Secure QR Code Transmission Using Permutation Scrambling, Share Splitting, and Private Blockchain Channels

Mentor : Dr. B. Thanasekhar

TEAM MEMBERS (A19):

ABHINAVH P(2022503059)

PRABHAKARA ARJUN R(2022503003)

BUVANES SRIVARDAN K(2022503037)

Problem Statement

QR codes are widely utilized in various fields for quick and efficient information sharing. However, they are vulnerable to security threats such as information leakage, data tampering, and unauthorized access, especially during transmission and storage. Ensuring the confidentiality and integrity of QR code data in such scenarios is crucial to prevent misuse and maintain trust.

Objective

The primary objective of this project is to

- Ensuring Secure Data Transmission with help of permutation scrambling , share splitting algorithm.
- Enhancing Data Integrity and Decentralization with help of private blockchain.

Problems faced in QR Code Transmission System in Industries

- Security threats
- Lack of encryption
- Single point of failure
- No secure storage mechanism

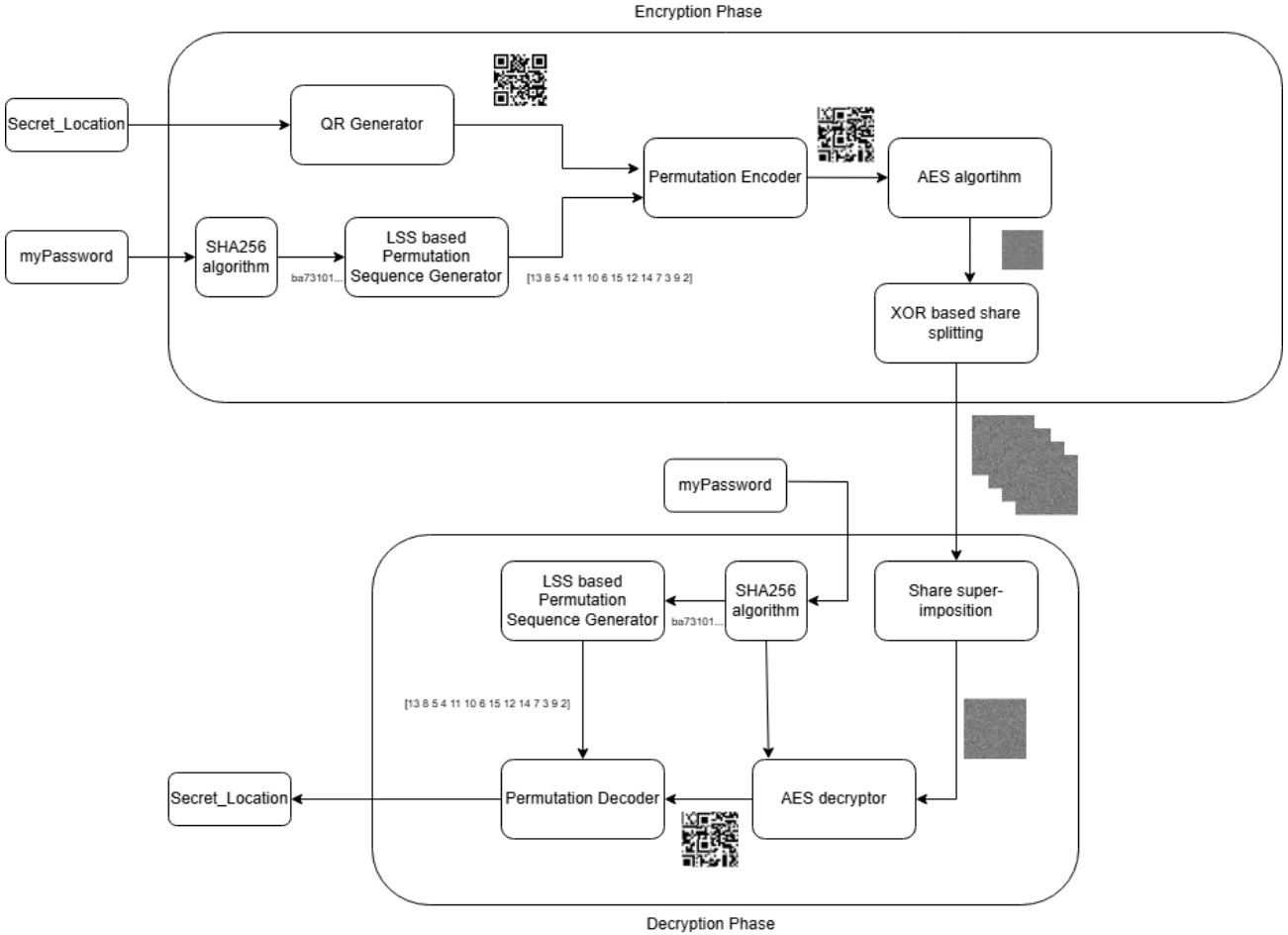
Importance of our project

- Enhanced security with encryption(AES) and scrambling.
- Data confidentiality and share splitting(all shares is needed to reconstruct QR).
- Decentralization and tamper resistant

Project modules

1. QR code generation and encryption.
2. Decryption and data retrieval.
3. Transmission and evaluation metrics.

Architecture diagram



MODULE-1: QR code generation and encryption.

```
# Generate QR Code
qr = qrcode.make(qr_input)
qr.save("qr.png")
```

```
Enter a string for SHA-256-based permutation: myPassword
Enter a string to generate a QR code: 40.7128, -74.0060
Generated QR Code:
```

QR code generation

Permutation scrambling using LSS and sha256

```python
def lss_permutation(seed_string, n, r=3.9, s=3.0):
    """
    Generate a valid and chaotic permutation sequence using LSS.
    :param seed_string: String input to generate SHA-256 hash seed.
    :param n: Length of the permutation sequence.
    :param r: Control parameter for the logistic map (default: 3.9).
    :param s: Control parameter for the sine map (default: 3.0).
    :return: A list of permuted indices.
    """
    x = sha256_to_float(seed_string)  # Convert hash to seed value
    sequence = [(x := (r * x * (1 - x) + s * np.sin(np.pi * x)) % 1, i) for i in range(n)]
    permuted_indices = [i for _, i in sorted(sequence, reverse=True)]
    return np.array(permuted_indices)
```

```
Enter a string for SHA-256-based permutation: myPassword
Enter a string to generate a QR code: 40.7128, -74.0060
Enter a string to generate a QR code: 40.7128, -74.0060
Generated QR Code:
Permutation Sequence: [13  0  8  5  4 11 10  6 15 12 14  7  3  9  2  1]
```

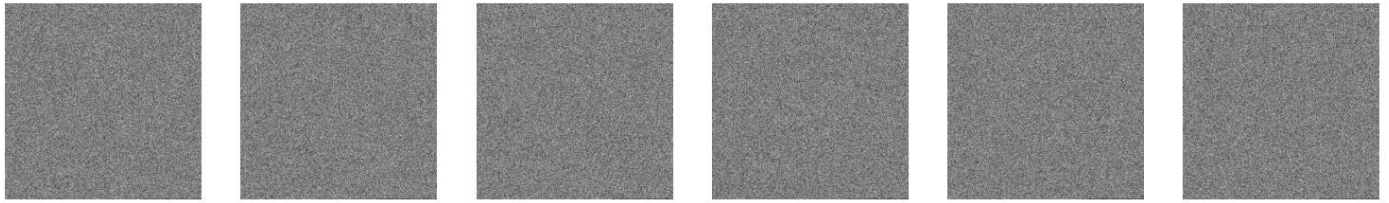
```python
def scramble_qr(blocks, permutation):
    """
    Scramble QR code blocks.
    :param blocks: List of divided blocks.
    :param permutation: Permutation sequence
    :return: Scrambled blocks.
    """
    return blocks[permutation.tolist()]
```

AES and XOR based share splitting algorithm

```python
def encrypt_aes(data, key):
    """Encrypt data using AES-256 in CBC mode."""
    cipher = AES.new(key, AES.MODE_CBC)
    ct_bytes = cipher.encrypt(pad(data, AES.block_size))
    return cipher.iv + ct_bytes
```

```python
def generate_shares(encrypted_data, n, k):
    """
    Generate n shares using XOR-based diffusion.
    :param encrypted_data: Encrypted data (1D byte array).
    :param n: Total number of shares.
    :param k: Minimum number of shares required for reconstruction.
    :return: List of shares.
    """
    shares = [np.random.randint(0, 256, size=len(encrypted_data), dtype=np.uint8) for _ in range(n-1)]
    last_share = np.frombuffer(encrypted_data, dtype=np.uint8).copy()
    for share in shares:
        last_share ^= share
    shares.append(last_share)
    return shares
```

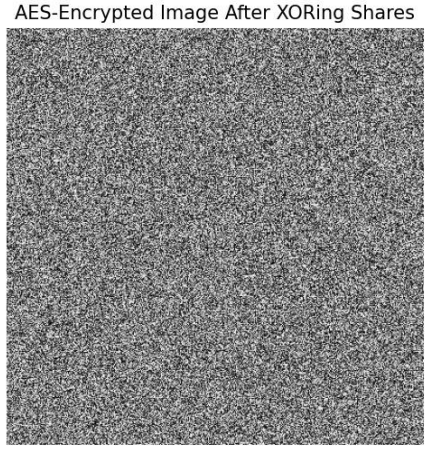
```
Input the number of shares images you want to create for encrypting (min is 2, max is 8): 6
```



MODULE-2:Decryption and data retrieval.

XORing all the n shares to obtain AES image

```python
def reconstruct_image(shares):
    """
    Reconstruct the encrypted data from shares.
    :param shares: List of shares.
    :return: Reconstructed encrypted data (1D byte array).
    """
    return np.bitwise_xor.reduce(shares).tobytes()
```



Obtaining scrambled image from AES

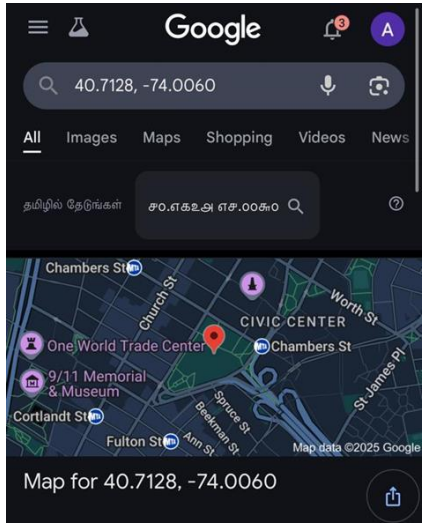
```python
def decrypt_aes(encrypted_data, key):
    """Decrypt data using AES-256 in CBC mode."""
    iv = encrypted_data[:AES.block_size]
    ct = encrypted_data[AES.block_size:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(ct), AES.block_size)
```

```
Enter the SHA-256-based password again to reconstruct the QR code: myPassword
Password verified. Reconstructing the QR code...
```

Scrambled Reconstructed Image

Descrambling using LSS and sha256

```python
def descramble_qr(blocks, permutation):
    """
    Reverse scrambling using inverse permutation.
    :param blocks: Scrambled blocks.
    :param permutation: Permutation sequence.
    :return: Descrambled blocks.
    """
    inverse_perm = np.argsort(permutation)
    return blocks[inverse_perm.tolist()]
```



Evaluation metrics

1. **Normalised Cross Correlation**
   a. NCORR Measures the similarity between the original and reconstructed QR codes

2. **Peak-Signal-To-Noise Ratio**
   a. Measures the difference between the original and reconstructed QR codes in terms of pixel intensity distortion.
   b. Higher PSNR means better quality that means less distortion.

3. **Structural Similarity Index (SSIM)**
   a. Compares luminance, contrast, and structure between images.
   b. More sensitive to human visual perception than PSNR.

MODULE-3:Evaluvation metrics

```python
def psnr(original, reconstructed):
    mse = np.mean((original - reconstructed) ** 2)
    if mse == 0:
        return float('inf')
    max_pixel = 255.0
    return 20 * np.log10(max_pixel / np.sqrt(mse))

def normxcorr2D(original, reconstructed):
    corr = np.corrcoef(original.flatten(), reconstructed.flatten())
    return corr[0, 1]
```

```python
psnr_value = psnr(original_qr, descrambled_qr_resized)
ncorr_value = normxcorr2D(original_qr, descrambled_qr_resized)
```

```
Evaluation metrics:
PSNR value is 36.71926405743969 dB
Mean NCORR value is 0.023155902450364453
```

References

[1] Haroon Rashid ,Hammood Al Dallal and Wijdan Noaman Marzoog Al Mukhtar, “*A QR Code Used for Personal Information Based On Multi-Layer Encryption System* ”, *Int. J. Interact. Mob. Technol.*, vol. 17, no. 09, pp. pp. 44–56, May 2023.

[2] Okubo, I.; Ono, S.; Kim, H.-W.; Cho, M.; Lee, M.-C. “*A Study on the Simple Encryption of QR Codes Using Random Numbers*.” Electronics 2024, 13, 3003.

[3] X. Wu, N. An and Z. Xu, "*Sharing Multiple Secrets in XOR-Based Visual Cryptography by Non-Monotonic Threshold Property*," in IEEE Transactions on Circuits and Systems for Video Technology,

[4] S. Sharma, S. Shivani and N. Saxena, "*A Self-Authenticating Multi-Tone Secret Sharing Scheme Using Meaningful Shares for Satellite Images*," in IEEE Access, vol. 12, pp. 101563-101591, 2024,

[5] S. Saha, A. K. Chattopadhyay, A. K. Barman, A. Nag and S. Nandi, "*Secret Image Sharing Schemes: A Comprehensive Survey*," in IEEE Access, vol. 11, pp. 98333-98361, 2023,

References

[6] P. Li, J. Ma, L. Yin and Q. Ma, *"A Construction Method of (2, 3) Visual Cryptography Scheme*," in IEEE Access, vol. 8, pp. 32840-32849, 2020,

[7] A. Jolfaei, X. -W. Wu and V. Muthukkumarasamy, "*On the Security of Permutation-Only Image Encryption Schemes*," in IEEE Transactions on Information Forensics and Security, vol. 11, no. 2, pp. 235-246, Feb. 2020,

[8] Zhengxin Fu, Liguo Fang, Hangying Huang, Bin Yu,"*Distributed three-level QR codes based on visual cryptography scheme*,",Journal of Visual Communication and Image Representation,Volume 87,2022,

[9] Bhardwaj, C., Garg, H., & Shekhar, S. (2022). ":*An Approach for Securing QR code using Cryptography and Visual Cryptography."* 2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)

[10] Xiaohe Cao, Liuping Feng, Peng Cao and Jianhua Hu "*Secure QR Code Scheme Based on Visual Cryptography*" 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE2016)