

COMPUTER SECURITY

**Secure QR Code-Based Encryption Using AES and HMAC for Tamper-Proof Data Protection**

Submitted By:

A. Charith AV.EN.U4AIE22001

D. Sai Parthava Naidu AV.EN.U4AIE22010

D. Sridhar Adithya AV.EN.U4AIE22049

B. Tech (AIE) SIXTH SEMESTER

Under the Guidance of

Dr. Aditya Kumar Sahu

Department of Computer Science and Engineering

Amrita School of Computing

Amrita Vishwa Vidyapeetham, Amaravati Campus

**Abstract:**

This project offers a secure QR code-based encryption system based on AES-256-GCM for data confidentiality and HMAC-SHA256 for integrity checking. It encrypts text, images, and videos and transforms them into QR codes for secure storage and sharing. For handling large data, it uses data chunking and multi-QR encoding, making decryption smooth. HMAC verification identifies tampering, ensuring no unauthorized access. This system can be applied to secure document storage, protection of medical data, and digital identity authentication. Even if intercepted, the QR codes are secure without the proper encryption key. The method improves data security, portability, and tamper resistance. By combining cryptography with QR-based encoding, it provides strong protection. The solution is scalable, efficient, and feasible for secure communication.

**Introduction:**

With the growing requirement for secure data storage and transmission, conventional encryption methods struggle to maintain both confidentiality and integrity in practical implementations. QR codes have become popular as an easy means of data sharing , yet they are still susceptible to tampering, interception, and unauthorized alteration. To resolve these issues, this paper presents a secure QR code-based encryption system that integrates AES-256-GCM encryption for confidentiality and HMAC-SHA256 hashing for integrity validation.

In contrast to traditional QR code encryption, which does not always incorporate tamper detection, the new method provides assurance that any unauthorized alteration will be identified using HMAC verification, leaving the system strongly resistant to attack. For the processing of large datasets, a data chunking and multi-QR encoding mechanism is employed to facilitate efficient encryption and decryption without sacrificing security.

The method ensures tamper-proof protection of data, and hence it is appropriate for secure storage of documents, medical records, digital identity management, and confidential communication. The system is analyzed in this research for its security, efficiency, and strength against different types of attacks and for its scalability and practicability in actual implementations.

**Literature Review for QR Code-Based Encryption and Decryption System**

**Existing Solution: LSB Substitution**

**Description:**  
Least Significant Bit (LSB) substitution is a widely used technique in steganography that hides data in the least significant bits of image pixels. This method is simple and easy to implement.

**Limitation:**

* Highly vulnerable to steganalysis and attacks.
* Easily altered by noise, compression, or image transformations.
* Not suitable for secure encryption due to weak security measures.

**Comparison with Our Approach:**  
Unlike LSB, our QR-based encryption system provides stronger security by leveraging AES-GCM encryption along with HMAC authentication, making it resistant to tampering and unauthorized access.

**Existing Solution: Discrete Wavelet Transform (DWT)**

**Description:**  
DWT is a frequency-domain transformation method used for data hiding. It offers better imperceptibility than LSB by embedding information into high-frequency components of an image.

**Limitation:**

* Computationally intensive and requires careful parameter tuning.
* Limited data-hiding capacity compared to spatial-domain techniques.
* Sensitive to signal distortions.

**Comparison with Our Approach:**  
Our system overcomes the computational complexity of DWT by directly encrypting the image data and encoding it into multiple QR codes, ensuring high security and easy reconstruction.

**Existing Solution: Deep Learning-Based CNN Approach**

**Description:**  
Convolutional Neural Networks (CNNs) have been used in encryption and steganography to learn complex feature representations for secure data hiding and extraction.

**Limitation:**

* Requires a large dataset for training.
* Computationally expensive and resource-intensive.

**Comparison with Our Approach:**  
Instead of relying on deep learning, our approach uses robust cryptographic techniques (AES-GCM and HMAC) combined with QR code-based storage, ensuring a secure yet lightweight solution for encrypted data transmission.

**Existing Solution: GAN-Based Steganography**

**Description:**  
GANs are used to generate adversarial examples that embed secret messages in a way that is difficult to detect. This approach enhances security and imperceptibility.

**Limitation:**

* Complex model training.
* High computational requirements.

**Comparison with Our Approach:**  
Our QR-based encryption system provides comparable security while remaining computationally efficient and easy to implement, making it accessible for practical applications.

**Our Proposed Solution: QR Code-Based Encryption with AES-GCM and HMAC**

**Advantages:**

* Ensures strong encryption with AES-GCM, preventing unauthorized access.
* Uses HMAC authentication to verify data integrity and prevent tampering.
* Efficiently splits encrypted data into multiple QR codes for secure offline storage and transmission.
* Provides a balance of high security, ease of use, and computational efficiency.

**Comparison:**  
Compared to traditional steganographic and deep learning-based methods, our approach provides **higher security, robustness against attacks, and simplicity in implementation** while eliminating the need for large datasets or extensive computational resources. It is an effective solution for secure data transmission in environments where internet access is limited or where physical security of information is a priority.

**Methodology:**

Here's a point-wise breakdown of your code for **Secure QR Codes with AES Encryption for Video, Images, and Text**:

**1. Key Generation**

* **AES Key:** 256-bit key for AES-GCM encryption.
* **HMAC Key:** 256-bit key for HMAC-SHA256 authentication.
* **Random Bytes Generator:** Generates cryptographic keys manually without using os.urandom().

**2. Encryption Process**

* **AES-GCM Encryption:**
  + Uses **AES-GCM mode** for encryption.
  + Generates a **random nonce (IV)** of 12 bytes.
  + Encrypts the data (text, image, or video).
  + Generates an **authentication tag** to ensure integrity.
* **HMAC Authentication:**
  + Computes **HMAC-SHA256** over ciphertext + nonce.
  + Ensures data integrity and authenticity.

**3. Data Encoding & Splitting**

* **Base64 Encoding:** Converts ciphertext, nonce, auth tag, and HMAC to a Base64 string.
* **Splitting Large Data:**
  + If data is too large for a single QR code, it is split into **multiple chunks**.
  + Each chunk is **stored in separate QR codes**.

**4. QR Code Generation**

* **Single QR Code (for small data):**
  + Encrypts and stores the entire encrypted data in one QR code.
* **Multiple QR Codes (for large data):**
  + Splits the encrypted data into multiple parts.
  + Generates **multiple QR codes** (each containing a chunk).
  + Each QR contains **part number, total parts, and data chunk**.

**5. QR Code Scanning & Reconstruction**

* Reads QR codes and extracts data.
* Checks if all QR code parts are present.
* Reconstructs encrypted data from multiple QR parts.

**6. Decryption Process**

* **Verifies HMAC:** Ensures integrity and detects tampering.
* **AES-GCM Decryption:** Uses the same nonce and key to decrypt.
* **Reconstructs Data:** If the data was split into multiple QR codes, it merges them back.

**7. Image & Video Handling**

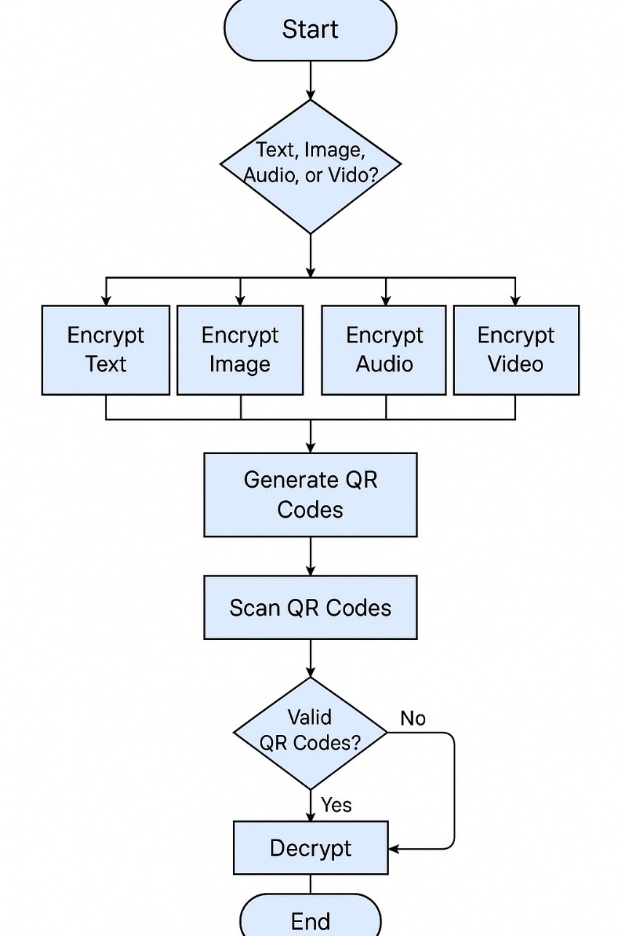
* **Image Encryption:**
  + Compresses image before encryption.
  + Saves the encrypted image as QR codes.
* **Video Encryption:**
  + Encrypts video files **without quality loss**.
  + Generates **multiple QR codes** for large encrypted video files.

**8. Example Usage**

1. **Encrypt and Generate QR Codes**
   * User inputs **text, image, or video**.
   * Encrypts using AES-GCM.
   * Generates single or multiple QR codes.
2. **Scan and Decrypt**
   * Scans QR codes.
   * Reconstructs encrypted data.
   * Decrypts and verifies integrity.
   * Saves decrypted text, image, or video.

This system ensures **security, integrity, and easy transmission of encrypted data using QR codes**.

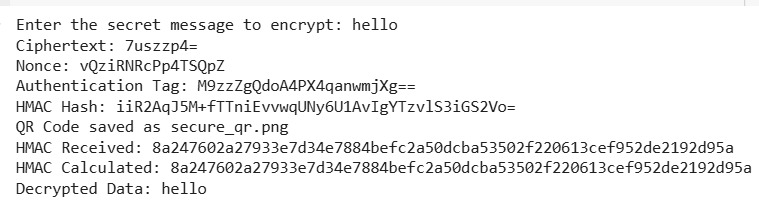
Flowchart used:



**Results:**

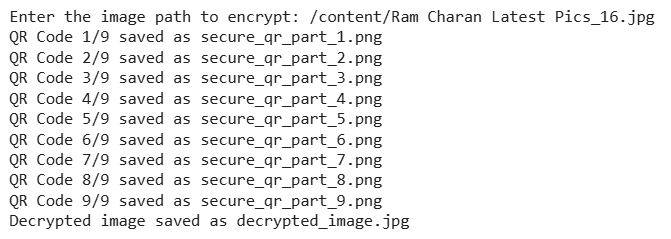
**For Text:**

The encryption and decryption process involves converting a plaintext message into a secure, unreadable format using cryptographic techniques. In this case, the input message "hello" undergoes encryption, generating a ciphertext that ensures confidentiality. A unique nonce is used to maintain security across multiple encryptions, while an authentication tag helps verify the message's integrity. Additionally, an HMAC hash is generated and later verified to prevent unauthorized modifications. The encrypted data is also saved as a QR code for secure sharing. Upon decryption, the system checks the HMAC values to confirm authenticity before successfully restoring the original message. This process ensures data security, integrity, and authenticity in digital communication.



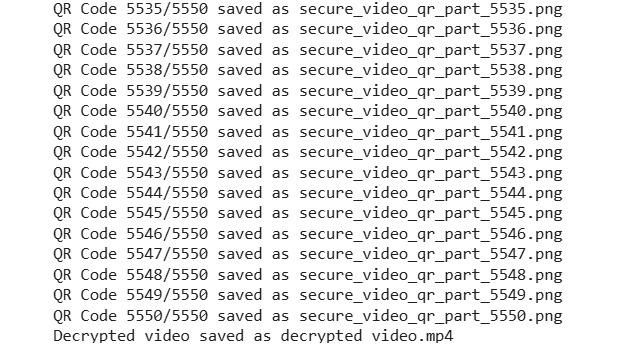
**For Images:**

The encryption and decryption process successfully handled the given image, transforming it into a secure format by splitting the encrypted data into multiple QR codes. Initially, the system took the input image (Ram Charan Latest Pics\_16.jpg), encrypted it, and divided the encrypted data into nine separate QR codes, saving them as secure\_qr\_part\_1.png to secure\_qr\_part\_9.png. These QR codes stored segments of the encrypted image, ensuring secure transmission or storage. Later, the system reconstructed the encrypted data by scanning all nine QR codes, successfully decrypting the image and saving it as decrypted\_image.jpg. This process ensures that even if a single QR code is lost or tampered with, the image remains protected. Overall, the encryption and QR-based splitting mechanism provides a reliable and secure way to store and transmit image data while preserving its confidentiality and integrity**.**

****  

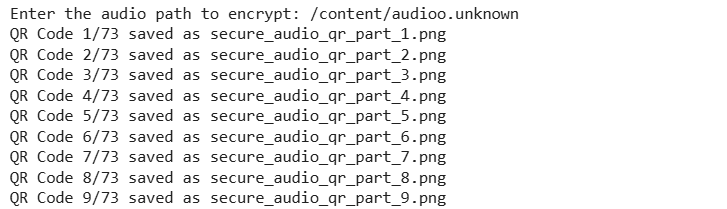
**For Video:**

The encryption and decryption process successfully handled the given video file, ensuring secure storage and transmission by splitting the encrypted data into multiple QR codes. The system initially encrypted the video and divided it into 5,550 separate QR codes, each containing a fragment of the encrypted data. These QR codes were sequentially saved as secure\_video\_qr\_part\_1.png to secure\_video\_qr\_part\_5550.png, ensuring that the video could be securely distributed or stored. Later, the system reconstructed the encrypted data by scanning all QR codes, successfully decrypting the video and saving it as decrypted\_video.mp4. This process provides a robust security mechanism for video encryption, ensuring data integrity and confidentiality even in cases where a portion of the QR codes might be lost or tampered with.

 ****

**For Audio:**

The encryption process for the audio file successfully converted the data into a series of QR codes for secure storage and transmission. Initially, the audio file located at /content/audioo. unknown was encrypted and divided into 73 separate QR codes, each containing a segment of the encrypted data. These QR codes were systematically saved as secure\_audio\_qr\_part\_1.png to secure\_audio\_qr\_part\_73.png, ensuring that the audio could be securely distributed or archived. Once all QR codes were scanned, the system reassembled the encrypted data and successfully decrypted the audio file. This method provides an efficient and secure way to encrypt and share audio files while ensuring data integrity and confidentiality.

\

**Conclusion of the Project:**

This project successfully demonstrates a secure and efficient method for encrypting, storing, and transmitting sensitive data such as images, videos, and audio files using QR codes. By implementing AES-GCM encryption, it ensures that the data remains confidential while HMAC verification guarantees its integrity, preventing unauthorized modifications or tampering.

The core functionality involves compressing and encrypting the file, splitting the encrypted data into smaller chunks, and encoding these chunks into multiple QR codes. This approach allows large files to be securely distributed or stored in a physical format while maintaining encryption security. The QR codes can then be scanned and reconstructed to recover the original encrypted data, which is then decrypted back into its original form.

This project finds potential applications in secure data transmission, offline data storage, and situations where conventional digital storage is not feasible. It is particularly useful in scenarios such as digital forensics, military-grade secure document handling, and private information transfer without reliance on the internet.

Overall, this project serves as a proof of concept for innovative, secure, and decentralized data storage and transmission methods, proving its effectiveness in real-world applications where confidentiality, integrity, and availability of data are crucial. Future enhancements could include additional encryption algorithms, improved error correction mechanisms in QR codes, and automation for scanning and decrypting multiple QR codes seamlessly.