Cryptanalysis of Caesar using Quantum Support Vector Machine

Hyun-Ji Kim*, Gyeong-Ju Song*, Kyung-Bae Jang*, Hwa-Jeong Seo*
*IT Department, Hansung University, Seoul
{khj1594012, thdrudwn98, starj1023, hwajeong84}@gmail.com

Abstract

Recently, artificial intelligence-based cryptanalysis techniques have been researched. In this paper, we find the key of the Caesar cipher, which is a classical cipher, by using a quantum machine learning algorithm that learns by parameterized quantum circuit instead of a classical neural network. In the case of 4-bit plaintext and key, results could not be obtained due to the limitations of the cloud environment. But in the case of 2-bit plaintext and key, an accuracy of 1.0 was achieved, and in the case of 3-bit plaintext and key, an accuracy of 0.84 was achieved. In addition, as a result of cryptanalysis for a 2-bit dataset on IBM's real quantum processor, a classification accuracy of 0.93 was achieved. In the future, we will research a qubit reduction method for cryptanalysis of longer-length plaintext and key, and a technique for maintaining accuracy in real quantum hardware.

Keywords: Cryptanalysis, Quantum support vector machine, Quantum computer

1. Introduction

Cryptographic systems are designed to be secure so that the key cannot be inferred or recovering plaintext through confusion and diffusion. Cryptoanalysis has various methods such as differential cryptanalysis and linear cryptanalysis. Recently, due to the development of artificial intelligence technology, studies on encryption analysis through artificial neural networks have been actively conducted[1,2]. In this paper, we intend to perform cryptographic analysis using an artificial intelligence model on a quantum computer rather than a conventional neural network.

2. Related Works

2.1 Support Vector Machine and Quantum SVM

A support vector machine (SVM) is one of the supervised machine learning algorithms that finds the optimal boundary between data points through a hyperplane. The hyperplane is $n-1$ dimensional, and is used to separate the n-dimensional space. A kernel is used to separate data points, and the kernel arranges various hyperplanes so that the data points can be divided well. To find the hyperplane, a nonlinear function must be applied to the data, which is called a feature map. The functions used include various kernel functions such as polynomial, gaussian, and sigmoid functions. A quantum support vector machine (QSVM)[3] performs the kernel operation of classical SVM on quantum computer and proceeds in the same way as the existing process. Because QSVM is advantageous for working with more dimensional data, it benefits from kernel optimizations, which are difficult for SVMs to handle, and generally outperforms classical SVMs.

2.2 Quantum Circuit [4]

Quantum circuits are constructed through bits and gates like classical logic circuits, and instead of classical bits and gates, qubits and quantum gates that utilize the principles of superposition and entanglement of quantum mechanics are used to operate on a quantum computer. A qubit performs the same role as a classical bit, but through the superposition state, all values exist as probabilities and are determined as a single value when observed. In addition, for one logical qubit (without an error), several physical qubits are required, and an error correction technique is required. However, there is still a lot of overhead with these techniques.

Figure 1 shows some of the quantum gates used in quantum circuits. The Hadamard (H) gate overlaps so that it can have the states of 0 and 1 at the same time in the initial state of the qubit. Also, when the same qubit goes through the gate again, it returns to its original state. The X gate changes the state of a qubit. And in a superposition state, it changes the probability. The $CNOT$ gate applies a NOT operation to the second qubit when the first qubit is 1, and the $Toffoli$ gate applies a NOT operation to the last qubit when both the preceding two qubits are 1. That is, both gates are gates that can observe the entanglement state. Finally, the $SWAP$ gate exchanges the values of two qubits, and $CSWAP$ gate exchanges the values of the following two qubits by the state of the first qubit. When designing quantum circuits, it is necessary to design faster and more efficient circuits by reducing the use of expensive gates such as $CNOT$ or $Toffoli$ gates and reducing the circuit depth (the width of the circuit). In addition, as the depth decreases, the quantum coherence decreases, resulting in higher accuracy.

Quantum circuits can be implemented using host languages such as Python, Java, and C# and quantum

language QASM in various frameworks such as Qiskit, ProjectQ, and Q#.

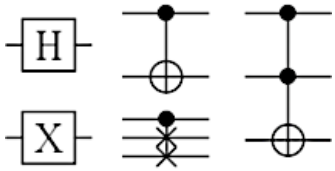

Figure 1 Quantum gate

2.3 Quantum Neural Networks (QNN)

A quantum neural network is an artificial intelligence that utilizes quantum mechanics phenomenon (entanglement and superposition). Quantum neural network consists of qubits and quantum gates on a quantum computer. Therefore, it learns quantum state data (parameterized quantum circuit) by encoding the classical data into quantum data. The parameters of the circuit are set using the input data, and each qubit passes through gates and then the value changes. When qubits are observed, the state of the qubits is determined. Through this process, a quantum neural network works.

3. Proposed Method

In this paper, we propose a cryptanalysis method on a quantum computer using QSVM for Caesar cryptography. Since the system performs a known-plaintext attack, it is a structure that finds the used key through a pair of plaintext and ciphertext. The overall process is shown in Figure 2. As mentioned in Section 2.3, after encoding classical data into quantum data, training is performed through a quantum circuit. That is, the circuit runs, measures the qubits, and changes the parameters over several iterations. Through this process, a probability value indicating which class the input data will be classified into is returned, enabling data classification through a quantum circuit.

3.1 Dataset

As mentioned earlier, since we perform a known plaintext attack, the data set configuration is shown in Figure 2. Each plaintext and ciphertext is represented by bits. And since the key value is used as a label, it is expressed as a decimal number. That is, the plaintext and ciphertext bits become input data, and the key becomes the label.

Plaintext bit		Ciphertext bit		Key
0	1	0	0	3
⋮	⋮	⋮	⋮	⋮
1	1	0	1	2
		Data		Label

Figure 2 Dataset for cryptanalysis of the Caesar cipher

3.2 Data encoding

Data encoding is the process of transforming classical data ($\vec{x_i}$) into a quantum state ($\Phi(\vec{x_i}) >$) in Hillbert space. Since the input data is expressed as a parameterized quantum circuit, the parameters of the corresponding quantum circuit are affected by the input data. In addition, the circuit acts as a kernel of QSVM because a non-linear function is applied to the input data. Therefore, the quantum circuit construction process is the kernel construction process of QSVM, and this quantum circuit becomes a feature map. Qiskit provides three feature maps, and among them, ZZFeaturemap is used in the proposed method. Currently, a gate that can express an expression representing this feature map is not provided, so a combination of two types of gates is used. If the plaintext, ciphertext and key are 2-bit, the parameterized quantum circuit is configured as shown in Figure 3. First, a Hadamard gate (H) is applied to all qubits so that the input data is in a superposition state. Then, the input data is assigned to the classical nonlinear function (P) according to Equation 2. In addition, each qubit is entangled through the $CNOT$ gate, and the values of other qubits are determined according to the state of one qubit. In addition, the qubits can be controlled through the linear option that makes the qubit entangled with one next qubit, or the full option that makes the qubit entangled with all the qubits following the qubit. In the case of linear, since the circuit depth is smaller, the execution time is shorter, and as mentioned earlier, higher accuracy can be obtained due to less depth, so the linear option is used. Through the above process, a quantum circuit that serves as the kernel of the QSVM is constructed.

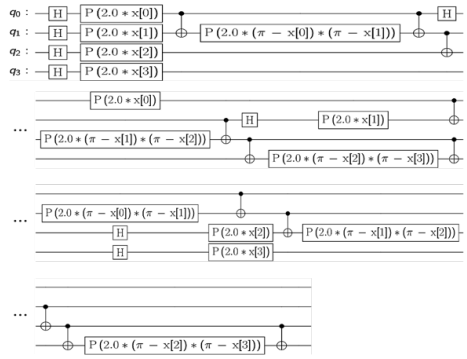

Figure 3 Quantum circuit of ZZfeaturemap with 4 qubit (plaintext, ciphertext and key are 2-bit)

$$\Phi_{u,v}(\vec{x}) = (\pi - x_u)(\pi - x_v) \tag{2}$$

3.3 Training and classification

By repeatedly executing the designed quantum circuit, the parameters of the circuit are updated. Measurements are performed on each qubit to determine the state of the qubit with a single value, and the measurement can be performed multiple times to classify them with high probability. The trained

quantum circuit can act as a classifier. Therefore, when test data is input, inference is performed.

4. Experiment and evaluations

In this experiment, Google Colaboratory, a cloud-based service, was used, and it supports Intel Xeon CPU (25GB RAM), Nvidia GPU (25GB RAM) and Ubuntu 18.04.5 LTS. As the programming environment, Python 3.7.11 and Qiskit library were used. In Qiskit, real quantum hardware from IBM can be used, but a token is required, so the experiment was conducted through a simulator.

4.1 Dataset

As described in Section 3.1, plaintext and ciphertext are expressed in bits and then concatenated and used as input data, and the key value for the data becomes a label. That is, the data set is configured so that the plaintext and ciphertext pairs are classified into a class corresponding to the key value used for encryption. In the case of 2-bit plaintext and keys, it is a problem of classifying into four classes, and in the case of 3-bit plaintext and keys, it becomes a multi-classification problem of classifying into eight classes. Figure 4 shows part of the actual dataset, and Figure 5 shows the distribution of data points in 3-bit plaintext and key. In Figure 5, dots of the same color are data with the same class. As shown in Figure 4, data is composed of 0 and 1, but the left side in Figure 5 is normalized to a value between -1 and 1 that quantum states can have. Even the right side in Figure 5, dots of the same color are data having the same class, but they have values of 0 or 1 without normalization. In addition, plaintext and ciphertext are concatenated, and each bit must be assigned to each qubit. Therefore, $2n$-qubits are required for n-bit plaintext and key.

	0	1	2	3
0	0	1	0	1

	0	1	2	3	4	5
0	0	1	1	0	0	1

Figure 4 2-bit(left) and 3-bit(right) dataset (plaintext bit and ciphertext bit)

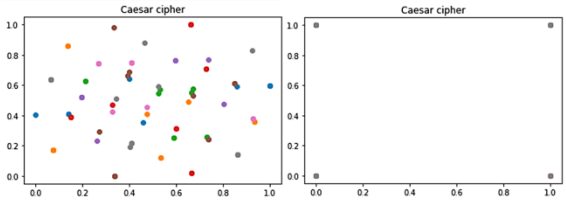

Figure 5 Visualization of data points in 3-bit dataset (left: normalized, right: unnormalized)

4.2 Quantum circuit

After composing the data set as in Section 4.1, the encoding process of converting classical data into quantum state data is performed. In other words, it is necessary to design a parameterized quantum circuit whose parameters change according to the input data. And it should be configured as a non-linear operation so that it can perform the function of the feature map (kernel) of QSVM. In the case of a 2-bit plaintext and key, each bit of the input data was input by using 4 qubits to correspond to the qubits, and the case of 3-bit is the same. Figure 3 is a quantum circuit for 2-bit plaintext and key, and Figure 6 is a quantum circuit for 3-bit plaintext and key. The circuit configured as described above is repeatedly performed as many times as the number of repetitions, and each qubit is measured by repeating as many times as the set shots value. Through this process, data encoding, training, and inference are performed.

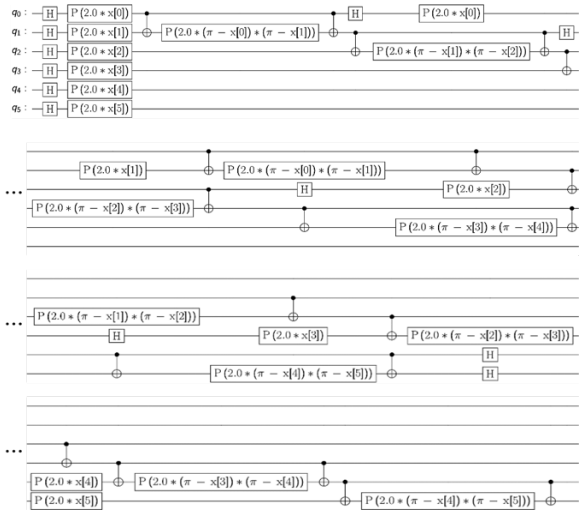

Figure 6 Quantum circuit (3-bit plaintext and key)

4.3 Evaluation

Table 1 shows the details of the quantum circuit when the encryption key is found with the highest accuracy, and is shown for two bit-type datasets. We performed cryptanalysis with a quantum circuit composed of these hyperparameters, and Table 2 shows the results. In the case of the 2-bit dataset, when the number of shots, which is the number of times to measure the state of each qubit, was 5, accuracy of 1.0 was achieved, and all keys were found. In the case of the 3-bit dataset, the highest classification performance was achieved when the shots value was 150, with 0.84. When the length of the plaintext and key increased from 2-bit to 3-bit, the accuracy decreased by about 0.16. In addition, as the value of shots increased, the accuracy tended to increase, and when it was measured more than a certain number of times, the accuracy decreased.

However, in the bit-type dataset, when the plaintext increases by 1-bit, the number of required qubits increases by two, so it was not possible to experiment with the 4-bit dataset due to the limitations of the cloud environment (runtime shutdown and RAM usage exceeded) and the token problem. Therefore, after expressing the hexadecimal plaintext and the ciphertext in decimal, the data set of the float type was constructed by normalizing it to a value between -1 and 1. In this method, only 2-qubit is

required even if the size of the plaintext and key increases. As can be seen in Table 3, the classification performance is worse than the proposed method.

Table 1 Details of quantum circuits (bit type dataset : 2-bit dataset (2-bit plaintext and key), 3-bit dataset (3-bit plaintext and key))

	2-bit dataset	3-bit dataset
Gate	H, $CNOT$, P (non-linear)	
Repetition	2	2
Shots	5	150
The number of qubits	4	6
The number of classes	4	8
Optimization function	Simultaneous Perturbation Stochastic Approximation	

Table 2 Accuracy of classification for cryptanalysis (with bit type dataset)

Shots	2-bit dataset	3-bit dataset
1	0.66	0.6
5	1.0	0.7
100	-	0.81
150	-	0.84

Table 3 Accuracy of classification for cryptanalysis (with float type dataset)

Shots	2-bit dataset	3-bit dataset
5	0.33	0.19
100	0.46	0.33
150	0.53	0.46
500	0.46	0.4

Table 4 shows the execution time for each data type. First, execution time is also affected by shots, but the difference is not large. It can be seen that it is determined according to the number of qubits used. Even if only 2-qubit increases from 4-qubit to 6-qubit, there is a difference of about 1600 seconds in execution time. The float type dataset takes less time to execute because the number of qubits is small, but we think that the performance is not good due to the small amount of data information. On the other hand, the bit type dataset has a high data dimension, so the number of qubits is relatively large. Therefore, it takes a long time to perform, but achieves high classification accuracy.

Table 4 Execution time according to qubits and shots (unit : s)

			The number of shots	
			5	150
Bit type	The number of qubits	4	142.21	144.94
		6	1618.06	1867.55
Float type		2	110.27	114.09

Table 5 shows the results performed on IBM's actual quantum processor. Currently, we do not have tokens, so we can only use basic hardware, so we cannot use more than 5-qubit. The actual hardware used is 'ibmq_bogota', which provides 5-qubits and 32 quantum volumes. Therefore, we performed the experiment only on the 2-bit dataset. This dataset showed the highest accuracy when the shots were 5, so we tested that case. When the circuit was executed on the simulator, it took 142.21 seconds, but when the actual quantum processor was used, it took about 5.5 times (780 seconds). In addition, the accuracy decreased by 0.07 due to errors such as noise generated in the operation process in the real processor.

Table 5 Execution time and accuracy on real quantum hardware (unit : s, the number of shots : 5, only 2-bit dataset : 4-qubits)

Execution time	Accuracy
780	0.93

5. Conclusion

In this paper, we propose a cryptanalysis technique to find the key of the caesar cipher, a classical cipher, through the Quantum Support Vector Machine, a machine learning algorithm on a quantum computer. Therefore, it uses quantum state data and quantum circuits (which perform non-linear operations like the kernel of SVM) rather than classical data. Due to the limitations of the experimental environment, experiments were performed on 2-bit and 3-bit plaintext and keys, and accuracies of 1.0 and 0.84 were achieved, respectively. In a real quantum processor, an accuracy of 0.93 (for 2-bit dataset) was achieved. In addition, as the number of qubits increased, the execution time increased or the execution was not completed. We will perform cryptanalysis on longer-length plaintext and key in the future. Finally, we think that additional control algorithms should be used or the number of required qubits should be reduced to maintain accuracy on the real processor.

References

[1] Gohr, Aron. "Improving attacks on round-reduced speck32/64 using deep learning." *Annual International Cryptology Conference.* Springer, Cham, 2019.

[2] Jain, Aayush, Varun Kohli, and Girish Mishra. "Deep Learning based Differential Distinguisher for Lightweight Cipher PRESENT." *IACR Cryptol. ePrint Arch.* 2020 (2020): 846.

[3] Rebentrost, Patrick, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification." *Physical review letters,* 113.13 (2014): 130503.

[4] Sleator, Tycho, and Harald Weinfurter. "Realizable universal quantum logic gates." *Physical Review Letters* 74.20 (1995): 4087.