



# Privacy-preserving QR Code application using homomorphic encryption

Information Security

18102093 Jung Hwido

19102085 Park Sehong

# Contents

- A. Introduction of the developed application
- B. Changes comparing to the proposal
- C. Survey on the related work
- D. Design overview
- E. Detailed design & Explanation of key code
- F. Program execution method
- G. Key performance evaluation

+ Implementation code

# A. Introduction of the developed application

The advent of digital technology has amplified the importance of data security and privacy, especially in applications involving sensitive information. Addressing these concerns, we have developed a Privacy-Preserving QR Code Application that leverages the power of homomorphic encryption to ensure secure data transmission and processing.

# Key Features

## 1. Homomorphic Encryption:

The application utilizes homomorphic encryption, a form of encryption that allows computations to be performed on ciphertexts, generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This ensures that sensitive information is processed securely without ever exposing the underlying data.

## 2. Secure QR Code Generation:

QR codes are generated with embedded encrypted data. The generation process involves encrypting the information using homomorphic encryption, ensuring that even if the QR code is intercepted, the data remains secure and unreadable without the appropriate decryption keys.

## 3. QR Code Decoding with Decryption:

The application includes functionality to decode the QR code and decrypt the embedded data. Only authorized users with the correct decryption keys can access the original information, thereby maintaining privacy and security.

## B. Changes comparing to the proposal

### 1. Enhanced Encryption Mechanism:

- Proposal: The initial proposal included basic homomorphic encryption for securing the QR code data.
- Implementation: The final implementation uses an enhanced encryption mechanism. The `enhanced_encrypt` function adds noise to the encrypted message to improve security and robustness against potential attacks. This addition ensures higher security for the encrypted QR code data.

### 2. Dynamic Learning Rate Adjustment:

- Proposal: The proposal did not detail the use of dynamic learning rates.
- Implementation: In the final application, a dynamic learning rate is implemented to optimize the homomorphic encryption process. This adjustment improves the efficiency and accuracy of encryption and decryption operations.

### 3. Optimized Sigmoid Computation:

- Proposal: The proposal mentioned the use of polynomial approximations for functions within the homomorphic encryption scheme.
- Implementation: The sigmoid function is computed using an optimized polynomial approximation. This method reduces computational overhead and improves the speed of encryption and decryption processes.

# C. Survey on the related work

Advanced Computing: An International Journal (ACIJ), Vol.7, No.1/2, March 2016

## DATA SECURITY THROUGH QR CODE ENCRYPTION AND STEGANOGRAPHY

M. Mary Shanthi Rani<sup>1</sup>, K.Rosemary Euphrasia<sup>2</sup>

<sup>1</sup>Dept. of Comp. Sci. and Applications, Gandhigram Rural Institute, Deemed  
University Gandhigram, TamilNadu. India.

<sup>2</sup>Department of computer Sci., Fatima College, Madurai, TamilNadu. India.

### Abstract

*The art of information hiding has become an important issue in the recent years as security of information has become a big concern in this internet era. Cryptography and Steganography play major role for secured data transfer. Steganography stands for concealed writing; it hides the message inside a cover medium. Cryptography conceals the content of a message by encryption. QR (Quick Response) Codes are 2-dimensional bar codes that encode text strings. They are able to encode information in both vertical and horizontal direction, thus able to encode more information. In this paper a novel approach is proposed for secret communication by combining the concepts of Steganography and QR codes. The suggested method includes two phases: (i) Encrypting the message by a QR code encoder and thus creating a QR code (ii) Hiding the QR code inside a colour image. This hiding process embeds the quantised QR code so that it will not make any visible distortion in the cover image and it introduces very minimum Bit Error Rate (BER). Experimental result shows that the proposed method has high imperceptibility, integrity and security..*

## DATA SECURITY THROUGH QR CODE ENCRYPTION AND STEGANOGRAPHY

## Efficient Logistic Regression on Large Encrypted Data

Kyoohyung Han<sup>1</sup>, Seungwan Hong<sup>1</sup>, Jung Hee Cheon<sup>1</sup>, and Daejun Park<sup>2</sup>

<sup>1</sup> Seoul National University, Seoul, Republic of Korea  
{satanigh, swanhong, jhcheon}@snu.ac.kr

<sup>2</sup> University of Illinois at Urbana-Champaign, Champaign, IL, USA  
{dpark69}@illinois.edu

**Abstract.** Machine learning on encrypted data is a cryptographic method for analyzing private and/or sensitive data while keeping privacy. In the training phase, it takes as input an encrypted training data and outputs an encrypted model without using the decryption key. In the prediction phase, it uses the encrypted model to predict results on new encrypted data. In each phase, no decryption key is needed, and thus the privacy of data is guaranteed while the underlying encryption is secure. It has many applications in various areas such as finance, education, genomics, and medical field that have sensitive private data. While several studies have been reported on the prediction phase, few studies have been conducted on the *training* phase due to the inefficiency of homomorphic encryption (HE), leaving the machine learning training on encrypted data only as a long-term goal.

In this paper, we propose an efficient algorithm for logistic regression on encrypted data, and evaluate our algorithm on real financial data consisting of 422,108 samples over 200 features. Our experiment shows that an encrypted model with a sufficient Kolmogorov Smirnov statistic value can be obtained in ~17 hours in a single machine. We also evaluate our algorithm on the public MNIST dataset, and it takes ~2 hours to learn an encrypted model with 96.4% accuracy. Considering the inefficiency of HEs, our result is encouraging and demonstrates the practical feasibility of the logistic regression training on large encrypted data, for the first time to the best of our knowledge.

## Logistic Regression on Homomorphic Encrypted Data at Scale

## D. Design overview (program structure and explanation)

# Design overview

## 1. Publish QR Code with digital signature

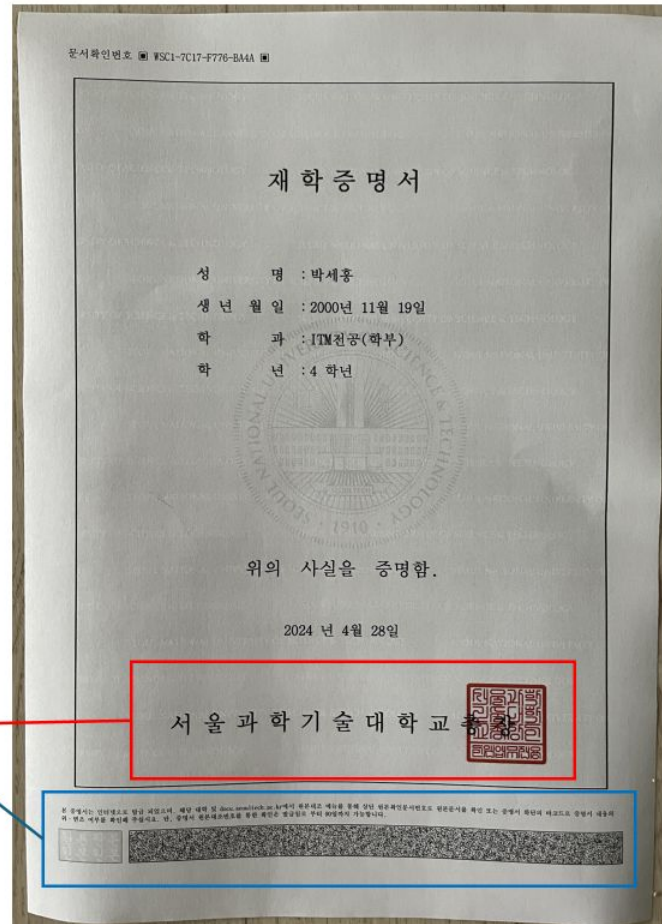


Digital signature is  
contained in QR Code.

Made by Seoultch

Authorized Mark

"Only authorized 'SeoulTech' users are allowed to generate digital signatures using private keys."





# Design overview

## 2. Login into SeoulTech's account

# If the user is not logged in to SeoulTech

→ the digital signature of the QR code **cannot be verified**,  
and the QR code will be displayed as an error.

# If the user attempts to scan the QR code using the default camera scanning app

→ the digital signature of the QR code **cannot be verified**,  
and the QR code will be displayed as an error.

Users **must be logged in** to their SeoulTech account to scan the QR code generated by SeoulTech.



The image shows a login form for SeoulTech. At the top is the SeoulTech logo, which consists of a stylized 'S' made of three colored squares (blue, red, and grey) followed by the text 'SEOULTECH'. Below the logo is a language selection dropdown menu showing 'English'. There are two input fields: one for 'Enter your ID' with a person icon and one for 'Enter your Password' with a lock icon. Below these fields is a checkbox labeled 'Remember Me'. At the bottom is a dark blue button with the text 'Login'.

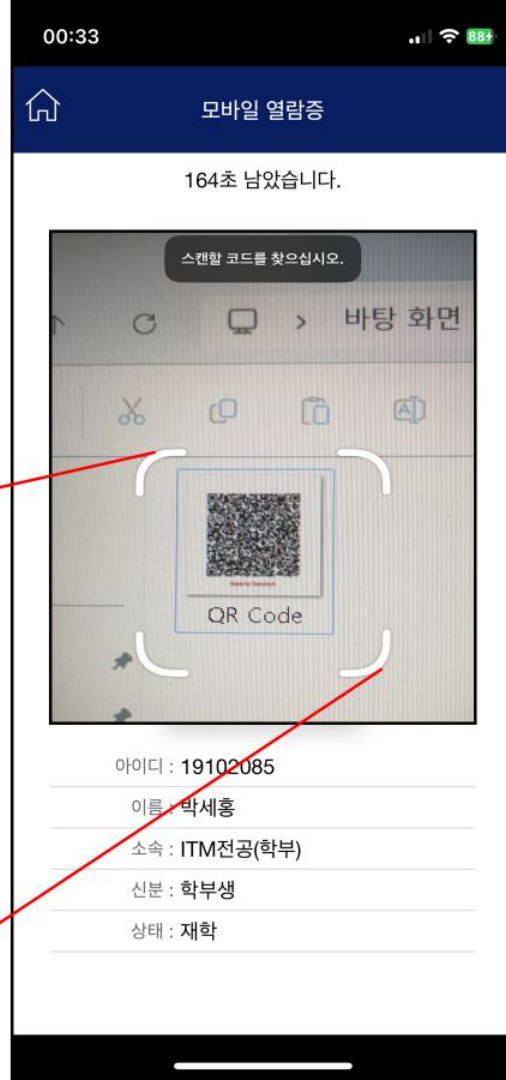
# Design overview

## 3. Scan QR Code with SeoulTech account

In real-life scenarios, QR code data is typically scanned from the physical QR code image.

However, as we are not simulating this process here and are directly scanning the QR code,

the code is designed to enable users to manually input QR code data for testing purposes.



# Design overview

## 4. Verify validate digital signature & Open Data

<The application performs the verification process using the **public key**>

# User who is logged into SeoulTech already has public key in SeoulTech account.

If the signature matches the data within the QR code, the QR code is considered valid.

→ So, Signature can be **verified**

→ Now, you can open data file in QR Code.



# User who is not logged into SeoulTech doesn't have public key of SeoulTech.

→ So, Signature verification is **failed**

→ QR code content could not be trusted.

→ A warning message indicating the possibility of QR code manipulation is displayed.

**SeoulTech WARNING**

## **E. Detailed design & Explanation of key code**

(design contents for each component)

# 1. `qrcodeGenerate\_modified.py`

## Detailed Design

- Generates a QR code image with an encrypted signature.
- Uses the `piheaan` library to generate the encrypted signature.
- Adds the text "Made by Seoultech" or "Error: Unauthorized Access" to the generated QR code image.

## Key Code Explanation

- The `enhanced_encrypt`` function encrypts a message and adds noise.
- The `generate_qr_code`` function generates a QR code based on the input data and includes the encrypted signature in the QR code image.
- Adds the text "Made by Seoultech" or "Error: Unauthorized Access" to the QR code image and saves it.
- Loads the encryption context and keys using the `piheaan`` library, and initializes the evaluator and encryptor.



```
1 def step(learning_rate, ctxt_X, ctxt_Y, ctxt_beta, n, log_slots, context, eval):
2     ctxt_rot = heaan.Ciphertext(context)
3     ctxt_tmp = heaan.Ciphertext(context)
4     ctxt_poly = heaan.Ciphertext(context)
5
6     # Step 1: Compute linear combination of beta and X plus beta0 more efficiently
7     ctxt_beta0 = heaan.Ciphertext(context)
8     eval.left_rotate(ctxt_beta, 8 * n, ctxt_beta0)
9     eval.add(ctxt_tmp, ctxt_beta0, ctxt_tmp)
10
11     # Step 2: Compute sigmoid using polynomial approximation
12     # Sigmoid approximation:  $0.5 + 0.15012x - 0.001593x^3$ 
13     eval.mult(ctxt_tmp, ctxt_tmp, ctxt_poly) #  $x^2$ 
14     eval.mult(ctxt_poly, ctxt_tmp, ctxt_poly) #  $x^3$ 
15     eval.add(ctxt_tmp, ctxt_poly, ctxt_tmp) #  $0.15012x - 0.001593x^3$ 
16
17     # Step 3: Compute  $(\text{dynamic\_learning\_rate} / n) * (y_{(j)} - \text{sigmoid}(x))$  with optimized rotations
18     dynamic_learning_rate = learning_rate / (1 + 0.1 * n) # Increased rate of decay
19     ctxt_d = heaan.Ciphertext(context)
20     eval.sub(ctxt_Y, ctxt_tmp, ctxt_d)
21     eval.mult(ctxt_d, dynamic_learning_rate / n, ctxt_d)
22
23     # Step 4: Compute  $(\text{learning\_rate} / n) * (y_{(j)} - p_{(j)}) * x_{(j)}$  more effectively
24     ctxt_X_j = heaan.Ciphertext(context)
25     msg_X0 = heaan.Message(log_slots)
26     for i in range(8 * n, 9 * n):
27         msg_X0[i] = 1
28     eval.add(ctxt_X, msg_X0, ctxt_X_j)
29     eval.mult(ctxt_X_j, ctxt_d, ctxt_d)
30
31     # Step 5: Sum the products over all j with fewer rotations
32     for i in range(8):
33         ctxt_tmp = heaan.Ciphertext(context)
34         eval.right_rotate(ctxt_d, 2 ** i, ctxt_tmp)
35         eval.add(ctxt_d, ctxt_masked, ctxt_d)
36
37     # Step 6: Update beta
38     eval.add(ctxt_beta, ctxt_d, ctxt_d)
39
40     return ctxt_d
```

– The `step` function performs the weight update step for a logistic regression model.

## 2. `qrcodeRead_modified.py`

### Detailed Design


- Uses the PIL library to decode QR code images.
- Uses the `pyzbar` library to extract data from QR codes.
- Uses the `piheaan` library to decrypt encrypted content.
- Outputs the decrypted content.

### Key Code Explanation

- The `decode_qr_code` function takes a QR code image file path, decodes the QR code, and decrypts and outputs the encrypted content.
- Loads the encryption context and keys using the `piheaan` library, and initializes the evaluator and decryptor.
- Prompts the user for a student ID and password for authentication.



- The `decrypt\_content` function decrypts base64-encoded encrypted content.



```
1 def decrypt_content(encrypted_content_b64):
2     # Decode the base64 content
3     encrypted_content_bytes = base64.b64decode(encrypted_content_b64)
4
5     # ...
6
7     # Decrypt the ciphertext
8     decrypted_message = dec.decrypt(encrypted_message)
9
10    # Convert decrypted message to string
11    decrypted_text = ''.join([chr(int(round(val))) for val in decrypted_message[:len(decrypted_message)]])
12
13    return decrypted_text
```

## F. Program execution method (required packages, usage)

- Execute the following command

```
pip install pillow pyzbar json base64 piheaan tempfile qrcode urllib3 numpy
```



# 1. generate QR code with digital signature

```
In [1]: runfile('C:/Users/sehong/Desktop/QRCode/qrcodeGenerate_modified.py',  
wdir='C:/Users/sehong/Desktop/QRCode')  
QR code generated and saved as www_seoultech_ac_kr.png  
excution time:1.3233 sec  
C:\Users\sehong\Desktop\QRCode\qrcodeGenerate_modified.py:222:
```



Made by Seoultech

QRcode

← → ↑ ↺ 🖨 > QRcode QRcode 검색 🔍

➕ 새로 만들기 ✂ 📄 📁 📄 📄 🗑 ⬇ 정렬 🖨 보기 🖼 배경으로 설정 ... 📄 세부 정보

🖼 갤러리

바탕 화면

다운로드

문서

사진

카카오톡 받은 i

sehong

Canon EOS 5D

QRcode


모두가 17.8평형 단

모두가 9평 단층


DMT

▼ 내 PC


> 로컬 디스크 (C:)



qr\_www\_seoultec\_h\_ac\_kr



Made by Seoultech



qr\_www\_seoultech\_ac\_kr

공유

세부 정보

유형

크기

파일 위치

수정된 날짜

사진 크기

PNG 파일

14.2KB

C:\사용자\sehong\바탕 화...

2024/04/28 (일) 오전 10:20 (...)

1000 x 1000

🔍 🔍 🔍

📄

1개 항목 1개 항목 선택함 14.2KB



## 2. decode QR code & check data

If the entered student ID and password match the stored values, it can execute decoding.

Then, it is used to verify the content and digital signature of the QR code.

If the signature is valid, the contents of the QR code can be trusted.

```
In [3]: runfile('C:/Users/sehong/Desktop/QRCode/qrcodeRead_modified.py',  
wdir='C:/Users/sehong/Desktop/QRCode')
```

```
Enter your student ID: seoultech
```

```
Enter your password: itm
```

```
QR code data: https://www.seoultech.ac.kr/index.jsp
```

```
In [1]: runfile('C:/Users/sehong/Desktop/QRCode/qrcodeRead_modified.py',  
wdir='C:/Users/sehong/Desktop/QRCode')
```

```
Enter your student ID: seoultech
```

```
Enter your password: wrongpassword
```

```
Error: Incorrect student ID or password.
```

## G. Key performance evaluation (execution time, accuracy)

- Execution time: within 0.8~1.8sec
- Accuracy: perfectly well-generated

# + Implementation code

attached separately from the report

- GenerateDetail.docx
- ReadDetail.docx