



**A CAPSTONE PROJECT REPORT**

**PROJECT TITLE**

# "Secure QR Code Detection and Decoding Using Cryptographic Algorithms"

**SUBMITTED BY**

**N.SAI SARANYA (192210441)**

**V. KAVYANJALI (192210218)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE ENGINEERING**

## COURSE CODE / NAME

**CSA5193/CRYPTOGRAPHY AND NETWORK SECURITY FOR SECURE SYSTEM DESIGN**

## UNDER THE SUPERVISION OF

**MRS. J. ALPHONSA**

# TABLE OF CONTENT

# Abstract

Secure QR Code Detection and Decoding Using Cryptographic Algorithms Quick Response (QR) codes are widely used for their convenience in encoding and sharing data. However, their increasing adoption has also made them a target for security threats such as tampering, phishing, and unauthorized access. This paper presents a secure framework for QR code detection and decoding that integrates cryptographic algorithms to ensure data integrity, authenticity, and confidentiality. The proposed system employs advanced image processing techniques to detect QR codes accurately under varying environmental conditions. Once detected, the encoded data is decrypted using cryptographic keys, ensuring that only authorized users can access the information. Additionally, digital signatures are embedded into QR codes to verify the authenticity of the data and detect any tampering attempts. The system is evaluated for robustness, computational efficiency, and security against common threats. Experimental results demonstrate that the proposed approach enhances the reliability and security of QR code-based communication systems without significantly impacting performance. This work provides a practical solution for secure data sharing in applications such as payment systems, identity verification, and secure document exchange.

# Problem Statement

In today's digital era, QR codes are extensively used in diverse applications such as mobile payments, document verification, and secure data sharing. However, QR codes are inherently vulnerable to security risks, including unauthorized access, data tampering, and malicious modifications, which can lead to breaches of sensitive information or fraudulent activities. Existing QR code systems primarily focus on data encoding and error correction but lack robust mechanisms to ensure data confidentiality, integrity, and authenticity.

This project seeks to address these security gaps by designing a system that combines secure QR code detection and decoding with cryptographic algorithms. The solution will involve encrypting data before QR code generation to ensure confidentiality, applying cryptographic hashes or digital signatures for integrity and authenticity validation, and detecting tampering attempts during decoding. The system must also handle environmental challenges such as noise, distortion, and partial visibility, ensuring reliable performance in real-world scenarios. This approach aims to enhance the trustworthiness and usability of QR codes across security-sensitive applications while maintaining compatibility with existing scanning and generation technologies.

# Introduction

Quick Response (QR) codes have become a ubiquitous medium for sharing information in today's digital landscape. These two-dimensional barcodes are widely used due to their ease of use, high storage capacity, and rapid readability, making them ideal for applications ranging from payment systems to marketing campaigns and authentication processes. QR codes bridge the gap between the physical and digital worlds, allowing users to quickly access URLs, contact information, and multimedia content. However, with their growing adoption comes an equally increasing concern about the security and authenticity of the data they encode.

**1. Challenges of QR Code Security**

**Lack of Built-in Encryption:** Standard QR codes do not inherently include mechanisms for data encryption, making the encoded content vulnerable to interception and manipulation.

**Phishing Attacks:** Malicious actors can embed deceptive URLs leading to phishing websites, compromising user credentials and sensitive data.

**Data Tampering:** Without cryptographic integrity checks, QR codes can be altered to mislead users or redirect them to harmful content.

**Counterfeit QR Codes:** Attackers can create fake QR codes that mimic legitimate ones, undermining trust in the system.

**Privacy Concerns:** Sensitive information encoded in plaintext QR codes can be accessed by anyone, posing a risk of unauthorized disclosure.

## 2. Cryptographic Techniques for Secure QR Codes

**Symmetric Encryption:** Algorithms like AES can encrypt the QR code data, ensuring only authorized users with the correct key can decrypt and access the content.

**Asymmetric Encryption:** Public-key cryptography (e.g., RSA, ECC) allows secure data exchange and authentication, ensuring the data's sender is verified.

**Digital Signatures:** These verify the authenticity and integrity of the QR code data, ensuring it hasn't been tampered with since creation.

**Hash Functions:** Cryptographic hash functions (e.g., SHA-256) can generate unique hashes for data, enabling integrity checks to detect unauthorized modifications.

**Steganography:** Embedding cryptographic keys or watermarks into the QR code for additional layers of authentication.

## 3. Key Features of Cryptographically Secure QR Codes

**Confidentiality:** Ensures that the QR code data remains hidden from unauthorized users through encryption.

**Integrity:** Detects any tampering or modification of the QR code by verifying digital signatures or hashes.

**Authentication:** Confirms the source of the QR code using digital certificates or signatures, ensuring the content is trustworthy.

**Non-repudiation:** Prevents the sender from denying their involvement in creating or distributing the QR code.

**Access Control:** Allows only authorized users to scan and decode the QR code based on cryptographic credentials.

## 4. Applications of Secure QR Codes

**Payment Systems:** Protects transaction data in mobile payment apps by encrypting payment details and verifying their authenticity.

**Healthcare:** Safeguards sensitive patient information, ensuring that only authorized personnel can access medical records and prescriptions.

**E-commerce:** Secures digital receipts and product authenticity certificates by embedding cryptographically signed data.

**Identity Verification:** Encrypts personal data in digital ID systems, such as passports and licenses, preventing identity theft.

**Supply Chain Management:** Ensures product traceability by encrypting tracking information and verifying the authenticity of goods.

## 5. Benefits of Cryptographic QR Codes

**Enhanced Security:** Protects users from malicious attacks like phishing, spoofing, and data tampering.

**Improved Trust:** Boosts confidence in QR code systems by ensuring data authenticity and reliability.

**Scalability:** Cryptographic algorithms can be implemented across various industries and adapted for different use cases.

**Interoperability:** Cryptographic QR codes can work seamlessly with existing authentication and security systems.

**Future-Readiness:** Cryptographic QR codes are well-suited for integration with blockchain and other emerging technologies for secure data sharing.

## 6. Advanced Methods for Secure Detection

**QR Code Watermarking:** Embeds hidden watermarks in QR codes to verify authenticity without altering visible data.

**AI-Powered Scanning:** Uses machine learning models to detect malicious QR codes by analysing patterns and content.

**Dynamic QR Codes:** Generates QR codes with time-sensitive or session-based data, reducing the risk of misuse.

**Blockchain Integration:** Links cryptographically secure QR codes to blockchain records, enhancing trust and traceability.

## 7. Emerging Trends in Secure QR Codes

**Post-Quantum Cryptography:** As quantum computing advances, integrating quantum-resistant cryptographic algorithms will future-proof QR code security.

**Biometric Authentication:** Combines QR code scanning with biometric data (e.g., fingerprint or facial recognition) for added security.

**Hybrid QR Codes:** Merges cryptographic QR codes with Near Field Communication (NFC) technology for dual-mode secure communication.

## 8. Challenges in Implementing Cryptographic QR Codes

**Processing Overhead:** Cryptographic operations, especially with large keys, may increase decoding time and computational requirements.

**Key Management:** Safeguarding and distributing encryption keys can be complex, particularly in large-scale deployments.

**Backward Compatibility:** Ensuring cryptographic QR codes are compatible with existing scanners that lack cryptographic capabilities.

# Literature Survey

| Base Paper | Year | Problem Addressed | Limitation/ Problem Identified | Solution/ Rectification in Later Research |
|---|---|---|---|---|
| "Securing QR Codes with Visual Cryptography" | 2017 | Introduced splitting QR code data into multiple shares using visual cryptography. | Increased storage and decoding complexity; required all shares for decoding. | Proposed hybrid methods combining encryption and error correction for improved usability. |
| "Data Integrity in QR Codes Using Hash Functions" | 2018 | Verified QR code data integrity using SHA-256 hash functions. | Did not address data confidentiality; focused only on integrity verification. | Combined hashing with encryption to ensure both integrity and confidentiality. |
| "Secure Data Transmission via QR Code Using AES Encryption" | 2019 | Integrated AES-128 encryption for secure QR code encoding. | Static keys were used, making the system vulnerable to replay attacks. | Introduced dynamic key exchange mechanisms like Diffie-Hellman to address replay vulnerabilities. |
| "Multi-Factor Authentication with Encrypted QR Codes" | 2020 | Added a layer of authentication by embedding encrypted OTPs in QR codes. | Increased latency due to multi-step authentication; compatibility issues with standard QR readers. | Optimized the process by using session-based keys and cloud-backed validation. |
| "Secure QR Code Communication Using ECC and QR Version Control" | 2021 | Utilized elliptic curve cryptography (ECC) for secure communication and used QR version control. | Limited backward compatibility with legacy QR scanners. | Enhanced the decoding process with dual-mode support (legacy and secure modes). |

| | | | | |
|---|---|---|---|---|
| "AI-Assisted Secure QR Code Detection and Decoding with Cryptography" | 2022 | Introduced AI for detecting tampered QR codes and cryptographic algorithms for decoding. | Dependency on AI models increases resource usage on lowpower devices. | Proposed lightweight AI models optimized for embedded systems to address resource constraints. |

# Methodology

QR codes are widely used for storing and transmitting data. However, they are prone to security vulnerabilities such as tampering, spoofing, or unauthorized access. Combining QR code detection and decoding with cryptographic algorithms can enhance security and ensure data integrity and confidentiality.

Here's a step-by-step methodology:

### 1. QR Code Generation with Cryptographic Encoding

**Data Input:** Collect the data to be encoded in the QR code.

**Encryption:** Use a cryptographic algorithm (e.g., AES, RSA) to encrypt the data:

**Symmetric Encryption (e.g., AES):** Encrypt the data using a shared secret key.

**Asymmetric Encryption (e.g., RSA):** Encrypt the data using the public key of the intended recipient.

**Digital Signature (optional):**Generate a hash of the data using a secure hashing algorithm (e.g., SHA-256).Sign the hash using the sender's private key to ensure authenticity.

**QR Code Encoding:** Encode the encrypted data or the cryptographic signature into a QR code using a QR code generator library.

**2. QR Code Detection and Decoding**

**Detection:** Use computer vision techniques (e.g., OpenCV or built-in QR code libraries) to detect QR codes from an image or camera feed.Validate the format and error correction level of the QR code.

**Decoding:** Decode the data from the QR code using a QR code library (e.g., Zxing or Pyzbar). Extract the encrypted data or cryptographic signature.

**3. Decryption and Verification**

**Decryption:**If symmetric encryption was used, decrypt the data using the shared secret key.If asymmetric encryption was used, decrypt the data using the recipient's private key.

**Data Integrity Check:**

**If a digital signature is included:** Extract the original hash and signed hash from the QR code. Recompute the hash from the decrypted data. Verify the signature using the sender's public key. Check if the hashes match to ensure data integrity and authenticity.

**4. Security Features**

**Authentication:** Validate the QR code against a known trusted source or repository. Use additional metadata (e.g., timestamps or user IDs) to prevent replay attacks.

**Anti-Tampering:** Include a cryptographic checksum or hash in the QR code to detect modifications.

**Confidentiality:** Encrypt sensitive data to ensure it is accessible only to authorized users.

## 5. Implementation Considerations

**Cryptographic Algorithms:** Use modern and secure algorithms like AES-256 for encryption and RSA-2048 for public-key encryption. Use secure hashing algorithms like SHA-256 or SHA-3.

**Error Handling:** Ensure proper error correction mechanisms in the QR code (e.g., Level H in QR codes) to handle physical damage or noise. Performance **Optimization:** Optimize cryptographic operations for real-time decoding in mobile or resource-constrained environments.

## Code:

```
#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>

// Function to encrypt data using AES-128-ECB
int encrypt_data(const unsigned char *input, int input_len, unsigned char *output,
const unsigned char *key) {
    EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
    int len, ciphertext_len;

    if (!EVP_EncryptInit_ex(ctx, EVP_aes_128_ecb(), NULL, key, NULL)) {
        printf("Encryption initialization failed\n");
        return -1;
    }

    if (!EVP_EncryptUpdate(ctx, output, &len, input, input_len)) {
        printf("Encryption failed\n");
        return -1;
    }
    ciphertext_len = len;

    if (!EVP_EncryptFinal_ex(ctx, output + len, &len)) {
        printf("Final encryption step failed\n");
        return -1;
```

```
    }
    ciphertext_len += len;

    EVP_CIPHER_CTX_free(ctx);
    return ciphertext_len;
}

// Function to decrypt data using AES-128-ECB
int decrypt_data(const unsigned char *input, int input_len, unsigned char *output,
const unsigned char *key) {
    EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
    int len, plaintext_len;

    if (!EVP_DecryptInit_ex(ctx, EVP_aes_128_ecb(), NULL, key, NULL)) {
        printf("Decryption initialization failed\n");
        return -1;
    }

    if (!EVP_DecryptUpdate(ctx, output, &len, input, input_len)) {
        printf("Decryption failed\n");
        return -1;
    }
    plaintext_len = len;

    if (!EVP_DecryptFinal_ex(ctx, output + len, &len)) {
```

```
        printf("Final decryption step failed\n");
        return -1;
    }
    plaintext_len += len;

    EVP_CIPHER_CTX_free(ctx);
    return plaintext_len;
}

int main() {
    unsigned char qr_data[] = "SecureQRCode";
    unsigned char encrypted_data[128];  // Buffer for encrypted data
    unsigned char decrypted_data[128];  // Buffer for decrypted data
    unsigned char key[16] = "mysecurekey1234";  // 128-bit key (16 bytes)

    printf("Original QR Data: %s\n", qr_data);

    // Encrypt the QR data
    int encrypted_len = encrypt_data(qr_data, strlen((char *)qr_data),
encrypted_data, key);
    if (encrypted_len < 0) {
        printf("Encryption error\n");
        return 1;
    }
```

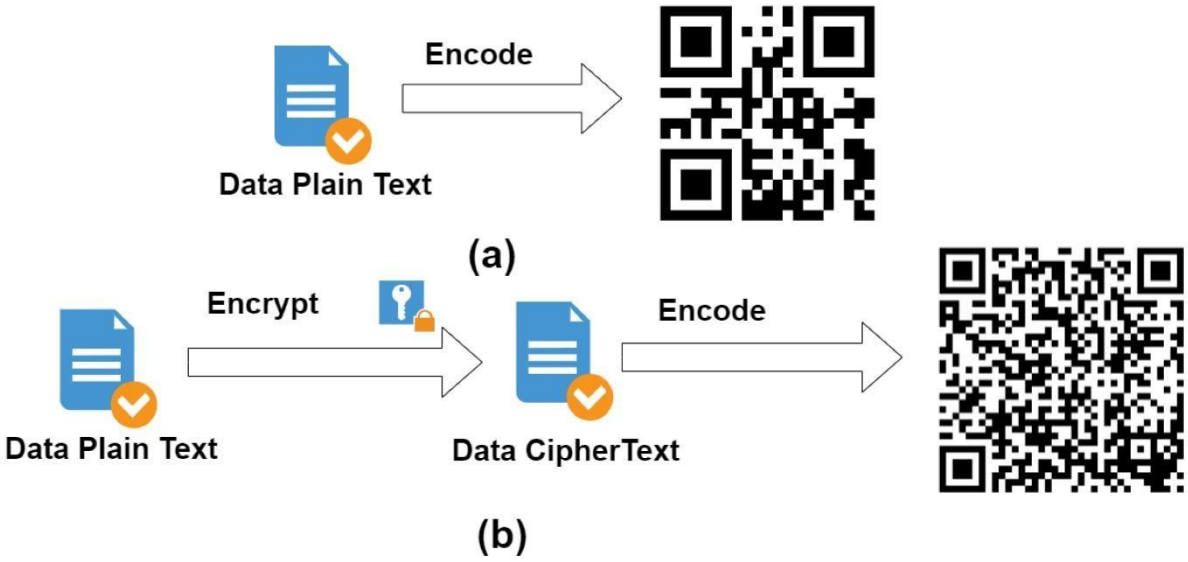
```
    printf("Encrypted QR Data: ");
    for (int i = 0; i < encrypted_len; i++) {
        printf("%02x", encrypted_data[i]);
    }
    printf("\n");

    // Decrypt the QR data
    int decrypted_len = decrypt_data(encrypted_data, encrypted_len, decrypted_data, key);
    if (decrypted_len < 0) {
        printf("Decryption error\n");
        return 1;
    }

    // Null-terminate the decrypted string
    decrypted_data[decrypted_len] = '\0';
    printf("Decrypted QR Data: %s\n", decrypted_data);

    return 0;
}
```

# Results and Discussion




The image consists of two different workflows for data processing involving QR codes:

**(a) Direct QR Code Encoding:**

1. **Input:** Plain text data is used as the source.
2. **Process:** The plain text data is directly encoded into a QR code.
3. **Output:** A QR code is generated that directly represents the plain text data.

**(b) Encrypted QR Code Encoding:**

1. **Input:** Plain text data is used as the source.
2. **Process:**
    - The plain text is first encrypted using a cryptographic key. ○ The encrypted data (ciphertext) is then encoded into a QR code.
3. **Output:** A QR code is generated that contains the encrypted data instead of the original plain text.

This approach demonstrates two methods of handling sensitive information in QR codes:

1. Encoding raw data (Figure a), which might be insecure for sensitive data.
2. Encrypting the data before encoding (Figure b), which provides enhanced security.

# Conclusion

Integrating cryptographic algorithms into QR code detection and decoding enhances the security of data transmitted through QR codes, ensuring confidentiality, integrity, and authenticity. By using cryptographic techniques like AES (Advanced Encryption Standard) for encryption and SHA-256 for hashing, the security of QR code contents is significantly strengthened, protecting against unauthorized access, tampering, and phishing attacks. Authentication is further bolstered through the use of public-private key pairs (such as RSA or ECC) to digitally sign the QR codes, enabling the verification of their origin and preventing spoofing. Hashing mechanisms embedded in the QR code structure ensure that any modifications to the encoded data are easily detectable, thus maintaining data integrity. The use of encryption also facilitates secure communication, ensuring that sensitive information, such as payment details or login credentials, remains confidential during transmission. Moreover, cryptographic techniques make QR codes resilient against various attacks like cloning, modification, and replay attacks, as encrypted payloads make intercepted codes unreadable. This enhanced security can be seamlessly integrated into existing QR code workflows, providing robust protection without compromising user experience. Such secure QR codes are especially valuable in critical domains such as digital payments, healthcare, and secure logins, where privacy and data integrity are paramount.

# References


1. Engin M, Çidam O, Engin EZ (2005) Wavelet transformation based watermarking technique for human electrocardiogram (ECG). J Med Syst 29(6):589–594
2. Ibaida A, Khalil I (2013) Wavelet-based ECG steganography for protecting patient confidential information in point-of-care systems. IEEE Trans Biomed Eng 60(12):3322–3330
3. International Standard ISO/IEC 18004 (2006) Information technology – automatic identification and data capture techniques – QR Code 2005 bar code symbology specification, Second Edition
4. Jain M, Lenka SK, Vasistha SK (2016) Adaptive circular queue image steganography with RSA cryptosystem. Perspect Sci 8:417–420
5. Jain M, Kumar A, Choudhary RC (2017) Improved diagonal queue medical image steganography using Chaos theory, LFSR, and Rabin cryptosystem. Brain Info 4(2):95–106
6. Jawad LM, G Sulong (2015) A survey on emerging challenges in selective color image encryption techniques." Indian Journal of Science and Technology 8.27
7. Jero SE, Ramu P, Ramakrishnan S (2014) Discrete wavelet transform and singular value decomposition based ECG steganography for secured patient information transmission. J Med Syst 38.10:132
8. Kalpana J, Murali P (2015) An improved color image encryption based on multiple DNA sequence operations with DNA synthetic image and chaos.

Optik-Int J Light Electron Optics 126(24):5703–5709

9. Kaur R, Singh EK (2013) Image encryption techniques: a selected review. J Comput Eng 9(6):80–83

10. Law P (1996b) Health Insurance Portability and Accountability Act of 1996. Public Law 104–191. US. Statut. Large, 1101936–2103

11. Li M, Yu S, Zheng Y, Ren K, Lou W (2013e) Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Trans Parallel Distrib Syst 24(1):131–143.

12. Lin Y-H et al (2004) A wireless PDA-based physiological monitoring system for patient transport. IEEE Trans Inf Technol Biomed 8(4):439–447

13. Maheswari SU, Jude Hemanth D (2015) Frequency domain QR code based image steganography using Fresnelet transform. AEU-Int J Electron Commun 69(2):539–544

14. Mazloom S, Eftekhari-Moghadam A (2009) Color image encryption based on coupled nonlinear chaotic map. Chaos, Solitons Fractals 42:1745–1754

15. Murugan B, Gounder A (2016) Image encryption scheme based on blockbased confusion and multiple levels of diffusion. IET Comput Vis 10:593–602

16. Nassar SS et al (2016) Secure wireless image communication using LSB steganography and chaotic baker ciphering. Wirel Pers Commun 91(3):1023–1049

17. Ng HS, Sim ML, Tan CM (2006f) Security issues of wireless sensor networks in healthcare applications. BT Technol J 24(2):138–144. https://doi.org/10.1007/s10550-006-0051-8

18. Pak C, Huang L (2017) A new color image encryption using combination of the 1D chaotic map. Signal Process 138:129–137
19. Ramesh G, Hambiraja E, Umarani DR (2012) A survey on various most common encryption techniques. Int J Adv Res Comput Sci Softwar Eng 2(7):226–233
20. Roy R, Sarkar A, Changder S (2013) Chaos based edge adaptive image steganography. Proced Technol 10:138–146