR codes might have brought user privacy into dilemma. On one side the machine-readable QR symbol seems to have protected user privacy by concealing and encoding data behind the symbol. While on the other side the information is exposed after decoding, alternatively in some cases those pieces of data which are expected to be disposable could be failed into being revealed and exploited by people of malicious purposes after a one-time successful scan/decoding. To release this sort of concern, a QR code generated from the developed solution shows GUID which can barely be understood by our software and points to valid storage blob data, as well as gives access to data behind only when the count of scanned/decoded times is less than or equivalent to the limitation. To be more considerate, it is also possible to remove the entire content of data behind, ceasing any attempt to find the one which is not expected to be reused after a one-time usage. Hence the user privacy is protected and sensitive data can be erased when there is a need.

Another contribution of our solution to QR-related ethical issues in the society is QR-code attack mitigation. Users of QR code might be afraid of their data being stolen or damaged by attacks. The two analysed attacking methods, i.e. QR code as attack vector and QR code itself, now become much hard to launch compared to traditional QR scheme armed with insufficient or no protection.

What is more, the stronger the blacklist and filter are, the greater mitigation can be.

QR code is being applied in a growing number of aspects where data must be well protected and kept secret. The solution developed by us opens possibility to be adopted into social usages like payment, ticket system [3]. Take the latter as an example, if a ticket company took use of our developed QR-code scheme for both generation and decoding of QR codes, it would lead into win-win. For the company, QR codes printed on tickets bring convenience, stability and reusability. For the customers, after they consume the ticket service their private data would not risk of being exploited.

5.2 Cons

- Time and money costs

Due to the employment of cloud. It is obvious that such a system has to run on a sustainable support of platform as such costs expenses. On the other hand, average time to generate and decode a QR code becomes longer because of extra security procedure. The latter can be less significant than the former since the speed of generation or decoding do not make a great difference and they are not of remarkable increases on working time. In comparison to existing products, this could be a financial issue and therefore makes it more difficult to compete with other QR-code apps in the market since that those existing QR-code apps appear to be cost-free and open to users [27].

- Network access

Again for that data is stored online on cloud except that the QR code symbol of the GUID is offline, network access is a must. Though network access broadly covers many area, it can still be a defect when the data is badly needed but no steady connection to network is available. If this happens frequently enough,

people might lose confidence of such a promoted scheme and turn back to the traditional ones, which enable users to decode offline.

- Breakthrough into the central

All blobs together with the metadata are stocked at the same place, i.e., the cloud storage containers. Once any attacking attempts to enter the administrate centre of cloud succeeds, the data is revealed and no longer protected. While, this is due to that we take the data onto cloud and make QR code more dynamic. Enhancement on security layer may more or less help.

5.3 Possible Improvements in Future

Initially this system is not designed to make profit however for the sake of convincing the public to have a try, certain measures are necessarily taken to gain financial supports. One of the ideas can be to announce the trustable-brand service provided by this system and the fee shall be charged by the use of GUID. As discussed earlier, a GUID can be reused to pointing to different data. So the envisage has possibility to come into reality.

With the metadata, it opens possibility to add other pieces of information to enhance security. One can think of generating a strong enough encrypted password and attach it to the blob as metadata. Alternatively there could be a log-in function to confirm user authentication before manipulation to blobs, leading to that only authorized users can benefit from the solution including generation, consumption and reuse of the QR codes.

Another meaningful possible improvement in future is to encrypt all metadata mentioned above and save them in separated storage locations or containers. Since MD5 is stored together with the real data behind, count of scanned times and limitation of scans, there is a risk for them of being tampered at the same time without detection. Whereas the encryption of MD5 might be of most significant importance. For this reason, we recommend adoption of MAC (Message

Authentication Code, a piece of code offering integrity and authenticity assurances on the message [32]) instead of MD5, which is keyed cryptographic hash function and has better performance in enhancement of QR code security. For separation of data into different locations, use of GUID might help to match real data with corresponding MD5s or MACs. However there might be some difficulties to implement. First, in order to synchronize and couple data stored in separated locations the decoding software needs to connect to more than one storage containers on the cloud and to update data accordingly. Second, how to allocate and store the key for MAC in a secured manner is of most importance and ought to be carefully considered, since that the generation process is conducted on the cloud but some verification work is done at the decoding terminal of which protection might be relatively weaker. Yet if the two obstacles can be overcome the QR code tend to be much more secure and persuasive in practice.

When confronting the discussed drawbacks that come with the enhancement of security in the section of cons. There are a variety of technologies we have thought of. First of all, more restricted guarantee on information integrity might be accomplished by adopting digital certificate or digital signature, which seem to work upon identification of website owners and protection against phishing [43]. Those technologies can lead to difficulty and/or complexity of implementation yet they are probably taken into account when security is a priority exceling other factors. Likewise to attain confidentiality, one may consider advanced cryptography or steganography which are referred to and recommended in the section of QR code encoding and decoding [13] [14] [15]. Finally there are techniques like colourful QR code [26] to enlarge capacity of QR code, which can be regarded as a solution to problems emerging along with embed of extra security information or data bits.

Last but not least, the responsibility to use our technique in the society needs to be under provision and protected by laws, in case when this QR-code scheme is exploited for malicious or illegal purposes such as being used for disclosure of confidential against agreement. The developed QR-code scheme should only be utilized either by individual or organization within the law.

6 Conclusion

QR code is a novel way to convey data. Its versatility and convenience have made QR code much more appealing. Nevertheless insufficient emphasis put on security of QR code causes problems such as leak of sensitive data and weak protection against malicious attacks. In this paper, we have analysed both QR code itself and related issues, as well as sought approaches to resolve these sorts of problems.

From research conducted on modules of QR code, it proves that once upon generation the QR code is hardly able to be changed according to demands thus it is not dynamic enough to keep updating data. Through study on two categorizes of attacking methods where malicious QR code either acts as attacking vector or does harm itself to systems, we have found there a need to take control over both generation and consumption of QR code rather than adding intermedia as a tool of mitigation. Along with an appropriate choice on error correction level of QR code and what we learned from existing software, a systematic scheme as solution is carried out.

The system we put forward takes advantages of cloud, cryptography and improvement on software. It tackles problem via ensuring information security and setting limit to how many times a QR code should be decoded. The cloud helps to reach our goals to a great degree by providing a relatively safe platform where the essential data of QR code is stored. At the same time, adoption of GUID and metadata has the prospect of being applied into practise to enhance QR code security to a higher stage and to forbid QR code abuse.

Despite the solution provides a list of advantages, finance-, network- and management factors might hinder the development. It can be a long-term journey for the solution to be accepted by the public. However we believe that the pros weigh over the cons in the case that security plays a key role, since that there are so many potential opportunities to employ more and multiple technologies into this solution. We recommend study on how watermarking, steganography and advanced cryptography in encoding and decoding procedure can enhance the QR code security, as such to bring people less concerns on usage of QR code.

7 Bibliography

[1] DENSO WAVE INCORPORATED. *History of QR Code*. http://www.qrcode.com/en/history/ (accessed 15 May 2015)

[2] *QR CODE MONKEY*. http://www.qrcode-monkey.com/#wifi (accessed 15 May 2015)

[3] Speed T, Nykamp D, Heiser M, Anderson J, Nampalli J. *Mobile Security: How to Secure, Privatize, and Recover Your Devices*. 1st ed. United Kingdom. Packt Publishing; 2013

[4] Kieseberg P, Leithner M, Mulazzani M, Munroe L, Schrittwieser S, Sinha M, Weippl E. QR code security. *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*.2010; MoMM \'10:Pages 430-435

[5] Dabrowski A, Krombholz K, Ullrich J, Weippl ER. QR Inception: Barcode-in-Barcode Attacks. *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*.2014; SPSM \'14:3-10

[6] Kapsalis I. Security of QR Codes. *Norwegian University of Science and Technology, Department of Telematics*.2013

[7] DENSO WAVE INCORPORATED. *About 2D Code*. http://archive.is/20120915040049/http://www.qrcode.com/en/aboutqr.html (accessed 15 May 2015).

[8] DENSO WAVE INCORPORATED. *Information capacity and versions of the QR Code*. http://www.qrcode.com/en/about/version.html (accessed 15 May 2015).

[9] DENSO WAVE INCORPORATED. *The maximum data capacity for each version*. http://archive.is/NONi#selection-246.0-246.1 (accessed 15 May 2015).

[10] Thonky.com. *Format and Version Information*. http://www.thonky.com/qr-code-tutorial/format-version-information#example-of-version-7-information-string (accessed 15 May 2015).

[11] RedTitan Technology Ltd. *QRCODE layout*. http://www.pclviewer.com/rs2/qrtopology.htm (accessed 15 May 2015).

[12] Bobmath. *QR Character Placement*. http://commons.wikimedia.org/wiki/File:QR_Character_Placement.svg (accessed 15 May 2015).

[13] Peng K, Sanabria H, Wu D, Zhu C. *Security Overview of QR Codes*. https://courses.csail.mit.edu/6.857/2014/files/12-peng-sanabria-wu-zhu-qr-codes.pdf (accessed 23 May 2015).

[14] Chen JH, Chen WY, Chen CH. Identification Recovery Scheme using Quick Response (QR) Code and Watermarking Technique. *Applied Mathematics & Information Sciences*.2014; 8(2):585-596

[15] Muthaiah RM, Krishnamoorthy N. An Efficient Technique for Data Hiding with use of QR Codes-Overcoming the Pros and Cons of Cryptography and Steganography to Keep the Hidden Data Secretive. *International Journal of Computer Applications (0975 – 8887)*.2014; 100(14):1-5

[16] Rivest R. The MD5 Message-Digest Algorithm. *MIT Laboratory for Computer Science and RSA Data Security*; RFC 1321:1-20

[17] Patange T. *How to defend yourself against MITM or Man-in-the-middle attack*. http://hackerspace.lifehacker.com/how-to-defend-yourself-against-mitm-or-man-in-the-middl-1461796382 (accessed 15 May 2015).

[18] Kim SH, Choi D, Jin SH, Lee SH. Geo-location based QR-Code authentication scheme to defeat active real-time phishing attack. *Proceedings of the 2013 ACM workshop on Digital identity management*.2013;DIM \'13 :51-62

[19] Narayanan AS. QR Codes and Security Solutions. *International Journal of Computer Science and Telecommunication*.2012; 3(7):69-72

[20] Krombholz K, Frühwirt P, Kieseberg P, Kapsalis I, Huber M, Weippl ER. QR Code Security: A Survey of Attacks and Challenges for Usable Security. *International Conference on Human Aspects of Information Security, Privacy and Trust at the 16th International Conference on Human-Computer Interaction (HCI)*.2014; 8533(LNCS):79-90

[21] Official ZXing ("Zebra Crossing") project home. *zxing*. https://github.com/zxing/zxing (accessed 15 May 2015)

[22] letswibe.com. *Short Payment Descriptor*. http://www.letswibe.com/wibe/Short-Payment-Descriptor/ (accessed 15 May 2015)

[23] Microsoft by Neudesic, LLC.. .*NET*. http://www.asp.net/ (accessed 15 May 2015)

[24] Microsoft. *Microsoft Azure: Cloud Computing Platform & Services*. http://azure.microsoft.com/en-us/ (accessed 15 May 2015).

[25] DENSO ADC. *QR Code® Essentials*. http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo%3D&tabid=1426&mid=4802 (accessed 15 May 2015)

[26] Pandya KH, Galiyawala HJ. A Survey on QR Codes: in context of Research and Application. *International Journal of Emerging Technology and Advanced Engineering*.2014; 4(3):258-262

[27] Yao H, Shin D. Towards preventing QR code based attacks on android phone using security warnings. *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*.2013; ASIA CCS \'13:341-346

[28] Zhang Y, Egelman S, Cranor L, Hong J. Phinding Phish: Evaluating Anti-Phishing Tools. *In Proceedings of the 14th Annual Network and Distributed System Security Symposium*.2006; NDSS 2007

[29] Egelman S, Cranor LF, Hong J. You\'ve been warned: an empirical study of the effectiveness of web browser phishing warnings. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.2008; CHI \'08:1065-1074

[30] Microsoft. *2.5.5 Globally Unique Identifiers (GUIDs)*. https://msdn.microsoft.com/en-us/library/cc246025.aspx (accessed 15 May 2015)

[31] Myers T. *How to use Blob storage from .NET*. http://azure.microsoft.com/sv-se/documentation/articles/storage-dotnet-how-to-use-blobs/ (accessed 15 May 2015).

[32] ZyTrax, Inc. *Survival Guide - Encryption, Authentication*. http://tilt.lib.tsinghua.edu.cn/docs/books/ProDNSandBind/www.zytrax.com/tech/survival/encryption.html (accessed 11 June 2015)

[33] Katz J, Lindell Y. *Introduction to Modern Cryptography*. 1st ed. Chapman & Hall/CRC; 2007

[34] Google. *Norton Snap qr code reader*. https://play.google.com/store/apps/details?id=com.symantec.norton.snap&hl=en (accessed 23 May 2015).

[35] Kaspersky Lab. *Kaspersky Lab*. http://www.kaspersky.com/ (accessed 15 May 2015

[36] James J. *SQL Injection*. https://technet.microsoft.com/en-us/library/ms161953(v=SQL.105).aspx (accessed 23 May 2015)

[37] Smithline N. *Top 10 2013-A1-Injection*. https://www.owasp.org/index.php/Top_10_2013-A1-Injection (accessed 23 May 2015)

[38] URI Planning Interest Group, W3C/IETF. URIs, URLs, and URNs: Clarifications and Recommendations 1.0. *W3C Note*.2001;

[39] Cisco Systems, Inc. Using Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS). In: Cisco Systems, Inc. (eds.) Cisco CallManager Security Guide. 8th ed. United State: Cisco Systems, Inc.; 2010. p2.1-2.12

[40] Microsoft. *Enable or Disable Directory Browsing in IIS 7*. https://technet.microsoft.com/sv-se/library/cc731109(v=ws.10).aspx (accessed 12 June 2015

[41] Microsoft. *Service Level Agreements*. http://azure.microsoft.com/en-us/support/legal/sla/ (accessed 11 June 2015)

[42] NeoMedia. *NeoReader*. http://www.neoreader.com/ (accessed 11 June 2015)

[43] Aldridge E. *Using digital certificates to identify website owners and protect against phishing*. http://www.infosecwriters.com/text_resources/pdf/Exploiting_PKI.pdf (accessed 12 May 2015).

8 Appendix

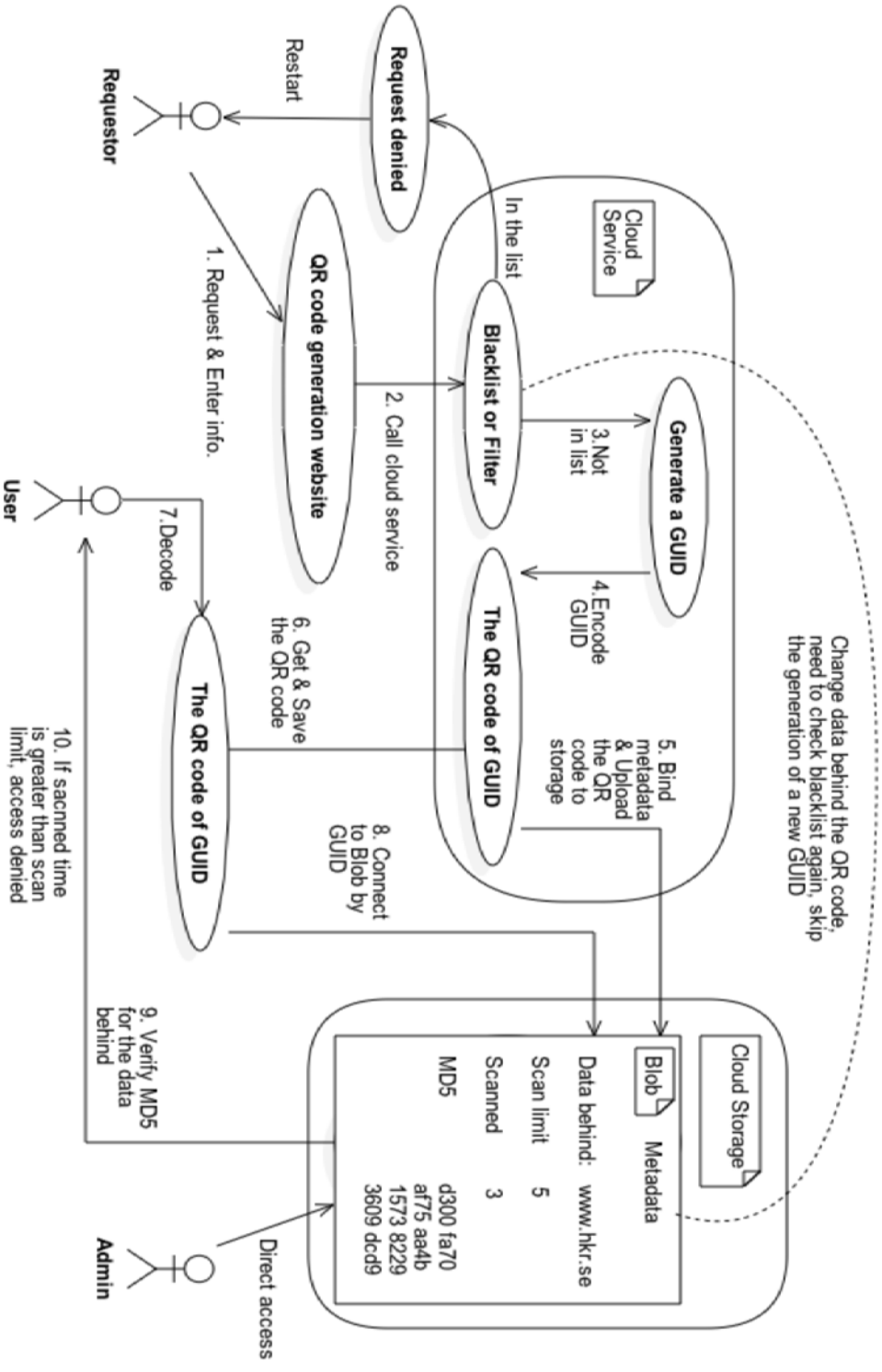

Figure 12. Enlarged schematic diagram of solution