ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

A novel Two-Factor HoneyToken Authentication Mechanism

ΒΑΣΙΛΗΣ ΠΑΠΑΣΠΥΡΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Επιβλέπων
ΘΕΟΔΩΡΟΣ ΤΣΙΦΤΣΗΣ

Λαμία, 19 ΜΑΡΤΙΟΥ έτος 2021

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

INFORMATICS AND COMPUTATIONAL BIOMEDICINE

A novel Two-Factor HoneyToken Authentication Mechanism

VASILIS PAPASPIROU

Master thesis

THEODOROS TSIFTSIS

Lamia

19 MARCH YEAR 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ ……..

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

A novel Two-Factor HoneyToken Authentication Mechanism

ΒΑΣΙΛΗΣ ΠΑΠΑΣΠΥΡΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων/σα
ΘΕΟΔΩΡΟΣ ΤΣΙΦΤΣΗΣ
Λαμία, 19 ΜΑΡΤΙΟΥ έτος 2021

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

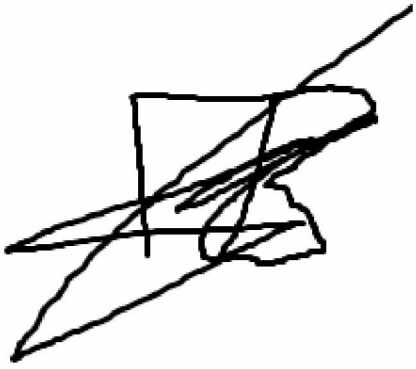
Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο [«τίτλος εργασίας»] αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η ΔΗΛΩΝ/-ΟΥΣΑ

ΒΑΣΙΛΗΣ ΠΑΠΑΣΠΥΡΟΥ

Ημερομηνία 19 ΜΑΡΤΙΟΥ 2021

Υπογραφή



A novel Two-Factor HoneyToken Authentication Mechanism

ΒΑΣΙΛΗΣ ΠΑΠΑΣΠΥΡΟΥ

Τριμελής Επιτροπή:

ΘΕΟΔΩΡΟΣ ΤΣΙΦΤΣΗΣ

ΚΩΝ/ΝΟΣ ΚΟΛΟΜΒΑΤΣΟΣ

ΛΕΑΝΡΟΣ ΜΑΓΛΑΡΑΣ

Επιστημονικός Σύμβουλος:

ΛΕΑΝΡΟΣ ΜΑΓΛΑΡΑΣ

ABSTRACT

The majority of computer systems rely on user authentication on passwords, but passwords have so many weaknesses and widespread use that easily raise significant security concerns, regardless of their encrypted form. Users hold the same password for different accounts, administrators never check password files for flaws that might lead to a successful cracking, and the lack of a tight security policy regarding regular password replacement are a few problems that need to be addressed.

The proposed MSc Thesis aims at enhancing this security mechanism, prevent penetrations, password theft, and attempted break-ins towards securing computing systems. The selected solution approach is two-folded; it implements a two-factor authentication scheme to prevent unauthorized access, accompanied by Honeyword principles to detect corrupted or stolen tokens. Both can be integrated into any platform or web application with the use of QR codes and a mobile phone.

A prototype system with 3 qrimages and 3 OTP (one time password) which give the user one more layer of protection is proposed. When the user logins at the server then 3 qrimages and one sms at the same moment, with the 3 OTP codes, will be shown at the user. Next, the user will have the options to scan the qrimage or choose the right OTP password from the sms. In either way, the user should choose to fill the box properly in order to have access to the rest of the system.

Contents

CHAPTER 1

1.0 Introduction

A wide range of areas, such as banks, government applications, the pharmaceutical sector, military organizations, educational establishments, etc., security issues are of great importance. Government institutions set guidelines, pass regulations, and compel organizations and agencies to conform with these standards with vast variery implications of non-compliance.

In these various and varied industries with a common, not strong-weak, link being passwords, there are many challenges when it comes to security issues. To identify the user, most applications today rely on static passwords. These keys, though, come with serious security issues for administrators. Users prefer to use easy-to-guess passwords, use different accounts with the identical password, write passwords or save them on their computers unencrypted. Moreover, although dedicated systems, called password managers, can offer secure password storage and retrieval, only a small fraction of users use. In addition, hackers have the choice of using many password stealing methods, such as shoulder surfing, snooping, sniffing, guessing, etc. Several best practices have been suggested for the use of passwords.

Most of them have great degree of difficulty of complexity and to use and others do not fulfill the security needs of the organization. To overcome the password problem, Two-factor authentication was advised utilizing tools including tokens and ATM cards and has been shown to be difficult to hack. There are several limitations of two-factor authentication, including the purchase, issuance, plus processing costs of tokens or cards. According to user's perspective, having more than one two-factor authentication methods demands the purchase of several tokens/cards that are likely to be misplaced or stolen.

Traditionally, cell phones have been considered a device for mostly making phone calls. However presently, the use of cell phones has been generalized to send calls, review addresses, shop contacts, etc., provided the developments in hardware and software. Also, opportunities for smartphone access have expanded.

Cell phones combine infra-red, Bluetooth, and WLAN connectivity, on top of normal 4G/5G connectivity.

For contact purposes, the majority, not all though, hold mobile phones. Several accessible Mobile banking services enjoy the benefits of mobile computer enhancement capacities. From the ability to collect account balance information in the form of SMS messages to the use of WAP and Java along with GPRS to enable transfers of funds between transactions, stock trading, and direct payment confirmation through the phone's micro browser.

The principle of using passwords and smart cards to authenticate customers is an old idea going back forty years now. Since then, many systems with two-factor authentication mechanisms were developed. However, because the smart card could be intercepted and the data contained in the smart card may be duplicated, the reliability of two-factor authentication may be breached, and the number of potential passwords can be limited and users could forget or lose their passwords.

Biometric authentication was adopted to verify consumers with the use of their biometric data due to those issues.

Scholars have suggested biometric authentication system since back in 1999 which enhances some facets of two-factor authentication since biometric features have greater entropy and cannot be missed and are rarely lost. One of some drawbacks, though is that biometric characteristics are not entirely confidential because one can "steal", biometric functionality from some other for example, the fingerprint can be retrieved from a mug used by the suspect and the facial features can be obtained from an image of a user. Combining all these three variables together is a way to mitigate these concerns. This technique is often referred to as three-factor authentication, and Cloud-based systems have been extensively optimized.

SIM cards are available in varying storage sizes. Related memory utilization of the SIM card connected with it plays a part in deciding the effectiveness of cloning the SIM card, more memory stored on the original SIM card than the longer the Ki A8 algorithm cracking process on the SIM card. Problems resulting from the above perspective relating to the inclusion of the A8 algorithm inserted in any SIM card used by telecommunications users to duplicate or replicate the SIM card are detrimental to the privacy and protection of cell phone users on either side.

The purpose of the SIM card cloning research is to provide an alert to consumer safety and provide a dedicated SIM card to tackle SIM card cloning criminal investigations along with their abuse of data. Authentication of subscribers Based on IMSI (Stored on SIM) and

Random Number Generator/RAND (Provided by Network), SIM card cloning authentication will be further investigated by comparing the network login response of the recipient to the mobile service network. The Random Number Generator (RAND) includes an algorithm A3 (Provided by Network) such that RAND participates in the process of cloning the SIM card in order to adapt the algorithms contained in the SIM card A8 to A3 algorithms contained in the user data of the connected network authentication.

Scholars have already demonstrated that an intruder will damage another system by performing a cross-platform infection attempt, either a PC or a cell phone. Prototypes of proof-of-concept demonstrate that all these assaults are viable and thus it is not fair to preclude them from the mobile 2FA scheme design Competitor. The intruder will snatch all authentication tokens as well as the rightful recipient impersonate, when both 2FA devices are infected, regardless of what individual smartphone 2FA instantiation is used. We carry out in order to justify our claim, actions towards different instantiations of mobile 2FA schemes implemented by banks and mainstream Internet service providers.

Schemes with 2FA OTPs created on the client side, such as Google Authenticator (GA), depend on secrets that are pre-shared. The configuration process of the GA app, used across hundreds of providers, including Google Mail, Facebook and Outlook.com, was evaluated. When the user allows GA-based authentication in his profile setup, the GA initialization begins. A QR code is created by the service provider and shown to the user also reviewed by the user's smartphone (mostly on Computer). The QR code stores all of the necessary information to configure GA with user-specific account specifics and pre-shared secrets. Academics evaluated the QR code supplied by Facebook and Google during the initialization stage and described the form of the QR code. This contains information like that of the type of scheme (counter-based vs. time-based), the identity of the service and account, the counter (counter-based mode only the OTP period formed and the shared secret identifier).

In addition, all this material is provided in plain text. To check if GA supports any alternate initialization system, scholars with the help of the JEB Decompiler make an engineered that is reversible and evaluated the internal apps. We have not found any alternate initialization patterns, suggesting that this initialization protocol is used by 32 service distributors. which is all, using GA.

The initialization message may be intercepted by a PC-residing malware (pretty obvious content stored in QR code format). The intruder will therefore trigger the GA's own copy and will be able to create valid OTPs.

The use of 'honeywords' was presented in order to track if or not the password file was stolen, i.e., a series of false passwords that are combined with the original password of the user and Such passwords (authentic passwords and honeywords) have their hash values in the password directory.
The adversary also does not know which one is the true password if the specific file is corrupted and all the hash values in the file system are hacked. Note that LS identity and password are submitted by the customer or the adversary to request login.LS then checks if a password is requested, among the honeywords of a user, but even if this search achieved, LS needs to review another protected component, HC, to see if the index of the honeyword reclaimed corresponds to the actual password of the user. HC warns the administrator otherwise, as a honeyword signal has been detected that the password file might have been corrupted

Based on these findings and trying to combine the strengths of honeywords and 2FAs while at the same time keeping the system simple and easily integrated in any existing platform or system, we present in this paper a prototype of a novel security mechanism. We develop and propose an innovative security mechanism for web applications that produces both passwords and QR codes covering different login modes. The proposed system entitled "Two-Factor Honeytoken Authentication (2FHA)", combines the strengths of two-factor authentication and Honeyword technologies. In the developed prototype a SMS with 3 OTP passwords that correspond to 3 QR codes is sent to the user. Only one of these three elements is the correct token that can be used in order to continue. This induces an extra layer of security adding more safety to the system. The proposed system offers enhanced security to the user while at the same time is simple and doesn't impose additional overhead during login.

Two-factor authentication (2FA) is a trusted entity where certain users utilize two distinct authentication keys to validate themselves, often linked to as two-step verification or dual-factor authentication. One such phase of this procedure is carried out to better secure each the user's credentials and the software that the user may use. Two-factor authentication

offers a stronger degree of defense than one-factor authentication (SFA)-dependent authentication schemes where only one factor, most of times are a password or passcode or something else, is given by the user. Two-factor security methods are based on a system that includes a password, along with a second element, either a safety token or a biometric variable, such as a fingerprint or a facial scan, are typically. Two-factor authentication provides an extra layer of encryption to the authentication process by making it harder for attackers to obtain access to the machines or internet domains of a person, because knowing the victim's password by itself is not sufficient to complete the authentication check.

Two-factor authentication has traditionally been shown to control access to confidential apps and documents, and 2FA is now being used by network operators to shield the identity of their users from being used by attackers who have compromised a password database or used phishing scams, cyberattacks to extract user passwords.

It is clear from Polybius' explanation that the scheme for the delivery of watchwords in the Roman army that, since ancient times, passwords or watchwords have been used. The password scheme functions in the military tradition as a pair of hidden words or phrases; one question and one answer.

For e.g., the US 101stAirborne Division paratroopers used the password flash in the starting days in war of the Battle of Normandy, which was posed as a big challenge and a mount that must be climbed, and replied with the correct answer, that was thunder. Every three days, the challenge and answer were altered.
Similarly, instead of a password scheme, the US paratroopers used a gadget identified as a 'cricket' on 'D-Day'(Tuesday, 6 June 1944by 6:30am), one metallic click provided by the gadget was to be met by two clicks in reaction instead of a password contest as a momentarily unique method of identity.

Since the early days of computing, passwords have been used on computers. MIT's Compatible Time-Sharing System (CTSS), was among the first time-sharing operating systems, was introduced in 1961. A login command was installed, which demanded a password for the user. The machine would turn down the printing process when the user typed in a password, so that the user could type in his password with anonymity.

Passwords represent the top and the start line of protection of most computer-based data management systems as a fundamental means of access control. Studies have found that much of the issues associated with the care-free mindset of users have a lot to do with and user's various required passwords.

Experience indicates that a regular user of the Internet has more than 60 passwords and PINs for multiple apps and services; up to 25 percent will not be able to memorize those with the strongest memory. Storage, password length and composition are thus the resulting concerns.

As a result, password users switch to views adverse to passwords protection in order to rid the brain of excessive tension. When a survey found that fifty percent of users wrote down their passwords, the security vulnerability associated with such attitudes is common. Researchers are also mixed as to whether it is easier to type down codes or not.

A vast formation of password use protection standards reveals that passwords do not have a common standard; different implementations have not the same specifications. If this scenario is examined against the background of that reality, that an ordinary user has many passwords, all of which are supposed to be solid, in accordance with inherent the Corruptibility of Mankind, all the circumstances associated with the password scheme are evidently unpractical to detect by any individual entity.

At the University of Bradford a research was developed, is therefore intended to suggest a potential way out of the password protection purgatory phenomenon, by dreaming of passwords that will take into account such as human and security considerations, as it is the security of the overall mechanism that is relevant.

This MSc Thesis would address some of the efforts to solve the password protection problem, a report on password security knowledge in developed countries, the issues with password protection, plus a suggestion for a proposed approach in an effort to accomplish the goal mentioned above.

1.2 Problems related to the encryption of passwords

A Consideration in Password System Protection the protection in a process that's password shielded is based on many factors. Among them is the need for sound protections to be planned for the overall framework, including protection from malware, eavesdroppers and similar risks. It is also important to take care of physical protection against dangers such as shoulder surfing, video recorder and keyboard sniffers. Passwords can also be selected such that they are difficult to guess and also difficult to find by an attacker using any of the automated attack schemes available. Hiding passwords while they are being entered as a precaution toward bystanders reading the passwords is already standard procedure for the device. Since this practice will lead to mistakes and tension, causing users to use poor and easily breakable passwords, specialists are of the belief that the mechanism should be configured such that users can reveal or conceal the passwords when they are typed in avoiding guessing or any similar factors and brute-force attacks, password strength is a metric of how powerful a password is; Duration, maziness and randomness are a part of it.

1.3 Multiple logins and related problems

It's incredible the measure of carelessness involved with using passwords. Studies have shown, however, that most of the concerns associated with the care-free approach of users with regard to the use of passwords have a lot to do with the multiplicity of passwords used by a person experience has demonstrated that for multiple apps and utilities, an average Internet user may have over sixty passwords and PINs; of these, up to 25 percent will not be able to memorize those with the strongest memory. Storage, password length and composition are thus the resulting concerns. As a consequence, password users turn to attitudes that are counterproductive to the protection of passwords and, by default, the security of the device they were built to secure in order to rid the brain of unnecessary stress. Even these negative stereotypes provide: all passwords in a diary are written using the same password for all applications; comparing the password to the individual application, e.g., using the number of the room and the initials of the occupant as keys to the door of the office; using very basic combinations such as 34343434, 12345678 or 1q2w3e4r; pasting passwords on the

wall, screen or device, etc. As a survey found that 50 percent of users write their passwords down, the security risk associated with these activities is common.

1.4 Repositories with Codes.

The proliferation of passwords has generated the question of password storage. This has given rise to numerous implementations of technologies intended to make password handling simpler. Collectively, they are called wallets and are in two distinct types. The first is a directory for username/password; an encrypted file saved on one's computer that contains information that needs to be signed into one's different accounts. The most prominent of these is Darn! Passwords It has a password generator that can generate passwords for multiple apps and helps one to drag one's passwords into the website or program that one uses. It helps one, instead of many, to recall only one password. Password Secure and Q*Wallet, both for windows, are related programs. On the Macintosh and PalmOS, Selznick PassWallet offers similar features. For UNIX or LINUX, obviously, no related product exists.

1.5 Login Access Compliance Recommendations

If necessary, it is typically easier to have passwords centrally managed. Whatever the case, users are generally recommended to follow certain recommendations in order to improve the protection of security entry , including It should be kept completely concealed.; it should not be revealed to any other user; It must not be recorded or saved in a way where other people will see it; If there is even the slightest hint or suspicion of a solution, it must be changed; it must be altered if a member of the company leaves the community or alterations the task; It must have at least eight characters. (mixed case/symbols alpha-numeric); it should not be created by any apparent source, e.g. Username or group/company/project name; it should be updated monthly or at least bi-monthly; the higher the risk or the more fragile the properties being secured, the

better. must be changed more frequently; It must not be used, i.e. not contained in a macro function, in an automatic log-in procedure; it should not be a dictionary phrase.

1.6 Solid Password recommendations

Guidelines for selecting appropriate passwords are designed through clever guessing to allow passwords less easily found. Popular guidelines include: minimum password length, if allowed, between 12 and 14 characters; generation of passwords randomly where obtainable; Avoid passwords based on redundancy, dictionary terms, sequences of letters or numbers, usernames, relative or pet names, etcetera; use numbers and symbols in passwords if permitted by the system; use capital and lower-case letters if the system considers case as important; prevent doing something that the general public or your peers learn you want or not like; Use mnemonic word/phrase acronyms; provide a non-keyboard option (for example, spoken or biometric passwords); include some authentication mechanisms, such as 2-factor authentication (knowing and possessing something); fill up the passwords, as can be seen from the above, researchers are split on whether it is best to write down passwords or otherwise. Certain recommendations warn against writing down passwords, while others recommend writing down passwords as frequently as the login lists written down are stored in a secure location, such as a wallet or safe, considering the vast number of passwords protected systems users must access. Not attached to a computer or in the drawer of a desk which is not locked.

Moreover, others also argue that, for the following reasons, the idea of password expiration is now outdated: requiring users to alternate passwords also facilitates easy and poor passwords; If you have a very strong password, there is no sense in modifying it-switching passwords that are already strong poses a chance that the new password will be less powerful; an attacker is likely to use a compromised password to install a backdoor instantly, sometimes by privilege escalation. Once this is done, potential intruder entry will not be hindered by login modifications; It doesn't gain much protection, mathematically speaking, such as never altering a password to switching it

with any authentication try (pass or fail attempts), before successfully guessing the password in a brute-force attack, just half the amount of attempts the attacker would make on average-one achieves even more protection just extending the password length by one character than switching the password on each demand.

1.7 Guidelines on Password Management

A password management scheme is an institutional structure aimed at delivering an efficient digital resource that
guarantees password consistency and enforces its use in compliance with the policy of the protection manager. Password protection can usually facilitate protected login procedures and defend passwords from unintended usage and access. This requires steps that
guarantee that passwords are saved using a one-way encryption algorithm in directories that are distinct from the main operating framework details. These precautions have some protection against password cracker systems and dictionary attacks. As part of the
customer-producer separation process, the manufacturer's original (default) passwords must be replaced following installation of the equipment.

1.8 Training and education for protection awareness

Each organization should have a program of safety awareness training that guarantees that organizations are accountable for training not just their own employees, but also some agents and contractors which they belong to them and also have admission to their facilities. Primary planning will need to provide a revision of the criteria and customized training needs for the organization's unique safety protocols, practices and technologies, depending on the level of security obligations for various consumer groups. A safety training curriculum should provide sensitivity training addressing internal security policies, management of codes, notification of accidents, and viruses; daily safety alerts performed as updates to basic safety education; User education on the safety of viruses, including detection, notification and prevention measures; user

education on the value of tracking the success/failure of log-ins and how to report inconsistencies, and the duty of workers to ensure information security; Password security and user education, with meticulously planned out operational guidelines to be practiced in establishing, modifying, and maintaining password confidentiality. Staff should also be aware of the need for the numerous techniques employed as an efficient means to monitor human hacking or social hackers in the company's password security architecture (socio-cryptanalysts). Let everybody know that there can be no technological hacking in the void (independent of human hacking), and that combating the attacks of the SE is still a purgatory venture (very difficult, complex and complicated endeavor). This underlines the value of developing education in social engineering in all facets of human activities, especially in the security arena.

CHAPTER 2

2.1 QR CODES

The first to propose the notion of secret exchange is Blakley and Shamir. At their emergence, hidden sharing networks have been critical instruments that are used in cryptography and distributed computing with many purposes. The following can be described as a definition of a general k-out-of-n, or (k, n), threshold secret sharing scheme. Information in the structure of a secret, D, are divided into n pieces, $D_1, D_2, \ldots, D_n$, where $n > 1$. The n pieces of the secret are called as shares. The secret is splitted into shares in a way that recognition of any k, or more shares, will allow D to be reclaimed, whereas knowledge of only $k - 1$, or fewer shares, makes the recovery of D not possible.

The following Is a threshold secret sharing pattern that has base on polynomial interpolation:

Definition 1. In a two-dimensional plane, for k points $(x1, y1), \ldots, (xk, yk)$, a unique and only one can be polynomial q(x) of degree k − 1 where $\mathrm{q(xi)} = \mathrm{yi}$ for all i. Let D be a number. D can be split into shares, D_i, by choosing a random k − 1 degree polynomial $q(x) = a0 + a1x + \ldots + ak - 1xk - 1 \ (mod\ p)$, such that a_0 = D and $D1 = q(1), \ldots, Di = q(i), \ldots, Dn = q(n)$, and p is a prime where $\mathrm{p} > \max(\mathrm{n}, |\mathrm{D}|)$.

With information of any k, or more, shares D_i along with the respective indices, i.e., $(\mathrm{xi}, \mathrm{q(xi)})$ where $\mathrm{i} = \{1, \ldots, \mathrm{k}\}$, the coefficients of q(x) can be interpolated to obtain the value of D = q (0). This can be done using Lagrange polynomial interpolation:

$$D = q(0) = \sum_{i=1}^{k} q(xi)\, \Pi_{j=1, j\neq i}^{k} \frac{-x_j}{x_1 - x_j} \ (mod\ p) \quad (1)$$

Conversely, it is difficult to evaluate the value of D with knowledge of just k – 1 or less "Di" since there is inadequate data.

For several years, a series of varied secret exchange systems of varying degrees of difficulty have been suggested. In the course of producing shares and/or retrieving the secret, certain schemes require complicated computational computation. Some systems, on the other hand, involve little to no computation. Digital cryptography, for example, is a widely recognized form of visual secret exchange in which the shares are binary icons composed of black and white pixels. A hidden picture is broken into a variety of shares. Each share is intended to have a transparency printed separately. When piling the predefined number of transparencies, the key is exposed. This is because the human visual process is able to interpret the black and white pixel inputs to achieve the secret without needing any sort of calculation by stacking the transparencies.

One of some disadvantages of conventional visual cryptography, though, is that each fragment is an image of a random black and white pixel pattern. A graphic share of cryptography is apparent even to a novice observer due to the apparently insignificant sequence of pixels. Extended visual cryptography is a process of encoding shares inside cover images to address this problem. Each share is, thus, a meaningful picture. The value of encoding shares into meaningful "innocent-looking" cover photos is that it helps to reduce the possibility of drawing attackers' unwelcome interest. Nevertheless, each share is a very noisy looking picture in extended visual cryptography.

In this research, the strategy analyzed aims to deter attackers from paying attention by circulating shares using innocent-looking QR codes that include both public and secret knowledge.

2.2 Quick Response Codes

A simple answer, (QR) code consists of many light and dark squares, known as modules, and is a two-dimensional code. The International Organization for Standardization (ISO) has established a standard for the QR code: ISO/IEC18004. The specification specifies forty

variations of QR code of which each version consists of a varying number of units, resulting in varying storage capacities. Various data types, including integer, alphanumeric, binary and kanji, can be encoded by QR codes. An intrinsic aspect of QR codes is error correction. And when part of a QR code is dusty, blurred or harmed, it makes for proper decoding. There are a total of four levels of error correction (i.e., low ~ 7 percent, medium ~ 15 percent, quartile ~ 25 percent and high ~ 30 percent), and a separate amount of error correction is given for each level. Higher amounts of raising the quantity of error that can be allowed when using error correction., but the size of the encoded data is also increased. This error correction mechanism makes it possible to decipher the contents of a QR code even though part of it is blurred, as long as the error number does not surpass the error correction power.

As such, by intentionally making a mistake, such as adding an image into a QR code, many people have abused this property for different purposes. An instance of a QR code system is seen in Figure1. It can be shown that, as well as feature patterns, the structure consists of encoding regions. Method patterns do not encrypt data and are used to extract basic QR code information, e.g., the version and degree of error correction. Data codewords and error correction codewords make up the encoding area. There are eight bits of a codeword. Data is encoded as a stream of bits in a QR code, which is separated into a series of codewords.

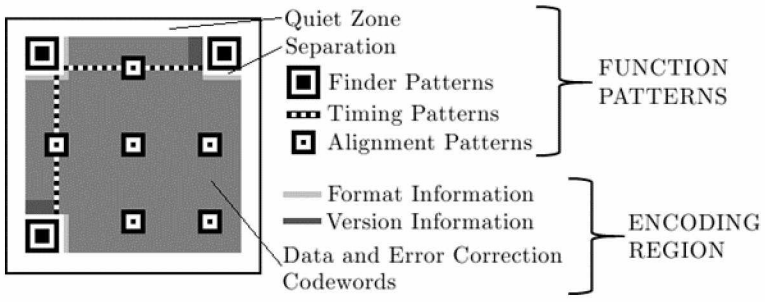


Figure 1 QR code Version 7 structure [20]

The codewords for data and error correction in a QR code are broken into one or more blocks. Based on the QR code version and error correction stage, the unique number of blocks is calculated along with the overall number of data and error correction codewords. The error correction codewords are added to the bottom of the data codeword chain for each block and encoded in an interleaved fashion within the blocks. The aim of this is to

decrease the possibility that an undecidable QR code will result in localized injury. Figure 2 indicates the structure inside a Version 4 QR code of data and error correction codewords, with error correction level H.

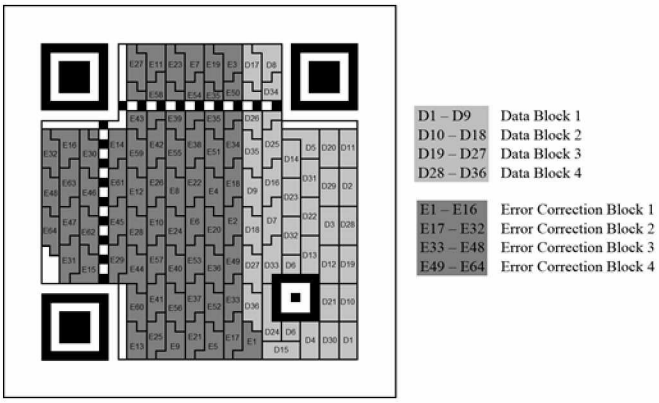


Figure 2 Arrangement of the data and error correction codewords within a Version 4 QR code, with error correction level H [20]

2.3 Related Work

For a number of different reasons, the use of QR codes in information security has been suggested, such as for authenticating visual cryptography shares [22], e-voting verification [21], optical watermarking, data hiding and secret sharing. This section reveals many studies in this paper that are relevant to the work. Using QR codes, experts have previously suggested steganography and data hiding strategies. Wu *et al.* proposed a technique [23] for embedding data in the form of a QR code inside a digital image. Their aim was to mask the presence of a QR code inside a digital image in a way that limits the visual quality deterioration of the image. By embracing the use of QR codes, Chen and Wang [24] proposed an imaging steganography system. The aim of their framework was to combine two forms of hidden data, text data (lossless) and image data, into a cover image (lossy). The text data was encoded in a QR code form. A relatively similar approach was also defined by Chung et al. [25]. In order to hide hidden data in the cover QR code, Lin et al. [26] capitalized on the error correcting replication mechanism of QR codes.

The amount of hidden data that can be shielded is determined by the QR code's data power, which in turn depends on the version of the QR code that is being used and its degree of error correction. Bui *et al.* [27] introduced a way of concealing a hidden message inside a QR code using Reed-Solomon code which can be used for error correction and list

encoding in QR codes in a separate method. Their goal was to solve the vulnerabilities of other methods to change, attacks that use bit embedding to conceal hidden messages in QR codes. In Chow et al [28], a way of covering a QR code containing hidden information was explored in a public QR code in previous work. Covert QR codes were named for this method.

The aim of the strategy was to use visual subterfuge in a QR code to embed a coded secret message and only approved entities could extract the secret. This will encourage a dealer to spread a hidden message through public media to a party of authorized persons. A distributor, for example, might put a poster in a public place with a QR code. The random observer would only be able to reclaim the public message, while the key will be accessed by all authorized persons. In the case where a dealer needs to broadcast the similar hidden message to all approved recipients, this strategy is beneficial.

On the other hand, the approach examined in this paper [20] is intended to give each approved participant a separate share, and the secret can be retrieved only by coordination between the threshold numbers of participants. It has also suggested a variety of strategies for using QR codes in hidden sharing schemes. A process of embedding visual cryptography shares inside cover QR codes was demonstrated by Wan et al. [29]. This technique was called a (k, n) visual hidden sharing system, based on QR codes. The key could be retrieved by the humane visual system based on visual cryptography concepts by adding shares in QR codes while a qualified number of cover QR codes were stacked. Although the value of this approach is that no measurement is required to decipher the key, the downside is that in order to include substantial visual cryptographic shares, the scale of the cover QR codes must be quite big. Cheng et al. [30] have also suggested a further solution focused on QR codes and visual cryptography. The access framework of visual cryptography schemes is used in their approach to transmit and embed hidden data into shares of QR code. This is achieved by splitting a cell into 3-3 sub-modules where the public message resides in the middle sub-module, but the surrounding sub-modules are used to embed a framework of access to visual cryptography. While the resulting QR code shares are all true QR codes and can be scanned by a standard QR code reader, due to the embedding of hidden data by changing the sub-modules, they don't look like normal QR codes. The decryption is based on an XOR (exclusive or) operation.

Lin [31] has investigated a distributed secret exchange solution using QR codes. This technique is a hidden exchange system (n, n), where a secret is broken into shares and encrypted until they are stored inside QR codes labelled. When all the shares are open, the secret can be recovered. In this method, a cheater avoidance framework was also implemented. In Chow *et al. [28],* a particular (n, n) QR code key exchange strategy was presented. By transmitting and embedding data from a hidden QR code within n cover QR code shares, this method often takes advantage of the error correcting mechanism that is a characteristic feature of the QR code structure. The data from the QR code shares can be obtained to access a restored hidden QR code in order to retrieve the secret, which can then be decoded to expose the hidden message. The value of this technique is that it does not require any encryption keys.

2.4 Protocol and Adversarial Models

This segment explains the protocol and adversarial models, along with the underlying principles, in order to analyze the security of the proposed protocol.

2.4.1 Protocol Model

The protocol is modelled on the basis of the entities mentioned below: a dealer who knows all participants' identities and has the ability to access to the scheme executing the stage of encryption and concealment as defined in Section 2.4.4. Each of the participants is fitted with a QR code reader system that holds the cryptographic keys and performs the export and decryption process as defined in Section 2.4.4. The dealer will submit QR codes over public not secure networks, even on printed media, to spread a password.

The protocol ensures that no information that can be used to evaluate the secret or other valuable data relating to the secret is revealed by the secret sharing system introduced in accordance with the protocol. In addition, any k-1, or fewer, shares will also not disclose any confidential information, thus precluding any sort of collusion between participants. Moreover, the protocol implies that the dealer and participants have a stable channel for exchanging a symmetric key without compromising generality. This also means that a public key infrastructure is in place where the dealer can receive digital credentials of the participants from a reputable verification authority, who knows the identity of all the participants.

2.4.2 Adversarial Model

The adversarial model presumes that an intruder will access any contact requiring the use of QR codes over public networks, even on printed media, based on the protocol model described above. It also means that the attacker can retrieve from the QR codes the secret message and can collect details on any improvements made to the QR codes of the cover.

Since the intruder may not have the cryptographic keys, it is presumed that the secret message will not be decrypted by the attacker and will not receive a legitimate portion. An intruder may, however, attempt to impersonate a member by giving the party an unlawful share.

2.4.3 Secret Sharing Protocol Using QR codes

A summary of the proposed secret sharing protocol using QR codes and symmetric keys is given in Figure 3. The goal of the proposed protocol is to encrypt and hide shares within QR codes in order to be able to disperse the shares without raising suspicion through public

channels. To obtain the shares, only approved persons who have the necessary credentials may retrieve and decrypt the content. Subsequently, before the secret can be exposed, a qualified number of community members must comply. The outline shown in Figure3 divides the overall process into three stages, namely the phase of encryption and concealment, the phase of extraction and decryption, and the phase of cooperation. In the subsections to follow, the specifics of the stages will be defined.

2.4.4 Encryption and Concealment Phase

A dealer splits a hidden message into a number of shares in this process. Each member of the community has a private key that is available only to them, and a matching public key. The dealer knows the public key of each party member, or has a means of accessing it. The individual shares are then encrypted using the respective public keys of the group members and a symmetric key, identified by the dealer and all the approved members. This results in an encrypted message for each share.

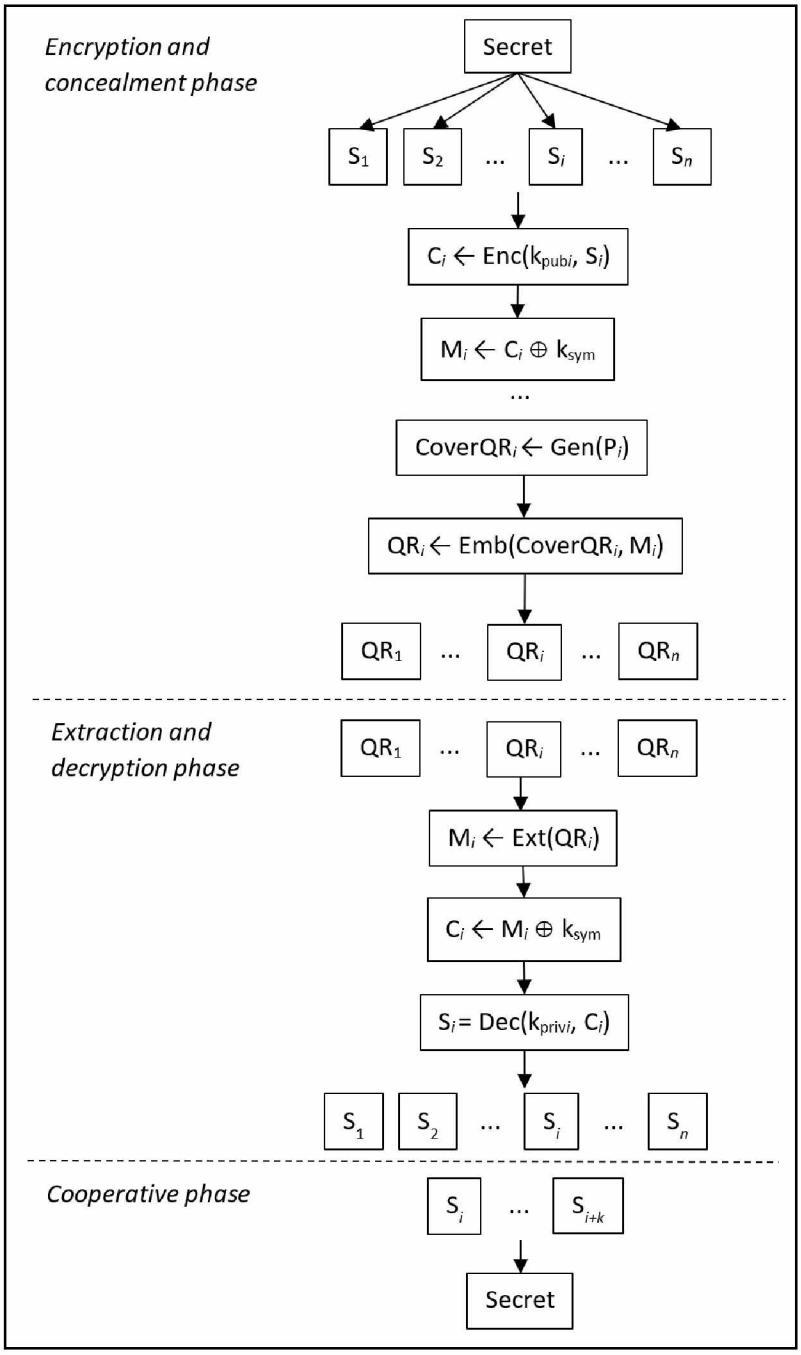


Figure 3 Overview of the proposed approach [20]

A number of covers QR codes are then created, which contain public information. The number of QR codes for the cover is equal to the number of shares. Inside a QR code, each encrypted share is then embedded. By distributing them to the participants over public media, the resulting QR codes can then be transmitted. The QR codes seem inconspicuous to a casual observer and can be searched to obtain public information. The shares can be accessed only by registered person.

This process is described formally as follows:

Definition 2

Let the hidden message be S and the individual shares be Si, where I = {1, . . , n} and n is the cumulative number of members of the party. In addition, let kpubi denote the public key of a community member and kprivi be the private key corresponding to that. Ksym is a symmetric key that the dealer and all permitted members of the party are aware of. These can be used in the algorithms that follow.

Let S be the secret message and S_i be the individual shares, where i = {1, . . . , n} and n is the total number of group members. Furthermore, let k_{pub_i} represent a group member's public key and k_{priv_i} be the corresponding private key. k_{sym} is a symmetric key that is known to the dealer and all the authorized group members. These will be used with the following algorithms.

- $C_i \leftarrow Enc(k_{pub_i}, S_i)$: A share, Si, is taken by this algorithm and encrypted using the public key of a community member, kpubi . The encrypted letter, Ci, results in this.
- $M_i \leftarrow C_i \oplus k_{sym}$: The encrypted message, Ci, with a symmetric key, ksym, is the XOR method to create the encrypted secret message, Mi, to be encoded in a QR code.
- $CoverQR_i \leftarrow Gen(P_i)$: This algorithm produces a cover QR code, CoverQRi, by taking a public message, Pi.
- $QR_i \leftarrow Emb(CoverQR_i, M_i)$: In order to generate a QR code, QRi, this method embeds the encrypted secret letter, Mi, into the cover QR code, CoverQRi.

The resulting QR codes, QRi, provide both public data and secret encrypted information. This QR codes should be spread over public networks, since the secret information can be decrypted only by people with the requisite credentials.

The cover QR code generation process, $Gen(P_i)$, can be implemented using any deterministic QR code generator that is based on the QR code standard. The aim is to

produce a valid QR code One of the inputs is an encrypted secret message for the embedding process, Emb(CoverQRi, Mi), which can be scanned and decoded by a regular QR code reader to access the public message, Pi. Mi, to be inserted into the QR code of the cover, CoverQRi. The cumulative quantity of data in Mi. The error correction ability of CoverQRi cannot be surpassed, or the resulting QR code, QRi, will not be true and will be unusable. Therefore, the option of version and error correction capability of CoverQRi would depend on the size of Mi. Due to the error correction redundancy in QR codes, the embedding process is feasible and can be carried out by deleting any of the redundant codewords in CoverQRi, as long as it preserves the credibility of the error correction system. This is analogous to how one can overlay an icon on a QR code, and it is still possible to decipher the partly blurred QR code correctly.

2.4.5 Extraction and Decryption Phase

The QR codes containing both public and secret data will be circulated to the members of the party. An approved member of the party who has all the requisite credentials, namely his/her private key and the symmetrical key, would be able to access a share of the QR code. This is accomplished by removing the secret message from the QR code first, decoding the hidden message using the symmetric key, then decrypting it using the individual private key of the community member. This will encourage the member of the community to receive his or her respective share properly.

Definition 3. *In this method, the following algorithms are used.*

- $M_i \leftarrow Ext(QR_i)$*): Provided the QR code with the secret knowledge, QRi, this algorithm extracts from it the hidden encrypted message, Mi.*
- $C_i \leftarrow M_i \oplus k_{sym}$*: Mi, with the symmetric key, ksym, the encrypted code, Ci, can be accessed by 'XOR'ing the encrypted secret message.*

- $S_i \leftarrow Dec(k_{priv_i}, C_i)$: *By decrypting the encrypted letter, Ci, using the private key, kprivi, this method obtains the share Si .*

Group member I will be able to acquire the corresponding share, Si, by extracting and decrypting the secret information from the QR code.

QR codes built on the basis of the norm of the QR code are deterministic. Therefore, if a true, but changed, QR code is scanned and decoded by a standard QR code reader, the public message can be accessed by the recipient. To replicate the original, unmodified, QR file, the public message may then be used. Thus, it is possible to remove the difference between the original and modified QR code, Ext (QRi). Although this can also be accessed by an opponent, by combining the two, the secret knowledge is calculated. This is the fundamental principle that governs the encrypted secret information which cannot be decrypted without the key.

2.4.6 Cooperative Phase

The group members will have already received their respective shares during this process. An eligible number of group members would collaborate by combining their shares together in order to disclose the hidden message that was transmitted by the distributor. K or more shares must be available in a (k, n) threshold secret sharing scheme for the retrieval of the secret message before the secret can be disclosed. The mystery will not be disclosed by any k-1, or fewer, shares. The recovery strategy is based on the system of hidden sharing that is used. For starters, if the secret sharing scheme defined in Definition 1 is used, then Equation (1) is used to retrieve the secret message.

2.5 Analysis and Discussion

2.5.1 Implementation Guidelines

Usage of a QR code embedding technique in combination with a hidden sharing scheme would include the introduction of the proposed protocol. It should be remembered that correct QR codes must be the resulting QR codes, i.e., QRi. In other words, it is sufficient for someone with a regular QR code reader to scan QRi and receive the public message. In addition, in the cover QR code, the public message should be the same as the message.

The extraction method, i.e., Mi ?? Ext(QRi), capitalizes on this fact, in that the cover QR code can be replicated by producing a QR code using the public message once the public message is received. Therefore, to retrieve the hidden details, the distinction between the cover QR code and the QRi may be used. Provided that the QR code error correction function is abused by this embedding process, this suggests that the size of the encrypted share should not be greater than the size of the error correction capability of the cover QR code. Otherwise, it would not be true for the resulting QR code and cannot be decoded. This allows the size of the QR code for the cover to be considerably greater than the size of the encrypted share.

As such, the required version of the cover QR code and degree of error correction depends on the size of the shares that need to be embedded. As follows, an appropriate version and error correction level can be calculated. In QR codes, the data is stored inside one or more lines. An instance of this was previously seen in Figure 2. As specified in the QR code standard, the number of blocks, “b”, depends on the version of the QR code and the error correction stage. Every block consists of a sequence of codewords, “c”, consisting of data codewords, “d”, and codewords for error correction, e.g. Each block also has an error correction capability, “r”, which is the maximum number of error-capable codewords in

that block. If the number of codewords within a block that are in error exceeds r, it is difficult to decipher the data in that block, and thus the entire QR code cannot be correctly decoded.

This includes embedding secret ones, Otherwise, by embedding the encrypted share, i.e. QRi ←Emb(CoverQRi, Mi), information inside a QR code, the size of the embedded letter, "m", the error inserted must be such that m < b X r. The error correction functionality of that QR code version and error correction level would be overwhelmed within CoverQRi, resulting in an invalid QRi. A strategy that can be used to mask the shares is the covert QR code solution illustrated in Chow et al. [28]

This strategy has the advantage of having a QR code on the hidden knowledge itself, which ensures that both the hidden QR code and the cover QR code provide redundancy for error correction. This minimizes the chances that the secret QR code will be undecoded due to the covert QR code being dirty or damaged. An example of how this approach can be used to embed an encrypted share is given in Figure 4. A QR code that was created with a secret message to be hidden in a cover QR code is shown in Figure 4a. As seen in Figure 4b, only the data and error correction codewords should be embedded because the feature patterns can potentially leak information. After the embedding process, Figure4d reveals the resulting QR code, i.e., the covert QR code. Both the public message and the secret information are in the covert QR language. Figure 4e indicates the discrepancy between the cover QR code and the corresponding QR code with the embedded detail. Note that both of the QR codes seen in Figure4c, d are legitimate QR codes that a regular QR reader will decipher.

In addition, they can all be decoded into the same public post. The use of a deterministic random number generator is an alternative to applying Mi ← Ci + ksym and its reverse Ci ← Mi + ksym. Since the size of Ci varies due to the length of the encrypted letter, a random number generator may create a deterministic series of random bits using the

same seed, which can be used for the XOR process. This will mean the dealer and all participants with the same random bit sequence will be able to execute the XOR operation.

A competitor, however, will not be able to access this sequence of bits and will therefore not be able to decode the results.

2.5.2 Figures of qrimages

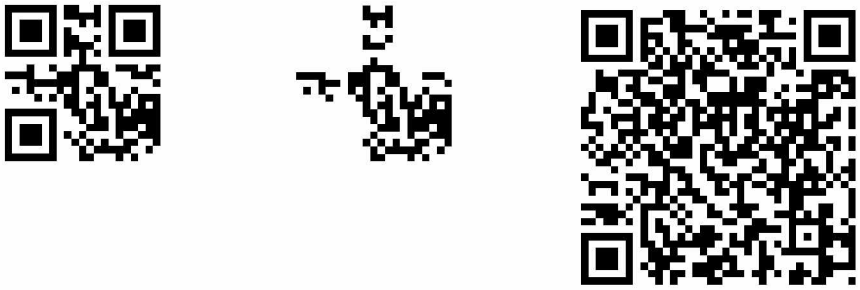


Figure 4, QRimages a, b, c

Figure 5 QRimages d, e

Figure 4 examples of embedding a secret QR code message in a QR code cover; (a) QR code containing a message to be inserted; (b) QR code modules presented in (a); (c) QR code containing a public message; (d) QR code containing a public message and an embedded hidden message; (e) Module variations between the QR codes displayed in the QR code cover; (c, d).

2.5.3 Security

In tandem with the proposed protocol, various hidden sharing systems can be used. For example, the secret sharing scheme given in Definition 1 can easily be inserted into the protocol. Since the protocol's protection depends on the procedures applied, the security of the proposed protocol is addressed as follows.

Theorem 1. *Shares in the proposed secret sharing protocol using QR codes as defined in Section 5 are secure.*

Proof of Theorem 1: The protection of the protocol shares is based on the underlying symmetric key and public key infrastructure that are used before transmission to encrypt the shares. CoverQRi, the resulting QR file, QRi, will be changed and different from CoverQRi because the shares are contained within a cover QR code. This ensures that using the public information decoded from QRi, an opponent who believes that QRi holds a secret message will decrypt CoverQRi and subsequently discover the gap between them.

Nevertheless, the only detail that will be released is the future sharing length, depending on the number of changed QR code codewords. No other information can be extracted, because the adversary cannot decrypt the encrypted information without the keys. Let C^J represents the number of modified codewords in QR_i, and the probability of success of a brute force attack is governed by $\frac{1}{8 \text{ X } [C']_r}$

The protocol mentioned in Section 5 implies that a public key infrastructure is in place where the dealer can receive from a trustworthy validation authority the digital credentials of the community members. Otherwise, an identity-based encryption technique will easily substitute the encryption of the master key instead of using a public key solution.

Theorem 2. *The key of the protocol set out in Section 5 is secure if the underlying secret communication mechanism introduced in the protocol is protected. This suggests that no knowledge about the secret would be disclosed by any k−1 or fewer shares.*

Proof of Theorem 2: In accordance with the protocol, a number of hidden sharing schemes may be implemented. The protection of the secret in the protocol is dependent on the security of the secret sharing system, so that if n shares are produced and transmitted, no information about the secret is revealed by any k-1 or less shares. This is a simple prerequisite of a hidden sharing scheme of (k, n). As such, without data on k, or more, shares, the proposed protocol is secure. In the proposed protocol mentioned in Section5, if, despite having k or more shares, the group members are unable to correctly recover the password, this means that there is at least one unauthorized and malicious individual within the group A verifiable secret sharing method such as Feldman's scheme may be used where the implementation needs cheater detection, which refers to the ability to detect whether a community member is malicious or an unauthorized party is attempting to tamper with the shares. Such a verifiable secret sharing scheme would allow the other legitimate community members to recognize the malicious individual(s) who did not have valid share (s).

A realistic concern when using the protocol is that, because shares do not disclose valuable information on their own, an authorized recipient cannot know whether the share has been correctly decrypted. As such, in fact, the share should be attached to a human-readable string and encrypted along with it. In this way, if a readable string is generated by decrypting the secret message, this would mean that the share was decrypted successfully as well.

CHAPTER 3

3.1 What are authentication factors?

There are many different ways in which more than one authentication mechanisms are used to authenticate anyone. Most authentication mechanisms usually rely on factors of information, such as a traditional password, whereas two-factor authentication methods incorporate either a possession factor or a factor of inherence

Authentication factors, listed in approximate order of adoption for computing, include the following:

1. A knowledge factor is when the user knows something, such as a password, a personal identification number (PIN) or some other sort of mutual secret.
2. A possession factor is when a user has to accept authentication requests, the user has something, such as an ID card, a protection key, a cell phone, a mobile computer or a smartphone app.
3. An inherence factor refers to anything intrinsic to the physical self of the individual is more generally considered a biometric element. This may be personal characteristics, such as fingerprints authenticated by a fingerprint scanner, are mapped to physical features. Facial and speech recognition are other widely used inherence variables. There are also the biometrics of behavior, such as keystroke dynamics, variations of gait or voice.
4. A location factor typically denoted by the location from which an authentication attempt is made, can be implemented by restricting authentication attempts to specific devices in a specific location or more commonly, by monitoring the geographical source of an authentication attempt based on the Internet Protocol (IP) source address or some other geolocation detail, such as data from the Global Positioning System (GPS),
5. A time factor limits user authentication to a fixed time frame where it is allowed to log in and limits access to the device beyond that window.

It should be remembered that the vast majority of two-factor authentication mechanisms rely on the first three authentication factors, while multifactor authentication (MFA), which may rely on two or more separate passwords for more reliable authentication, can be used by systems that demand greater security.

3.2 How does two-factor authentication work?

In this section we briefly describe the process of a typical two factor authentication system

- The user is asked by the program or by the website to log in.
- The user enters what he or she knows—usually a username and password. Then a match is made by the site's server and the user is remembered.
- The website creates a special authentication key for the user for processes that don't need passwords. The authentication function processes the key and it is checked by the site's server.
- Then the site asks the user to start the second stage of login. While a variety of ways can be taken through this step, users must show that they only have what they will have, such as an identification key, ID card, smartphone or other mobile device. This is the factor for ownership.
- During phase four, the user enters a one-time code created.
- The customer is authenticated and given access to the program or website after supplying all variables.

In technical terms, two authentication factors are required to obtain access to a device or facility at any point.

Using two variables from the same group, though, would not constitute 2FA; for instance, it is always called SFA to require a password and a mutual secret since both belong to the

same class of authentication factor: information. The user ID and password are not the most reliable as far as SFA services. One concern with password-based authentication is that generating and recalling good passwords requires awareness and diligence. Passwords need protection against many internal attacks, such as carelessly kept login credential sticky notes, old hard drives and vulnerabilities in social engineering. Passwords are often vulnerable to external threats, such as hackers using brute-force, dictionary or rainbow table attacks.

An intruder will typically break password-based protection mechanisms and steal corporate data, including personal information of users, provided ample time and money. Because of their low cost, ease of execution and familiarity, passwords have remained the most common type of SFA. Depending on how they are applied, several challenge-response questions can provide more security, and stand-alone biometric authentication approaches can also provide a more reliable SFA process.

3.3 Types of two-factor authentication products

There are several different 2FA deployment equipment and utilities — from tokens to radio frequency identification (RFID) cards to applications for smartphones It is possible to separate two-factor authentication devices into two categories: tokens that are provided to users to use while signing in and infrastructure or software that detects and authenticates entry for users who correctly use their tokens. Physical devices, such as key fobs or smart cards, may be authentication keys, or they may exist in applications like mobile or web apps that produce authentication PIN codes.

These authentication codes are normally created by a server, often known as one-time passwords (OTPs), and can be recognized by an authentication system or app as authentic. The authentication code is a short sequence connected to a specific computer, user or account that can be used once as part of an authentication process. To accept, process and authorize — or reject — access to users who authenticate with their tokens, organizations need to install a framework. This may be implemented in the form of cloud applications, a dedicated hardware server, or supplied by a third-party provider as a service.

A significant feature of 2FA is ensuring that the authenticated user is granted access to all services the user is allowed for — and only those resources. As a consequence, one of 2FA's main functions is to connect the authentication method with the authentication data of an entity. Microsoft offers some of the required infrastructure for Windows 10 2FA service organizations through Windows Hello, and will work with Microsoft accounts, as well as authenticate users with Microsoft Active Dii.

3.4 How 2FA hardware tokens work

Hardware tokens for 2FA are available that support numerous authentication approaches. The YubiKey, a small Universal Serial Bus (USB) system that supports OTPs, public key encryption and authentication, and the Universal 2nd Factor (U2F) protocol developed by the FIDO Alliance, is a common hardware token. YubiKey tokens are sold by Palo Alto, California-based Yubico Inc.

When YubiKey users log in to an OTP-supported online site, such as Gmail, GitHub, or WordPress, they insert their YubiKey into their device's USB port, enter their password, select the YubiKey field, and then tap the YubiKey icon. YubiKey produces and inputs an OTP into the field. The OTP is a 44-character, single-use password; a special ID defining the authentication key associated with the account is the first 12 characters. The remaining 32 characters contain information that is encrypted using a key only known to the computer and the servers of Yubico that was generated during the initial registration of the account. An OTP is submitted from an online service to Yubico for verification of authentication. The Yubico authentication server sends back a message verifying that this is the correct token for this user until the OTP is checked. Two authentication criteria have been given by the user: the information factor is the password, and the possession factor is the YubiKey.

3.5 Two-factor authentication for mobile device authentication

For 2FA, smartphones provide a number of possibilities, encouraging organizations to choose what suits best for them. A built-in camera can be used for face recognition or iris detection, and the microphone can be used for speech recognition. Certain applications are able to recognize fingerprints. GPS-equipped smartphones will check the location as an extra consideration. Also, Speech or Short Message Service (SMS) may be used as an out-of-band authentication channel. For receiving authentication codes by text message or automatic phone call, a trustworthy phone number may be used. To participate in 2FA, a person needs to check at least one trustworthy phone number. Both applications that support 2FA are available for Apple iOS, Google Android and Windows 10, allowing the phone itself to function as the physical interface to satisfy the ownership aspect. Duo Defense, headquartered in Ann Arbor, Mich., and acquired for \$2.35 billion by Cisco in 2018, is a 2FA software provider whose solution allows 2FA consumers to use their trusted products. Before checking that the mobile device can still be trusted to authenticate the customer, Duo's platform first determines that a user is trusted. The need to acquire an authentication code through text, voice call or email is replaced by authenticator apps. For example, users type in their username and password to access a website or web-based application that supports Google Authenticator — a knowledge factor. Users are then asked to type a number of six digits. Instead of having to wait a few seconds to answer a text message, an Authenticator produces the number for them. Every 30 seconds, these numbers alter and are different with every login. Users complete the authentication process by entering the correct number and show custody of the correct unit — an ownership element

3.6 Is two-factor authentication secure?

There are several limitations of two-factor authentication, including the cost of purchasing, issuing, and handling tokens or cards. From the point of view of the user, having more than

one two-factor authentication method allows several tokens/cards to be held that are likely to be misplaced or stolen. Although two-factor authentication improves security—because access privileges are no longer dependent solely on a password's strength, —two-factor authentication systems are just as reliable as their weakest part. Hardware tokens, for instance, depend on the security of the issuer or manufacturer. In 2011, when the technology firm RSA Security announced its SecurID authentication tokens had been stolen, one of the most high-profile examples of a compromised two-factor device occurred. If it is used to circumvent two-factor authentication, the account recovery mechanism itself can often be subverted because it sometimes resets the existing password of a user and e-mails a new password to allow the user to log in again, bypassing the 2FA process. The corporate Gmail accounts of the chief executive of Cloudflare were compromised in this way. Although 2FA is cheap, simple to implement and user-friendly based on SMS, it is vulnerable to multiple attacks. In its special publication 800-63-3, the National Institute of Standards and Technology (NIST) has discouraged the use of SMS in the 2FA services Due to cell phone number portability attacks, such as the Signaling System 7 hack, against the mobile phone network and malware, such as Eurograbber, that can be used to intercept or divert text messages, NIST concluded that OTPs sent via SMS are too vulnerable. From all the above factors the idea of 2HFA is created.

CHAPTER 4

4.1 HONEYWORDS

The fundamental principle behind the Honeywords scheme is to adjust the password storage mechanism in such a way that a password and a series of false passwords are associated with each account. The phony passwords are called honeywords. Sweetwords are the union of both honeywords and the password. As soon as the password is entered during the authentication process, the password database is immediately detected to have been compromised. Therefore, unlike traditional schemes, implementations focused on honeywords can effectively detect violations of password databases.

User ID	Username		Hashes	
1	u_1	$H(sw_{1,1})$	,…,	$H(sw_{1,n})$
2	u_2	$H(sw_{2,1})$	,…,	$H(sw_{2,1})$
…	…		…	
m	u_m	$H(sw_{m,1})$	,…,	$H(sw_{m,n})$

Table 1 Credentials database of a LS in the Honey-words system

User ID	Password Index
1	c_1
2	c_2
…	
m	c_m

Table 2 Data stored on a HC

4.2 The method of Honeyword is as follows.

During the authentication process, users select a username and a password, as with many traditional schemes. Then, the Login Server (LS) produces honeywords for the password and maintains a record in the database of passwords. The ordering of the sweetwords is randomly selected by the LS in each record.

In addition, LS sends the corresponding user ID and actual password index to Honeychecker (HC), the auxiliary server built to store the password index. Let ui and H() denote respectively the user name of user I and the hash function used in the method. H(swi,j) denotes the hash of user "I". "jth sweetword". A standard example of a table of qualifications is illustrated in Figure. HC saves the user IDs and the password index between the honeywords. During the authentication, no username or password itself is sent to HC. In comparison, HC is built as a hardened server that can only be reached by LS.

A standard structure of the HC data is seen in Figure 8. Notice that only two kinds of messages are accepted by HC: Check and Set To verify "if j=ci", check(i, j)} implies that uf "j=ci" , HC returns True, otherwise False is returned and a warning is activated.

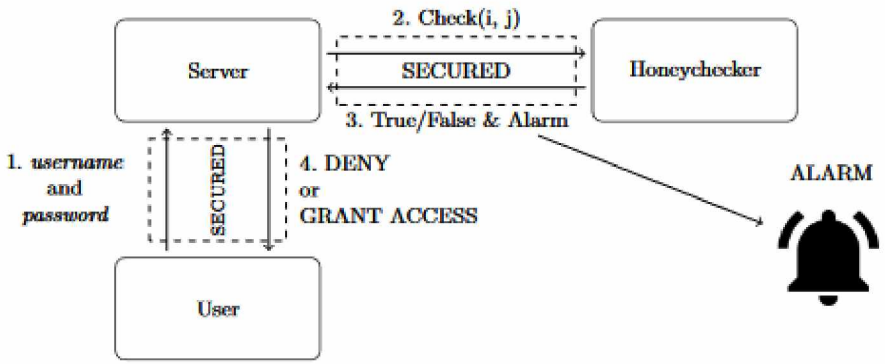


Figure 6 Login scheme of a system using honeywords [6]

The command set is structured as: Set (I j) indicates setting ci=j}. The user submits its username and password.LStries during the authentication process to locate the corresponding record for that username in the credentials database. If a record exists, LS computes the hash of the password sent and attempts to find a match in the sweetword hashes. If no match occurs, then the password sent is incorrect and access is refused.LS sends the respective user ID and the corresponding index to HC if there is a match.

First, HC seeks the record that fits the user ID and compares the index value obtained with the one stored in its database. If the outcome is valid, then access is provided. Otherwise the HC returns incorrect, generates an alert and notifies the administrators of the device policy.

Originally, the Honeywords scheme was constructed with the expectation that the opponent could steal the hashed passwords and invert the hashes to obtain the passwords. It is therefore presumed that both LS and HC will not be abused by the attacker within the same time frame. The Honeywords mechanism defends passwords from brute-force and dictionary attacks mentioned in Section 3.2. The method attempts to prevent violations of the password database and seeks to prevent only offline dictionary attacks where the adversary is believed to have taken the hashes of the password and abandoned the system.

CHAPTER 5

5.1 The proposed Two-Factor HoneyToken Authentication (2FHA) Mechanism

In this Section we introduce an alternative authentication method, for enhancing systems' security. The system combines two factor authentication with honeywords in order to make impossible for an attacker to bypass the authentication mechanism of the system. Even in the occasion that the attacker has access to the device that receives the token, e.g. by sim cloning, the proposed 2FHA method makes the authentication bypass unfeasible if not impossible.

In order to demonstrate the proposed system, we created a website that includes a login page and have developed a prototype. The user in order to enter the system must fill the correct username and password, which is the first authentication factor.

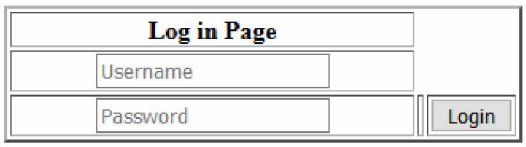


Figure 7 Login page

Then the system sends to the user a number M that indicates the token that is correct on every login attempt in the future. When logging into the system from a new device, the user must enter the correct OTP. The user receives a number of tokens N. He can choose with what platform wants to be alerted for the token, to get it (e-mail, SMS, phone call etc.).

Then we must enter the second authentication factor. The prototype of the 2FHA mechanism produces 3 qrcodes, each one of those is represented with a password and sends a SMS message to the mobile phone of the user. The SMS includes all 3 OTPs (One Time Password) passwords corresponding to each of the qrcodes. One is the correct and the others 2 are fake.

Figure 8 *Page with the qrimages and OTP fill box*

The user now has to choose what it's more suitable method for him to continue in order to fill the OTP box and proceed in the website. We have to highlight here that the number of produced tokens is kept to 3 only for demonstrating purposes but can be generalized to a number N.

If the user chooses to scan the qrcodes, the process is simple. He scans the correct qrcode and then he fills the OTP box. The qrscanner is free software and most of them are suitable for any device. If the user doesn't have qrscanner then, the option of SMS is more convenient for him. The SMS message as presented in Figure, will be sent to the user the time he logins to the system.

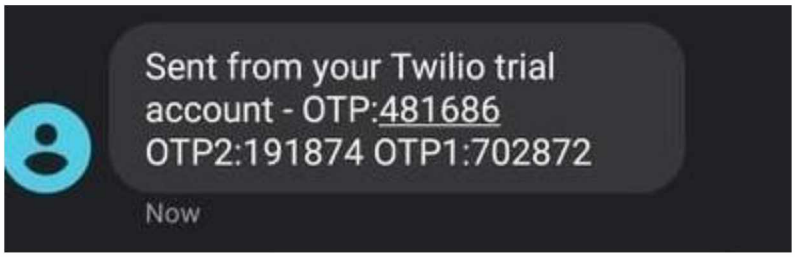


Figure 9 *Screenshot of the SMS message*

As you can see in Figure 11 the message contains 3 OTP passwords (OTP, OTP1, OTP2). These are the produced from the qr codes. Each user knows that only one of the 3 qrcodes is the correct while the other 2 are fake.

If the user fills the OTP box correctly, he will continue to the system. If not, then he will be sent back to the initial login page and has to follow the procedure again. Also, for precaution reasons the account of the user can be suspended. The OTPs must follow some rules when created; they can't be very similar among them in order to avoid misspelling mistakes.

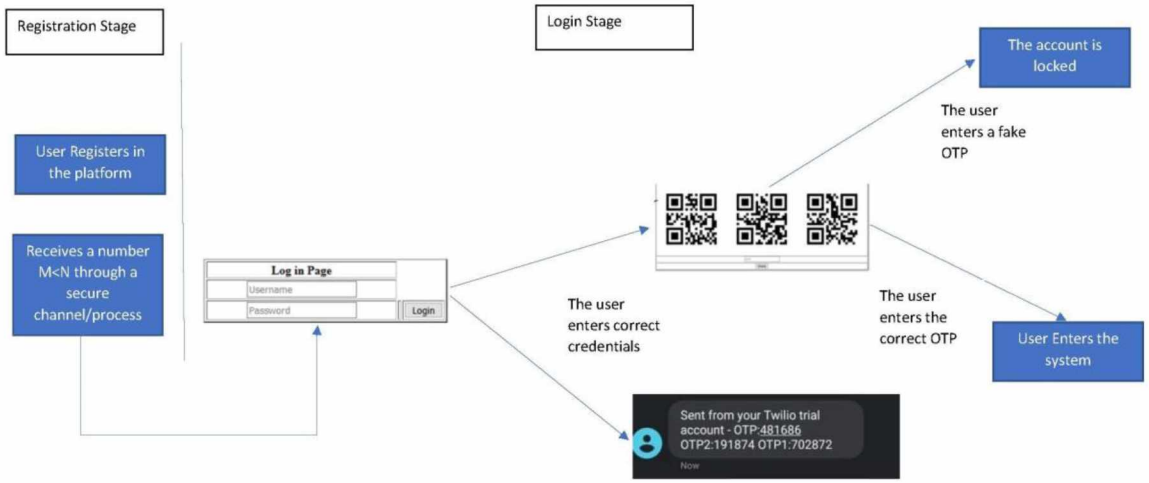


Figure 10 *Block diagram of our proposed system architecture*

5.2 HOW WE BUILD IT

For the creation of the project, we had free tools like PyCharm and free account for the first 10 euros of Twilio. Of course, we could build with other sources like PHP and HTML, instead of python and modules of it, but for more convenient we chose python.

We start the build up from the python and then we are going step by step with the HTML to finish it. We choose python cause of the big variety of libraries and modules that excels in speed with the other free source languages. Time and speed are two crucial factors for our project. We want when the user pushes the login button the sms with the 3 OTP codes reach the exact time as the user sees the 3 qrimages.

Python is a globally used programming language that is translated, high-level and general-purpose. With its prominent use of substantial white space, Python's architecture style stresses code readability.

Python has many privileges but most used for is:

1. Data Science/Scientific Computing
2. Startups
3. General web development
4. Machine learning
5. Fintech and financial industry

Some fun fact about Python is the reason behind it's name. The producer Guido Van Rossum was reading a script from "Monty Python and the flying circus" and without any others thoughts, Python name won his mind.

5.2.1 DATA SCIENCE

Computing and scientific research is used from python and is also friendly for the user of science. It has libraries which is specific for science such as:

- Astropy for astronomy
- Biopython for biology
- Grapg-tool for statistical analysis
- Pcychopy for neuroscience and experimental psychology

Parsing data is one of many roles of Python and it has big impact in knowledge. Because of the rise of data science more roles will be created in the future involving it.

5.2.2 STARTUPS

Python is easy to use and is scalable and that's the best reason tech startups want and use python so much. One big example is Dropbox. Dropbox started from Drew Houston when he kept forgetting his flash drive (USB) when he was student. By the year 2012 and November month, Dropbox was being used from 100 million people.

5.2.3 GENERAL WEB DEVELOPMENT

Simplicity is one of the biggest advantages in Python and that is said for good reason. According to the Medium article "Unlike other programming languages, Python, gives emphasis at code readability and instead of punctuation allows the user to use English words. The update of the software can be done with less extra time and effort because of the readable and clean code. Django, Flask and Pyramid are some of the many pre-built libraries and web frameworks. It's very good for back end web development projects(like ours) and can lowering the amount of time the user spends for coding.

5.2.4 MACHINE LEARNING

Machine learning needs many code and modules which python has that all in the libraries that they are specific for machine learning including scikit-learn and TensorFlow.

FINTECH AND FINANCIAL INDUSTRY

Bank of America, Venmo and many big industries are using python for their graphs and financial algorithms.

5.2.5 BENEFITS OF USING PYTHON

1. Third Party Modules Presence: There are various third-party modules in the Python Package Index (PyPI) that render Python able to communicate with most other languages and platforms.
2. Libraries with Comprehensive Support: Python includes a broad standard library, covering

fields such as internet protocols, string procedures, tools for online servers and interfaces for the operating system. Many computing functions of high use have already been scripted into the standard library, which greatly decreases the length of code to be written.

3. Creation of Open Source and Community: The Python language is developed under an open source license licensed by OSI, which allows it to be used and distributed free of charge, even for commercial purposes. In addition, its growth is driven by the culture that collaborates by hosting conferences and mailing lists for its code, and provides for its various modules.
4. Ease and help for learning Available: Python provides excellent readability and an easy-to-learn, uncluttered syntax that encourages newcomers to use this programming language. The standards for the coding type, PEP 8, include a set of rules to promote code formatting. In addition, the large base of users and engaged developers has contributed to the growth and sustained acceptance of the language by a rich internet capital bank.
5. User-friendly Systems of Data: In order to create fast runtime data structures, Python has built-in list and dictionary data structures that can be used. In addition, Python also offers the option of high-level dynamic data typing that reduces the length of support code required.
6. Productivity and Velocity: Python has a simple object-oriented architecture, offers improved process management functionality, and has good capabilities for integration and text analysis and its own unit testing platform, both of which add to its speed and usability increase. For developing complex multi-protocol network applications, Python is considered a viable alternative.

5.2.6 DISADVANTAGES OF PYTHON

1. Speed: In contrast to C/C++ or Java, Python is interpreted as a language and is slow. Unlike C or C++, since Python is a high-level language, it is no closer to hardware. As we all know, compilation and execution helps to operate naturally, but in this situation, Python execution takes place with the aid of an interpreter rather than the compiler, as we have seen the Python code is executed line by line, allowing it to slow down. For the project needed by every programmer, pace is a focal point. In the other hand, for many web apps, it can be seen that it is easy too.
2. Production for Smartphone: Python, though, is powerful on desktop and server systems, which is a wonderful language on the server side, but Python is not a really good language for mobile development, which means it is a poor language for mobile development. For mobile growth, it is very rarely used. This is the reason why, like Carbonnelle, which is designed in python, very few smartphone apps are built into it.
3. Hungry for memory: Python is not a good option for any memory-intensive tasks. That is why

for that reason it is not used. The memory use of Python is also high, owing to the data types' versatility.

4. Database Access: With limited tension and issues, Python is a versatile programming language. This vocabulary is, however, incredibly insecure and should only be used at one's own risk. There are some drawbacks on Python's access to databases. Compared to common technology such as JDBC and ODBC, the Python database access layer is considered to be somewhat underdeveloped and basic. When big companies search for a vocabulary that guarantees seamless interaction with complicated legacy data, it serves as a significant obstacle. However, the Python database access layer is not applied to organizations that require seamless interaction with complex legacy data. That is, in large companies, it is less commonly implemented.
5. Runtime Errors: One of this language's big disadvantages is that there are several problems with its architecture. There are many problems facing Python programmers about the architecture of the language. This language needs further research and it often has errors that turn up only when the language is typed dynamically at runtime.
6. Difficulty in Using Other Languages: Python lovers are so used to its functionality and vast libraries that they face difficulties understanding other programming languages or operating on them.
7. Simplicity: Python is a basic language for programming, which is perhaps the main downside. Indeed, it may be a problem. Its syntax is very basic, which makes a programmer more of a python person and may make them feel unnecessary with hard language code like Java. It concludes that it is difficult to migrate to a different language from Python with its late-binding dynamic templates and vast libraries, as the user finds it difficult to adapt to its vulnerable existence and take everything on a light note.

5.2.7 DJANJO

Django is a high-level Python web platform that allows stable and maintainable websites to be rapidly created. Django, designed by seasoned developers, takes care of much of the web creation hassle, so you can concentrate on writing the software without having to reinvent the wheel.

We choose to build our project with Django for 3 big reasons.

1. Django is very fast and is designed to help program developers complete application as soon as possible.
2. Django is very safe web framework and takes security issues seriously and is helping to bypass many common security problems.
3. Is very scalable. It has big flexibility and many big sites are trusting Django to build their sites like Instagram, Spotify, Dropbox, Mozilla, Reddit and many others.

5.2.8 PYCHARM

We use PyCharm compiler to make our py files and HTML files. PyCharm is a dedicated Python Integrated Programming Environment (IDE) offering Python developers a wide variety of important tools, closely integrated to create a convenient environment for the development of efficient Python, network, and data science.

5.2.9 Twilio

We used Twilio for the SMS application. As a service provider headquartered in San Francisco, California, Twilio is an American cloud communications network. Twilio enables software developers to use its web application APIs to programmatically make and receive phone calls, send and receive text messages, and execute other connectivity functions. Twilio has conquered communications channels as voice, chat, video and e-mail.

Twilio has a Library for python that is used for that program language specially. It's called Twilio Python Helper Library and is very easy to use and interact. The library supports Python applications that are written in Python version 2.7 and above.

5.3 PYTHON CODE

First, we make a new project and we give it a name, qrapp in our case. At python we need to install some libraries and a web framework to help us to construct our project. We use

Django in ours. At PyCharm we go to the console which is located at the bottom of the program and we write the command “Django-admin startproject qrapp”. This will create a qrapp directory.

The command creates these files:

The outer root/mysite directory for your project is a container. Django doesn't care about its name; you can rename it anything you want.
Manage.py: A command-line tool that lets you communicate in different ways with this Django project. In Django-admin and manage.py, you can read all the information on manage.py. The real Python package for your project is the inner mysite/directory. Its name is the name of the Python package that you would use to import something inside it (e.g., mysite.urls).

mysite/__init__.py: An empty file that informs Python that it should be called a Python package for this directory. Read more about the kits in the official Python Docs if you're a Python novice.
mysite/settings.py: For this Django project, settings/configuration. Django settings are going to teach you all about how the settings work.
mysite/urls.py: URL declarations for this Django project; the Django-powered site's "table of contents." In the URL Dispatcher, you can learn more about URLs.
mysite/asgi.py: An entry-point for web servers compliant with ASGI to support your project. For more details, see How to Deploy with ASGI.
mysite/wsgi.py: An entry-point to serve the project with WSGI-compatible web servers. For more details, see How to Deploy with WSGI.
To verify that our system works we insert the command “python manage.py runserver”

The next step is to make our app. In Django, each program you write consists of a Python package that follows a specific convention. Django comes with a feature that creates an app's simple directory structure automatically, meaning you can concentrate on writing code instead of building files. We write the command “python manage.py startapp qrapp”. With this we created the qrapp directory.

These files will be created:

“qrapp/”

“__init__.py”

“admin.py”

“apps.py”

“migrations/”

“__init__.py”

“models.py”

“tests.py”

“views.py”

After that we start to code.

We will write all the code and will explain the most key parts.

Views.py

The views.py file contains the code of our project and sum of the applications we will use for our cause. Here we write the code of each function individually. Here is like a store of functions code and when we need them at the other pages we call them.

```
from django.shortcuts import render
import random
import qrcode
from twilio.rest import Client
from PIL import Image
from pyzbar import pyzbar


otp = 0


def openLoginPage(request):
```

```
    return render(request, "login.html")


def validateuser(request):
    username = request.POST.get("t1")
    password = request.POST.get("t2")
    account_sid = '***************'
    auth_token = '***************'

    if username == "1" and password == "1":
        rno = random.randint(100000, 999999)
        rno1 = random.randint(100000, 999999)
        rno2 = random.randint(100000, 999999)
        global otp
        otp = rno
        im = qrcode.make(" OTP:"+str(rno))
        im.save(r"C:\Users\Vasilis\PycharmProjects\qrapp\qrapp\static\qrimages\vasilis.jpg")
        im1 = qrcode.make(" OTP1:" + str(rno1))

im1.save(r"C:\Users\Vasilis\PycharmProjects\qrapp\qrapp\static\qrimages\wrong_password
1.jpg")
        im2 = qrcode.make(" OTP2:" + str(rno2))

im2.save(r"C:\Users\Vasilis\PycharmProjects\qrapp\qrapp\static\qrimages\wrong_password
2.jpg")
        client = Client(account_sid, auth_token)

        decoded_list =
pyzbar.decode(Image.open(r"C:\Users\Vasilis\PycharmProjects\qrapp\qrapp\static\qrimages
/wrong_password1.jpg"))

        decoded_list2 = pyzbar.decode(

Image.open(r"C:\Users\Vasilis\PycharmProjects\qrapp\qrapp\static\qrimages/wrong_passw
```

```
ord2.jpg"))

    decoded_list3 =
pyzbar.decode(Image.open(r"C:\Users\Vasilis\PycharmProjects\qrapp\qrapp\static\qrimages
/vasilis.jpg"))

    message = client.messages.create(
        from_='+12014827179',
        body=(decoded_list3[0].data + decoded_list2[0].data)+decoded_list[0].data,
        to='+**********'

    )
    print(message.sid)
    return render(request,"qrcode_page.html ")

  else:

    return render(request, "login.html", {"message": "Invalid User"})

def validateOTP(request):
  user_otp = request.POST.get("otp")
  if user_otp == str(otp):
    return render(request, "welcome.html")
  else:
    return render(request, "login.html", {"message": "Wrong OTP"})
```

The openLoginPage is a function is a function which calls the login page of our project. The login.html is the page which we created.

The validateuser function is the function which we wrote our code and define our username and password. We use static HTML so we didn't need PHP and any more

database functions. Of course, we could do that but the result will be the same. Database doesn't play big role here. Username and password, we put to get the value of t1 and t2, in our case t1=t2 =1, for hastier login.

As we scroll down, we see the Account Sid and Auth_token. These are for our SMS sending procedure. At Twilio they give us their numbers which they are unique. We need them so the Twilio can recognize us. For security reasons I don't show the numbers, cause if someone have them it could be a problem.
To determinate, Twilio, which project an API request used and is coming from, is using two credentials. One is Account SID that it plays the role of the username and Auth Token which acts like a password and they both are very strong tools.
For the above reason many hackers and fraudsters will try to use tactics to get access on them. They will try to use phishing and others methods to find them and so they can get the Twilio project. A main problem is when users public code at sites (Github) and forgot to take away these parameters.
If you let them get them these parameters then the consequences can be devastated. They can impersonate you and commit many a big bill at your account with thousand of dollars in a short amount of time.

After that we have the if function and the parameters. We write that if our username and password are equal as "1" then do the rest of the if function. This is to give our system the requirements to procced the login page. This tells to our system what username and password is needed so the user to procced. It's static, that's why we put that line. The rno = random.randint (100000,999999) is a python module which generates pseudo-random numbers, 6 in our case. We use those numbers for our OTP (one time password) codes. The rno will generate a six-digit code, rno1 another and rno2 another, 3 in total. I put a six-digit code because the bigger the number the more time and bigger the chance the user do a wrong number.

5.3.1 RANDOM NUMBERS IN PYTHON

Some words about the random library in Python because it's great factor and plays the biggest role for the security. The random library produces pseudo-random numbers. Usually, this module is used for projects that they don't have big impact in security. It's not

so safe to use random module for security because it's not so random as it seems. In the scientific sense of the term, most random data generated using Python is not entirely random. Instead, it is pseudorandom: generated using a pseudorandom number generator (PRNG), which is simply any algorithm for generating data that appears to be random but can still be replicated. For better results in security, we could use the module "secrets" also.

5.3.2 RANDOM OR CLOSE TO RANDOM?

Next, it includes a prominent disclaimer. In the scientific sense of the term, most random data generated using Python is not entirely random. Instead, it is pseudorandom: generated using a pseudorandom number generator (PRNG), which is simply any algorithm for generating data that appears to be random but can still be replicated.

You guessed it, a real random number generator will produce "True" random numbers (TRNG). One example is to pick up a die repeatedly from the floor, chuck it in the air, and let it fall as it can.

You just have no idea what percentage the dice would fall on, assuming the toss is impartial. Rolling a dice is a primitive way of producing a number that is not deterministic at all by using hardware. (Or, you can make the dice-o-matic do this for you.) TRNGs are out of this article's reach, but nonetheless worth noting for the sake of contrast.

PRNGs function somewhat differently, typically performed with software rather than hardware.

A succinct summary is here:
They begin with a random number which they call it the seed and then use an algorithm to produce a series of pseudo-random bits that they have a base on it.

In Python you have probably seen random.seed(892094), random.seed(22482) and others like that. The underlying random number generator used for the Python random module is seeded by this function call. This is what makes subsequent call deterministic to create

random numbers: input A still generates parameter output B. When it is used maliciously, this gift may even be a problem. Perhaps the words "random" and "deterministic" appear to be unable to work alongside each other.

5.3.3 IS IT SECURE AND CRYPTOGRAPHY STRONG?

For the "RNG" acronyms, if you haven't had plenty, let's add one more into the mix: a CSPRNG or cryptographically stable PRNG. For creating confidential data such as passwords, authenticators, and tokens, CSPRNGs are suitable. Provided a random string, in a series of random strings, there is no practical way for Malicious Joe to decide what string came before or after that string.

Entropy is one more word that you can see. This applies, in a nutshell, to the amount of randomness added or needed. For example, one Python module you're going to cover here specifies DEFAULT ENTROPY = 32, which is the default number of bytes to return. This is known by the developers as "enough" bytes to be a decent amount of noise.

CSPRNGs has a key point that they are still pseudo-random. In a way that is internally deterministic, they are engineered, but they incorporate some other variable or have some property that makes them "random enough" to prevent backing into whatever function enforces determinism.

5.3.4 WHAT AREAS RANDOM COVERS

Practically this means that for mathematical modelling, simulation and to make random data reproducible, you can use simple PRNGs. As you can see later on, they're actually considerably quicker than CSPRNGs. For safety and cryptographic applications where data sensitivity is imperative, use CSPRNGs.

In addition to building on the use cases above, you can dig into Python tools for using both PRNGs and CSPRNGs in this tutorial:

- The PRNG solutions include the Python standard library random module and its array-based NumPy equivalent, numpy.random.

- The os, secrets, and uuid modules in Python provide functionality to create cryptographically protected objects.

5.3.5 QRCODE.MAKE AND QRIMAGES

Now we need to put these numbers - codes in qrimages so eventually will be qrcodes. This will happen with qrcode.make module.
Qrcode.make is a python module which generates qrimages and we put the data in there. For our project (" OTP:"+str(rno)) is the data we say to insert at our qrimage, we name it OTP and we say that is from the rno module. The same happens for the im1 and im2.
We need to save our image somewhere, at our static directory in our case and specific at static/qrimages directory. With the code line im.save we give the command to do the above action. In brackets we fill the location where we want to save the image. We do the same action for im1 and im2.

Qrcode.make(data) returns a PIL image that contains a qr code with the data we gave.

We have 3 decode_lists. The decoded lists are decoding the image we created above into a number One decoded list has the right number and the other two have the wrong. How we separate the wrong for the right number?? With the code after each decoded_list (0) line. The “im1 = qrcode.make(" OTP1:" + str(rno1))” converts our number into a qrimage. Above we had the line “otp = rno” which defines to our system which otp will take as right, in our system is Vasilis.jpg. The other images will be saved as wrong_password1 and wrong_password2.

The pyzbar.decode module is to read one-dimensional barcodes and QR codes from Python 2 and 3 using the zbar library. The basic idea here is to make a qrimage, save it somewhere in our system and then decode the qrimage.

The message paragraph-code is from Twilio (they give for each program language the code individually). The “from” line is the number which is unique and comes with the choice of the user. When you register at Twilio the user chooses one umber which represents a phone number. At body we insert the data we want; in our case we want the sum of the

decoded_list [0] +decoded_list2[0] + decoded_list3[0]. We want our SMS message come with all three numbers and at our website shows three qrimages.

The validateOTP is the validate function that tells the system when otp is right to proceed to the other page which is the “welcome.html. In other case will go to the “login.html” with a message “wrong OTP.

urls.py

At urls.py we name all urls and functions. We need to write at urlpatterns what functions we use and how we named them.

```
"""qrapp URL Configuration
"""
from django.contrib import admin
from django.urls import path
from qrapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.openLoginPage),
    path('validate', views.validateuser, name="validate_login"),
    path('validate_otp', views.validateOTP, name="validate_otp"),
    #path('sms', views.sendsms, name="SMS"),
]
```

settings.py

At settings.py we have all options about the system like database path, installed apps and a lot more. Most of them is default and we don't change something except we put at installed apps ours, "qrapp".

```
"""
Django settings for qrapp project.
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '6*(88h9!64rd#1f*o0-q!=f#k@@i(hp(53+rvopj@-5b$-tz_^'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
    'qrapp',
    'rest_framework',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'qrapp.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [r"C:\Users\Vasilis\PycharmProjects\qrapp\templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'qrapp.wsgi.application'
```

```
# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
```

5.4 HTML files

5.4.1 WHAT IS HTML??

HTML is a markup language that describes the content structure. HTML consists of a collection of items that you use to enclose, or wrap, various portions of the material in order to make it look or behave in a certain manner.The enclosing tags can create a hyperlink to something else with a word or image, can italicize words, can make the font bigger or smaller, etc.

5.4.1.1 HTML ELEMENT AND THE ANATOMY

Our major component pieces are as follows:

1. Opening tag: This indcludes of the element name (in this case, p), wrapped in the brackets of the opening and closing angles. This states where the feature starts or begins to take place, where the paragraph begins, in this case.
2. Closing tag: That's the same as the opening tag, except that before the name of the feature, it requires a forward slash. This states where the aspect stops, where the paragraph ends, in this case. One of the typical beginner errors is that failure to add a closing tag will lead to odd effects.
3. Content: Material: This is the content of an element, which is just text in this case.
4. Element: The element is composed of the opening tag, the closing tag and the material.

Attributes provide additional details that you do not want to show in the actual content of the element. Here, the attribute name is “class”, and the attribute value is “editor-note”. The class attribute helps you to assign the element a non-unique identifier that can be used to target it with style details and other items (and all other elements with the same class value).

An attribute is often expected to get the foregoing:

1. A space in between it and the name of the element (or the previous attribute, if the element already has one or more attributes).
2. The name of the attribute is accompanied by an equivalent symbol.
3. Worth of the attribute wrapped by opening and closing quote points.

5.4.1.2 HTML DOCUMENT

The following is the anatomy of the HTML document:

- `<!DOCTYPE html>`: It is a preamble which is necessary. Doctypes were intended to serve as links to a series of guidelines that the HTML page had to obey in the mists of time when HTML was young (around 1991/92), to be called decent HTML, which may mean automated error checking and other useful stuff. They don't do much these days, though, and are essentially only necessary to ensure that the paper behaves

properly.

- `<html></html>`: This feature wraps the whole page with all the material and is often referred to as the root element.
- `<head></head>`: This feature serves as a container for all the items that you want to put on the HTML page that are not the material that you present to the visitors on your page. This includes items you want to appear in search results, such as keywords and a page overview, CSS to format our text, character set declarations, and more.
- `<meta charset="utf-8">`: This element adjusts your document's character set to UTF-8, which comprises most of the letters in the large number of written languages. Essentially, whatever textual material you might place on it can now be treated. There is no excuse not to set this up, because later on, it will help prevent any issues.
- `<title></title>`: It determines the page title, which is the term that shows in the browser tab where the page is loaded. When you bookmark/favorite it, it is often used to define the page.
- `<body></body>`: This includes all the material you want to present to web visitors as they visit your website, whether it's text, photos, videos, sports, audio tracks that can be played, or something else.

Login.html

This is the first page at our project which tell the user to put the username and password. At form `<form action="{% url 'validate_login' %}" method="post">` we give the necessary action in our page to link it with the function validate_login, we had written at urls.py. The line with crsf_token is for cross site request forgery protection. Cross-site request forgery (often identified as CSRF) is a loophole to network protection that causes users to be coerced by an intruder to perform acts they do not wish to do. It enables an attacker to partially exploit the very same principle of origin, which also is intended to avoid intrusion between separate websites. After we create a table with two lines we fill. The align is for the decoration and where the table will be at our page. We chose to put it in the center. Border =”2” is for the how dark will be the lines in our table. The number 2 is a medium

black color. The “tr” tag is to define one row at HTML pages. It has >tr> for start and </tr> to end it. The <th width=”250”Log in Page> is to determinate the width of our headline in the table. The headline will get what we write after the “>” symbol. The “th” tag defines the header in a table or cell. In <input type="text" placeholder="Username" name="t1" required> we give at our table one row which the name “username” is shown at the background. This is one cell that user must fill for the element username he has. We name it as “t1” so we can determinate it and put the variable we want at views.py, like username = request.POST.get("t1"). To continue more at our code in our HTML page we do the same for the password cell. We make one more cell, two in total one for username and one for password, so the user has to fill it. With that we complete the table we want for the user has to fill so to procced further. Now we want one button so the variables user writes at them to send it in our system or database. That will happen with the command line <button type="submit" >Login</button>. This line constructs a button element that has the “Login” name upon it and has the action of submit, which means when the user press it the data of what he wrote at the boxes of username and password will be send at our system, so to configure if they are the correct to let him proceed. The system has from the function validateuser at views.py the required username and password and will check them when the data will be sent. This is all only for filling the cells. Now we must show something the user to see when he got something not correct, a message for example. The “% if message %” will make that happen. This line tells the system what will do when the user wrote something wrong at above cells. If he did that then, with the command line <h1 style="color:red">{{message}}</h1> we will warn him. The “h1 style= “color:red” is to determinate the color of our message. The {{message}} is to show what message will the user see. The message is connected with our validateuser at views.py and specially return render(request, "login.html", {"message": "Invalid User"}). This explains that at “login.html” when the he fills something wrong at username or password cells the message “invalid user” will be shown to him. With the </html> we finish our login page.

```
<html>
  <body>

<form action="{% url 'validate_login' %}" method="post">
```

```
    {% csrf_token %}
    <table align="center" border="2">
        <tr>
            <th width="250">Log in Page</th>
        </tr>
        <tr>
            <th>
            <input type="text"  placeholder="Username" name="t1" required>
            </th>

        </tr>
        <th>

            <input type="password"  placeholder="Password" name="t2" required>

        </th>
        <th>

        </th>

         <th>



            <button type="submit" >Login</button>
        </th>


    </table>

</form>

   {% if message %}
```

```
    <h1 style="color:red">{{message}}</h1>
    {% endif %}
    </body>
</html>
```

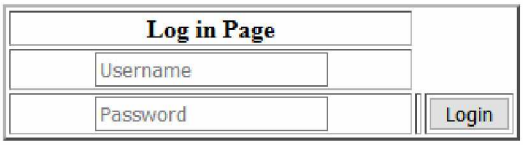


Figure 11 This is how it looks at user

Qrcode_page.html

This is the page if the user successfully writes the username and password to login page. Is the page which shows our 3 qrimages and tell the user to put the right otp to proceed. Also, instantly when this page shows up, the user will get the SMS containing the 3 OTP passwords. Let's explain it line by line. At the start in our page, we see {% load static %}. This is to load all elements at our static directory, in our case the images. After that, we have the form action. Here we give a command to our system to link it with the validate_otp of the views.py. The method = post, means how to send these elements in our page. The line with the crsf_token is for cross site request forgery. The reason, we explained it above at "login.html.". After this we must create a table and insert to it, our qrimages. First, we create the table with the "table align" line. At "center" we tell the system to put the table at the center of our page and the "border=2" to give order, how much deep or bright the lines will be at the table. The <tr> tag we explained it above so we will not explain

it again. Now we reached the <img src="{% static 'qrimages\vasilis.jpg' %}"> line. The "img" tag describe that we insert an image. The "src" tag means the path we insert of the image is located. We must put all three images so we do the same procedure three times in total, one for Vasilis.jpg (which is the image that contains the right OTP code), wrong_password1 and wrong_password2. The order of the images is shown to the user from left to right will be Vasilis.jpg.wrong_password1 and wrong_password2. Now we must create a cell to fill the OTP code. With the line <input type="password" placeholder="OTP" required name="otp"> we create a cell with the name OTP at the background. The type=password we say the system what element the user will write so it can do the authentication process and defines a password field. The otp code here will be checked at the validate_otp function at views.py. The lines if user_otp == str(otp): and return render(request, "welcome.html") tell the system that if the number, the user wrote at OTP cell is the same as we produce, then let the user procced to the "welcome.html" page. Now we need a button so we can send the data. That will happen with the <button type="submit">Check</button>. Here we create a button element which has the name "Check" upon it.

```
{% load static %}
<html>
  <body>
  <form action="{% url 'validate_otp' %}" method="post">
    {%csrf_token %}

  <table align="center" border="2">
      <tr>
        <th>
            <img src="{% static 'qrimages\vasilis.jpg' %}">
             <img src="{% static 'qrimages\wrong_password1.jpg' %}" >
             <img src="{% static 'qrimages\wrong_password2.jpg' %}">

          </th>
        </tr>
      <tr>
```

```
        <th>
        <input type="password"  placeholder="OTP"   required name="otp">
        </th>
        </tr>
        <tr>
          <th>
            <button type="submit">Check</button>
          </th>

        </tr>
      </table>
      </form>
    </body>
</html>
```

Figure 12 How qrcode_page.html looks like to the user

Welcome.html

Is the final page of our system and it shows up when the user put correctly the OTP number. Is a message telling the user “welcome” . Here we write only the command line `<h1 style="color: blue">Welcome User</h1>` which create the words “Welcome User” with font color “blue”.

```
<!DOCTYPE html>
<html lang="en">
<h1 style="color: blue">Welcome User</h1>

<body>

</body>
</html>
```


Figure 13 Welcome user that tell the user he fill the OTP box successfully

Figure 14 Wrong OTP message when user fills the OTP box false

5.5 STATIC FILES

At static files we have the file directory named qrimages. There the 3 our images are save which they produced by the “im.image” function. As we can see we have 3 images, 1 image named vasilis which is the correct qrimage and has the correct password. The other 2 have the wrong password and we have name them wrong_password1 and wrong_password2.

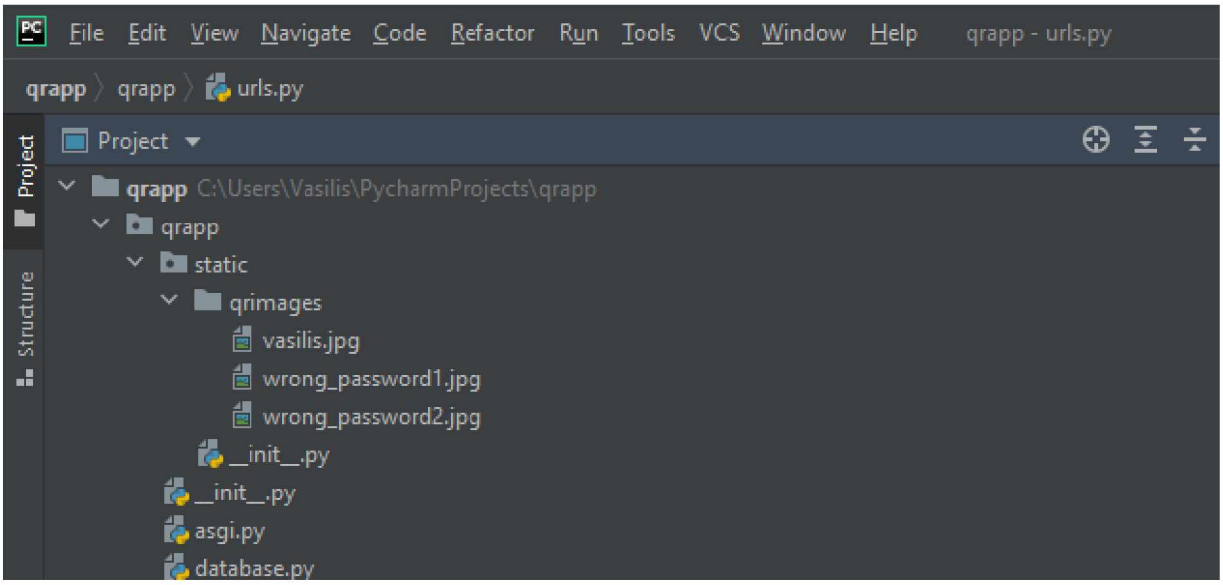


Figure 15 Static files at PyCharm

5.6 QRImages

Here are the qrimages which are produced from the code qrcode.make.

Figure 16 Vasilis.jpg

Figure 17 wrong_password 1

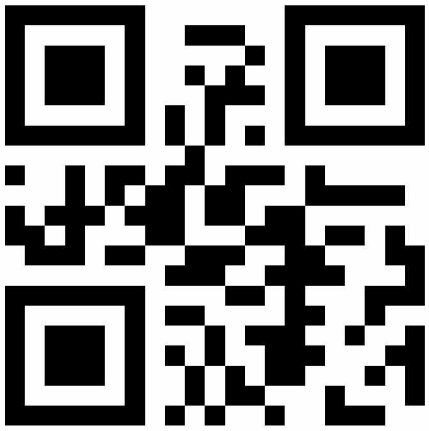


Figure 18 wrong_password 2

CHAPTER 6

Conclusion – Discussion

In this thesis we have taken actions to strengthen the security of a system against stolen tokens and penetration attempts. The proposed mechanism combines 2FA and Honeyword principles and can be integrated in any existing platform or web application. We plan to improve the system in the future by producing a higher number of qrcodes and passwords that will increase the security. In the prototype of the proposed system OTP's are sent them through SMS. In the near future we plan to integrate the proposed 2FHA with google and Microsoft authenticators. We also plan to enhance the registration phase in order to make it more secure by encrypting the initial information. We could also set a time limit to wait the user to fill the code.

REFERENCES

[1] S. A. Zhang, S. Pearman, L. Bauer, and N. Christin, “Why people (don’t)use password managers effectively,” inFifteenth Symposium on UsablePrivacy and Security ({SOUPS}2019), 2019.

[2] F. Aloul, S. Zahidi, and W. El-Hajj, “Two factor authentication usingmobile phones,” in2009 IEEE/ACS International Conference on Com-puter Systems and Applications. IEEE, 2009, pp. 641–644.

[3] J. Yu, G. Wang, Y. Mu, and W. Gao, “An efficient generic frameworkfor three-factor authentication with provably secure instantiation,”IEEEtransactions on information forensics and security, vol. 9, no. 12, pp.2302–2313, 2014.

[4] N. Anwar, I. Riadi, and A. Luthfi, “Forensic sim card cloning usingauthentication algorithm,”International Journal of Electronics and In-formation Engineering, vol. 4, no. 2, pp. 71–81, 2016.

[5] A. Dmitrienko, C. Liebchen, C. Rossow, and A.-R. Sadeghi, “Onthe (in) security of mobile two-factor authentication,” inInternationalConference on Financial Cryptography and Data Security.Springer,2014, pp. 365–383.

[6] Z. A. Genc, S. Kardaş, and M. S. Kiraz, “Examination of a newdefense mechanism: Honeywords,” inIFIP International Conference onInformation Security Theory and Practice.Springer, 2017, pp. 130–139.

[7] D. Wang and P. Wang, “Two birds with one stone: Two-factor authenti-cation with security beyond conventional bound,”IEEE transactions ondependable and secure computing, vol. 15, no. 4, pp. 708–722, 2016.

[8] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, “Au-thentication protocols for internet of things: a comprehensive survey,”Security and Communication Networks, vol. 2017, 2017.

[9] M. A. Ferrag, L. Maglaras, A. Derhab, and H. Janicke, “Authenticationschemes for smart mobile devices: Threat models, countermeasures, andopen research issues,”Telecommunication Systems, vol. 73, no. 2, pp.317–348, 2020.

[10] T. Limbasiya, M. Soni, and S. K. Mishra, “Advanced formal authenti-cation protocol using smart cards for network applicants,”Computers &Electrical Engineering, vol. 66, pp. 50–63, 2018.

[11] J. Reynolds, N. Samarin, J. Barnes, T. Judd, J. Mason, M. Bailey, andS. Egelman, “Empirical measurement of systemic 2fa usability,” in29th{USENIX}Security Symposium ({USENIX}Security 20), 2020, pp.127–143.

[12] P. A. Grassi, M. E. Garcia, and J. L. Fenton, “Draft nist specialpublication 800-63-3 digital identity guidelines,”National Institute ofStandards and Technology, Los Altos, CA, 2017.

[13] Z. A. Genc, S. Kardas¸, and M. S. Kiraz, “Examination of a newdefense mechanism: Honeywords,” inIFIP International Conference onInformation Security Theory and Practice.Springer, 2017, pp. 130–139.

[14] A. S. Narayanan, “Qr codes and security solutions,”InternationalJournal of Computer Science and Telecommunications, vol. 3, no. 7,pp. 69–72, 2012.

[15] TWILIO. https://www.twilio.com/docs/sms/send-messages. [Online].Available: https://www.twilio.com/docs/sms/send-messages

[16] A. M. R Varadarajan, “Using qr codes for authenticating users to atmsand other secure machines for cardless transactions,” 2014.

[17] A. Dmitrienko, C. Liebchen, C. Rossow, and A.-R. Sadeghi, “Securityanalysis of mobile two-factor authentication schemes.”Intel TechnologyJournal, vol. 18, no. 4, 2014.[18] G. Ali, M. Ally Dida, and A. Elikana Sam, “Two-factor authenticationscheme for mobile money: A review of threat models and countermea-sures,”Future Internet, vol. 12, no. 10, p. 160, 2020

[18] ADEKA, Muhammad; SHEPHERD, Simon; ABD-ALHAMEED, Raed. Resolving the password security purgatory in the contexts of technology, security and human factors. In: *2013 International Conference on Computer Applications Technology (ICCAT)*. IEEE, 2013. p. 1-7.

[19] Rani, M. Mary Shanthi, and K. Rosemary Euphrasia. "Data security through qr code encryption and steganography." *Advanced Computing: An International Journal (ACIJ)* 7.1/2 (2016): 1-7.

[20] CHOW, Yang-Wai, et al. Cooperative secret sharing using QR codes and symmetric keys. Symmetry, 2018, 10.4: 95.

[21] Falkner, S.; Kieseberg, P.; Simos, D.; Traxler, C.; Weippl, E. E-voting Authentication with QR-codes. In Human Aspects of Information Security, Privacy, and Trust; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8533, pp. 149–159.

[22] Weir, J.; Yan, W. Authenticating Visual Cryptography Shares Using 2D Barcodes. In Lecture Notes in Computer Science, Proceedings of the 10th IWDW 2011 International

Workshop, Atlantic, NY, USA, 23–26 October 2011; Springer: Berlin, Germany, 2011; Volume 7128, pp. 196–210.

[23] Wu, W.C.; Lin, Z.W.; Wong, W.T. Application of QR-Code Steganography Using Data Embedding Technique. In Information Technology Convergence; Springer: Berlin, Germany, 2013; pp. 597–605.

[24] Chen, W.Y.; Wang, J.W. Nested image steganography scheme using QR-barcode technique. Opt. Eng. 2009, 48, 057004

[25] Chung, C.H.; Chen, W.Y.; Tu, C.M. Image hidden technique using QR-barcode. In Proceedings of the Fifth IEEE IIH-MSP'09 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kyoto, Japan, 12–14 September 2009; pp. 522–525.

[26] Lin, P.Y.; Chen, Y.H.; Lu, E.J.L.; Chen, P.J. Secret Hiding Mechanism Using QR Barcode. In Proceedings of the 2013 IEEE International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Kyoto, Japan, 2–5 December 2013; pp. 22–25

[27] Bui, T.V.; Vu, N.K.; Nguyen, T.T.; Echizen, I.; Nguyen, T.D. Robust Message Hiding for QR Code. In Proceedings of the 2014 IEEE Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Kitakyushu, Japan, 27–29 August 2014; pp. 520–523.

[28] Chow, Y.; Susilo, W.; Baek, J. Covert QR Codes: How to Hide in the Crowd. In Lecture Notes in Computer Science, Proceedings of the 13th International Conference Information Security Practice and Experience (ISPEC), Melbourne, Australia, 13–15 December 2017; Liu, J.K., Samarati, P., Eds.; Springer: Berlin, Germany, 2017; Volume 10701, pp. 678–693

[29] Wan, S.; Lu, Y.; Yan, X.; Wang, Y.; Chang, C. Visual secret sharing scheme for (k, n) threshold based on QR code with multiple decryptions. J. Real Time Image Process. 2018, 14, 25–40.

[30] Cheng, Y.; Fu, Z.; Yu, B.; Shen, G. A new two-level QR code with visual cryptography scheme. In Multimedia Tools and Applications; Kluwer Academic Publishers: Boston, MA, USA, 2017.

[31] Lin, P. Distributed Secret Sharing Approach With Cheater Prevention Based on QR Code. IEEE Trans. Ind. Inform. 2016, 12, 384–392.

[32] https://realpython.com/python-random/ : Last checked 14/2/2021

[33]https://www.invensis.net/blog/benefits-of-python-over-other-programming-languages/: Last checked 14/2/2021

[34] https://www.geeksforgeeks.org/disadvantages-of-python/ : Last checked 14/2/2021

[35] https://en.wikipedia.org/wiki/HTML: Last checked 14/2/2021

[36] Vassilis Papaspirou, Leandros Maglaras, Mohamed Amine Ferrag, Ioanna Kantzavelou, Helge Janicke, Christos Douligeris (2020). A novel Two-Factor HoneyToken Authentication.