PQC-NN: Post-Quantum Cryptography Neural Network

Abel C. H. Chen
Information & Communications Security Laboratory,
Chunghwa Telecom Laboratories
Taoyuan, Taiwan
ORCID: 0000-0003-3628-3033; Email: chihua0826@cht.com.tw

***Abstract*—In recent years, quantum computers and Shor's quantum algorithm have been able to effectively solve NP (Non-deterministic Polynomial-time) problems such as prime factorization and discrete logarithm problems, posing a threat to current mainstream asymmetric cryptography, including RSA and Elliptic Curve Cryptography (ECC). As a result, the National Institute of Standards and Technology (NIST) in the United States call for Post-Quantum Cryptography (PQC) methods that include lattice-based cryptography methods, code-based cryptography methods, multivariate cryptography methods, and hash-based cryptography methods for resisting quantum computing attacks. Therefore, this study proposes a PQC neural network (PQC-NN) that maps a code-based PQC method to a neural network structure and enhances the security of ciphertexts with non-linear activation functions, random perturbation of ciphertexts, and uniform distribution of ciphertexts. The main innovations of this study include: (1) constructing a neural network structure that complies with code-based PQC, where the weight sets between the input layer and the ciphertext layer can be used as a public key for encryption, and the weight sets between the ciphertext layer and the output layer can be used as a private key for decryption; (2) adding random perturbations to the ciphertext layer, which can be removed during the decryption phase to restore the original plaintext; (3) constraining the output values of the ciphertext layer to follow a uniform distribution with a significant similarity by adding the cumulative distribution function (CDF) values of the chi-square distribution to the loss function, ensuring that the neural network produces sufficiently uniform distribution for the output values of the ciphertext layer. In practical experiments, this study uses cellular network signals as a case study to demonstrate that encryption and decryption can be performed by the proposed PQC neural network with the uniform distribution of ciphertexts. In the future, the proposed PQC neural network could be applied to various applications.**

Keywords—post-quantum cryptography, McEliece cryptography, neural network

I. INTRODUCTION

In the recent years, the development of quantum computers and Shor's quantum algorithm [1] have been used for effectively solving NP (Non-deterministic Polynomial-time) problems (e.g. prime factorization and discrete logarithm problems). Therefore, the current mainstream asymmetric cryptography methods (e.g. RSA based on the prime factorization problem and Elliptic Curve Cryptography (ECC) based on the discrete logarithm problem) may be attacked by quantum computing. Therefore, the National Institute of Standards and Technology (NIST) in the United States has called for Post-Quantum Cryptography (PQC) methods [2] and collected latticed-based cryptography methods [3], code-based cryptography methods [4-5], multivariate cryptography methods [2], and hash-based cryptography methods [6] based on different NP problems for improving security. For instance, McEliece cryptography method (i.e. one of code-based cryptography methods) generates several matrices as the private key and the multiplication of these matrices as the public key, and the non-negative matrix factorization (NMF) problem (i.e. one of NP problems) supports the security of McEliece cryptography method. Moreover, some noises can be added into ciphertexts for improving security, and these noises can be detected and corrected through the error-correcting phase in McEliece cryptography method for resisting quantum computing attacks.

Furthermore, neural networks have been a popular tool to provide encoding and decoding. Therefore, this study proposes PQC neural network (PQC-NN) based on the ability of encoding and decoding in the neural network [7]. The proposed PQC neural network can map a code-based PQC method to a neural network structure and enhance the security of ciphertexts with non-linear activation functions, random perturbation of ciphertexts, and uniform distribution of ciphertexts.

The contributions and innovations of this study are highlighted and summarized as follows.

- Constructing a neural network structure that complies with code-based PQC, where the weight sets between the input layer and the ciphertext layer can be used as a public key for encryption, and the weight sets between the ciphertext layer and the output layer can be used as a private key for decryption.
- Adding random perturbations to the ciphertext layer, which can be removed during the decryption phase to restore the original plaintext.
- Constraining the output values of the ciphertext layer to follow a uniform distribution with a significant similarity by adding the cumulative distribution function (CDF) values of the chi-square distribution to the loss function, ensuring that the neural network produces sufficiently uniform distribution for the output values of the ciphertext layer.

This manuscript has five sections. Section II describes a code-based PQC method and presents the principle and calculation examples of McEliece cryptography method. Section III proposes PQC neural network from a neural network for McEliece cryptography method to the advanced neural network for PQC method. For generating secure ciphertexts, Subsection III.C illustrates the added random perturbations in the ciphertext layer and the added CDF values in the loss function for the chi-test of ciphertext uniform distribution. In Section IV, practical cellular network signals are selected as a case study to be encrypted by the proposed PQC neural network, and the mean-square errors and CDF values are evaluated under different hyperparameter values.

Finally, Section V concludes the contributions of this study and discusses the future work.

II. Code-Based Post-Quantum Cryptography

This section presents McEliece cryptography method (i.e. one of code-based PQC methods) and uses Hamming code method as the error-correcting method in McEliece cryptography method. Section II.A gives a calculation example to explain encoding and decoding by Hamming code method. Section II.B gives a calculation example to explain encryption and decryption by McEliece cryptography method.

A. Hamming code method

This subsection illustrates the initial phase, encoding and decoding phase, and error-correcting phase.

1) Initial phase

In the initial phase, Hamming code method generates three matrices that include a generator matrix (i.e. an encoder) *G*, an error-detector matrix *H* based on *G*, a decoder matrix *R* based on *G*. This manuscript shows a calculation example to present the matrices *G*, *H*, and *R* in Equations (1), (2), and (3).

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

2) Encoding and decoding phase

In the encoding phase, the generator matrix *G* can be multiplied by the plaintext *x* to obtain the encoded text *y* (shown in Equation (4)). Furthermore, the decoding phase, the decoder matrix *R* can be multiplied by the encoded text *y* to obtain the plaintext *x* (shown in Equation (5)). In the calculation example, the plaintext $x = [1 \quad 0 \quad 0 \quad 0]$ is selected as a case for explanation, and the encoded text *y* can be calculated as $[1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$ (shown in Equation (6)). Furthermore, the plaintext *x* can be obtained by the multiplication of the encoded text *y* and the decoder matrix *R* (shown in Equation (7)).

$$y = xG \tag{4}$$

$$x = yR \tag{5}$$

$$\begin{aligned} y &= xG \\ &= [1 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \\ &= [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0] \end{aligned} \tag{6}$$

$$\begin{aligned} x &= yR \\ &= [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= [1 \quad 0 \quad 0 \quad 0] \end{aligned} \tag{7}$$

3) Error-correcting phase

For the demonstration of error-correcting phase, a random number r is added into the encoded text *y* to obtain the text *y*' with a noise (shown in Equation (8)). The positioning of error bit *z* in the text *y*' can be detected by the error-detector matrix *H* (shown in Equation (9)) for correction. In the manuscript, the random number $r = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$ is selected to be a noise for generating the text *y*' (shown in Equation (10)). Furthermore, the error-detector matrix *H* can be multiplied by the text *y*' to obtain the positioning of error bit $z = [1 \quad 1 \quad 1]$ (shown in Equation (11)). The seventh bit is incorrect and detected by the positioning of error bit *z*, so the seventh bit of the text *y*' can be corrected to obtain the corrected encoded text $y = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$.

$$y' = y + r \tag{8}$$

$$z = y'H \tag{9}$$

$$\begin{aligned} y' = y + r &= [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0] \\ &\quad + [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1] \\ &= [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1] \end{aligned} \tag{10}$$

$$z = y'H = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = [1 \quad 1 \quad 1] \tag{11}$$

B. McEliece cryptography method

This subsection illustrates the key generation phase, encryption phase, and decryption phase.

1) Key generation phase

In the key generation phase, McEliece cryptography method generates three matrices that include a scrambler *S*, a generator matrix *G*, and a permutation matrix *P*. These three matrices (i.e. {*S*, *G*, *P*}) are selected as a private key, and the multiplication of these three matrices *G*' is selected as public key (shown in Equation (12)). For the explanation of McEliece cryptography method, a calculation example of *S*, *G*, and *P* is given in Equations (13), (1), and (14). Furthermore, the public key *G*' can be obtained by Equation (15).

$$G' = SGP \tag{12}$$

$$S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \tag{13}$$

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{14}$$

$$G' = SGP$$
$$= \begin{bmatrix}1&1&0&1\\1&0&0&1\\0&1&1&1\\1&1&0&0\end{bmatrix} \times \begin{bmatrix}1&1&1&0&0&0&0\\1&0&0&1&1&0&0\\0&1&0&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix} \times \begin{bmatrix}0&1&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&1\\1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\end{bmatrix} \times \begin{bmatrix}0&1&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&1\\1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\end{bmatrix} \quad (15)$$
$$= \begin{bmatrix}0&1&1&0&1&0&1\\1&0&0&0&1&0&1\\1&0&1&0&1&1&0\\1&0&1&1&0&0&1\end{bmatrix}$$

2) Encryption phase

In the encryption phase, the public key *G*' can be multiplied by the plaintext *x* to obtain the ciphertext *y* with a random number *r* (shown in Equation (16)). For random perturbations, the same plaintext can be encrypted for generating different ciphertexts based on different random numbers. The plaintext $x = [1\ \ 0\ \ 0\ \ 0]$ and the random number $r = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$ are selected as a calculation example, and the ciphertext $y = [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1]$ can be obtained by Equation (17). Due the random number $r = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$ for the explanation of McEliece cryptography method, the error-correcting phase in Subsection II.A.3 is not necessary in this case. If other random numbers are used in the encryption phase, the ciphertext with a noise could be denoted as *y*'; furthermore, the noise in *y*' can be corrected and detected by the steps in Subsection II.A.3 for removing the influence of random number *r*.

$$y = xG' + r = xSGP + r \quad (16)$$

$$y = xG' + r$$
$$= [1\ \ 0\ \ 0\ \ 0]\begin{bmatrix}0&1&1&0&1&0&1\\1&0&0&0&1&0&1\\1&0&1&0&1&1&0\\1&0&1&1&0&0&1\end{bmatrix} + [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0] \quad (17)$$
$$= [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1]$$

3) Decryption phase

The ciphertext is obtained in accordance with the public key *G*' (i.e. the multiplication of *S*, *G*, and *P*), so the ciphertext can be decrypted by the multiplication of the matrix P^{-1}, the decoder matrix *R*, and the matrix S^{-1} for obtaining the plaintext *x*. Firstly, the inverse matrix of the permutation matrix *P* (i.e. P^{-1}) can be multiplied by the ciphertext *y* to remove the influence of the permutation matrix *P* (shown in Equation (18)). Furthermore, the error-correcting phase in Subsection II.A.3 can be performed to remove the influence of the random number *r* (i.e. rP^{-1} in Equation (18)) and get the matrix xSG. The decoding phase can be performed by Equation (19), the decoder matrix *R* can be multiplied by the matrix xSG to remove the influence of the generator matrix *G* for getting the matrix xS. Finally, the inverse matrix of the scrambler *S* (i.e. S^{-1}) can be multiplied by the matrix xS to remove the influence of the scrambler *S* (shown in Equation (20)) for obtaining the plaintext *x*.

$$yP^{-1} = xG'P^{-1} + rP^{-1}$$
$$= [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1] \times \begin{bmatrix}0&0&0&1&0&0&0\\1&0&0&0&0&0&0\\0&0&0&0&1&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&1&0&0&0&0\end{bmatrix} \quad (18)$$
$$= [1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1]$$
$$= xSGPP^{-1} + rP^{-1}$$
$$= xSG + rP^{-1}$$

$$xSGR = [1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1]\begin{bmatrix}0&0&0&0\\0&0&0&0\\1&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad (19)$$
$$= [1\ \ 1\ \ 0\ \ 1] = xS$$

$$xSS^{-1} = [1\ \ 1\ \ 0\ \ 1]\begin{bmatrix}1&1&0&1\\1&1&0&0\\0&1&1&1\\1&0&0&1\end{bmatrix} \quad (20)$$
$$= [1\ \ 0\ \ 0\ \ 0] = x$$

For the explanation of McEliece cryptography method with a calculation example, Hamming code method (i.e. a linear error-correcting method) is used, but the non-linear error-correcting methods could be used for improving security levels. Furthermore, this section uses small matrices, but bigger matrices could be considered for practical applications. For instance, the McEliece cryptography method which was submitted to NIST for PQC standardization has different sizes of bigger matrices for the requirements of security levels in the standard of NIST.

III. Post-Quantum Cryptography Neural Network

This section presents detail explanation of the proposed PQC neural network from step-by-step. Subsection III.A presents the structure of neural network for Hamming code method based on the case in Subsection II.A, and Subsection III.B presents the structure of neural network for McEliece cryptography method based on the case in Subsection II.B. Finally, the proposed PQC neural network is illustrated in Subsection III.C.

A. Neural network for Hamming code method

Fig. 1 presents the structure of neural network for Hamming code method based on the case in Subsection II.A. The neural network has four neurons in the input layer (i.e. $X = \{x_1, x_2, x_3, x_4\}$), seven neurons in the hidden layer (i.e. $Y = \{y_1, y_2, \ldots, y_7\}$), and four neurons in the output layer. Each neuron has no bias, and the linear function is adopted as the activation function. The outputs of the neural network is fitting the inputs of the neural network (i.e. an auto-encoder network). Therefore, the weights between the input layer and the hidden layer is a 4 x 7 matrix *G* (i.e. the generator matrix *G* in Subsection II.A), and the value of each element in the matrix *G* is denoted in Equation (1). The weights between the hidden layer and the output layer is a 7 x 4 matrix *R* (i.e. the decoder matrix *R* in Subsection II.A), and the value of each element in the matrix *R* is denoted in Equation (3). In this subsection, the red line is represented as the weight value of

one, and the black line is represented as the weight value of zero in the structure of neural network. Therefore, Hamming code method in Subsection II.A can be represented as the neural network in Fig. 1 for encoding and decoding. When the values of the input X are $[1 \quad 0 \quad 0 \quad 0]$, the values of the neurons in the hidden layer Y are $[1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$ (i.e. the encoded values). Furthermore, the encoded values can be decoded as $[1 \quad 0 \quad 0 \quad 0]$ through the output layer.

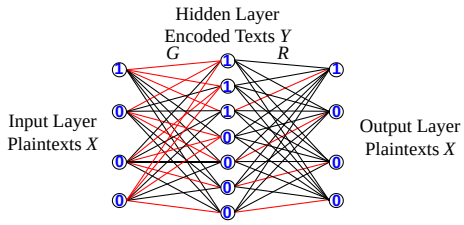

Fig. 1. Neural network for Hamming code method

B. Neural network for McEliece cryptography method

Fig. 2 presents the structure of neural network for McEliece cryptography method on the case in Subsection II.B. The neural network has four neurons in the input layer (i.e. $X = \{x_1, x_2, x_3, x_4\}$), four neurons in the first hidden layer, seven neurons in the second hidden layer, seven neurons in the third hidden layer (i.e. the ciphertext layer) $Y = \{y_1, y_2, \ldots, y_7\}$, seven neurons in the fourth hidden layer, four neurons in the fifth hidden layer, and four neurons in the output layer. Each neuron has no bias, and the linear function is adopted as the activation function. In this subsection, the red line is represented as the weight value of one, and the black line is represented as the weight value of zero in the structure of neural network.

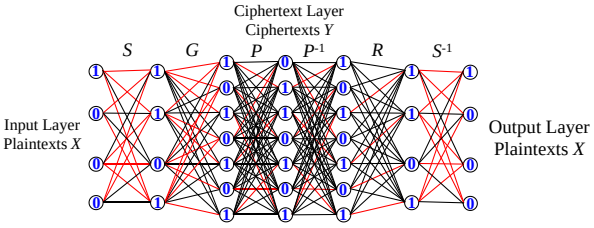

Fig. 2. Neural network for McEliece cryptography method

Encryption can be performed from the input layer to the third hidden layer (i.e. the ciphertext layer). In the neural network, the weights between the input layer and the first hidden layer is a 4 x 4 matrix S (i.e. the scrambler S in Subsection II.B), and the value of each element in the matrix S is denoted in Equation (13). The weights between the first hidden layer and the second hidden layer is a 4 x 7 matrix G (i.e. the generator matrix G in Subsection II.A), and the value of each element in the matrix G is denoted in Equation (1). The weights between the second hidden layer and the third hidden layer is a 7 x 7 matrix P (i.e. the permutation matrix P in Subsection II.B), and the value of each element in the matrix P is denoted in Equation (14). When the values of the input X are $[1 \quad 0 \quad 0 \quad 0]$, the values of the neurons in the third hidden layer (i.e. the ciphertext layer) Y are $[0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]$ (i.e. the values of ciphertexts).

Decryption can be performed from the third hidden layer (i.e. the ciphertext layer) to the output layer. In the neural network, the weights between the third hidden layer and the fourth hidden layer is a 7 x 7 matrix P^{-1} (i.e. the inverse matrix of the permutation matrix P in Subsection II.B). The weights between the fourth hidden layer and the fifth hidden layer is a 7 x 4 matrix R (i.e. the decoder matrix R in Subsection II.A), and the value of each element in the matrix R is denoted in Equation (3). The weights between the fifth hidden layer and the output layer is a 4 x 4 matrix S^{-1} (i.e. the inverse matrix of the scrambler S in Subsection II.B). When the values of the neurons in the third hidden layer (i.e. the ciphertext layer) Y are $[0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]$ (i.e. the values of ciphertexts), the values of the neurons in the output layer $[1 \quad 0 \quad 0 \quad 0]$ (i.e. the decrypted values).

C. Post-quantum cryptography neural network

Subsection III.B shows that the neural network can be constructed for McEliece cryptography method, so this study proposes an advanced PQC method based on neural networks. Non-linear functions can be adopted as the activation functions, and random numbers can be added to the neurons in the ciphertext layer (i.e. the third hidden layer in Fig. 3) for random perturbations and security improvement. From the ciphertext layer and the output layer, the random perturbations can be removed through the calculation of neural network to obtain plaintexts for decryption.

For the explanation of the proposed PQC neural network, the case in Subsection III.B is used to construct the neural network with an input layer, five hidden layers, and an output layer. For encryption, the structure of neural network includes four neurons (i.e. $X = \{x_1, x_2, \ldots, x_c\}$ and the number of inputs c is four) in the input layer, four neurons in the first hidden layer, seven neurons in the second hidden layer, and seven neurons in the third hidden layer (i.e. the ciphertext layer). The weights between the input layer and the first hidden layer is a 4 x 4 matrix S; the weights between the first hidden layer and the second hidden layer is a 4 x 7 matrix G; the weights between the second hidden layer and the third hidden layer is a 7 x 7 matrix P. Furthermore, the notation $\otimes$ is denoted as the non-linear activation functions in each layer for generating the n-dimension ciphertexts (i.e. n neurons in the ciphertext layer $Y = \{y_1, y_2, \ldots, y_n\}$) by Equation (21). For improving the security, random numbers can be added to the output of the ciphertext layer. For instance, the i-th neuron in the ciphertext layer y_i can add the random number r_i based the weight α by Equation (22). For decryption, the structure of neural network includes seven neurons in the fourth hidden layer, four neurons in the fifth hidden layer, and four neurons in the output layer. The weights between the third hidden layer and the fourth hidden layer is a 7 x 7 matrix L; the weights between the fourth hidden layer and the fifth hidden layer is a 7 x 4 matrix M; the weights between the fifth hidden layer and the output layer is a 4 x 4 matrix N for decrypting the ciphertexts and obtain the c-dimension plaintexts (i.e. $O = \{o_1, o_2, \ldots, o_c\}$) by Equation (23). Furthermore, the Mean-Square Error (MSE) function is adopted as a part of loss function to reduce the errors of decrypted data.

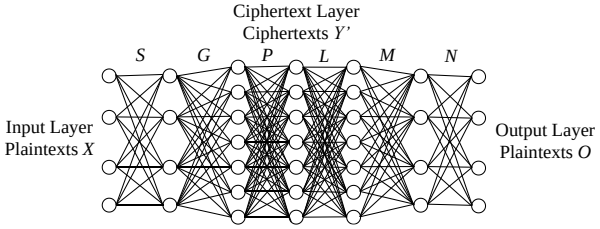

Fig. 3. PQC neural network

$$Y = X \otimes S \otimes G \otimes P \tag{21}$$

$$y_i' = y_i + \alpha \times r_i \tag{22}$$

$$O = Y' \otimes L \otimes M \otimes N \tag{23}$$

For improving the security of ciphertexts, the outputs of the ciphertext layer subject to an uniform distribution in this study. The ciphertext layer has n neurons, and the value of each neuron is normalized. The normalization of the i-th neuron value y_i' with a random number r_i is performed to be the variable h_i. This study considers m intervals and groups each variable into a range. Furthermore, the probability of each range can be estimated; for instance, the probability of the j-range is denoted as p_j. This study adopts the CDF value of chi distribution (i.e. (1 – p-value in chi-squared test) to analyze the outputs of the ciphertext layer for testing the subjection to uniform distribution. The chi-squared value χ^2 and the CDF value θ of chi distribution based on degrees of freedom (i.e. $m - 1$ in this case) are measured by Equations (24) and (25). The gamma function is denoted as $\Gamma\left(\frac{m-1}{2}\right)$, and the incomplete gamma function is denoted as $\gamma\left(\frac{m-1}{2}, \frac{\chi^2}{2}\right)$. If the CDF value is smaller than 0.5, the distribution of ciphertexts is similar to an uniform distribution with a significance. Therefore, this study adopts the CDF value θ in Equation (25) as a part of loss function $F(X)$ to improve the security of encrypted data by Equation (26).

$$\chi^2 = \sum_{j=1}^{m} \frac{\left(p_j - \frac{1}{m}\right)^2}{\frac{1}{m}} = m \sum_{j=1}^{m} \left(p_j - \frac{1}{m}\right)^2 \tag{24}$$

$$\theta = \frac{\gamma\left(\frac{m-1}{2}, \frac{\chi^2}{2}\right)}{\Gamma\left(\frac{m-1}{2}\right)} \tag{25}$$

$$F(X) = \theta + \frac{1}{c} \sum_{i=1}^{c} (o_i - x_i)^2 \tag{26}$$

In the proposed PQC neural network, the matrix set {S, G, P} is adopted as the public key for encryption, and the other matrix set {L, M, N} is adopted as the private key for decryption. The ciphertexts have random perturbations based on random numbers, and the distribution of ciphertexts is subject to an uniform distribution for obtaining secure ciphertexts. Furthermore, the random perturbations can be removed through the decryption neural network for restoring plaintexts. In this subsection, a small neural network is given as a case, but deeper and wider PQC neural networks with billion neurons can be designed for encrypting more data and provide higher security.

IV. Practical Experimental Results and Discussions

For the evaluation of the proposed PQC neural network, the data from the previous study [8] is adopted to perform practical experimental results. The cellular network signals are adopted as training data and testing data, and the encrypted cellular network signals can be obtained by the proposed PQC neural network. The number of inputs and outputs is 361 (i.e. c = 361), and the number of neurons in the ciphertext layer is 64 (i.e. n = 64).

For adding random perturbations, the random number r_i is added based on the weight α. This section evaluates the MSEs and CDF values under different weights, and the results are shown in Table 1. For comparison, the same random seed is considered to generate random numbers, so the CDF values of uniformed random numbers is 0.009227 (i.e. 0.009227 < 0.05) under different weights. In the practical experimental results, the distribution of ciphertexts is subject to an uniform distribution with a higher significance when the weight α is bigger than 0.4; for instance, the value of weight α is 0.4, the CDF value of ciphertexts Y' is 0.038992 with a significance (i.e. 0.038992 < 0.5). However, if the weight α is smaller, the MSEs of the output layer is lower (i.e. precise decrypted values); for instance, the MSE of the output layer is 3.74E-05, but the CDF value of ciphertexts Y' is 0.303414 with no significance (i.e. 0.303414 > 0.5) when the value of weight α is 0.1. Therefore, in this case, the value of weight α is recommended as 0.4 to provide uniformed ciphertexts and precise decrypted values.

TABLE I. The MSEs and CDF values under different values of α

Weight α	The MSEs of output layer	The chi-test CDF values of random numbers	The chi-test CDF values of ciphertexts Y'
0.1	3.74E-05	0.009227	0.303414
0.2	5.86E-05	0.009227	0.202276
0.3	6.84E-05	0.009227	0.089382
0.4	8.27E-05	0.009227	**0.038992**
0.5	0.0001	0.009227	**0.018034**
0.6	0.0001	0.009227	**0.013643**
0.7	0.0002	0.009227	**0.016572**
0.8	0.0002	0.009227	**0.012751**
0.9	0.0003	0.009227	**0.017519**
1	0.0003	0.009227	**0.017685**

V. Conclusions and Future Work

This study proposes a PQC neural network based on a code-based PQC method (i.e. McEliece cryptography method). In the proposed PQC neural network, non-linear activation functions, random perturbations, and uniformed ciphertexts with a significance are performed for improving the security of ciphertexts. In experiments, the practical cellular network signals are analyzed and encrypted by the proposed PQC neural network, and the distribution of the encrypted cellular network signals is subject to an uniform distribution (i.e. a lower CDF value) with precise decrypted values (i.e. a lower MSE). Although the small neural networks are presented in this study, deeper and wider PQC neural networks with billion neurons can be designed for encrypting more data and provide higher security.

In the future, more efficient and more secure activation functions can be designed for encryption and decryption with less computation time. Furthermore, the proposed PQC neural network can be applied to a wide variety of applications.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484-1509, 1997.

[2] G. Alagic et al., "Status report on the third round of the NIST post-quantum cryptography standardization process," *NIST Special Publication*, NISTIR 8413, 2022.

[3] H. Nejatollahi et al., "Post-quantum lattice-based cryptography implementations: a survey," *ACM Computing Surveys*, vol. 51, no. 6, p. 129, 2019.

[4] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *JPL Deep Space Network Progress Report*, vol. 42-44, pp. 114-116, 1978.

[5] J. -P. Thiers, J. Freudenberger, "Code-based cryptography with generalized concatenated codes for restricted error values," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 1528-1539, 2022.

[6] T. Sahraneshin et al. "Securing communications between things against wormhole attacks using TOPSIS decision-making and hash-based cryptography techniques in the IoT ecosystem," *Wireless Networks*," vol. 29, pp. 969-983, 2023.

[7] C. Guo et al., "De-correlation neural network for synchronous implementation of estimation and secrecy," *IEEE Communications Letters*, vol. 27, no. 1, pp. 165-169, 2023.

[8] L. Wu, C. -H. Chen, Q. Zhang, "A mobile positioning method based on deep learning techniques. *Electronics*, vol. *8*, no. 1, p. 59, 2019.