PRIVocular: Enhancing User Privacy Through Air-Gapped Communication Channels

Anastasios N. Bikos *

Posted Date: 25 February 2025

doi: 10.20944/preprints202502.1952.v1

Keywords: Metaverse security; OCR; privacy; privacy-preservation; QR; Secure visual key exchange; virtual proof; virtual-reality cybersecurity; visual cryptography

 doi:10.20944/preprints202502.1952.v1

Article

PRIVocular: Enhancing User Privacy Through Air-Gapped Communication Channels

Anastasios N. Bikos

Department of Computer Engineering and Informatics, University of Patras, 26504 Patras, Greece; mpikos@ceid.upatras.gr; Tel.: +30-6978130329 (G.R.)

Abstract: Virtual Reality (VR)/Metaverse is transforming into a ubiquitous technology by leveraging smart devices to provide highly immersive experiences at an affordable price. Cryptographically securing such augmented reality schemes is of paramount importance. Securely transferring the same secret key, i.e., *obfuscated*, between several parties is the main issue with symmetric cryptography, the workhorse of modern cryptography because of its ease of use and quick speed. Typically, asymmetric cryptography establishes a shared secret between parties, after which the switch to symmetric encryption can be made. However, several SoTA (State-of-The-Art) security research schemes lack flexibility and scalability for industrial Internet of Things (IoT)-sized applications. In this paper, we present the full architecture of the PRIVocular framework. PRIVocular (i.e., PRIV(acy)-ocular) is a VR-ready hardware-software integrated system that is capable of visually transmitting user data over three versatile modes of encapsulation, encrypted –without loss of generality– using an asymmetric-key cryptosystem. These operation modes can be Optical Characters-based or QR-tag-based. Encryption and decryption primarily depend on each mode's success ratio of correct encoding-decoding. We investigate the most efficient means of ocular (encrypted) data transfer by considering several designs and contributing to each framework component. Our pre-prototyped framework can provide such privacy preservation (namely *virtual proof of privacy (VPP)*) and visually secure data transfer promptly (<1000 msec), as well as the physical distance of the smart glasses (∼50 cm).

Keywords: Metaverse security; OCR; privacy; privacy-preservation; QR; Secure visual key exchange; virtual proof; virtual-reality cybersecurity; visual cryptography

1. Introduction

Virtual Reality (VR)/Extended reality (XR) technology has significantly advanced over the last few years. It is indicated that 2016 was the year when VR went from virtual to reality [1]. VR encompasses a collection of technologies (3D displays, input devices, UHD cameras, wireless network protocols, software frameworks, etc.) that aim to create an interactive medium that offers human users the feeling of being immersed. Hence, the evolution of consumer-grade hardware (such as Oculus Rift and HTC Vive), as well as the application development flexibility and portability of software platforms, i.e., Android, to create and display VR content, strongly suggest that this field could be the next big success wave of computer technology [1]. Interestingly, consumer prices for such VR devices have also been dropping steadily, creating a huge potential for more mass availability to the public [2].

Such immersive devices can be smart glasses, tablets, or smartphones. As part of an integrated VR framework, these devices can utilize the visible spectrum, and together with their UHD cameras, they can easily capture and further process visual data in various formats. Data transmission on the visible spectrum can be tedious due to various aspects affecting optical performance, i.e., room lighting conditions, display reflections, and contrast. Combined with the specific format (optical characters, images, code tags, etc.) of the visually displayed data, there is a trade-off between the performance of correct (error-free) optical transmission and the amount of data to be transmitted optically.

 doi:10.20944/preprints202502.1952.v1

The use of the most optimal means of visual transfer of ocular data has several areas of applicability. Provided we receive a relatively large amount of data. At the same time, via the photon carrier, or visible domain, between a computer display and a UHD camera of a VR device, we can then process user information into several applications or vertical industries (e.g., finance, medical, and military activities). We can manipulate an image code tag or optical characters from a display screen to retrieve industrial codes for product tags via camera sensors or to tag patient codes in hospitals for medical history queries. However, the security of the data transmitted through optical means can be compromised. Hence, this data could be considered confidential or user-private in most applicable cases [31,32].

Therefore, we need a method of securing privacy-sensitive data sent to the optical carrier by selecting a strong encryption scheme. The Paillier cryptosystem, proposed by Pascal Paillier in 1999, is a probabilistic asymmetric algorithm for public key cryptography [3]. We could provide VR systems with such confidentiality by Utilizing this scheme with strong encryption/decryption keys (ideally as much as 1024 bits). Encryption for optical data on visual transmission should not adhere to the limited resources on the VR device.

Therefore, we need a method of securing sensitive data sent to the user via the visible spectrum. To this end, this paper describes the architecture, component functionality, and design analysis of the PRIVocular open-source framework[1]. The PRIVocular framework is a Private VR hardware-software integration system that aims to encode data from several different defined optical representation methods visually, encrypt the encoded data, and then visually capture (transmit/receive on the visible spectrum) the image ciphertext. Finally, the framework should successfully retrieve the initial data via the reverse cycle of integrity-correct and confidentiality-preserving decoding and corresponding decryption methods. PRIVocular possesses a client-server architecture approach and is based on the Android framework. Encryption/decryption is performed only on VR devices with appropriate security priority constraints and manageable performance, with the Paillier cryptosystem utilized. The primary goal of PRIVocular is threefold: (1) *performance* (i.e., end-to-end transceive the maximum possible amount of optical data), (2) *integrity* (i.e., reconstruct original data, 100% accurate), and (3) *confidentiality* (i.e., privacy preservation of user data). Concerning the previously mentioned cryptosystem, successful decryption depends on the key size the end-to-end user defines for his/her data to be transmitted and retrieved correctly. **The most novel design motivation behind PRIVocular is based on user client-detection, meaning only the user who possesses the correct key can ideally retrieve the visually encoded original data.**

This paperwork attempt marks the following highlighted innovative research contributions:

1. We prototype PRIVocular, an open-source framework *(that works for any type of asymmetric/symmetric key encryption scheme)* that aims to operate as a **virtual proof of privacy**, for establishing strong cybersecurity constraints into industrial-level (IoT) vertical applications, that demand extremely low latency requirements.
2. We integrate PRIVocular inside a Metaverse-applicable immersive reality platform architecture.
3. We *a priori* design, implement, & incorporate MoMA tag inside our framework. The MoMA tag contains 61% more capacity than the QR tag (version 40).

The rest of the paper is organized as follows: Section II briefly discusses some important preliminary topics and the most relevant work in the literature. Section III describes the implementation of the proposed system. Section IV presents experiments and testing for the application, whereas Section V concludes the paper along with future contributions to the paperwork.

[1] While the framework is built on the Paillier cryptosystem due to earlier work of the authors in [11], the framework can be used with any underlying cryptographic scheme.

 doi:10.20944/preprints202502.1952.v1

2. Background

This section will discuss basic theories and technologies about optical data encoding formats, namely Optical Character Recognition (OCR), the Quick-Response (QR) code tag, and the Paillier encryption scheme. Finally, some relevant work on VR frameworks and applications will be briefly presented.

2.1. Tesseract OCR Engine

Optical character recognition (also optical character reader, OCR) is the electronic conversion of images of typed, handwritten, or printed text into machine-encoded text, whether from a scanned document, a photo of a document, or a digital image photo. There are currently many engines and techniques to perform OCR. One of the most prominent is Tesseract.

Tesseract is an open-source OCR engine developed at HP as a Research Prototype between 1984 and 1994. Tesseract architecture assumes its input is a binary image with defined optional polygonal text regions. Processing, then, follows a traditional step-by-step pipeline. The first step, which is the most computationally expensive, is a connected component analysis in which outlines of the components are stored. At this stage, several outlines are gathered together, purely by nesting into so-called Blobs. Blobs are then organized into text lines, and the lines and regions are post-analyzed for fixed pitch or proportional text. Text lines are immediately broken into words differently according to the kind of character spacing. Fixed pitch text is separated by character

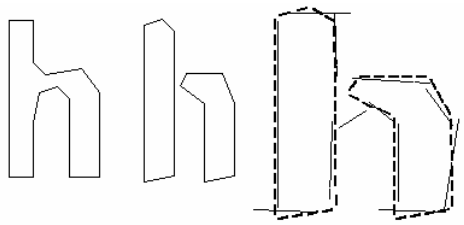

Figure 1. (a) Pristine 'h', (b) broken 'h', (c) features matched to classifier prototypes [7].

cells. Finally, the proportional text is broken into words using definite and fuzzy spaces [7].

The Recognition phase then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word adjacently. Each well-recognized word is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to recognize text throughout the page segment more accurately. Since the adaptive classifier may have learned better next time, words not recognized well enough in the previous stage are recognized again. A final phase resolves fuzzy spaces and re-captures x-height coordinates to locate small-cap text [7].

Recognition of Latin script and typewritten text is still not 100% accurate, even with clear imaging. One study on recognition of 19th- and early 20th-century newspaper pages concluded that character-by-character OCR accuracy for commercial OCR software varied from 81% to 99% [4]. Optical Character Recognition is not a computationally cost-free operation, especially for mobile devices like smartphones and tablets. Because the computational cost of correcting errors dominates the optical document conversion process, the most important characteristic of an OCR device is accuracy.

Thus, several performance considerations have been suggested for the Tesseract OCR engine. [9] discusses a novel, cost-effective method to eliminate background images/watermarks to improve OCR performance. One well-known critical procedure in OCR is to detect text characters from a document image. To address this potential issue, the authors first enhance the document images before OCR by utilizing brightness and chromaticity as contrast parameters. Then, they convert color images to grayscale and threshold it. In this way, as they claim, background images can be removed effectively without losing the quality of recognized text characters. In another study [8], the authors emphasize the default lighting conditions, the position object tilt, and the camera focus settings as

 doi:10.20944/preprints202502.1952.v1

external parameters to increase OCR text accuracy rather than re-design the Tesseract engine itself. Still, the results, as they claim, improve by an amount of 5% more in recognition accuracy.

As previously mentioned, the cryptographic performance of a VR framework, such as PRIVocular, relies on the visual accuracy of the captured information. Although an error-prone technique, OCR is the natural way of optical data encoding and transmission method. Thus, despite its potential negative effect on *integrity*, PRIVocular includes OCR detection as the baseline for privacy-preserving data transfer through air-gapped communication channels.

2.2. QR Code

QR (abbreviated from Quick Response Code) is the trademark of the matrix barcode (or two-dimensional barcode). A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to store data efficiently; extensions may also be used. A QR code consists of black squares arranged in a square grid on a white background, which can be read by an imaging device such as a camera and processed using Reed-Solomon error correction until the image can be appropriately interpreted. The required data are then extracted from patterns present in the image's horizontal and vertical components [5].

The amount of data that can be stored in the QR code symbol depends on the data type (mode or input character set), version (1, ..., 40, indicating the overall dimensions of the symbol), and error correction level. The maximum storage capacities occur for 40-L symbols (version 40, error correction level L) [5]. Codewords are 8 bits long and use the Reed-Solomon error correction algorithm [6] with four error correction levels. The higher the error correction level, the less storage capacity. Table 1 enlists the approximate error correction capability at each level.

Table 1. QR Error-Correction Levels

ECC Level	Amount of correctable data
Level L (Low)	7% of codewords can be restored
Level M (Medium)	15% of codewords can be restored
Level Q (Quartile)	25% of codewords can be restored
Level H (High)	30% of codewords can be restored

Figure 2 illustrates the structural analysis of the QR tag. The *version parameter* specifies the size and data capacity of the code. Version ranges between 1 and 40, where version 1 is the smallest QR code and version 40 is the largest. If this parameter is left unspecified, then the contents and error correction level will be used to guess the smallest possible QR code version that the content will fit inside.

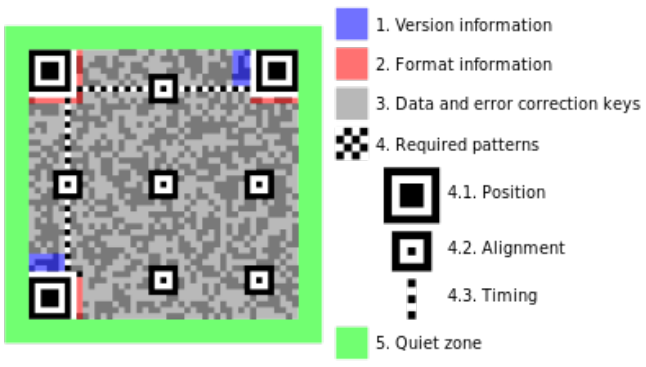

Figure 2. Structure of a QR code, highlighting functional elements [5].

 doi:10.20944/preprints202502.1952.v1

While the OCR technique often lacks optical accuracy and stable performance, QR is a stronger candidate for more time-critical machine-readable applications. Thus, PRIVocular further utilizes QR tags as an additional data transmission mode through air-gapped channels. Moreover, the framework has also included custom extensions to the conventional QR-tag specification to encapsulate more data while retaining the same error-correction accuracy standards.

2.3. Paillier Encryption Scheme

As mentioned earlier, PRIVocular uses Paillier as the underlying cryptographic engine. Any cryptosystem can be used with PRIVocular, as its main contribution is the encoding/decoding of visual data, not the cryptosystem utilized. Any other cryptosystem (e.g., AES) can be easily used in PRIVocular. Paillier has been chosen because partial homomorphic encryption schemes allow data manipulation without the cryptographic key. In the context of PRIVocular, **the display source does not require the decryption key; It can still process encrypted data, but not decrypt it**. This section briefly discusses the Paillier cryptosystem to justify the performance overheads of encoding/decoding and encryption/decryption.

Let N be a cryptographic parameter equal to the product of two random primes, p and q. We consider m the plaintext and c the corresponding ciphertext. The Paillier encryption scheme is defined as a unique correspondence between a value c from $\mathbb{Z}^*_{N^2}$ and values m and r from $\mathbb{Z}^*_N$ and $\mathbb{Z}_N$ accordingly:

$$c = r^N g^m mod N^2 \tag{1}$$

where g is a generator in $\mathbb{Z}^*_{N^2}$. In our encryption scheme, r is the probabilistic part, while m is the plaintext value to be protected [11]. Moreover, decryption requires knowledge of ϕ that is the value of Euler's totient function of N, and $n = pq$:

$$\phi = \lambda = lcm(p-1, q-1) \tag{2}$$

Ensuring n divides the order of g is done by checking the existence of the following modular multiplicative inverse:

$$\mu = (L(g^\lambda mod n^2))^{-1} mod n \tag{3}$$

where function L is defined as: $L(x) = \frac{x-1}{n}$. So, finally, we compute the plaintext message as:

$$m = L(c^\lambda mod n^2) \cdot \mu \, mod n \tag{4}$$

Decryption is essentially one exponentiation modulo n^2 [3].

2.4. Related Work

Recently, there has been quite an extensive research effort to develop commercially available and open-source applications built into integrated VR-ready frameworks. These applications vary from performing OCR for translation and text-to-speech conversion through a user-interface interaction to image map recognition for geolocation services or even securing video streaming on smartphones. Next, we mainly focus on relevant VR-based applications, which utilize and process visually compressed data sources and any security constraints integrated with the Android API framework [12].

In [10], the authors have developed an Android application that combines the Tesseract OCR engine, Bing translator, and smartphones' built-in speech-out technology to perform text detection and translation from a captured image source. By using this application, the authors claim that travelers who visit a foreign country will be able to understand messages portrayed in different languages. Finally, visually impaired users could access important messages from printed text through speech features.

 doi:10.20944/preprints202502.1952.v1

From the QR encoding technique perspective, the authors in [13] build a framework using Java and Android that implements a digital signature technique for electronic prescription to prevent cybercrime problems such as robbery, modification, and unauthorized access. The prescription recipe is encoded into a conventional QR code and is encrypted using an asymmetric algorithm. To decrypt the QR image tag with the recipe, a third user needs to log in to the Android part of the framework application to gain access to the public key. If the verification process is successful, the application will display the recipe as decrypted from its corresponding QR tag. In [14], the authors suggest designing and implementing a two-factor identification authentication system using QR codes. Their system claims to provide another level of security, where the QR code acts as the first factor and the Android mobile system as the second. QR-TAN [26] is a smart authentication technique that relies on QR tags and smart cards to validate electronic transactions. QR-TANs authenticate transactions by using a trusted device, such as a smartphone. Finally, [15–19] discuss relevant work in the field, where QR code is being utilized to visually encode and transmit data correctly and provide cryptographic mechanisms for confidentiality or authentication purposes.

Ubic [1] is a perceptual framework based on head-mounted displays and computer terminals to perform a wide range of cryptographic primitives, such as secure identification, document verification using a novel physical document format, and content hiding. DARKLY [25] offers a privacy protection layer in untrusted perceptual applications over trusted devices. In [20], the authors propose an application for video streaming on Android smartphones that can capture video from a smartphone camera and then send it to the computer in real time. The contribution is that the video can be secured by selective encryption of critical data in the video. Selective encryption only selects important parts of the video that will be encrypted. Finally, a general framework for developing augmented reality-based, multiplayer adventure games is proposed in [21]. The framework can create online treasure hunt or scavenger hunt games for mobile devices, i.e., Android-based. It offers integrated image recognition support combined with GPS-based localization. Specifically, image recognition is used to determine the exact location of the player-user, and then a picture is displayed in augmented reality mode. Perhaps one of our solution's most recent and similar research efforts is [28]. Hereby, when visual tracking is enabled, a novel visual cryptography technique is used that is tolerant to users' head motion and slight misalignment of the two shares of encrypted visual information. However, the scheme is not very scalable because it only relies on generating a two-shares scheme, i.e., a one-time pad, and it remains ad-hoc only for this sole purpose, despite using virtual reality. Finally, the Author(s) in [27] extend the Physical Unclonable Function (PUF) and Virtual Proof (VP) sensors to prevent key exchange attacks based on hardware implementation rather than number theory. They claim the novel key exchange methodology is developed and demonstrated using experimental data based on a Virtual Proof of Reality.

To address the limitations of the existing SoTA works, i.e., lack of IoT flexibility, cybersecurity scalability, and security protocol interoperability, the unique contribution of the proposed PRIVocular framework is that it targets purely privacy-preserving information transfer through automatic detection of ciphertext data on display sources, providing a seamless experience to the user.

3. The PRIVocular Framework

3.1. Contributions

To make a practical contribution, we mainly build a resource-friendly, graphical user interface and easy-to-use system that can be deployed into the current Android-based mobile infrastructures. From a technological perspective, PRIVocular offers the following functionalities and novelties:

Authentication. PRIVocular's generic encryption scheme (Paillier) includes a cryptographic key generation and distribution phase. The framework realizes this technique through a high-definition camera from smart mobile devices and a computer node (key server). The previous allows users to

 doi:10.20944/preprints202502.1952.v1

authenticate themselves before creating, encrypting, sharing, and decrypting any type of written text. Specifically, the encryption and decryption keys produced during this phase are encoded as QR tags so that no eavesdroppers could potentially read them (blind eye) apart from the authenticated users themselves. It is implied, at this point, that the user(s) that will generate the keys (public & private) is the same user(s) that will eventually be able to obtain the plaintext through PRIVocular.

Content Hiding. We provide an end-to-end solution for ensuring privacy through air-gapped transmission channels. Rather than projecting the ciphertext(s) and public/private keys on the screen as is, we encode them to **hexadecimal** format and concatenate them so that no human user could brute-force and locate each encrypted group segment that belongs to a single ciphertext character. Only the authenticated user(s) possessing the required decryption key can perform the un-grouping successfully.

Efficient Data Encapsulation method(s). PRIVocular offers three different data encapsulation and transmission methods: *OCR*, *QR-based* and *custom-QR-based (MoMAtag)*. Each specific method possesses different design specifications and performance efficiency; however, for the proof-of-concept, we include a comparative performance study and comprehensive visual analysis for all in Section IV.A.

OCR Filtering Intelligence. Another novelty of the framework, inside the OCR mode of operation, is its software capability to distinguish between encrypted and non-encrypted text on computer displays. For instance, PRIVocular's OCR engine can separate the *ciphertext* and *non-encrypted* text and produce the resulting plaintext nominally.

MoMAtag. We have expanded the conventional storage amount properties of the traditional QR-tag version 40 to fit almost **double** the previous size for our encryption needs. This was particularly driven due to the incremental ciphertext data size on screen while the key size increased.

3.2. PRIVocular's Architecture

PRIVocular is a VR/XR framework that allows end-users (s) to choose among three visual data encoding techniques. Both the application's client and server-side parts have three operation modes. The data is encoded using one of the three encoding types, to be consequently encrypted and visually captured on conventional computer displays by a VR device UHD camera. Finally, decryption decoding on the server side is performed. The reverse cycle of encoding-encryption-transmission-decoding-decryption (see Figure 6) depends on the underlying cryptographic scheme. Specifically, if the end-user has not obtained the correct encryption/decryption keys, he/she cannot generate the correct ASCII-typed characters.

Thus, the goal of *confidentiality* should be satisfied by construction in the PRIVocular framework. *Integrity* and *performance* would be the most challenging parts.

PRIVocular system framework, as in Figure 3, consists of the following (hardware) key elements: (1) ***Prompting displays***, could range from wall projector sources, UHD monitors, LCD computer displays, to smartphone device monitors. The particular display sources serve the key role of displaying the end-user-typed characters and recognizing or detecting all the visually encoded ASCII characters from the server-side part of the application. (2) ***Client-side smartphone, or tablet***, that will allow the user(s) to type or input any ASCII character, as keystrokes, to PRIVocular. (3) ***Server-side smartphone, tablet, or VR smartglasses, equipped with UHD camera***, which can capture the visual encoding format and perform the decoding and decryption cycles to retrieve the correct data.

PRIVocular consists of the following (software) key element technologies: (1) ***Key Generation and Distribution Server***. A third node in the system generates the public/private cryptographic keys for encryption/decryption. The Key-Server is written in Java and could be deployed with the prompting-display computer node. (2) ***Android-based API*** for both the client-server software implementation side part of the framework.

 doi:10.20944/preprints202502.1952.v1

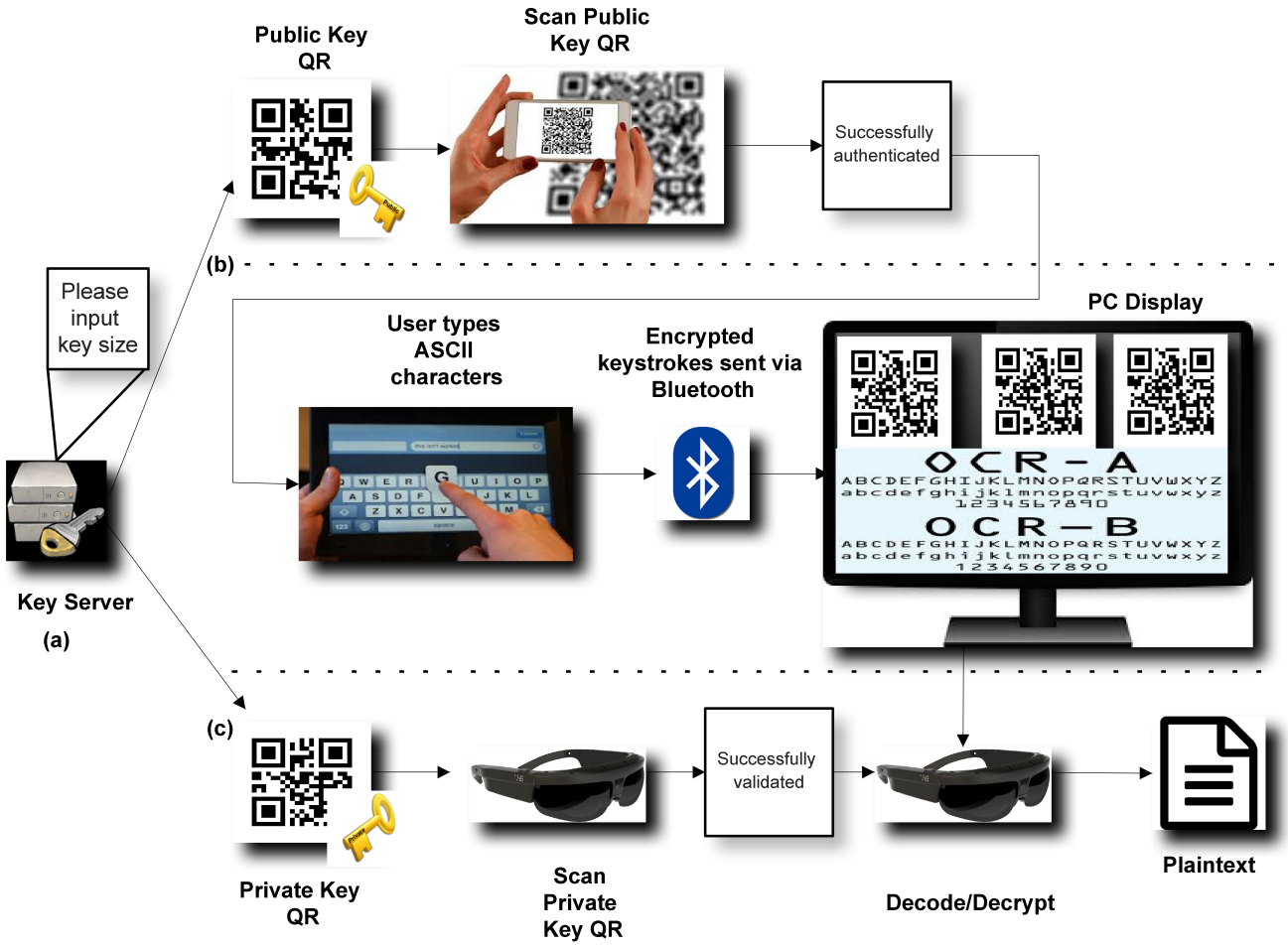

Figure 3. Overview of the PRIVocular stateflow functionality: (a) key generation phase, (b) encryption phase (client-side), and (c) decryption phase (server-side).

 doi:10.20944/preprints202502.1952.v1

PRIVocular encompasses the two following information transmission methods: (1) ***Bluetooth wireless technology***, to transmit the keystrokes from the client part of the framework to the prompting devices, which are, of course, connected to a Bluetooth adapter, and (2) ***visual transmission***, i.e., images and optical characters are transmitted on the visible spectrum (captured by smart-device cameras).

Finally, the three modes of operation for PRIVocular, both sides, end-to-end, are: (1) ***Optical Character Recognition-OCR***, which is of course achieved by displaying optical characters in Base16 format, in the prompting displays, (2) ***MoMAtag***, which is a customized QR-code tag based on the Quick Response code (see Section II.B), able to fit double the size of the QR version 40, and (3) ***Hybrid***, which is based on a grid-layout of conventional QR-codes, each QR dedicated to a sole encrypted ASCII character, any version from $1 \cdots 40$, depending on the size of the previous individual ciphertext.

Figure 3 also depicts the PRIVocular processing and interaction pipeline, end-to-end, for the different encapsulation modes. It is worth mentioning, at this point, that the framework can scale to multi-party environments, i.e., multiple authenticated users with the same public/private key, as long as they are all trusted parties.

3.3. Use Case

PRIVocular's main goal is to generate and optically capture-reconstruct the **maximum** amount of ASCII characters typed by end-to-end user(s). Because we select transmission on the visible spectrum, and due to the limited spacing of characters on any displays, there is always a significant balance between ocular performance and accuracy. As mentioned earlier, accuracy or integrity is vital for successful encryption-decryption. Due to the mathematical properties of the Paillier cryptosystem (i.e., uniqueness of plaintext-ciphertext space mapping), it is crucial to reconstruct or decode the original user data from any mode of operation with a 100% error-free state. Room lighting conditions can significantly affect visual performance as well. External factors, such as room brightness, visual distortion or reflexes on the screen, color interference from different lighting sources on the display, or even the physical distance between the smart device's camera and the monitor, play a key role. Another critical factor, especially for the QR/MoMAtag cases, is a malefic user trying to maliciously alter or destroy the image tag to forge the integrity part of the framework. By any means, if data is decoded erroneously, either due to deliberate or natural causes, decryption will instantly fail.

Thus, it is important to introduce error-correction-detection codes for all cases or modes, like in MoMAtag and Hybrid methods. Still, in the OCR case, we only allow raw data to be encoded and displayed without any error correction for the proof-of-concept. PRIVocular framework has been explicitly designed to perform source detection *(input visual content from multiple sources)* and client detection *(to which client to explicitly share visual content, like visual shared keys)*. For the first case, the framework can adapt to any custom display capabilities, i.e., detect the custom display resolution and graphic capabilities, and based on those specifications, decide the optimal display parameters (font size, zoom level, orientation) of the optical characters or QR-tags. For the second, PRIVocular functionality corresponds only to the end client that carries his/her specific encryption/decryption key. We could impose that authentication is another derived goal of PRIVocular, meaning only the correct key-holding client can successfully derive the original plaintext. Furthermore, because the software part of PRIVocular runs on smart-devices client hardware, the framework possesses a minimalistic design approach. The system framework picks security constraints as a top priority and manageable performance.

To conclude this subsection, a typical applicability case scenario for PRIVocular, based on Figures 3 and 4, should follow the next sequence of events:

Key Generation and Distribution phase

1. The end-user inputs his/her desired key size in the (software-based) Key Server's input prompt.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 February 2025 doi:10.20944/preprints202502.1952.v1

2. The Key Server creates the Public (Encryption) Key QR in addition to the Private (Decryption) Key QR based on *random* prime numbers p and q, each time, for the Paillier cryptographic scheme.

Data Encapsulation and Encryption phase (client-side)

3. The client-side application of PRIVocular reads the Public Key QR to generate the encryption parameters.
4. The end-user then selects a visual encoding technique, or data representation method, among OCR, Hybrid, or MoMAtag from the software GUI.
5. The end-user can now start typing ASCII characters from the client application. Encryption is performed in *stream mode* (i.e., the ciphertext is produced per character typed on the on-the-fly).
6. Through the Bluetooth communication interface, each character typed is sent to the prompting device screen via a Bluetooth (server) adapter. The previous implies that all PRIVocular devices should be paired via Bluetooth and become synchronized. Depending on the data encapsulation mode:
   - **Hybrid mode:** Each time a character is typed, a QR code is shown on the prompting screen in a grid layout, with the corresponding ciphertext of the typed character as its content.
   - **MoMAtag mode:** encryption is done in *block mode*, meaning after the user has typed an ASCII character, he/she will generate the MoMAtag with its ciphertext, a posteriori.

Data Decoding and Decryption phase (server-side)

7. The server-side application of PRIVocular reads the Private Key QR by the same authenticated end-user to generate the decryption parameters.
8. The end-user will then select the applicable mode of operation, depending on his/her generated or encoded form of plaintext data.
9. The end-user, provided he/she has obtained the correct decryption key format, can now successfully decode and decrypt, thus reconstructing the original message on the smart device's display.

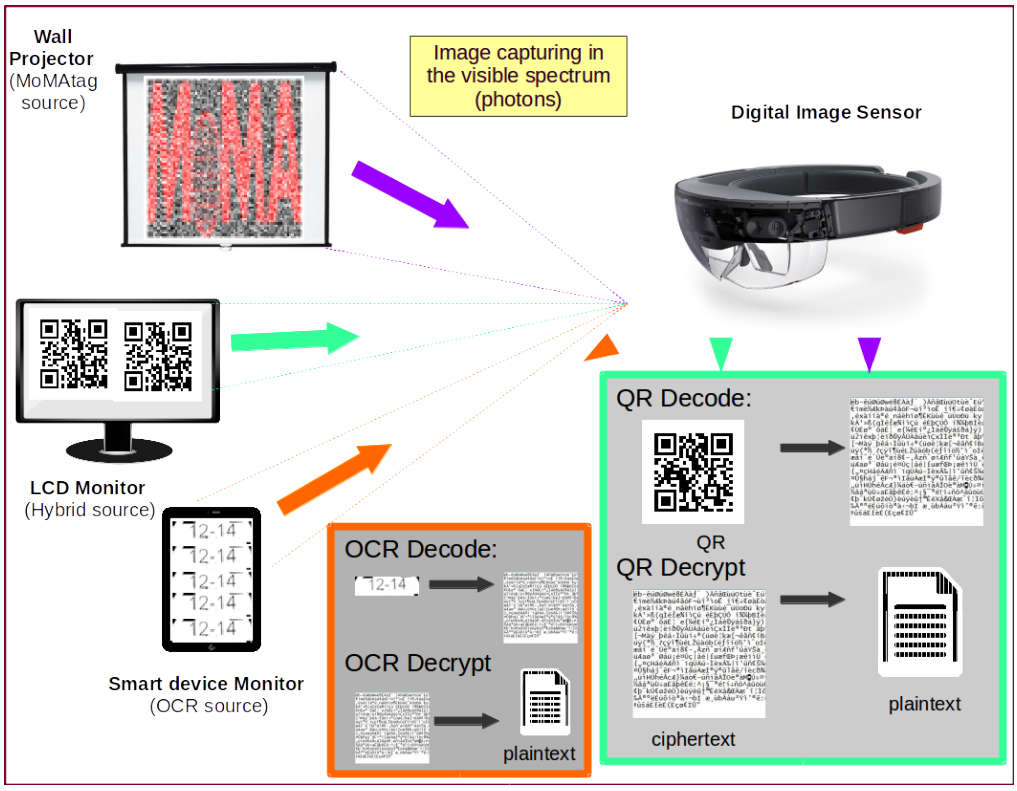

Figure 4. PRIVocular's typical applicability scenario

 doi:10.20944/preprints202502.1952.v1

Figure 5. PRIVocular combined with VR smart-glasses

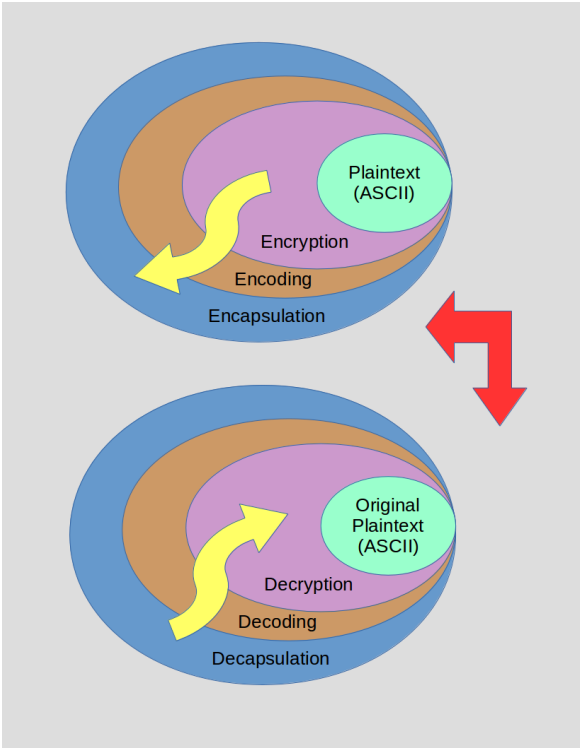

Figure 6. The encode-decode/encrypt-decrypt full cycle

3.4. PRIVocular Key Components

Inside this Section, we present, in more detail, the *Key Server* features and functionality for PRIVocular framework, as well as the three modes of operation, or data encapsulation means: *OCR*, *MoMAtag*, and *Hybrid*.

3.4.1. **Key Generation and Distribution Server**

The Key Server is a software-based key generation and distribution component for the necessary encryption and decryption procedures of the Paillier cryptographic scheme we are utilizing in PRIVocular. It mainly accepts as only input parameters from the end-user, the key size, which should be a power of 2, from 32 bits to 4096 bits, for the whole range of the framework functionality. The process of key generation distribution is directly illustrated in the next figure (Figure 7).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 February 2025 doi:10.20944/preprints202502.1952.v1

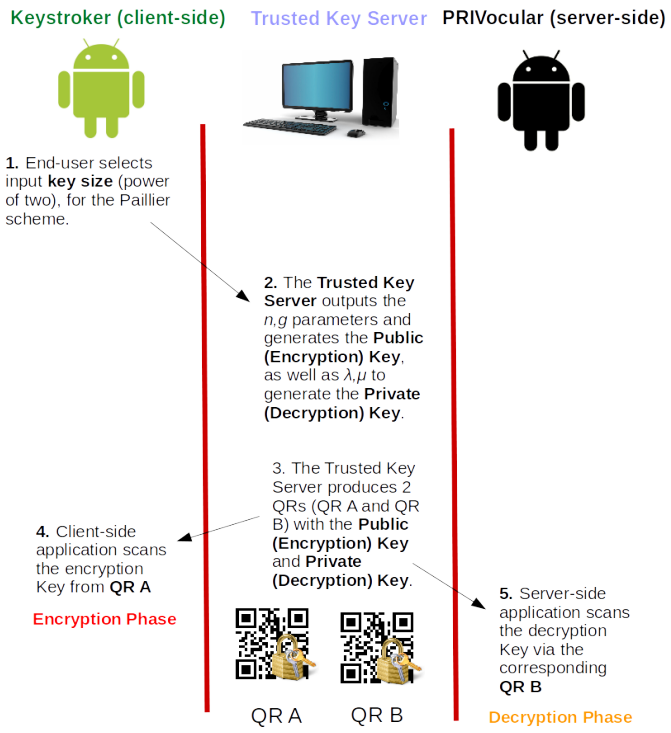

Figure 7. The key-generation and distribution technique

The Key Server inside the scope of PRIVocular's functionality is considered a fully trusted server. It should be emphasized that no cryptographic keys are being stored in the non-volatile memory of the server, and all cryptographic primitives being produced are being placed only in the RAM. Encryption and decryption keys are outputted as QRs, only displayed on the screen terminal as UNIX non-printable ASCII characters (black and white stripe blocks), without being saved or stored as image files [22]. Thus, after a few seconds, the QRs are flushed from the terminal screen regardless of whether the end-user has scanned them.

As discussed in Section II.D, the essential cryptographic parameters to construct the public (encryption) key is the pair: (n, g), whereas to derive the private (decryption) key the user needs: (λ, μ) [3]. Upon user (key size) input, the Key Server generates two random prime numbers based on the key length in bits and produces the two corresponding pairs (encryption and decryption) to be embedded inside the two Key QRs. It is worth noticing that even if the end-user inputs the same key size in bits each time, the encryption and decryption keys remain different due to the secure randomness of operations. Thus, successful decryption would be impossible each time, even with the same key size. Figure 8 demonstrates the content data, in packet illustration format, for the encryption and decryption Key QRs.

 doi:10.20944/preprints202502.1952.v1

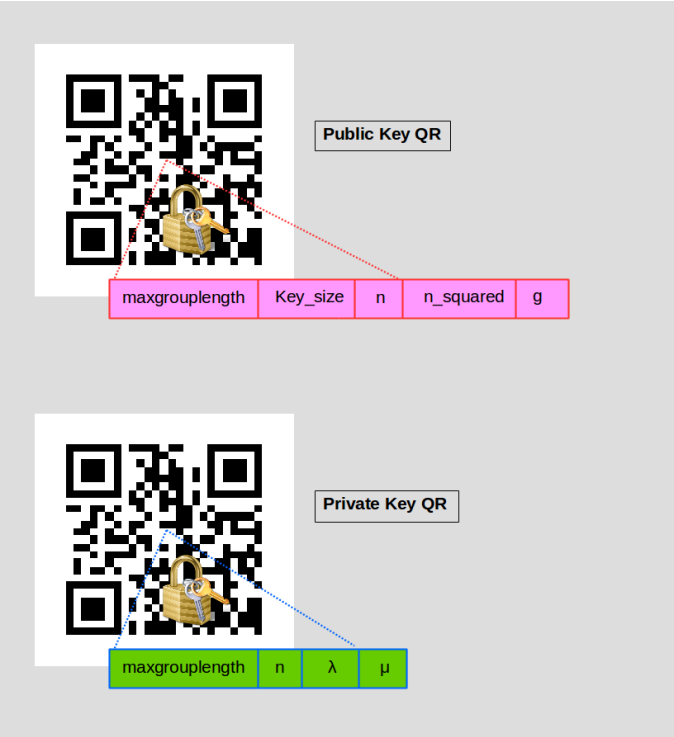

Figure 8. The Key QRs content data

Finally, as easily depicted in Figure 8, we include a new packet header inside the two Key QRs content data, which is named *maxgrouplength*. This parameter is not an actual cryptographic primitive essential for any Paillier encryption or decryption calculations; it mainly groups or separates into ciphertext segments all the encrypted ASCII characters (encoded in Base 16 format) in the OCR mode. The purpose of grouping is a vital process for post-decoding OCR decryption. As described in the next Section, when successful OCR recognition of all the individual OCR ciphertext(s) takes place, the application engine requires information on the correct group length for all ASCII encrypted characters based on the decryption key parameters to decrypt them. Thus, *maxgrouplength* is directly derived from the post-encryption process. It aids the application algorithm in segmenting and matching the OCR result text, which it decodes into the corresponding sole ASCII encrypted characters. Without the grouping parameter, locating each ciphertext, and/or finding the total number of ciphertexts, or even performing decryption would be challenging.

The method by which *maxgrouplength* is being calculated by the internal functionality of the Key Server is the following: Based on the encryption key parameters, already computed and derived, the Key Server performs an «internal» encryption between the minimum printable ASCII character, and the maximum printable ASCII character. The server will then be able to find the maximum hexadecimal length needed for any one of the output ciphertext characters based on sequential comparisons. The maxgrouplength parameter should now correspond to any encryption length range of the printable ASCII characters, provided they would be encrypted with the same key parameters. Finally, to cover the case when the produced ciphertext might have fewer digits length than the previous parameter (*maxgrouplength*), we perform zero padding in the most significant HEX digits, with the necessary number of zeros.

3.4.2. **OCR**

Optical characters are the first mode of visual data encapsulation for PRIVocular. Figure 9 illustrates the OCR process from data generation, encryption, encoding, encapsulation, and vice versa. Once the end-user types an ASCII character from the client-side part of PRIVocular, the ASCII character

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 February 2025 doi:10.20944/preprints202502.1952.v1

is zero-padded, based on the maxgrouplength parameter, then converted to the decimal numeral system to be encrypted by the Java-based Paillier library, and finally converted to Base 16 for the OCR representation method. Since transmission is being performed on the visible spectrum, OCR images are being captured and analyzed by the UHD camera of the server-side part. The OCR text is first recognized by the OCR engine; depending on the physical distance, lighting conditions, and various other optical parameters, successful recognition is not always 100% guaranteed, but it is a vital process for successful decryption to proceed.

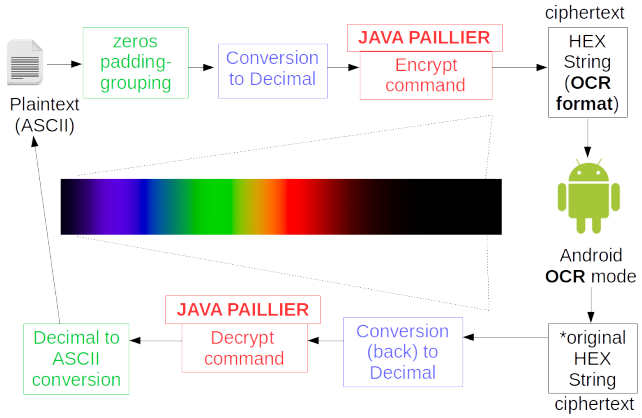

Figure 9. The OCR mode of operation functionality

The OCR procedure does not follow any error-correction-detection technique; thus, we have to introduce a confidence metric for the level of text errors introduced. For some OCR applications, like PRIVocular it may be important to know the reliability of the recognized text generated by the engine. Confidence metric, or *mean confidence* property, expresses the certainty of the character recognition and ranges between 0 and 100 [23]. A value of 100 means that the engine recognized the character with high confidence. Applications that examine character confidence information can use a threshold value. As demonstrated in Figure 10 below, the value of a character is treated as a suspicious result. Based on experiments, we identified that a value of 64 is best for this purpose. A value of 64 or more indicates high confidence that the character was recognized correctly. A value less than 64 marks that code as suspicious.

One main contribution functionality of PRIVocular for the OCR mode is the capability of the OCR engine to detect, filter, or isolate between encrypted and non-encrypted text. In most real-case scenarios, encrypted ASCII characters, represented in Base 16, can be perplexed with normal ASCII characters, numbers, computer graphics on screen, etc. Ideally, we would prefer to depict only the ciphertext(s) on the display. Still, very often, even a single character or digit not belonging to the ciphertext set could easily ruin our decryption process. That is because the OCR engine searches for the whole device's camera capture area to detect optical characters. The recognition effort might include several other 'foreign' digits, even in that range. Thus, as described by the below algorithm, the PRIVocular OCR engine has extra intelligence to distinguish between ciphertext(s) and non-encrypted text in the same visible area.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 February 2025 doi:10.20944/preprints202502.1952.v1

Algorithm 1 Encrypted OCR filter algorithm

```
procedure MAIN
    maxgl ← maxgrouplength
    OrigOCR ← OCR_TextResult
    HEX_OCR ← extractHEXchars(OCR_TextResult)
    for (i = 0; i <= (stringlenof[HEX_OCR] − maxgl); i += 1) do
        idx_start ← i
        idx_end ← (i + maxgl)
        ciphertextgroup ← HEX_OCR.SubString(idx_start, idx_end)
        ctg ← ciphertextgroup
        res ← decrypt(ctg)
        if (res is ASCII) then
            // Decryption successful
            pre_decrypt_results ← append(res)
            if (ctg in OrigOCR) then
                pos_index ← OrigOCR.FindIndexOf(ctg)
                decrypt_res_coordinates ← append(pos_index)
            end if
        end if
    end for
    postOCR(pre_decrypt_results, decrypt_res_coordinates)
end procedure
```

The key idea of the filtering algorithm is to extract the 'purely' hexadecimal digits from all OCR text that the engine detects on screen. That would eventually limit the results significantly to the scope of isolating only the Base 16 ciphertext(s). Still, though, even at this step, the algorithm might (wrongly) input decimal digits (0-9) that correspond to Base 16 or even some ASCII characters, i.e., Aa, Bb, Cc, Dd, Ee, Ff, that would be mistakenly considered to belong to the previous correct set. For that purpose, the algorithm performs a sliding window search, whose size is the *maxgrouplength* parameter (see Section III.B.1), for all the Base 16 filtered OCR text to check a priori if the corresponding groups of HEX values can be decrypted or not. Thus, for each elementary HEX group, if decryption succeeds, the application will immediately show its plaintext value in the correct XY coordinates on the screen where it specifically appears. If decryption fails for any group, this search will dynamically continue on the next OCR input text result. At this point, the previous process occurs on the fly, meaning the application will show any plaintext character that could be decrypted, regardless of the rest (see Figure 10).

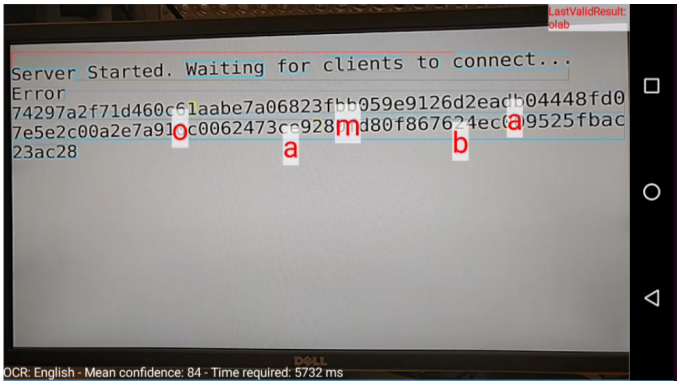

Figure 10. Example deployment scenario of OCR filter algorithm

3.4.3. **MoMAtag**

The main contribution of MoMAtag (MoreMAssivetag) is that it fits (almost) double the size of the conventional QR-code Version 40 in binary data encoding mode and with a High ECC level. It is basically an extension of the traditional maximum-sized QR-code tag. We constructed MoMAtag design specifications by expanding (1) *the data_capacity*, and (2) *Error Correction Code Words and Block Information* tables of the conventional QR [5]. Version_size, position_adjustment, and version_pattern tables were not altered. Furthermore, to increase the effectiveness of the Reed-Solomon error-correction and detection algorithm for such bigger data capacity QR-tag (MoMAtag), we introduced new values

 doi:10.20944/preprints202502.1952.v1

to the *generator polynomial*, as well as the log and antilog values used in GF(256) arithmetic of the algorithm [6]. The following table (Table 2) illustrates the comparison parameters between traditional QR-Version 40 and the MoMAtag.

Table 2. Comparison features between QR-40H & MoMAtag

	QR	MoMAtag
ECC Level:	Level H (High)	Level H (High)
Data encoding format:	binary/byte	binary/byte
bits/char:	8	8
max. characters:	1,273	2,049
EC Code Words Per Block:	30	30
Block 1 Count:	20	1
Block 1 Data Code Words:	15	2334
Block 2 Count:	61	0
Block 1 Data Code Words:	16	0

MoMAtag has been pre-selected, as a design assumption, to host **one** ASCII character encrypted with 4096-bit key size. Thus, such ciphertext string size (2048 HEX characters) would be impossible to fit inside the QR-code Version 40H, but with the MoMAtag, it is possible, combined with a HIGH ECC level. It would be possible to increase the size of MoMAtag by re-modifying the previous table parameters. That way, we could fit more ciphertext(s) with even bigger key sizes.

Figure 11 displays a MoMA tag. Figure 12 demonstrates the MoMAtag mode of operation, with the corresponding ASCII character (x) being detected, decoded, and decrypted on the smart device screen.

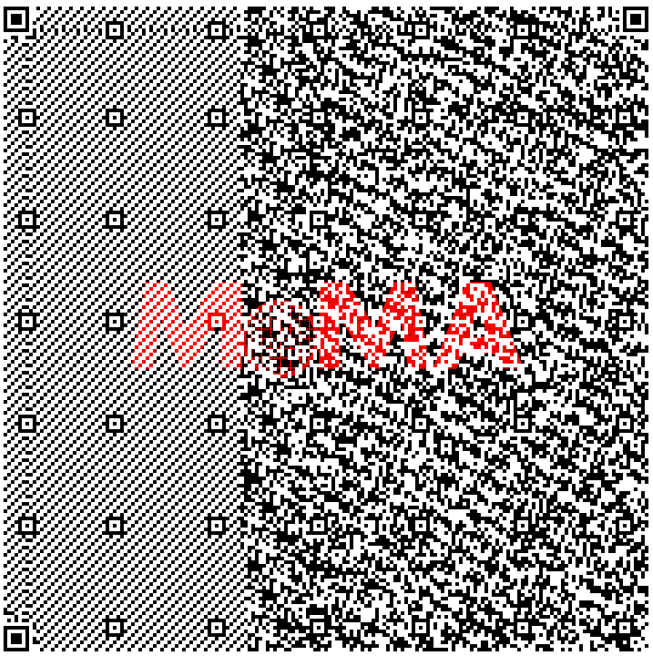

Figure 11. A MoMAtag image sample

 doi:10.20944/preprints202502.1952.v1

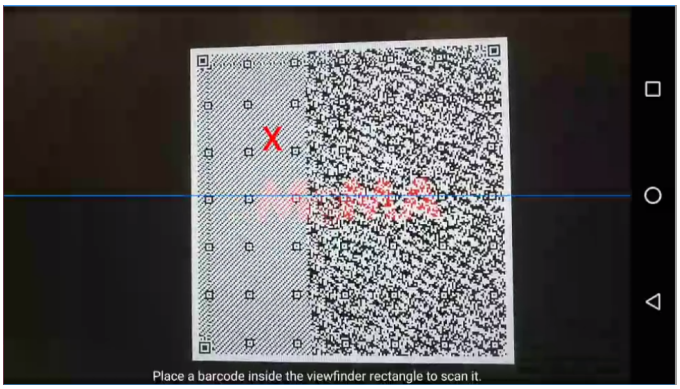

Figure 12. MoMAtag operation mode

3.4.4. **Hybrid**

Finally, we discuss the Hybrid mode of operation for PRIVocular framework, end-to-end. The main idea behind this operation mode is to illustrate on the prompting device one conventional QR-code (Version 1...40) dedicated for every typed ASCII character from the client side. Each QR code contains the ciphertext of the typed character, again with Base 16 encoding format. The QRs are displayed in a grid-style layout, in the same fashion a user types normal letters to form a sentence. Although PRIVocular's Hybrid mode engine decrypts what it sees on-screen (WYSIWYG), it is not possible to brute-force the total length of the plaintext because the terminal screen buffer might have more QR tags stored already from before, i.e., previously typed characters.

The Hybrid mode for PRIVocular works for all user-determined key sizes from 32 bits to 2048 bits. It is worth noticing that even with a key size 4096, one encrypted ASCII character could easily fit into a QR Version 40 but with a LOW ECC Level. Thus, we decided to utilize MoMAtag and separate the Hybrid and MoMAtag modes of operations, although they are both QR-based.

PRIVocular deploys the ZXing QR decoding libraries to detect and decode QR codes. ZXing ("zebra crossing") is an open-source, multi-format 1D/2D barcode image processing library implemented in Java, with ports to other languages [24]. PRIVocular implementation part, which is Android-based, modifies the previous libraries to perform decryption in bulk mode further, as well as to represent the original plaintext ASCII characters with their exact XY corresponding QR-specific coordinates on the device screen. The bulk-mode software option capabilities of the ZXing QR engine facilitate the operation of bulk QR decoding.

Finally, the QRs are displayed directly on the UNIX terminal screen as non-printable ASCII characters rather than image files [22]. The particular design criterion allows the easier manipulation of the codes, i.e., delete a QR when a user presses *Backspace* to remove a typed ASCII character. Furthermore, it is crucial to massively control the size of the displayed QRs, which changes on different key sizes; thus, by manipulating the QRs as *String* variables in the terminal, it proves more practical to alter their display dimensions, i.e., by changing the terminal font size, to increase performance.

Figure 13 depicts the Hybrid mode of operation, where the grid-layout (plaintext) display mode is presented.

 doi:10.20944/preprints202502.1952.v1

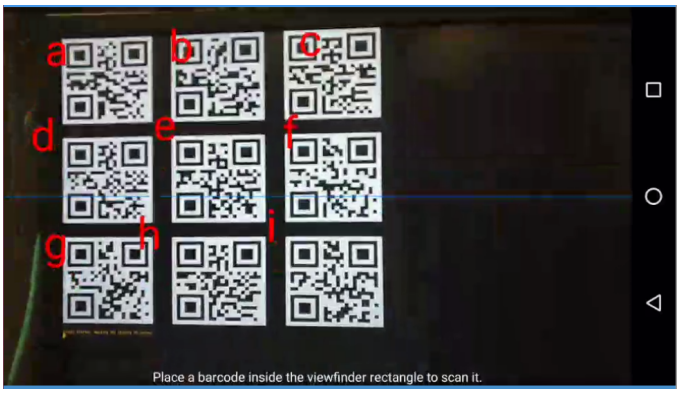

Figure 13. Hybrid mode of operation

4. Experimental Results and Analysis

4.1. Experimental Analysis

The experiment setup (for both analysis and final results) consists of utilizing a smart device (i.e., smart glasses or smartphone) camera (HD 720p) positioned at a stable viewing distance (approximately 43 cm) from the prompting device node. This node consists of a desktop PC monitor (with 1920x1080 native pixel resolution). The lighting conditions were considered the default, i.e., at normal room lighting. There were no visible obstacles between the camera and the source. Conditions such as the viewing angle and line of the site were not considered and will be explored in future work. An example of the experiment's functionality follows in Figure 14.

Figure 14. Experimental setup

 doi:10.20944/preprints202502.1952.v1

The (pre-final) visual experimental setup consisted of capturing two image types:

1. QR Low. A low-resolution QR Tag (372x359 pixels)
2. Text/OCR. A medium resolution Text-on-Screen file (1084x584 pixels)

To analyze the image statistics of each image type, the *skimage* python library was utilized, and additional analysis on various statistics, i.e., number of pixels per color channel, image entropy, and various histograms, was also performed. The goal of this initial setup is to understand which is the most efficient optical data transmission type through the visible spectrum. We also investigate image *Entropy*. In information theory, information entropy is the log-base-2 of the possible outcomes for a message. For an image, local entropy is related to the complexity of a given neighborhood, typically defined by a structuring element. The entropy filter can detect subtle variations in the local gray level distribution.

4.1.1. QR Low

In the first example, the image comprises two surfaces with slightly different distributions. The image has a uniform random distribution in the range [-14, +14] in the middle of the image and a uniform random distribution in the range [-15, 15] at the image borders, both centered at a gray value of 128. We compute the local entropy measure to detect the central square using a circular structuring element of a radius big enough to capture the local gray level distribution, as shown in Figure 15. The second example shows how to detect texture in the camera image using a smaller structuring element.

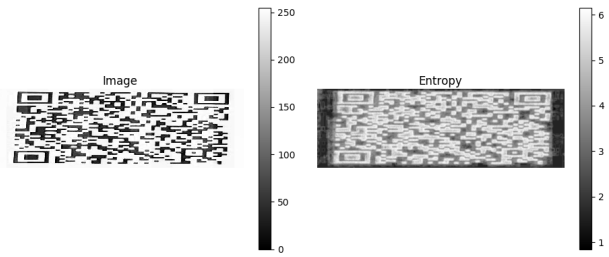

Figure 15. Image Type1 Entropy

The histogram in Figure 17 is interpreted as follows: The bins (0-255) are plotted on the X-axis. And the Y-axis counts the number of pixels in each bin. The majority of pixels fall in the range of 230 to 255. Looking at the right tail of the histogram, we see almost every pixel in the range 200 to 255. This means many pixels that are almost "white" are in the image. Based on this initial research, the preliminary conclusion is that the QR tag can be decoded successfully and error-free from the visible spectrum almost instantly. Thus, *it appears to be the most efficient means of optical transmission*.

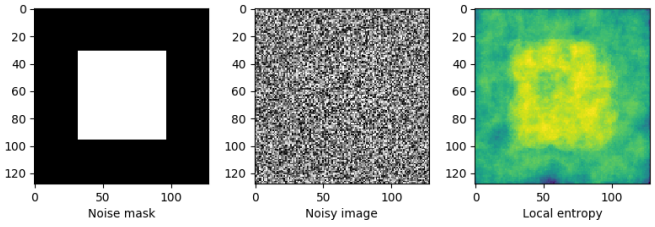

Figure 16. Image Type 1 Noise

 doi:10.20944/preprints202502.1952.v1

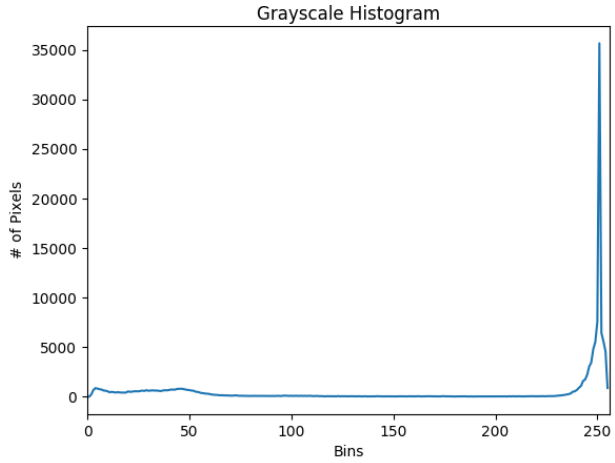

Figure 17. Image Type 1 Grayscale Histogram

4.1.2. Text/OCR

Again, we compute the entropy parameters for the OCR text file. The results are depicted in Figure 18. The figure shows that the entropy range now appears more restrained than the QR case. However, if we observe the histogram illustration (Figure 19), we notice that this time, the bins on the X-axis are more widely distributed. Thus, there is a wider majority of pixel range than the 'black-white' case of the QR. This diversity of the pixel colors, together with the particular image entropy or gray level distribution, implies that the OCR text cannot be visually detectable in an error-free state. Thus, it initially seems from the plots that OCR *is not the most effective means of ocular data transmission*.

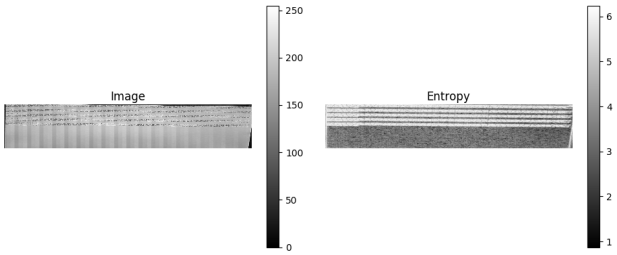

Figure 18. Image Type 2 Entropy

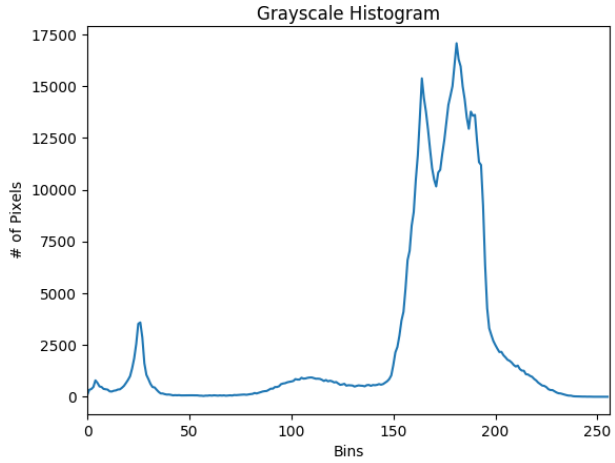

Figure 19. Image Type 2 Grayscale Histogram

 doi:10.20944/preprints202502.1952.v1

4.2. Performance Evaluation of PRIVocular Framework

For the scope of the main evaluation analysis of PRIVocular framework, we have conducted two sets of experiments: the first with a vertically oriented prompting PC display screen and the second with a horizontal or landscape orientation. For each case scenario, we have utilized the same smartphone device equipped with a UHD camera (16MP, 1080p) and the corresponding orientation mode (portrait or landscape).

Next follows the first results table (see Table 3). This visual analysis table corresponds to the vertical orientation mode for both end-to-end displays. The table includes all three input modes of operation, i.e., OCR, MoMAtag, and Hybrid. The aim of this experiment set, as well as the second, would be to ideally capture and successfully retrieve the **maximum** amount of optical data on screen while maintaining several other technical parameters such as physical distance, visual space of data on-screen and encoding types as observational dependant variables. We pre-select to run this setup with a 32-bit key size for OCR/Hybrid modes. MoMAtag mode is dedicated to a 4096-bit key size.

In Table 3, the second column, named *Num. of elements*, matches the number of HEX characters that can be displayed, captured, detected, and decrypted from the prompting screen. Of course, this number can be divided by the *maxgrouplength* parameter to uncover the total number of ASCII characters optionally recognized. The third column (i.e., *Max. characters*) is the maximum amount of (byte) data that can be successfully decoded and decrypted in the corresponding input mode. Finally, the column named *Square Pixels Per Element* could be considered as a visual space calculation metric for needs, i.e., to estimate the space (width #pixels X height #pixels) for each (corresponding input mode) individual element at the desktop PC monitor.

Table 3. Visual Analysis Table

Input Mode	Num. of elements	Max. characters	Bits/char.	Encoding	Square Pixels Per Element	Distance
OCR	1200	600	8	HEX/byte	1683	45cm
MoMAtag	2048	1024	8	Binary/byte	1176120	60cm
Hybrid	640	320	8	Binary/byte	49163	100cm

Obviously, each mode of operation performs differently, simply because due to the specific optical encapsulation method; QR-based encapsulation, like in MoMAtag/Hybrid cases, seems to be able to fit more raw data information, whereas decode and decrypt faster (due to ECC presence) than OCR. Other factors, such as the image entropy characteristics of an OCR-Text file compared to a QR image tag, as depicted in the previous subsection, contribute to this argument.

Using Table 3, we compare OCR-based and QR-based representation methods. Although OCR appears to consume less visual space on screen per individual element and successfully process 1200 HEX elements, or 600 bytes, it is the least effective means of optical transmission. To justify the previous argument, it is easily observed that QR-based methods can capture more byte data, even while consuming more area space on the display screen. Thus, we can conclude that MoMAtag appears to be the most effective method of ocular encapsulation and transmission among the QR-based techniques and, as a total, from the first set of experiments.

For the second set of evaluations, we conducted the same setup with the horizontal orientation mode for both devices and derived more technical parameters as output results. Table 4 matches the OCR operation mode, whereas Table 5 corresponds to the QR-based techniques (i.e., MoMAtag/Hybrid).

 doi:10.20944/preprints202502.1952.v1

Table 4. Visual Analysis (OCR) Table

Key Size (bits)	32	64	128	256	512	1024	2048	4096
Num. of recognized elements (ASCII)	47	22	9	4	1	NA	NA	NA
Max. characters (HEX)	752	704	576	512	256	NA	NA	NA
Square Pixels Per Element Character (pixels)	2184	2184	2184	2184	5332	NA	NA	NA
Physical Distance (cm)	43	43	43	43	43	NA	NA	NA
Mean confidence	75	84	85	81	91	NA	NA	NA
Time Required (msec)	30730	30137	22450	20825	13954	NA	NA	NA

Table 5. Visual Analysis (QR-based) Table

Key Size (bits)	32	64	128	256	512	1024	2048	4096
Num. of recognized elements (ASCII)	24	18	8	6	3	2	2	1
Max. characters (HEX)	384	576	512	768	768	1024	2048	2048
Square Pixels Per Element Character (pixels)	54756	74529	125316	190096	351649	625681	447561	2190400
Physical Distance (cm)	43	43	43	43	43	43	43	43
Mean confidence	-	-	-	-	-	-	-	-
Time Required (msec)	2140	1040	720	870	800	860	940	3500

In this setup, we maintain the key size (ranging from 32 to 4096 bits) as an independent variable and analyze six dependent parameters derived from the experiment. Mention at this point that for the OCR case, we have introduced the mean confidence metric (see Section III.B.2), as well as the time required for the engine to perform OCR detection (time metric is internally computed from the Android application environment). The latter applies to the QR-based methods as well, with the exception of the mean confidence metric. The reason is that the OCR deployment in PRIVocular does not encompass any ECC method. On the contrary, the Hybrid/MoMAtag methods have embedded error-correction codes inside their specifications (Reed-Solomon). Finally, physical distance is the ideal distance between the device camera and the prompting display for successful recognition.

Table 4 shows that OCR does not become functional for key sizes larger than 1024 bits. One individual ASCII element encrypted with a key size above 512 bits consumes much space on the screen. Therefore, it depends on at least 512 HEX characters for successful decryption; if one is mis-detected by the OCR engine, the whole decryption process fails. The same table shows that the number of recognized ASCII characters is minimized as the key size becomes larger towards 512 bits.

As the key size grows, each ciphertext group contains a bigger (segmented) HEX length, requiring more visual space. Less useful byte data remains visible on-screen, so the time to decode/decrypt becomes shorter. To enhance the previous argument, we can easily comprehend why, for the same reason, mean confidence becomes higher as the key size becomes larger.

Finally, we analyze the last table of our evaluation steps (Table 5). By only considering and comparing the number of recognized ASCII characters between OCR and Hybrid, it seems that OCR outperforms QR-based techniques at first glance. That is not an apparent and correct assumption, especially considering all further technical parameters. Although QR tags consume at least 25% more visual space, for the 32-bit key size, and as mentioned before recognizing 50% less ASCII characters, still OCR remains an erroneous technique, thus its detection time is tens of seconds with erroneous performance. On the contrary, in QR tags, the time to retrieve data is instant, performance is error-free, and with maximum availability.

Hybrid mode, with different-sized QR tags in each key size case, has the same smooth performance for all ranges of key sizes (32 bits to even 2048 bits). The case of 4096 bits, as mentioned earlier, although it could be practical for a conventional QR tag, is solely dedicated to MoMAtag, with HIGH-ECC.

 doi:10.20944/preprints202502.1952.v1

5. Conclusion and Future Work

PRIVocular framework explores the use of consumer smart devices for VR interaction inside an integrated software environment that allows encrypting, encoding, and visually encapsulating data. The reverse cycle of plaintext information retrieval mirrors the ciphertext display in most operation modes, except for OCR, which functions erroneously. Robust cryptographic security reliability and strong privacy preservation in air-gapped communications are practically attained for IoT devices. Thus, we achieve the lowest possible encryption-key-exchange-decryption visual latency (less than 1000 msec) and optimal user flexibility (physical distance from ciphered visual elements and human avatar user around 40cm). Here, we have presented general functionality and design components of PRIVocular. We have also presented several key novelties and contributions inside our framework functionality, such as the "expanded" QR-tag, MoMAtag, and the OCR ciphertext filtering capabilities. Finally, we conducted extensive evaluation and analysis experiments to test PRIVocular in real-case scenarios. The results demonstrate that QR-based techniques **are more efficient** than OCR-based encapsulation for on-the-fly image decryption.

PRIVocular framework could be further enhanced in various aspects. Initially, OCR with error correction detection could be introduced to increase detectability and accuracy for optical ciphertext characters. In that case, information denoising could be improved in the (visual) decryption. Furthermore, MoMAtag specifications could be altered to host more encrypted raw data so that the functionality of PRIVocular would work for even bigger key sizes. Perhaps we could even allow a grid layout of MoMAtags to be detected in bulk mode. Different encryption/decryption schemes could also be deployed, including symmetric-key cryptosystems. Finally, we did not perform investigation cryptanalysis against security-related attacks because we rely on the cryptographic strength of well-utilized asymmetric schemes; rather, we focused mainly on the visual quality of the air-gapped channel between the source and receiver.

The Metaverse's disruptive nature offers benefits, but traditional security solutions may be ineffective due to its immersiveness, hyper spatiotemporality, sustainability, interoperability, scalability, and heterogeneity. This challenges fast service authorization, compliance auditing, and accountability enforcement. The large-scale Metaverse's virtual worlds pose significant interoperability challenges. Privacy and security are key, with avatar two-factor authentication and data protection critical [30]. Conclusively, as proven above, PRIVocular is a holistic and technologically most innovative privacy-preserving VR-ready platform that can fully grant these extremely strict cybersecurity constraint requirements of the Metaverse.

Acknowledgments: The authors would like to thank Nektarios Tsoutsos and Anastasis Keliris, both members of MoMAlab (NYUAD), for their considerable efforts and helpful approach to producing this Paperwork.

Abbreviations

The following abbreviations are used in this manuscript:

OCR	Optical Character Recognition
QR	Quick Response Code
ECC	Error Correction Code

References

1. Simkin M., Schröder D., Bulling A., Fritz M. (2014) *Ubic: Bridging the Gap between Digital Cryptography and the Physical World.* In: Kutyłowski M., Vaidya J. (eds) Computer Security - ESORICS 2014. ESORICS 2014. Lecture Notes in Computer Science, vol 8712. Springer, Cham
2. Reality 51 team managed by Łukasz Rosiński, The Farm 51 Group S.A., *Report on the current state of the VR market*, 2015, http://thefarm51.com/ripress/VR_market_report_2015_The_Farm51.pdf

3. Paillier, Pascal, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, in EUROCRYPT 1999. Springer. pp. 223-238. doi:10.1007/3-540-48910-X_16
4. Holley, Rose, *"How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs"*, in April 2009, D-Lib Magazine. Retrieved 5 January 2014
5. From QR Code.com. Denso-Wave. Retrieved 23 May 2016, *"QR Code Standardization"*, http://www.qrcode.com/en/about/standards.html
6. Guruswami, V.; Sudan, M., *Improved decoding of Reed-Solomon codes and algebraic geometry codes*, in IEEE Transactions on Information Theory, 45 (6): 1757-1767, doi:10.1109/18.782097
7. R. Smith, *An Overview of the Tesseract OCR Engine*, in Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, 2007, pp. 629-633. doi: 10.1109/ICDAR.2007.4376991
8. T. Mantoro, A. M. Sobri and W. Usino, *Optical Character Recognition (OCR) Performance in Server-Based Mobile Environment*, 2013 International Conference on Advanced Computer Science Applications and Technologies, Kuching, 2013, pp. 423-428. doi: 10.1109/ACSAT.2013.89
9. Mande Shen and Hansheng Lei, *Improving OCR performance with background image elimination*, 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, 2015, pp. 1566-1570. doi: 10.1109/FSKD.2015.7382178
10. S. Ramiah, T. Y. Liong and M. Jayabalan, *Detecting text based image with optical character recognition for English translation and speech using Android*, 2015 IEEE Student Conference on Research and Development (SCOReD), Kuala Lumpur, 2015, pp. 272-277. doi: 10.1109/SCORED.2015.7449339
11. O. Mazonka, N. G. Tsoutsos and M. Maniatakos, *Cryptoleq: A Heterogeneous Abstract Machine for Encrypted and Unencrypted Computation*, in IEEE Transactions on Information Forensics and Security, vol. 11, no. 9, pp. 2123-2138, Sept. 2016. doi: 10.1109/TIFS.2016.2569062
12. *Android Developers*, https://developer.android.com/index.html
13. M. A. Sadikin and S. U. Sunaringtyas, *Implementing digital signature for the secure electronic prescription using QR-code based on Android smartphone*, in 2016 International Seminar on Application for Technology of Information and Communication (ISemantic), Semarang, 2016, pp. 306-311. doi: 10.1109/ISEMANTIC.2016.7873856
14. B. Rodrigues, A. Chaudhari and S. More, *Two factor verification using QR-code: A unique authentication system for Android smartphone users*, in 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 457-462. doi: 10.1109/IC3I.2016.7918008
15. D. Jagodić, D. Vujičić and S. Ranđić, *Android system for identification of objects based on QR code*, in 2015 23rd Telecommunications Forum Telfor (TELFOR), Belgrade, 2015, pp. 922-925. doi: 10.1109/TELFOR.2015.7377616
16. R. Divya and S. Muthukumarasamy, *An impervious QR-based visual authentication protocols to prevent black-bag cryptanalysis*, in 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2015, pp. 1-6. doi: 10.1109/ISCO.2015.7282330
17. D. Patil and S. K. Guru, *Secured authentication using challenge-response and quick-response code for Android mobiles*, in International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, 2014, pp. 1-4. doi: 10.1109/ICICES.2014.7033865
18. R. M. Bani-Hani, Y. A. Wahsheh and M. B. Al-Sarhan, *Secure QR code system*, in 2014 10th International Conference on Innovations in Information Technology (IIT), Al Ain, 2014, pp. 1-6. doi: 10.1109/INNOVATIONS.2014.6985772
19. S. Dey, S. Agarwal and A. Nath, *Confidential Encrypted Data Hiding and Retrieval Using QR Authentication System*, in 2013 International Conference on Communication Systems and Network Technologies, Gwalior, 2013, pp. 512-517. doi: 10.1109/CSNT.2013.112
20. D. T. Massandy and I. R. Munir, *Secured video streaming development on smartphones with Android platform*, in 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Bali, 2012, pp. 339-344. doi: 10.1109/TSSA.2012.6366079
21. Z. Bálint, B. Kiss, B. Magyari and K. Simon, *Augmented reality and image recognition based framework for treasure hunt games*, in 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, Subotica, 2012, pp. 147-152. doi: 10.1109/SISY.2012.6339504
22. *GitHub qrcode-terminal*, https://github.com/gtanner/qrcode-terminal
23. J. Kanai, T. A. Nartker, S. Rice and G. Nagy, *Performance metrics for document understanding systems*, in Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on, Tsukuba Science City, 1993, pp. 424-427. doi: 10.1109/ICDAR.1993.395703

 doi:10.20944/preprints202502.1952.v1

24. *Official ZXing ("Zebra Crossing") project home*, https://github.com/zxing/zxing
25. Jana, S., Narayanan, A., Shmatikov, V.: *A scanner darkly: Protecting user privacy from perceptual applications.* In: IEEE Symposium on Security and Privacy, pp. 349-363. IEEE Computer Society (2013)
26. Starnberger, G., Froihofer, L., Goeschka, K.M.: *Qr-tan: Secure mobile transaction authentication.* In: 2012 Seventh International Conference on Availability, Reliability and Security, pp. 578-583 (2009)
27. Gao, Y. (2015). Secure Key Exchange Protocol based on Virtual Proof of Reality. IACR Cryptol. ePrint Arch., 2015, 524.
28. Du, R., Lee, E., & Varshney, A. (2019). Tracking-Tolerant Visual Cryptography. 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), 902-903.
29. Park, S., & Kim, Y. (2022). A Metaverse: Taxonomy, Components, Applications, and Open Challenges. IEEE Access, 10, 4209-4251.
30. Canbay, Y., Utku, A., & Canbay, P. (2022). Privacy Concerns and Measures in Metaverse: A Review. 2022 15th International Conference on Information Security and Cryptography (ISCTURKEY), 80-85.
31. Ravi, R.V., Dutta, P.K., & Roy, S. (2023). Color Image Cryptography Using Block and Pixel-Wise Permutations with 3D Chaotic Diffusion in Metaverse. International Conferences on Artificial Intelligence and Computer Vision.
32. De Lorenzis, F., Visconti, A., Marani, M., Prifti, E., Andiloro, C., Cannavo, A., & Lamberti, F. (2023). 3DK-Reate: Create Your Own 3D Key for Distributed Authentication in the Metaverse. 2023 IEEE Gaming, Entertainment, and Media Conference (GEM), 1-6.