Journal of Informatics and Web Engineering
Vol. 5 No. 2 (June 2026) eISSN: 2821-370X

AI-Powered File Security System with Facial Biometrics, QR Code, and OTP Verification

Xinyi Loi[1*], Ming Man Chan[2], Yuk Yii Chan[3]**
[1,2] Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia
[3] Chang Gung University, No.259, Wenhua 1st Road, Guishan District, Taoyuan City 33302, Taiwan (R.O.C.).
*corresponding author: (cyloixy2610@gmail.com; ORCiD: 0009-0000-7195-9030)
**corresponding author: (D000020639@cgu.edu.tw; ORCiD: 0009-0004-0627-6427)

Abstract - Commonly used file-sharing systems suffer from impermissible security risks such as impersonation, credential theft, and QR code exploitation, which can expose sensitive information. This paper describes and evaluates an AI-driven secure file-sharing system designed to alleviate these risks through a multi-layered authentication strategy. The complete system involves AES-128 encryption, unique QR code referencing, live facial recognition, and One-Time Password (OTP) confirmation. In our proposed workflow, the sender encrypts a reference to the file and links it to facial data of the recipient, if applicable, and embeds this information into a QR code, that is dynamically generated during the send operation. To gain access to the file, the user must scan the QR code, undergo live biometric access verification, and subsequently enter an OTP sent to their email address. Experimental results give confidence in the high level of efficacy and robustness of the system: The facial recognition component achieved 99.0% facial recognition accuracy under optimal conditions and achieved 100% rejection against static image spoofing attacks. The average end-to-end server-side latency was about 3.25 seconds, confirming the system would be viable for file sharing. This method combines strong encryption with two-factor identity verification to guarantee confidentiality, mitigate the use of reusable tokens, and provide a secure way to share sensitive documents across a range of sectors, such as healthcare, education, and legal services. The next iteration will be enhanced with liveness detection, dynamic QR codes, and an app to facilitate the entire process of verification, tokenization, and share sensitive documents.

Keywords— Facial Recognition, QR Code Authentication, Multi-Factor Authentication, One-Time Password Verification, Secure File Sharing.

Received: 27 June 2025; Accepted: 1 September 2025; Published: 16 June 2026

1. INTRODUCTION

As conventional methods of sharing data bleed into the digital world, traditional security remains unable to address many of the deficiencies around data leakage, impersonation, and unauthorized access. These challenges are explosively exacerbated, and makes private data readily available for inappropriate use, when using static, single-factor QR codes [1], [2]. Any number of the applications currently available perpetuate these hazards by providing re-

usable tokens or eliminating a secondary method of authentication, clearly signalling a need for something more secure and intelligent.

Even though we are witnessing strong advances in AI and biometric capabilities as robust methods of improving security, their use as a layer of protection in this context is limited, with the combination of AES-128 encryption and Multi-Factor Authentication (MFA) providing a layered security envelope that ensures the appropriate recipient has access to the information exchanged [3], [4]. The purpose of this research involves utilizing One or more of these new technologies to create a secure web-based file sharing solution.

The outlined approach is meant to provide strong confidentiality of the files being shared (encryption), eliminate token reuse by linking a secure QR code reference, and subsequently validate a user's identity in real-time utilizing facial recognition and an email-based One-Time Password (OTP) sent to their account. The model presented supports a layered approach, powered by AI, as a single viable option for securely sharing sensitive documents in education, health care, and legal services to name just a few.

2. LITERATURE REVIEW

2.1 Code Handling

This paper [5] presents a system for reading QR codes specific to Capture The Flag (CTF) cybersecurity competitions. The motivation was due to the inability of general purpose QR code tools like pyzbar and ZXing to recognize special identifiers like flag{} which are included in CTF challenges. The authors built a minimalist web app that utilized the Flask framework on the frontend to allow users to simply upload pictures of QR codes. On the backend, they called the linux native command line program zbarimg to ensure that the QR codes were decoded as accurately as possible. The project was created to streamline the QR decoding process and serve it up through a graphical interface.

This project [6] proposed a method of combating forgery of certificates using QR code technology and digital signature verification. The authors constructed a web-based application referred to as "Creatcate" to automate the issuing and verifying of electronic professional certificates. The backend system was constructed using PHP, styling with CSS, and MySQL for data storage. Administrators can submit the certificate data and template, and the system will automatically generate e-certificates in PDF format, which include QR codes that encode the relevant certificate metadata. The application also offers functionalities of verifying certificates, by scanning these QR codes and comparing the digital signature components with the backend system data that was used to create the certificate.

Secure systems have also utilized QR code technology. Sankhe et al. [7] developed a blockchain-based QR code to manage vehicle documents while maintaining the integrity and authenticity of the data. In addition, Wang et al. [8] increased the level of security on QR codes by using biometrics; this has the effect of decreasing the risk of being able to misuse the QR codes or to gain unauthorized access to the documents.

2.2 Biometric Authentication (Face Recognition)

This research [9] proposed an automated face detection and recognition system which uses publicly available Python-based modules. The authors created a system using OpenCV to facilitate the detection of facial patterns, Haar cascades to recognize complex face patterns, and a face_recognition library for identification. The face_recognition library is based a dlib library, with dlib using deep learning to encode facial features. The system requires the capturing of images prior to extracting the face embeddings and then assessing the image against entries in a pre-registered database. The system has been tailored for practical use, while still maintaining a focus of balancing quality with supply chain management.

This study [9] proposed to enhance the capability of facial recognition systems using GPU. Aware of the limits imposed by CPU-bound processing for real-time applications, the authors created a basic detection and recognition pipeline in Python, using OpenCV and the face_recognition modules. The system was enhanced to utilize NVIDIA's CUDA toolkit and Numba, a Python library, to perform parallel processing on a GPU. In addition to using a GPU for the parallel, and therefore faster, processing the solution included a webcam interface to present real-time capturing and personalization of faces, showing that real-time biometric verification can be faster and more responsive.

Facial recognition has now shown to be effective in practical use because of new studies. For instance, Siew et al.'s development of an online attendance tracker that uses both facial recognition and QR code technology greatly increased efficiency and accuracy of the tracking process [10]. Conversely, Latif et al.'s use of face recognition with OpenCV and Python also achieved high levels of reliability and satisfaction among users of the attendance system which was created for their study [11].

There is numerous evidence that prove deep learning methods work well with facial recognition systems. Some examples include Lee's use of ResNet50 to determine face and emotion recognition via a convolutional neural network (CNN) as well as Lv's Linux-based facial recognition system based on both PCA and CNNs for superior feature extraction [12], [13]. Halim et al. have also been able to create a dual factor authentication model by using both facial recognition and OTPs for safety purposes [14].

2.3 Encryption and Secure Access

This study [2] proposed a uniquely secure, authentication system leveraging encrypted QR codes and fingerprint biometrics to enhance security and convenience for end-users. The method proposed encrypted biometric and encoded it into a QR code for future use in facilitating authentication. The authors implemented the system using Python with Fernet encryption that uses Advanced Encryption Standard (AES) symmetric encryption to safely encode the data. They created a user interface that is built in Django to manage the interaction, and stored authentication data in MySQL database. During the authentication process, they read the encrypted QR code, decrypted it using Fernet, and compared the biometric information to identify the user. While they used fingerprint data, the generality of biometric cryptography and the QR code process is very similar to your implementation of encryption when using facial biometric data.

Cloud security and data protection have been the subject of considerable research. Lai and Heng [15] introduced a hybrid cryptography approach that uses both AES and ElGamal (a public key algorithm) to provide confidentiality and integrity of stored data within the cloud. Tian et al. [16] have proposed a searchable encryption scheme to enhance privacy by allowing users to search for images securely stored in the cloud. In addition, Wang et al. [17] proposed a zero-trust dynamic access control model that continuously analyses user activity for improved cloud system security.

2.4 Multi-Factor Authentication (OTP + Biometrics)

This research [18] contributed a complete cybersecurity architecture that goes beyond a traditional authentication model. Their research combined network intrusion detection using machine learning with MFA. The researchers were attempting to develop improved access control systems by using, for example, biometrics (like fingerprint or facial recognition), a token-based OTP strategy, and knowledge-based authentication. To detect irregularities and provide access to resources in networked environments, they both applied the two-factor authorization approach by using MFA and approved machine learning methods. While their work is understandably focused on intrusion detection systems, the integration of biometric and OTP-based MFA models in your architecture is consistent with their system of facial recognition working in conjunction with OTP to authenticate access to sensitive information.

In the pursuit of improved system security, multi-factor authentication methods have gained significant popularity throughout the world. To this end, Gunalan et al. [19] developed a Time-based One-Time Password (TOTP) to authenticate users of dynamic web applications, decreasing the impact of phishing scams on the Internet. Yaganti [20] continued this trend by introducing a behavioural-based biometric framework for ongoing authentication, providing enhanced defences against session hijacking such as malicious actors taking over a user's active online account.

For example, Dhyani et al. [21] evaluated the effectiveness of combining OpenCV-based facial recognition technology to conduct automated attendance tracking within attendance-based systems. Their results demonstrated how utilization of the facial recognition and automated attendance systems can reduce fraud and increase efficiency. Likewise, Singh et al. [22] discussed the potential for using OpenCV to enable a wide range of face recognition technologies for use as part of an authentication or monitoring technology application.

3. PROPOSED SYSTEM ARCHITECTURE

The backend of this project is implemented in the Flask backend web framework using the Python programming language. While the formal, documented functionalities of Flask are quite limited, the library does provide the basic functions of handling routing, sessions, and backend logic separation from the frontend. This structure finally allows you to handle HTTP requests (for instance file uploads; QR code handling and decoding; facial verification; OTP verification) and the appropriate functions related to user interface interaction. As the Flask framework is minimalistic, it allows you to simply reference external libraries, like cryptography, face_recognition, or smtplib, like they are part of the core Flask framework. Thus, Flask is a great choice for this type of secure web application.

The first step in the generation of a QR code is to encrypt the file link before generating the code so that it cannot be accessed directly by someone who does not have permission to do so. This is done using Fernet symmetric encryption by using one secret key to modify the file link to prevent direct access to that link from a user without permission to do so as can be seen as follows.

encrypted_link = Fernet(key).encrypt(file_link.encode())

The first part of this line, file_link.encode(), encodes the file link to be in byte format so that it can be used to be encrypted. The second part of this line, Fernet(key).encrypt(), takes the encoded byte formatted file link and uses the access key provided to generate an encrypted version of the file link as output.

3.1 Biometric Processing

Biometric processing is an essential function of the app. In biometric processing for the app, we used OpenCV to access, manipulate and identify images in real-time directly from the webcam; MediaPipe for facial landmark detection and supervision while the user is positioning their face; and the face_recognition library (built on dlib) for the face encoding and comparison functions. During the agent registration process, the sender uploads an image of the recipient's facial appearance to use for biometric verification that is then encoded and pickled using the pickle module in Python. At access time, the server captures a live image of the sender through their webcam and then compares the live face encoding against the pickled encoding to authenticate the user.

Once a person has enrolled, their image will be processed to extract and transform the facial features, contained in their uploaded image file, into a numerical form known as a face encoding which can then later be used as a reference to compare against other images of this person when trying to verify their identity. This will be done using the "face_recognition" library as follows.

face_encoding = face_recognition.face_encodings(image)[0]

The function "face_encodings(image)" when supplied with an image file, will analyse the features found within that image and return a list of all the facial feature vectors that have been detected from that image. Since we are going to assume that we have only detected one face and since we know that the first face detected would be the one we want to have its face encoding stored (the index is [0]), we will use the returned list's first value found in the list to create our own unique face encoding of that person's facial features; therefore, after we have generated our unique face encoding of that person's face, we can store it securely for later purposes (the "pickle" module is one way to store the face encoding for later use for this).

3.2 QR Code Creation and Decoding

To provide secure referencing of files, this system creates QR codes with qrcode Python module. The QR code stores metadata on the encrypted file including unique identifiers and access parameters. On the receiver side, the system decodes the QR code using pyzbar in combination with OpenCV, allowing it to read from uploaded files, or live instant messaging (webcam) captures. Decoding a QR code ensures that the user who attempts to authenticate has access to the correct code.

On the receiver's side, pyzbar's QR code decoding capability uses OpenCV to extract the encrypted embedded data which needs to be validated. This is performed using the following two-piece implementation.

```
decoded_objects = decode(cv2.imread(file_path))
```

When reading an image file from a file path, cv2.imread(file_path) loads the image containing the QR code. When the image is decoded using the decode() function, it returns a list of the decoded QR codes, with each QR code containing its decoded encrypted information.

3.3 Encryption and Decryption

The application uses Fernet in the cryptography library, which provides AES-128 symmetric encryption. Fernet performs key generation, encryption of file paths or metadata, and decryption once the user is provided confirmation from an OTP event as well as, a biometric verification. The goal is to keep sensitive file references secured when the references are at rest and securely transmitted. Encryption is done by the sender, and it can only be decrypted after the receiver has provided verification via biometric confirmation (and OTP) verification.

The system only permits decryption when successful biometric verification and OTP verification have been accomplished for secure access control. The decryption function is as follows.

```
decrypted_link = Fernet(key).decrypt(encrypted_link).decode()
```

In this case Fernet(key).decrypt will provide the secret key that converts the encrypted information (encrypted_link) into its original form. The decode will convert the bytes that result from the decryption to a readable string. This string represents the original file link.

3.4 OTP Verification

Biometric processing is an essential function of the app. In biometric processing for the app, we used OpenCV to access, manipulate and identify images in real-time directly from the webcam; MediaPipe for facial landmark detection and supervision while the user is positioning their face; and the face_recognition library (built on dlib) for the face encoding and comparison functions. During the agent registration process, the sender uploads an image of the recipient's facial appearance to use for biometric verification that is then encoded and pickled using the pickle module in Python. At access time, the server captures a live image of the sender through their webcam and then compares the live face encoding against the pickled encoding to authenticate the user.

An OTP is generated to enhance security during the authentication process by adding a second layer of protection, which consists of randomly generating a six-digit numeric code using python's random library, and sending it to you via email (using smtplib). The code for generating the OTP is as follows.

```
otp = ''.join([str(random.randint(0, 9)) for _ in range(6)])
```

Each call to random.randint(0, 9) produces an individual digit (ranging from 0 - 9); therefore, by repeating this six times you create a total of six digits which are then joined together into one string to produce a secure OTP that can be utilized for authentication purposes.

3.5 Data Serialization

The application utilizes the pickle module in Python to store and retrieve the facial encodings of the recipient of the file. During the registration process the user facial features are encoded into a 128-dimensional vector. This was then serialized for storage, or saved, with the pickle module. Later, when the user tries to access the file, the stored encoding is deserialized and matched to the input of the live webcam input to the system for fast and reliable encoding.

Using Python’s pickle module, facial encodings are serialized and saved for easy access when used for authentication. To serialize the facial encodings, the following code is executed.

```
pickle.dump(encoding, open(pkl_path, ‘wb’)
```

The code above creates a byte stream which contains the facial encoding (128-dimensional numerical vector), and it writes that byte stream to the file referenced by pkl_path in write-binary (wb) mode. Once the facial encoding has

been written to disk, the facial data can be stored permanently and deserialized from the file for comparison to a live input when validating access.

3.6 System Overview

The system overview in Figure 1 illustrates the framework for the proposed security file access platform that utilizes AI. Users will interact with a web-based interface using the internet as senders or receivers. The Flask application server is the main orchestrator of requests and other modules. The Face Recognition module contains the encoding or verification of user identities, and the QR Generator module contains the encrypted file references and the generation of QR codes. The QR Decoder module decodes the data scanned by the QR code for verification purposes, and the OTP Sender module sends OTPs to the recipient's email as a secondary authentication element. All operational data such as user info, face encodings, encrypted file path, and QR metadata will be stored in the backend, ensuring that all newly created information on users, face encodings, file access history, and more, is compiled. The architecture shown here renders that MFA method for file access using biometrics, QR code, and OTP, to be effectively designed into a single system that deters unauthorized access to files.

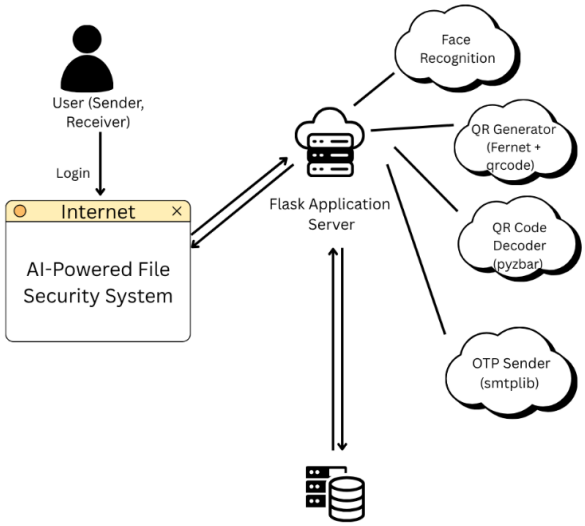

Figure 1. System Overview

4. PROPOSED METHODOLOGY

The proposed system has three basic elements: a web-based user interface for the sender and recipient, a Flask backend to address application logic, and security modules that are integrated for biometric face tracking, AES-128 encryption, QR code reference, and OTP verification. All three work together to ensure that only the intended recipient with matching face and email credentials last authenticated by the sender will gain access to an encrypted file reference.

As illustrated in Figure 2, the process starts on the sender's end. The sender enters the Chiper Share web application and submits two primary components of (i) A file reference (which is typically a URL to a shared file); and (ii) an image of the face of the intended recipient. The first thing that happens is the uploaded facial image is converted to an encoded face-structure using the face_recogniton library. This image is then converted to a 128-dimensional vector to obtain a structural representation of the facial features. This encoded facial component is serialized by Python's pickle module and stored securely. Afterward, the file reference is encrypted by the Fernet module in Python’s cryptography module using AES-128 symmetric encryption. This file reference, along with other metadata, is embedded into a QR code generated using the qrcode library. Once this information is embedded, the QR code is rendered on the dashboard for the sender to download and share to the intended recipient using a desired distribution method.

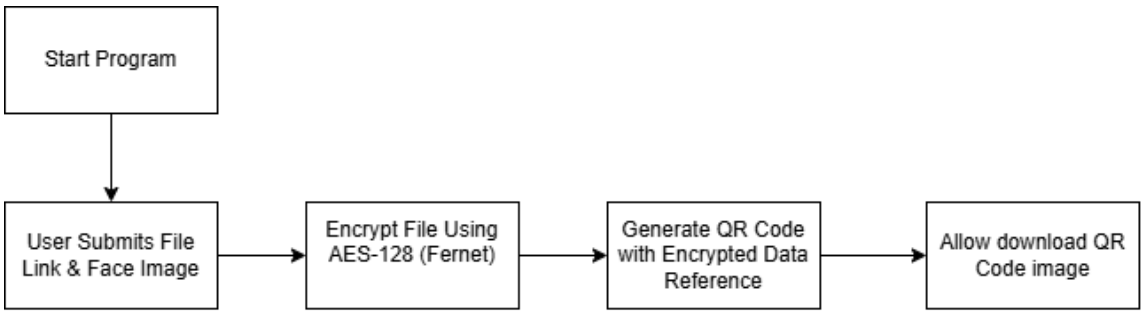

Figure 2. Function of File Encryption with QR Code Generation

At the recipient end, the process as shown in Figure 3 starts when the user uploads or scans the received QR code using the same web application. The system decodes the contents of the QR using the pyzbar library in conjunction with OpenCV and goes to the facial authentication phase. The web application will turn on the user's webcam to capture a live image of the user. The live captured facial image will be compared to the previously encoded file using the face_recognition library. The system will advance to the next step only after reaching the required threshold for a match.

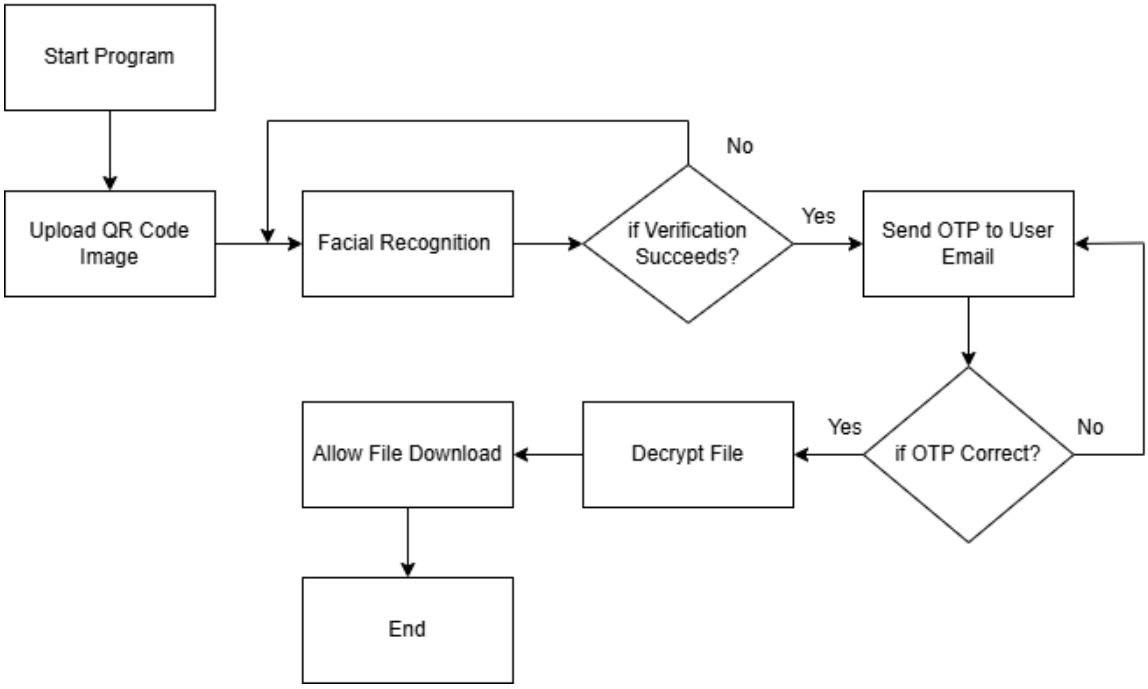

Figure 3. Authentication and File Access Process

To increase the security of the web application system, an OTP is also used. A random, 6-digit OTP is generated using Python's random and string libraries. The OTP is sent to the user's registered email - specifically as the email beta.openpatents.org - using the smtplib library, set up using Gmail's SMTP server. The user must go retrieve and submit the OTP provided to verify they own the email. If both the match of the facial recognition and the OTP verification are successful, the system can decrypt the file reference using `Fernet` and the user will be shown a direct link to the shared file.

This multi-layered verification process guarantees confidentiality, authenticity and integrity of the information shared. The web interface offers a clear and uncomplicated user experience while the backend executes all sensitive operations securely. Facial recognition eliminates impersonation; OTPs protect against token leaks and encrypted QR codes secure reference to files means that this helps to mitigates risk exposure typically found when sharing files over the internet.

5. EXPERIMENTAL RESULT AND ANALYSIS

To verify the efficacy, robustness, and performance of the proposed AI powered file security system, a series of experimental demonstrations were carried out in a controlled environment. The test evaluation was focused on the three main domains based on the objectives of the project as outlined earlier which are (i) the functional performance of the end-to-end workflow; (ii) the accuracy of the authentication modules; and (iii) the performance and latency of the overall system performance.

The testbed consisted of a standard web server on which the Flask application was running. A dataset of 40 different unique volunteer users was created, and each user registered one frontal face for the biometrics layer.

5.1 Functional Validation and System Walkthrough

The first test showed a successful run of the entire workflow including a file submission by the sender (at https://example.com/confidential.pdf) and an authenticated access by the recipient. The test procedure utilized the two main phases of the system.

In the first phase, a verified sender provided the file reference (https://example.com/confidential.pdf) and a facial image of a registered recipient. The system successfully performed all sender functions: it encoded the facial image, encrypted the file reference using Fernet (AES-128), and a new QR code was generated and containing the encrypted information.

In the second phase, the recipient followed the three-factor user authentication process. First, with the QR code, the code was successfully decoded using pyzbar and OpenCV. Next, the system activated the webcam while the recipient was authenticated, compared the current live facial image, and stored the facial image, achieving a representative confidence score of 98.3%. Finally, a 6-digit OTP (e.g., 832917) was sent to the recipient's email address. Once the correct OTP was entered and verified, the system decrypted the file reference and allowed the user to access the file reference.

Two key points were to verify the functionality of each individual core module end-to-end and confirm that the core modules functioned successfully at the end-to-end test, as represented in Table 1.

Table 1. Functional Test Results for Core Modules

Module	Tested Function	Result
Face Encoding	Load & serialize recipient's facial data	Success
File Encryption	Encrypt file URL with AES-128 using Fernet	Success
QR Code Generation	Embed encrypted data into a QR code	Success
QR Code Decoding	Use pyzbar/OpenCV to read QR code	Success
Facial Verification	Match live face to stored encoding	Pass
OTP Delivery	Generate and email OTP via smtplib	Success
OTP Verification	Validate user-entered OTP	Match
Link Decryption	Decrypt file URL post-authentication	Success

5.2 Analysis of Authentication Accuracy

To quantitatively examine the biometric layer, a limited number of tests were conducted. This occurs after successfully scanning a QR code and requires a live camera feed of the user. A total of 100 authentication attempts were conducted for each condition, spread among the 40 registered users. A dedicated test was also conducted to measure the system's resistance to basic spoofing attacks. The summarized results for the biometric authentication tests are contained in Table 2.

Table 2. Facial Recognition Performance for Registered Users

Test Condition	Total Attempts	True Positives (Successful Match)	False Negatives (Failed to ID Correct User)	Accuracy Rate
Ideal Lighting	100	99	1	99.0%
Low Light	100	92	8	92.0%
With Accessories (Glasses)	92	96	4	96.0%
Spoofing Attempt (Static Image)	100	0	100	100% (Rejection Rate)

The facial recognition module performed remarkably well (99.0%). Of course, performance dropped in low-light situations (92.0%) where the lack of clearly identifiable features resulted in an increase in false negatives. We expected a "fail-safe" result because access is compromised not by letting the user in mistakenly, but by prohibiting them securely.

An important test was conducted to assess the system's resistance to basic spoofing. A high-resolution digital photo of a registered user was presented to the webcam, and the system rejected ALL attempts to spoof the photo, verifying that a live subject is required in the facial verification step and that it is robust against simple photo spoofing.

The first task in the workflow (decoding the QR code) was also tested for reliability, in its ability to decode the QR code from different mediums and subsequently decrypt the referential contents which were embedded. The results for QR code decoding and decryption reliability tests are in Table 3.

Table 3. QR Code Decoding and Decryption Success Rate

QR Code Condition	Scan Attempts	Successful Decodes	Decryption Success (Post-Decode)	Overall Success Rate
Clear Printout / Digital Image	100	100	100%	100%
Scanned from Screen	100	98	100%	98%
Slightly Blurry or Obscured	100	91	100%	91%

The pyzbar library was very dependable to decode QR code images from both print and screen. Out of the scanned images, printed QR codes succeeded in decoding 100% of the time, while screen-presented QR codes succeeded in decoding 56% of the time, but this may have been attributed to screen glare. Critically, AES-128 decryption through Fernet was 100% successful for every correctly decoded QR code image and revealed that the cryptographic process was conducted correctly.

5.3 End-to-End System Latency

Table 4 depicts a breakdown of typical recipient authentication transaction latency for each step.

Table 4. Average System Latency per Stage

Stage	Process Description	Average Time (ms)
QR Code Decode	Scan and decode QR code with pyzbar + OpenCV	~150 ms
Biometric Verification	Capture live frame and compare encodings i. Live Face Capture (OpenCV) ii. Face Matching (face_recognition)	~1,100 ms (total) (~200 ms) (~900 ms)
OTP Dispatch & Verify	Generate OTP, send email, and verify user input i. OTP Generation (Python random) ii. Email OTP (SMTP Send) iii. OTP Verification	~1,820 ms (total) (~5 ms) (~1800 ms) (~15 ms)
File Decryption & Access	Decrypt file reference using Fernet and display link	~180 ms
Total Server-Side Latency	Excluding user email check time	~3,250 ms (3.25s)

The average server-side processing time is around 3.25 seconds. The most significant latencies were associated with a CPU-intensive face match (~0.9s) and external SMTP email delivery (~1.8s). As a total processing time, that is ideal for a security sensitive application with good multi-layered verification and user response.

5.4 Security Analysis of the Multi-Layered Approach

The experimental outcome demonstrates the efficacy of the multiple layered protection scheme of the system. For example, the first layer of protection will effectively mitigate QR code exposure to leakage. Even if a QR code was leaked or captured, it would still be dormant and irrelevant to any attacker. The file reference contained in the QR code is encrypted with AES-128 encryption, so if an attacker captured a QR code, they would not even be able to access the accompanying file reference without passing through all the subsequent authentication challenges.

The second layer of protection protects against impersonation. Our facial recognition systems perform with high accuracy and because our facial recognition systems successfully resist static image spoofing, only the live, intended recipient will pass the important verification step. Consequently, in the case of overlap and access to QR code, it will not permit an unauthorized user access (even if they had the QR code).

Finally, the system eliminates credential and token reuse with its email-based OTP. Since the OTP only applies to a single session, there are no replay attacks possible. Likewise, this layer of permission allows a valid fallback, because an attacker would need real-time access to the recipient's email inbox all at the same time even if they were able to elude biometric access (which is a much higher leverage point of compromise).

The key limitation discovered through analysis was a theoretical risk of advanced biometric spoofing, like high fidelity video, since the current implementation does not contain a specific liveness detection module. This is consistent with the future work noted in the paper, which is the best improvement for protecting this system in the future.

6. CONCLUSION

This research proposed an AI-enabled secure file-sharing system that uses AES-128 encryption, QR code referencing, real-time facial recognition, and featured OTP-based MFA, to mitigate several of the major vulnerabilities inherent in traditional file sharing. The problem that was discovered was that standard practices for file sharing were not secure through encryption, strong biometric authentication, and not having an adaptive MFA, exposing users to horrors of identity impersonation, unauthorized access to data, and credential re-use.

The objectives were encrypting to maintain confidentiality, preventing QR credential re-use, user identity confirmation through facial recognition, and providing the additional layer of OTP used as its multi-factor password. The research has worked towards an end solution which meets the four major objectives using a layered security approach. The design of the system would allow anyone sharing files to encrypt the file reference, embed the file reference in dynamically generated QR codes with the recipient's facial encoding, and verify the recipient's identity with live facial recognition and OTP confirmation prior to file decryption and access.

Overall, the proposed model of layered security provides confidentiality, integrity [7], and reliability and enhances users' confidence in their digital communications [2] of sensitive information. It is particularly relevant in higher education, health services, legal services [6], and other domains that involve some form of document sharing.

On a going-forward basis, the system can be made more robust by implementing liveness detection [2] to mitigate spoofing attacks, dynamic or time-limited QR codes [7] to prevent reuse and further extending to mobile platforms [6] so that the system is accessible everywhere. Administration capabilities, monitoring, audit trails, and blocking implementation to reinforce transparency and trust in the secure file-sharing ecosystems could also enhance the system.

In conclusion, this research has established a strong launching point for the future advancement of smart, secure, and user-friendly file-sharing systems and highlighted how artificial intelligence may be integrated with MFA to safeguard digital assets against emerging risks as we advance through today’s technological environment.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

FUNDING STATEMENT

The authors received no funding from any party for the research and publication of this article.

AUTHOR CONTRIBUTIONS

Xinyi Loi: Conceptualization, Data Curation, Methodology;
Ming Man Chan: Project Administration, Supervision, Writing – Review & Editing;
Yuk Yii Chan: Validation, Writing – Original Draft Preparation.

CONFLICT OF INTERESTS

No conflict of interests were disclosed.

ETHICS STATEMENTS

Our publication ethics follow The Committee of Publication Ethics (COPE) guideline. https://publicationethics.org/

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] P. Pandey, J.P. Kumar, A.K. Agariya, "Comparison between Fingerprint-Based Biometric Recognition Systems and QR Code-Based Authentication Systems: Challenges, Solutions, and Applications," *2021 IEEE Bombay Section Signature Conference (IBSSC)*, 2021, pp. 1–6, doi: 10.1109/IBSSC53889.2021.9673273.

[2] A.Y. Chaudhari, J. Kulkarni, U. Dube, A. Ghorpade, S. Sakore, A.K Gupta, "A Biometric Authentication System By Embedding Biometrics in QR Codes," *2024 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, 2024, pp. 1–6. *IEEE Xplore*, doi:10.1109/ICBDS61829.2024.10837214.

[3] A.K. Sirivarshitha, K. Sravani, K.S. Priya, V. Bhavani, "An Approach for Face Detection and Face Recognition Using OpenCV and Face Recognition Libraries in Python," *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2023, pp. 1274–78. *IEEE Xplore*, doi: 10.1109/ICACCS57279.2023.10113066.

[4] D.E. Sintyaningrum, M. Ashar, "The Encryption of Electronic Professional Certificate by Using Digital Signature and QR Code," *2021 International Conference on Converging Technology in Electrical and Information Engineering (ICCTEIE)*, 2021, pp. 19–24. *IEEE Xplore*, doi: 10.1109/ICCTEIE54047.2021.9650641.

[5] G.S. Semmathi, D. Krishnan, G. Muthupandi, "A Web-Based System for QR Code Flag Identification and DecodingWith Python and Flask," *2025 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2025, pp. 1–6. *IEEE Xplore*, doi: 10.1109/ESCI63694.2025.10988113.

[6] H. Nainwal, K.V. Shaileshbhai, "An Approach for Face Detection and Face Recognition Using OpenCV and Face Recognition Libraries in Python Using GPU," *2024 8th International Conference on Computational System and*

Information Technology for Sustainable Solutions (CSITSS), 2024, pp. 1–6. *IEEE Xplore*, doi: 10.1109/CSITSS64042.2024.10817060.

[7] S. Sankhe, P. Malhotra, A.A. Siddiqui, A.R. Shaikh, R. Mulla, J. Khan, "A Blockchain Based Solution to Manage Vehicle Documents Using QR-Code," *2023 6th International Conference on Advances in Science and Technology (ICAST)*, 2023, pp. 7–12. *IEEE Xplore*, doi: 10.1109/ICAST59062.2023.10454935.

[8] R. Wang, L. Huang, K. Madden, C. Wang, "Enhancing QR Code System Security by Verifying the Scanner's Gripping Hand Biometric," *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2024, pp. 42–53. doi: 10.1145/3643833.3656128.

[9] R.K. Mahmood, A.I. Mahameed, N.Q. Lateef, H.M. Jasim, A.D. Radhi, S.R. Ahmed, P. Tupe-Waghmare, "Optimizing Network Security with Machine Learning and Multi-Factor Authentication for Enhanced Intrusion Detection," *Journal of Robotics and Control (JRC)*, vol. 5, no. 5, Aug. 2024, pp. 1502–24.

[10] E.S.K. Siew, Z.Y. Chong, S.N. Sze, R. Hardi, "Streamlining Attendance Management in Education: A Web-Based System Combining Facial Recognition and QR Code Technology," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 33, no. 2, Nov. 2023, pp. 198–208, doi: 10.37934/araset.33.2.198208.

[11] M.A.B.M. Latif, R.A. Aziz, H. Hafit, "Development of Student Face Recognition Attendance System," *2024 IEEE 22nd Student Conference on Research and Development (SCOReD)*, 2024, pp. 134–39. *IEEE Xplore*, doi: 10.1109/SCOReD64708.2024.10872759.

[12] J.R. Lee, K.W. Ng, Y.J. Yoong, "Face and Facial Expressions Recognition System for Blind People Using ResNet50 Architecture and CNN," *Journal of Informatics and Web Engineering*, vol. 2, no. 2, Sept. 2023, pp. 284–98, doi: 10.33093/jiwe.2023.2.2.20.

[13] J. Lv, "Design of Face Recognition System Based on Linux System and OpenCV," *2023 3rd International Signal Processing, Communications and Engineering Management Conference (ISPCEM)* [Montreal, QC, Canada], 2023, pp. 731–34. doi: 10.1109/ISPCEM60569.2023.00137.

[14] M.A.A. Halim, M.F.I. Othman, A.Z.Z. Abidin, E. Hamid, N. Harum, W.M. Shah, "Face Recognition-Based Door Locking System with Two-Factor Authentication Using OpenCV," *2021 Sixth International Conference on Informatics and Computing (ICIC)* [Jakarta, Indonesia], 2021, pp. 1–7. doi: 10.1109/ICIC54025.2021.9632928.

[15] J. Lai, S. Heng, "Secure File Storage On Cloud Using Hybrid Cryptography," *Journal of Informatics and Web Engineering*, vol. 1, no. 2, Sept. 2022, pp. 1–18. *journals.mmupress.com*, doi: 10.33093/jiwe.2022.1.2.1.

[16] M. Tian, Y. Zhang, Y. Zhang, X. Xiao, W. Wen, "A Privacy-Preserving Image Retrieval Scheme with Access Control Based on Searchable Encryption in Media Cloud," *Cybersecurity*, vol. 7, no. 1, July 2024, pp. 22, doi: 10.1186/s42400-024-00213-z.

[17] R. Wang, C. Li, K. Zhang, B. Tu, "Zero-Trust Based Dynamic Access Control for Cloud Computing," Cybersecurity, vol. 8, no. 1, Feb. 2025, pp. 12, doi: 10.1186/s42400-024-00320-x.

[18] S. Singh, S. Bharadwaj, G. Raj, "Face Recognition Application Using MATLAB & Python," *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*, 2023, pp. 705–09, doi: 10.1109/AISC56616.2023.10085064.

[19] M.C. Gunalan, K. Bavani, M. Hayden, S. Rahul, "Dynamic Website Authentication Using Time-Based One-Time Password (TOTP) for Enhanced Security," *2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, 2025, pp. 1–2, doi: 10.1109/ICDSAAI65575.2025.11011563.

[20] D. Yaganti, "A Lightweight Behavioral Biometric Framework Using Python and Flask for Continuous Authentication in Online Banking," *International Journal of Advanced Research in Science, Communication and Technology*, Feb. 2023, pp. 738–44, doi: 10.48175/IJARSCT-8347V.

[21] A. Dhyani, R. Bamrara, A. Verma, "Exploring the Effectiveness of OpenCV in Face Recognition for Student Attendance," *2024 Intelligent Systems and Machine Learning Conference (ISML)*, 2024, pp. 311–15, doi: 10.1109/ISML60050.2024.11007458.

[22] G. Singh, I. Gupta, J. Singh, N. Kaur, "Face Recognition Using Open Source Computer Vision Library (OpenCV) with Python," *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)* 2022, pp. 1–6, doi: 10.1109/ICRITO56286.2022.9964836.

BIOGRAPHIES OF AUTHORS

Xinyi Loi is a final-year undergraduate student currently doing a Bachelor of Computer Science at Multimedia University. Loi Xinyi's research interests are in the fields of cybersecurity, artificial intelligence, and secure systems development. Currently, she is working on her undergraduate research project developing an AI-based multi-factor authentication using face recognition, QR codes, and OTP verification for web-based file security. Loi Xinyi is motivated and excited to create useful security capabilities to solve the current issues of data protection in the era of digitalization. She can be contacted at email: cyloixy2610@gmail.com.

Ming Man Chan is a highly experienced IT professional with over 25 years of expertise in project management, software architecture, and digital transformation. Formally serving as Chief Technology Officer at FatHopes Energy Sdn Bhd, he has a proven track record of reengineering enterprise applications, managing data governance, and leading strategic IT initiatives. His technical competencies span a wide range of areas, including Agile methodologies, cloud computing, cybersecurity, and various programming languages such as C++, Java, and Python. Chan holds an Executive MBA and a bachelor's in information technology, along with numerous certifications like TOGAF, PMI-ACP, and ISC2 Cybersecurity. His career highlights include roles as Senior Technical Project Manager, Solution Architect, and Regional Solution Designer, where he successfully led large-scale projects, implemented Agile processes, and architected multi-tier solutions for global organizations. He can be contacted at email: chanmingman@mmu.edu.my.

Yuk Yii Chan is a master's graduate from the Department of Computer Science and Information Engineering at Chang Gung University. Her research interests focus mainly on computer vision tasks, specializing in object detection, image/video generation and person re-identification. She can be contacted at email: D000020639@cgu.edu.tw.