3D CUBE Algorithm for the Key Generation Method: Applying Deep Neural Network Learning-Based

JUNGHA JIN¹ AND KEECHEON KIM²
¹Department of Computer, Information, and Communications Engineering, Konkuk University, Seoul 05029, South Korea
²Department of Computer Science and Engineering, Konkuk University, Seoul 05029, South Korea
Corresponding author: Keecheon Kim (kckim@konkuk.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) funded by the Korea Government (MSIT) under Grant 2017-0-00275-003, through the Development of protecting technologies of LPWAN for deploying secure private IoT network, and in part by the National Research Foundation of Korea (NRF) funded by the Korea Government (MSIT) under Grant NRF-2017M3C4A7083678, through the Next-Generation Information Computing Development Program.

ABSTRACT
Current encryption systems are run with a hybrid mode in which symmetric and asymmetric key methods are mixed. This hybrid mode is devised to employ the fast processing speed of the symmetric key while circumventing the difficulty of providing services due to computational complexity of the asymmetric key method by limiting its use to secured key exchange. To implement a secured symmetric key-based cryptographic system to circumvent the crack problem due to the emergence of quantum computers in asymmetric key exchanges, the problem of sharing a key transferred in the most secure manner should be solved while symmetric keys in use are kept up-to-date. This paper proposes a three-dimensional (3D) cube algorithm that can be used by creating the up-to-date key in a symmetric key encryption system to provide security resistance in a quantum computer environment. More specifically, it presents a solution to the secured key sharing method that can minimize the damage due to the pre-shared key (PSK) leakage. This is done by using the method of inducing the symmetric key creation based on deep neural network learning without sharing the PSK between systems to securely maintain the key while creating and using the key that is variably used through the symmetric key encryption system of the 3D cube algorithm. Thus, the proposed algorithm secures confidentiality and integrity of data transferred over a network because information that can be obtained by malicious attackers is small (because the symmetric key used in encryption and decryption is induced without exchanging the PSK with the application of deep neural network learning).

INDEX TERMS
Artificial neural networks, encryption, information security, public key, random number generation.

I. INTRODUCTION
Modern society has faced a rapid change in digital environments due to the Fourth Industrial Revolution. The destructive technological change throughout the whole of society is referred to as digital revolution and affects even the way individuals live. Amid this rapid change, it is necessary to analyze fundamental causes and have effective response systems by investigating application of innovative and creative technologies from diverse viewpoints, departing from existing approaches to research and development that focus on analyzing only advanced technologies.

The associate editor coordinating the review of this manuscript and approving it for publication was Shuihua Wang.

One of the most important reasons for protecting the data that are created and distributed through information technology (IT) is to deliver accurate information from sender to recipient while preventing data leakage to a malicious third party. The originality of the data, unchanged from the creation of the data, should be maintained to achieve reliable information delivery. To do this, data should have reliability during the data creation process, integrity in the distribution and management process, and authenticity (by which reliability and integrity can be proven). Thus, a method to guarantee data confidentiality and integrity should be reflected in the data distribution system to create, distribute, and store data with reliable value; to this end, encryption and decryption systems should be used.
This study proposes a method that strengthens security and guarantees distributed data confidentiality and integrity by implementing a secure key generation and management technique through the proposed three-dimensional (3D) CUBE algorithm in an environment where the symmetric cryptographic technique is used. The aim here is to provide a secure communication environment by implementing a symmetric key system without key exchange through the application of deep neural network (DNN) learning, an artificial intelligence (AI) technology, replacing the existing shared key exchange method to ensure secure exchange of keys used in the symmetric key encryption algorithm. To prove this, this study aims to demonstrate the execution of improved security strengthening through performance comparison with an existing hybrid-mode cryptographic system.

This paper is organized as follows: In the present chapter, the background and purpose of the study are explained. In Chapter 2, the key management technique using the 3D CUBE algorithm proposed in this study is explained. In Chapter 3, the performance of the encryption system, in which the 3D CUBE algorithm presented in Chapter 3 is implemented and applied, is verified, and the performance is compared with that using another method in the 3D CUBE algorithm system to verify the effectiveness of the proposed method. Finally, Chapter 4 derives conclusions regarding the performance verification of the system to which 3D CUBE algorithm is applied.

II. DEEP NEURAL NETWORK LEARNING-BASED 3D CUBE ALGORITHM

A. 3D CUBE ALGORITHM

In this study, the 3D CUBE algorithm is proposed to solve the problem of secure key exchange in the symmetric key encryption algorithm. A symmetric key encryption algorithm is normally used in current encryption and decryption systems because it shows excellent performance in terms of speed. When keys used in encryption and decryption are leaked, data confidentiality and integrity are damaged. Accordingly, a hybrid method, in which keys used in symmetric keys are delivered using an asymmetric key mode, is used in a mixed manner [1]–[6]. However, as the security of the asymmetric key is threatened due to the development of quantum computing, studies are needed to securely exchange keys to provide strengthened security in existing symmetric key encryption systems by reducing their vulnerability [7], [8].

The 3D CUBE algorithm proposed in this study is implemented based on DNN learning, which induces a secret key through learning, thereby minimizing the prior shared information in order to reduce the damage due to data leakage. In addition, it provides more secured encryption communication by excluding the sharing of secret keys and increasing the complexity of key management by inducing a key rather than using a method that transfers used keys [9]. This is a methodology that can acquire a secret key through artificial neural network (ANN) learning, in which the sending side transmits only a pattern sequence of 3D CUBE as a key used in encryption and, using this sequence, the receiving side induces a secret key by employing the ANN applied to the decryption system and utilizes the induced secret key to decipher the cipher text [10]–[14].

To solve the transmitted arbitrary 3D CUBE pattern, the DeepCube algorithm, which is run by DNN learning, is used, and this algorithm performs an operation to solve a transmitted arbitrary pattern by modifying the previous algorithm that restores a shuffled cube into the original state in the Rubik’s cube [15]–[20]. To solve the pattern, various cases occur, and the accurate secret key can be induced by managing the optimum policy reward table through the ANN learning.

B. TERM DEFINITION OF 3D CUBE ALGORITHM

In the 3D CUBE algorithm, a 3D cube is configured and used. Terms defining the cube surface used in this study and terms of operation that are used to shuffle the cube are given as follows:

1) TERM DEFINITION OF CUBE SURFACE IN THE 3D CUBE ALGORITHM

- Cube’s upper surface: Top (T)
  - This refers to the uppermost line on the cube’s reference surface in the horizontal direction.
- Cube’s middle surface: Middle (M)
  - This refers to the middle line on the cube’s reference surface in the horizontal direction.
- Cube’s lower surface: Core (C)
  - This refers to the lowermost line on the cube’s reference surface in the horizontal direction.
- Cube’s left surface: West (W)
  - This refers to the leftmost line on the cube’s reference surface in the vertical direction.
- Cube’s right surface: East (E)
  - This refers to the rightmost line on the cube’s reference surface in the vertical direction.
- Cube’s center axial surface: Axis (A)
  - This refers to the middle line on the cube’s reference surface in the vertical direction.

2) TERM DEFINITION OF CUBE MOVEMENTS IN THE 3D CUBE ALGORITHM

- Cube’s rotation in the left direction: Left (L)
  - This refers to the motion in the left direction in the cube’s reference surface line in the horizontal direction.
- Cube’s rotation in the right direction: Right (R)
  - This refers to the motion in the right direction in the cube’s reference surface line in the horizontal direction.
- Cube’s rotation in the upper direction: Upper (U)
  - This refers to the motion in the upper direction in the cube’s reference surface line in the vertical direction.
- Cube’s rotation in the lower direction: Down (D)
  - This refers to the motion in the lower direction in the cube’s reference surface line in the vertical direction.
3) TERM DEFINITION OF CONCEPTS IN THE 3D CUBE ALGORITHM
   ① Cube shuffle: Act of shuffling a cube
   ② Cube pattern: Information of entire coordinate values in the reference cube surface
   ③ Phase: Information about the order of shuffling a cube.
   ④ Sequence: Derived from phase and corresponding pattern information

C. OPERATING PRINCIPLE OF 3D CUBE ALGORITHM
In the 3D CUBE algorithm, the 3D cube structure is utilized to manage a key in the encryption communication in a novel way, and the operation is conducted based on the pattern created from the sending side rather than delivering the entire cube structure, thereby performing an ANN operation to solve the pattern delivered from the receiving side. A series of moves that solve the 3D CUBE pattern are used as a sequence, through which a key that is used in encryption and decryption is inferred. The idea of the 3D CUBE algorithm proposed in this study is based on a method that solves the Rubik’s cube pattern.

The sender (one of the participants in encryption communication) performs a shuffle rotation in relation to the randomly extracted number of 3D CUBE using an ANN consisting of DNN, and a secret key is created by having the order that each shuffle progresses and the pattern information at the time as a sequence. It also delivers the finally derived pattern information to the receiver. The receiver infers the key using a deep cube algorithm, which is an ANN consisting of DNN, to derive the delivered pattern, and transfers the hashed shuffled number using the inferred key to the sender, thereby completing the confirmation procedure. If the key information transferred from the receiver is not matched, the sender transmits the same pattern again to the receiver. If the key information is accurately matched, an OK signal is transmitted according to the protocol, and data to be delivered to the receiver are transmitted after being encrypted using the derived key. Here the process to perform the synchronization of the deep cube algorithm used by the sender and receiver is regarded as the process of solving the 3D CUBE pattern. Figure 1 shows the diagram depicting the process of executing 3D CUBE shuffle through the DeepCube algorithm ANN [21]–[26].

**FIGURE 1.** Cube shuffle move in DNN utilizing the back-propagation of MCTS.
The proposed method uses a key by deriving it via completing the shuffle of 3D CUBE based on AI to decrypt the cipher text delivered from the receiver, which resolves the problem of secured key exchange by omitting the key exchange process from the existing symmetric key encryption algorithm. The 3D CUBE algorithm can provide better security than the existing symmetric key encryption algorithm because it is difficult to infer a key used by a third party using AI-based learning. The operation after deriving the key in the 3D CUBE algorithm is the same as that in the symmetric key encryption system as it is used to manage keys and distribute them securely in the symmetric key encryption system.

Figure 2 shows the operation flow of the 3D CUBE algorithm. In the symmetric encryption system, the sender transmits a 3D array key pattern as a shuffled 3D CUBE pattern, by which the key can be induced, rather than sharing of keys used during communication between participants of initial encryption communication, and the receiver acquires the secret key by inducting it through AI-based learning. Here, the secret key is created based on the shuffled order as a sequence through performing XOR operation on the 3D CUBE pattern of the sequence.

1-1 The sender and receiver request communication, and a 3D array key pattern consisting of 3D CUBE pattern is transmitted.
1-2 The sender extracts the 3D CUBE pattern according to the shuffle order.
1-3 The secret key is created by performing XOR operation on the extracted output.
2-1 The receiver generates a key based on the result induced by learning through the DeepCube algorithm with regard to the delivered key pattern.
2-2 The result of XOR operation on the number of shuffles using the induced key is transmitted to the sender.

3-1 An OK signal is transmitted if the key delivered from the receiver is matched.
3-2 If the key is not matched, the key pattern is re-transmitted to allow the receiver to perform learning again.

By using the above method, the receiver can induce a secret key used in the symmetric key encryption system through the key pattern learning delivered by the sender, which can provide secured encryption communication by minimizing possible information leakage. This increases the complexity through the 3D CUBE’s shuffle, thereby providing more enhanced security, and faster data transmission in the symmetric key encryption system can be possible as a result of the induced key use after the initial key induction process. In addition, the key’s up-to-datedness can be maintained because key management can be done by setting the threshold of the induced key’s life cycle, thereby ensuring the secured key exchange and key up-to-datedness, which were problems in existing symmetric key encryption algorithms.

The key management techniques used in existing symmetric key encryption algorithms struggle both to provide strengthened security due to the simplicity of the structure and to apply an appropriate level of security according to information importance, computerized allowance of equipment and networks, and limitations of available resources. However, if the key management is performed through the application of the proposed 3D CUBE algorithm, flexible scalability can be provided in contrast with the formalized secret key management structure provided in existing symmetric key encryption systems. The CUBE’s shape provided in the 3D CUBE algorithm can be basically $3^3$, $2^2$, or $9^9$, which can be applicable to various cubic structures and selectively provides flexible complexity such as lightweight structure or reinforced structure.
D. DESIGN OF THE 3D CUBE ALGORITHM

The basic structure of the 3D CUBE algorithm proposed in this study is a $3^3$ structure. It is designed to generate a 256-bit key to be operated in AES-256 algorithms. The same 3D CUBE algorithm is used in the encryption and decryption, and DeepCube algorithm, which is an AI-based technique, is applied and used to solve the shuffled pattern of 3D CUBE.

1) BASIC CONFIGURATION OF THE 3D CUBE ALGORITHM

The 3D CUBE algorithm is a key management algorithm used to generate a 256-bit secret key to be used in the AES-256 symmetric key encryption algorithm [27]–[30]. This algorithm generates a pattern by manipulating the 3D CUBE array to create a 256-bit secret key, and transmits the generated pattern, and the receiver side employs a secret key by inducing it through AI-based ANN learning. In existing symmetric key encryption systems, a prior shared key is transmitted and used as a secret key, thereby achieving fast encryption and decryption performance. However, this causes a problem of secured transmission of the secret key, resulting in degraded data confidentiality and integrity. Thus, the proposed 3D CUBE algorithm is applied to strengthen the security by changing the key management method in the existing symmetric key encryption system into a key induction method rather than using a key exchange method.

The 3D CUBE algorithm proposed in this study was implemented as a $3^3$ size, but it can be expanded to any $n^3$ size due to the characteristics of 3D CUBE, which will increase the number of cases and complexity to create a key. In addition, because there is a pool of various methods to solve the pattern, even if a third party acquires the key pattern delivered to the receiver from the sender, securer key management can be possible due to the difficulties to induce the key accurately.

The existing Rubik’s cube considers only color, but the proposed 3D CUBE algorithm should have data that correspond to each coordinate value. The pattern created finally at the sending side is delivered utilizing the coordinate values, and Monte Carlo Tree Search (MCTS) configuration is operated at the receiving side through the ANN consisting of DNN to produce the delivered pattern [10]–[12].

The coordination array implemented according to the 3D CUBE generation algorithm is shown in Figure 3, in which the coordinate value is assigned as a combination of alphabet letters that indicates the same surface and a sequence number. The coordinate value for the inside other than the surface is not considered.

2) PROCESS TO GENERATE A SECRET KEY IN 3D CUBE ALGORITHM

The 3D CUBE algorithm proposed in this study extracts a pattern in relation to the move in each sequence and the order while performing shuffle operations of the initially arranged 3D CUBE, and the pattern is hashed through SHA 256 followed by XOR operation on each of the values, thereby finally creating a 256-bit secret key by performing XOR operation on each extracted data.

The difference in method between the existing Rubik’s cube solution and 3D CUBE algorithm is presented in Table 1, in which the existing Rubik’s cube solution is based on the colors of all sides, whereas 3D CUBE aims to solve an arbitrary pattern. Thus, the 3D CUBE increases the number of cases and complexity because it should input a coordinate value based on the array elements inside the cubic in the 3D CUBE and then derive a pattern that is matched with the corresponding coordinate value.

The 3D CUBE algorithm proposed in this study checks whether the pattern is matched with 54 3D array coordinates, in which an order of shuffling is defined as a sequence to solve the pattern, and a secret key is generated by mapping the data derived by XOR operation on the coordinate value for each move with the final pattern. At the receiving side, a secret key used in the symmetric key encryption system is induced through the pattern-solving moves of the 3D CUBE delivered from the sending side, and the result of XOR operation on the number of sequences and derived secret key is delivered to the sending side for verification. To generate a secret key in the 3D CUBE algorithm, the following details are defined and applied. Figure 4 shows the element names of the model generated by the 3D CUBE algorithm. Figure 5 shows the overview of the shuffle rotation move in the 3D CUBE algorithm.
FIGURE 4. Naming rule of 3D CUBE portions.

1. Face_Def-1: upper face is called Top (T)
2. Face_Def-2: middle plane is called Middle (M)
3. Face_Def-3: lower Plane is called Core (C)
4. Face_Def-4: left side is called West (W)
5. Face_Def-5: right side is called East (E)
6. Face_Def-6: middle axis is called Axis (A)

FIGURE 5. Rotation rule of 3D CUBE shuffle.

1. Turn_Def-1: Rotate Left (L)
2. Turn_Def-2: Rotate Right (R)
3. Turn_Def-3: Rotate Upper (U)
4. Turn_Def-4: Rotate Down (D)

The shuffle is performed through the random value of the number of shuffles via the deep cube algorithm ANN in the 3D CUBE algorithm using the defined rotation rule and portion names. Here, an MCTS is configured through the ANN while shuffling and is managed by setting a weight of corresponding shuffle moves by back-propagation of finally derived cube pattern results [13]–[16]. The MCTS derives a cube pattern while learning through iterating selection, expansion, simulation, and back-propagation several times, and a secret key is generated based on the information extracted through the sequence and pattern in each process. The finally derived pattern is transmitted to the receiving site to start operation.

If the shuffling result using the deep cube algorithm is two rotations of the upper surface in the left direction followed by three rotations of the left surface in the lower direction, one rotation of the middle surface in the right direction, and two rotations of the lower surface in the left direction, then the subsequent sequence pattern can be extracted. A secret key is generated based on the values extracted through XOR operation on the sequence extracted in the above and pattern information at each stage, which are then hashed through SHA 256, and the finally extracted entire values are subject to XOR operation. While the generated secret key is stored in the sending side, the receiving side performs learning through the deep cube algorithm (an ANN consisting of DNN) to solve the delivered cube pattern and transmits the data calculated by XOR operation on the derived key and the number of shuffles, n, based on the pattern derived through learning. Once the verification request of the secret key derived at the receiving side is sent to the sending side, the sending side compares the received data with the result of XOR operation on the secret key owned by the sending side and the number of shuffles to verify whether the key has been accurately derived. If the key is matched, the induced key is used as a secret key to perform data encryption and the result is transmitted to the receiving side. Then, the receiving side performs decryption of the cipher text using the induced key to acquire a plain text. Figure 6 shows the method to create a secret key at the sending side through the CUBE algorithm.

Since four rotations in the same direction in the shuffle rotation of 3D CUBE algorithm entails a rotation back to the original position, the shuffle logic was implemented to allow only one to three rotations in the same direction.

In the 3D CUBE algorithm, a secret key is generated based on the extracted value from each sequence and corresponding derived pattern.
pattern while performing shuffling. If a secret key is generated using the aforementioned sequence pattern example, it can be explained by the moves described below.

In the 3D CUBE algorithm, after hashing the CUBE face pattern derived through the shuffling results and a move sequence to make each pattern through SHA 256, respectively, a sequence key is generated using XOR operation, and once all sequences are complete, the stored sequence keys are XOR operated to generate a secret key. In the 3D CUBE algorithm, an arbitrary pattern is generated by shuffling the CUBE face pattern. Here, the shuffled order is extracted as a sequence and hashing is performed through SHA 256. Finally, a secret key used in the key encryption system is generated through XOR operation on CUBE face pattern values.

3) PROCESS TO INDUCE A SECRET KEY IN 3D CUBE ALGORITHM

The 3D CUBE algorithm proposed in this study induces a secret key without sharing the secret key. Thus, it is necessary for the receiving side to solve the 3D CUBE pattern delivered from the sending side from the initially arranged 3D CUBE to perform encryption communication. This is done using the DeepCube algorithm, which is an AI technique that solves a CUBE pattern.

In the 3D CUBE algorithm implemented and tested in this study, the basic pattern is defined as “A face surface” consisting of a coordinate values, and a task is performed to return the delivered CUBE pattern back to the A face surface. Here, the DeepCube algorithm that is used in the 3D CUBE algorithm to return the pattern evaluates whether the existing CUBE layout can be improvable after receiving the 3D CUBE to be solved, and iterates learning a process, starting from the CUBE completed through the automatic learning iteration system into a solved version of the CUBE to be solved. It searches possible movements to solve the 3D CUBE pattern using the standard search tree, and rotates the CUBE to solve the number of cases.

The DeepCube algorithm can solve any arbitrarily arranged CUBE pattern using fewer than 30 movements on average. The 3D CUBE algorithm that is completed by applying the learning method in the DeepCube algorithm in this study searches for a sequence that solves the pattern out of various cases when an arbitrary pattern is given in the cube model. A series of processes that induce a secret key through the 3D CUBE pattern at the receiving side is explained below:

1) Start solving of CUBE pattern delivered from the sending side.
2) Arrange the pattern in the reverse direction of the movements to solve the CUBE pattern.
3) Secret key is derived using the extracted sequence.
4) Hashing the derived secret key and sequence number \( n \) and sending the result to the sending side.
5) Termination of secret key derivation process if OK signal indicating that the secret key is matched is received from the sending side.
6) If the CUBE pattern is delivered again from the sending side, operations 1 to 5 are repeated.
7) Once the secret key setup is complete between sending and receiving sides, encrypted communication is performed in the AES-256-based symmetric key encryption system.

As described in the above series, the receiving side performs a process to derive the CUBE pattern solution and calculates the extracted sequence reversely to infer and induce the secret key to verify the secret key generated through the 3D CUBE algorithm at the sending side. The final pattern is derived by call and execution of the outcomes in the MCTS configuration via learning with the algorithm to solve the pattern through the DeepCube algorithm. Even if the pattern of inducing the key delivered from the sending side was leaked to a third party, it would be difficult to induce the precisely matched key because there are numerous solution processes to solve the pattern. In addition, as the secret key is used on confirmation between sending and receiving sides, stronger security is provided in the key generation algorithm.

In the algorithm that implements the induction of secret key to solve the shuffled pattern of the 3D CUBE algorithm at the receiving side, a secret key that is used finally in encryption communication is derived by performing an XOR operation on the hashing information of pattern’s coordinate values and hashing result of each corresponding sequence after processing a sequence performed in each phase derived as an output value using an inverse function. A stronger key generation algorithm is implemented through the operation by generating a sequence of pattern change cases in the 3D CUBE algorithm.

The 3D CUBE algorithm performs an operation that returns the shuffled CUBE pattern delivered from the sender back to the originally initialized CUBE pattern by applying the DeepCube algorithm which performs AI learning. The DeepCube algorithm learns an inference method independently solve the complex environment into a sole single-reward state using pure reinforcement learning. It employs asymmetric MCTS reinforced with the training neural network to solve a given cube into the initialization pattern. Starting from the search tree with the initial state, search trees are iteratively generated and then learning simulation is performed until it reaches the end node of \( T \). Each state is connected with the memory that stores the following variables and is managed:

\[ N_a(s): \text{Number of move } a \text{ taken at state } s \]
\[ W_a(s): \text{Maximum move value } a \text{ at state } s \]
\[ L_a(s): \text{Current virtual loss of move } a \text{ at state } s \]
\[ P_a(s): \text{Prior action probability } a \text{ from state } s \]

All learning simulations perform iterations of the tasks according to the MCTS tree policy starting from the root node until a leaf node is reached. The MCTS tree policy that solves the CUBE pattern in the DeepCube algorithm selects and determines a move for each time step as presented in equations (1), (2), (3), and (4).

\[
A_t = \arg \max_a U_t(s_a) + Q(s_a) \tag{1}
\]

\[
W_a(s) = \arg \max_a W(s) - L(s) \tag{2}
\]

\[
L_a(s) = \gamma L(s) + \sum_{a'} B(s') \tag{3}
\]

\[
P_a(s) = \frac{1}{N_a(s)} \tag{4}
\]}
Algorithm trains CUBE patterns through DNN and returns the delivered pattern from the sending side back to the initial state to extract a sequence, thereby generating a secret key. A DNN is an ANN consisting of input and output layers, and multiple hidden layers between input and output layers. The neural network structure of DNN has various types of DNN, which are derived from typical several neural network structures. There are limitations to comparing the performance of various types of neural network structures because some specific neural network structures are not implemented suitably to the given data set [31]–[33].

Figure 7 shows the DNN structure in the 3D CUBE algorithm.

4) 3D CUBE ALGORITHM-APPLIED ENCRYPTION AND DECRYPTION PROCESS

The 3D CUBE algorithm employs a secret key induced and generated through a CUBE pattern. It is implemented to generate a 256-bit secret key to apply and use the existing AES-256 algorithm. Thus, encryption and decryption using the 3D CUBE algorithm are operated the same as those of the AES-256 algorithm. This implies that the purpose of the 3D CUBE algorithm proposed in this study is to support a secured key exchange in the symmetric key encryption system and provide enhanced security through up-to-date key management. In addition, even if 512-bit secret key-applied systems are launched to cope with quantum computers, the code that generates a secret key in the 3D CUBE algorithm can be simply modified and applied due to the flexible structure of creating a secret key.

Although it is impossible to improve the information security of the encryption system using the 3D CUBE algorithm, it can guarantee transmitted data confidentiality and integrity by providing enhanced security in existing symmetric key encryption systems as the proposed method is used in securely managing a secret key used in the symmetric key encryption system. Figure 8 shows the process that performs encryption and decryption through the 3D CUBE algorithm, through which plain texts are ciphered and then deciphered.

\[
U_{s_t}(a) = cP_{s_t}(a) \sqrt{\sum_a N_{s_t}(a)/(1 + N_{s_t}(a))} \quad (2)
\]

\[
Q_{s_t}(a) = W_{s_t}(a) + L_{s_t}(a) \quad (3)
\]

\[
L_{s_t}(A_t) \leftarrow L_{s_t}(A_t) + v \quad (4)
\]

Equation (1) receives the results of (2) and (3) and processes them to calculate \( A_t \), which is the maximum of the optimal value at the child state with regard to time \( t \). Equation (2) presents that the search is performed using parameter \( c \) with regard to the number of taking move \( a \) in the shuffled CUBE state \( s_t \) to calculate \( U_{s_t}(a) \), which is the probability of prior action. Equation (3) acquires \( Q_{s_t}(a) \) by calculating the virtual loss with regard to performing \( a \) and maximum initial value at state \( s_t \). Here, the virtual loss \( L_{s_t}(a) \) is updated constantly using parameter \( v \) as presented in (4). This virtual loss calculation prevents the same state in the tree search from being searched more than twice, thereby preventing taking the same path only in the asymmetric task.

Once the state search with regard to \( s_t \) reaches the leaf node, the lower elements \( s_t \) and \( \{A(s_t, a), \forall a \in A\} \) are added to the search tree \( T \) to expand the state. After the state expansion, the memory of \( s' \) of each child node is initialized as follows: Here, \( P_{s'} \) in 4 indicates the policy output of \( f_0(s') \), which is a network function.

\[
\text{① } N_{s_t}(a) \rightarrow N_{s'}(\cdot) = 0
\]

\[
\text{② } W_{s_t}(a) \rightarrow W_{s'}(\cdot) = 0
\]

\[
\text{③ } L_{s_t}(a) \rightarrow L_{s'}(\cdot) = 0
\]

\[
\text{④ } P_{s_t}(a) \rightarrow P_{s'}(\cdot) = P_{s'}
\]

After the above procedures are complete, the maximum value of the policy estimate and the child state optimal value of the learning result are calculated using (5) to solve the CUBE pattern in the DeepCube algorithm. Here the calculated value is stored in all search states of the simulated paths.

\[
(v_{s_t}, \rho_{s_t}) = f_0(s_T) \quad (5)
\]

The memory update occurring over time in \( 0 \leq t \leq \tau \) can be defined by (6), (7), and (8).

\[
W_{s_t}(A_t) \leftarrow \max(W_{s_t}(A_t), v_{s_t}) \quad (6)
\]

\[
N_{s_t}(A_t) \leftarrow N_{s_t}(A_t) + 1 \quad (7)
\]

\[
L_{s_t}(A_t) \leftarrow L_{s_t}(A_t) - v \quad (8)
\]

The difference here from existing MCTSs is that only the maximum values that occurred along the search tree are stored in the memory, not the total sum. Since the method to solve the CUBE is not definite and the result is not mutually contradictory, calculating a mean value of reward when determining the move is not necessarily.

This calculation method, by deriving a corresponding sequence of CUBE pattern delivered to induce a secret key by applying the DeepCube algorithm, may have longer delay than existing encryption schemes. A secret key is generated using a variety number of cases since multiple solutions for the CUBE pattern solution are present. The DeepCube algorithm trains CUBE patterns through DNN and returns the delivered pattern from the sending side back to the initial state to extract a sequence, thereby generating a secret key.

Although it is impossible to improve the information security of the encryption system using the 3D CUBE algorithm, it can guarantee transmitted data confidentiality and integrity by providing enhanced security in existing symmetric key encryption systems as the proposed method is used in securely managing a secret key used in the symmetric key encryption system. Figure 8 shows the process that performs encryption and decryption through the 3D CUBE algorithm, through which plain texts are ciphered and then deciphered.
III. EVALUATION RESULTS
A. VERIFICATION OF THE 3D CUBE ALGORITHM IMPLEMENTATION

Once the 3D CUBE is generated through the 3D CUBE algorithm, it can be found that each coordinate has its value, which is different from the array structure of the existing Rubik’s cube (where only colors that correspond to each surface are arranged). However, coordinate values must be generated as an array in the proposed 3D CUBE algorithm because a secret key is generated by using the coordinate values of the corresponding surface, and whether the coordinate values are correctly inter-operated and moved during CUBE shuffling should be verified. The arrangement criteria of coordinate values generated through the 3D CUBE are explained in Figure 9 in Section 3. A coordinate value is assigned in each individual space of the component in a $3 \times 3 \times 3$ CUBE in the 3D CUBE algorithm. A pattern is recognized in the 3D CUBE algorithm through the assigned coordinate values, and key management with better security can be performed through the SHA 256 hashing process of the coordinate values.

Shuffling is conducted to make a secret key using the generated 3D CUBE. Here, the number of shuffles is delivered to the DeepCube algorithm-applied 3D CUBE algorithm as a random value and, based on this number, the algorithm performs learning and completes the MCTS configuration.

A sequence used to generate a secret key is produced during this shuffling of cube pattern. The shuffling order is determined using the following rule defined in the 3D CUBE algorithm. Here, the limitation on the rotation in the single direction minimizes the meaningless dummy data generation by preventing the rotation, and it solves the difficulties of AI algorithm learning due to the same pattern occurrence at the time of the four-rotation sequence. The number of shuffle moves to create a secret key in the 3D CUBE algorithm is determined through calling random()
TABLE 2. Example of secret key generation in the 3D cube algorithm.

| Sequence | Rotation 3D CUBE | Sequence generation code |
|----------|-----------------|--------------------------|
| 0        | Upper face 2 turns left | TL02 |
| 1        | Left side 3 turns down | WD03 |
| 2        | Midplane 1 turn right | MR01 |
| 3        | Lower plane 2 turns left | CL02 |

TABLE 3. Example of secret key induction in the 3D cube algorithm.

| Transmitting side | Rotation | Seq code | Solution | Error |
|-------------------|----------|----------|----------|-------|
| Seq 0             | Upper face 2 turns left | TL02 | Seq 3 | TR02 | Seq 3 | TL02 |
| Seq 1             | Left side 3 turns down | WD03 | Seq 2 | WU03 | Seq 2 | WD01 |
| Seq 2             | Midplane 1 turn right | MR01 | Seq 1 | ML01 | Seq 1 | MR03 |
| Seq 3             | Lower plane 2 turns left | CL02 | Seq 0 | CR02 | Seq 0 | CL02 |

without certain rules, rather than using only the designated number of moves. Here, the 3D CUBE algorithm where the DeepCube algorithm is applied (an ANN consisting of DNN) checks the sequence and pattern information generated during the cube pattern shuffling and delivers them to the module that generates a secret key, thereby performing the operation to generate a secret key. The sequence value extracted when each shuffling is complete and pattern coordinate values at the corresponding surface of the CUBE are processed by SHA-256 hashing, thereby generating the sequence data, which is an element necessary to produce a secret key.

Once the secret key generation code is performed with regard to the conditions presented in Table 2 through the 3D CUBE algorithm, the following results are derived. The final pattern is delivered to the receiving side, and the generated secret key is stored internally to be used in encrypted communication.

① Sequence 0: Upper face 2 turns left
→ TL02 ⊕ CUBE Face pattern: [C01, C02, C03, A04, A05, A06, A07, A08, A09]
→ Hash code for TL02: c46476ab9eb1e49116aaf4b5c64ec76725a1e97b92c05f96ea473114ba4e7
→ Hash code for CUBE Face pattern: 0xbc746ccaf84c61921d9c92346789c4ef8ad857c94ef7fb7aaf99184
→ Result of the TL02 ⊕ CUBE Face pattern: 0×78101a61232d07b5c8a38c0b5b5e8bfae6b561b043fbf6006c57e1d9ea23563

② Sequence 1: Left side 3 turns down
→ WD03 ⊕ CUBE Face pattern: [F01, C02, C03, F04, A05, A06, F07, A08, A09]
→ Hash code for WD03: e4218fcd031c4575a2e5e05c3f87226deecb86092082739cd8450780a8305dce
→ Hash code for CUBE Face pattern: 0xcb1ae3acc1967b8b8335e928e0c0f7f903723c86ec574172806570b76c34c
→ Result of the WD03 ⊕ CUBE Face pattern: 0xe89021f7cf0522cd1ad625bf71767ed9915bf5f2ae8ec4dadaec562f015469e90

③ Sequence 2: Midplane 1 turn right MR01
→ MR01 ⊕ CUBE Face pattern: [F01, C02, C03, B02, D05, D08, F07, A08, A09]
→ Hash code for MR01: 47bce861e96a4060c1da4f8262344c60a7ea591b5e5f204e7411f68b

④ Sequence 3: Lower plane 2 turns left CL02
→ CL02 ⊕ CUBE Face pattern: [F01, C02, C03, B02, D05, D08, C07, C08, E09]
→ Hash code for CL02: d579acc343bc1b91cdb803341257a268e8eb04a7a5b3d2635f04d8e8d3b13b84
→ Hash code for CUBE Face pattern: 0×3d13a87b923768b161f25322c99d2e97ea5f501b43d44710694a9c7a27e6c6e
→ Result of the CL02 ⊕ CUBE Face pattern: 0xe86a04b89f6d1ad6a726063ec7efc0f5552b8e8e062421f39071a271cf57ea

Final pattern and creation secret key
→ Receiving side transfer CUBE Face Pattern: [F01, C02, C03, B02, D05, D08, C07, C08, E09]
→ Generated secret key: 0xda413a0445d89e47724ce7109eafa6880a293b2de968e2a301bf4d898e300929

Once the 3D CUBE pattern is finally complete by extracting the sequence data, the generated sequence data are XORed again to generate a secret key. Here, the 256-bit secret key is used for application to AES-256 encryption systems.

The sequences that occurred during shuffling of 3D CUBE pattern extracted through the algorithm and coordinate value of each CUBE pattern are hashed and XORed to create a secret key, which is then used. The receiving side generates the CUBE using the coordinate value information of the 3D CUBE delivered from the sending side, and then returns this to the original state to infer the sequence. This is done to learn the CUBE’s pattern through the DNN used in the DeepCube algorithm and to derive the sequence to induce the secret key, thereby delivering the secret key to the sending side. Various cases occur to solve the pattern, through which it can be an advantage to prevent the leakage of the secret key. Table 3 summarizes other solutions to the previously performed sequence.

Table 3 verifies that the same result is derived through various solutions to the simply shuffled CUBE. Since the sequence itself changes, a different result is derived during the hashing process of the sequence code. Accordingly, verification errors occur because processing is conducted with
a different secret key in the 3D CUBE algorithm. To solve this problem, the evaluated value is stored internally in the DeepCube algorithm; based on this value, a weight to action is assigned and managed to provide the direction to reinforcement learning. In addition, the search criteria to the paths experienced through learning are firmly established, thereby inducing the CUBE pattern result more accurately between sending and receiving sides and supporting the generation of an accurate secret key. In the DeepCube algorithm, learning is performed over the an ANN consisting of DNN. Here, weights and policy estimates of normal nodes are managed while configuring the MCTS search tree. The given pattern is thereby solved while inducing and deriving a secret key.

B. APPLICATION OF 3D CUBE ALGORITHM TO AES-256 ENCRYPTION SYSTEM

The secret key generated through the 3D CUBE algorithm proposed and implemented in this study can be applied to an AES-256 symmetric key encryption system. In this study, the cipher text produced by performing encryption using the 256-bit secret key generated through the 3D CUBE from the arbitrary plain text was conducted, and the decryption of the cipher text was performed with the same secret key to verify that the first plain text value was produced correctly. The 3D CUBE algorithm is a system to generate and manage a secret key. Thus, since it does not change the encryption feature of AES-256 systems, whether the key generated through the 3D CUBE algorithm could be delivered to an AES-256 system and used was verified through the aforementioned encryption and decryption, although this study did not modify and implement the AES-256 symmetric key encryption system separately. Table 4 presents the comparison results of mean values by measuring encryption and decryption speeds 20 times for 100MB, 500MB, 1GB, 2GB, and 4GB sample data between encryption communication using a secret key according to the existing method in AES-256 symmetric key encryption system and encryption communication using a secret key derived through the 3D CUBE algorithm.

The environments were as follows: for the existing method, encryption and decryption were conducted with the presence of a prior shared key, and for the 3D CUBE algorithm, it was conducted without such to perform the simulation. The measurement results of encryption and decryption speeds by applying the 3D encryption algorithm to the AES-256 symmetric key encryption system were 1.136 sec slower on average than those of the existing method during encryption, and 4.532 sec slower during decryption. If the performance of the 3D CUBE algorithm is improved and a secret key can be induced by solving the CUBE pattern by utilizing prior trained data, this study expects performance comparable to that of the existing method.

Figure 9 shows the measurement results of encryption measured in Table 4 as a graph, in which the 3D CUBE algorithm did not affect the encryption performance. Figure 10 shows the measurement results of decryption measured in Table 4 as a graph.

In contrast with the encryption measurements, Figure 10 verifies that the degradation in speed according to the application of the 3D CUBE algorithm affected the decryption performance. The reason for the speed degradation during decryption was verified as being due to the difference in time used learning the 3D CUBE pattern.

This study verified in the experiment environment that a sudden difference did not occur, as there was no change in encryption method compared to the AES-256 symmetric key encryption system. However, a slight speed difference occurred in the system due to the time taken to learn in the initial operation of the 3D CUBE algorithm, but it was verified that this amount of delay did not cause any problem in the actual-use environment, where learning was complete through the measurement results of encryption and decryption speeds. Thus, this experiment proved that there was no issue in applying the 3D CUBE algorithm (which enables more secured key management) to AES-256 symmetric key encryption systems.
TABLE 5. Measurement of key generation speed between using 3D cube algorithm and GnuPG (GPG).

| Key generation forwarding | 3D CUBE algorithm | GnuPG (GPG) method |
|--------------------------|-------------------|--------------------|
|                          | 1.21 Sec          | 0.86 Sec           |

FIGURE 11. Graph of measurement on 3D CUBE algorithm-applied key generation speed.

C. PERFORMANCE COMPARISON OF THE 3D CUBE ALGORITHM

To simulate the performance evaluation on the key management of the 3D CUBE algorithm proposed and implemented in this study, an open source system, GnuPG (GPG), was selected out of the systems running with a hybrid manner using the AES-256 symmetric key algorithm, and the difference in speeds was analyzed. The simulation of the key management algorithms was performed in the following personal computer environment: Intel Core i5-8265U 2.4GHz 8core CPU, 8.00GB memory, and Nvidia GTX-1060 3GB external GPU. The simulation was conducted by measuring time taken to enable encryption communication through AES-256 symmetric key encryption system by creating and exchanging a secret key initially at the sending and receiving sides. The measurements were conducted except for delay due to communication in the experiment environment, and a mean of 20 measurements was recorded.

GnuPG adopted the Rivest–Shamir–Adleman (RSA) method to generate public and private keys in the server and performed encryption communication. This experiment generated and used a 2048-bit key with a security strength that corresponds to that of 112 bits in a symmetric key system. Thus, the performance of the 256-bit key generated through the 3D CUBE algorithm is much better. Figure 11 shows the results derived from 20 experiments as a graph. Initially, the speed of the 3D CUBE algorithm was degraded slightly compared to GnuPG method if learning was not properly done. However, after learning was complete, it showed a performance similar to that of the GnuPG method.

The processing speed increased with the complexity of shuffle in the 3D CUBE algorithm. When sending the shuffle patterns was conducted four times based on the example explained in Table 3, a mean time of 1.21 sec from 20 measurements was taken. For GnuPG used for key exchange in the hybrid method, 0.86 sec was taken, which verified no significant difference in performance compared to that of 3D CUBE method proposed in this study. This proves the effective applicability of the secret key induction method through the 3D CUBE algorithm compared to that of asymmetric key encryption used in GnuPG.

If 30 times or more shuffles are generated to increase the complexity of the 3D CUBE algorithm, the time taken to create a key would increase further, whereas the key stability would increase further due to the sequence complexity. Table 6 presents the results of measuring times taken according to the shuffle complexity. With regard to shuffle complexity, the possible combination number during one shuffle in the 3D CUBE face is 36. Thus, if an equation is made with n, the number of shuffles, as a coefficient, it can be defined as presented in (9). If the shuffle complexity is calculated four times, which is the number of simulations performed, it can be calculated as presented in (10).

\[(36P_1)^n\]  
\[(36P_1)^4 = \left(\frac{36!}{(36-1)!}\right) = 1,679,616 \approx 16 \times 10^5\]  

The key generation time was measured by performing the 3D CUBE algorithm method 20 times for each shuffle time, and a mean time is presented.

However, since the 3D CUBE algorithm is progressed by learning through the DNN, operations should start after the synchronization of the ANN between sending and receiving sides is complete to infer a pattern accurately. If the synchronization is not achieved accurately, normal sending and receiving cannot be achieved. Thus, this study measured the pattern recognition rate with regard to DNN learning and verified that the rate to recognize a pattern was degraded as the number of shuffles increased. Table 7 summarizes the accuracy of the 3D CUBE algorithm through DNN learning when the number of shuffles is four.

The above verification experiment result presents the recognition accuracy of the 3D CUBE algorithm with regard to the conversion of the number of hidden layers that perform 100 iterations of learning in the DNN model.
resources, thereby enabling the use of change with more
ried according to the difference in equipment and available
basically with $3^*3^*3$-size system. However, size can be var-
TABLE 7. Measurement of accuracy according to the number of shuffles in the 3D CUBE algorithm.

| CUBE Shuffle | Hidden Layer Count | 3D CUBE Learning Accuracy |
|--------------|--------------------|--------------------------|
| 4 times      | 2                  | 39.60%                   |
|              | 3                  | 86.50%                   |
|              | 4                  | 84.46%                   |
| 20 times     | 2                  | 20.09%                   |
|              | 3                  | 43.25%                   |
|              | 4                  | 41.32%                   |
| 40 times     | 2                  | 5.23%                    |
|              | 3                  | 21.50%                   |
|              | 4                  | 21.45%                   |

highest recognition rate, 86.50%, was exhibited in the opera-
tion of four shuffles that maintained the number of hidden
layers as 3, which was mainly set in this experiment.

IV. CONCLUSION
The 3D CUBE algorithm proposed and implemented in this
study was applied to a symmetric key encryption system, and
its usability was verified through experiments. The proposed
algorithm differs from existing symmetric key encryption
systems that exchange a secret key. It provides enhanced
secure management of keys by applying a method that
induced a secret key itself at the receiving side. In particular,
it provided better security than existing methods due to the
complexity of the 3D CUBE pattern solution and this study
proved the secure management of the secret key by imple-
menting and verifying the proposed algorithm.

Through comparison with the GnuPG method (a hybrid
method used to exchange a secret key securely in the
AES-256 algorithm, which was a symmetric key encryption
system), the effectiveness of the 3D CUBE algorithm was
proven. Although slight speed degradation occurred com-
pared to the existing method, stronger security could be pro-
vided.

The security issue in the asymmetric key methods has
emerged due to the emergence of quantum computers, which
is also closely related to the hybrid encryption methods such
as GnuPG. To address this, studies on secure key manage-
ment techniques in the quantum computer environment are
underway. The 3D CUBE algorithm proposed in this study
employs a pattern extraction method by configuring and using
the MCTS in the ANN consisting of DNN. Thus, it is novel
in terms of the fusion of AI technology and cryptography, and
proves that the security of existing symmetric key encryption
systems can be enhanced.

The 3D CUBE algorithm was implemented and tested
basically with $3^*3^*3$-size system. However, size can be var-
ied according to the difference in equipment and available
resources, thereby enabling the use of change with more
in-depth secured key management methods. In addition,
as the 3D CUBE algorithm is not fixed to a cubic shape, var-
ious shapes of the 3D CUBE can be generated and applied,
through which state-of-the-art security can be provided.
Moreover, this study generated a secret key through shuffles
by assigning a coordinate value to the surface of 3D CUBE,
but other security-enhanced methods that apply a coordinate
value mapped internally can be used, thereby providing better
security.

A secured symmetric key encryption system can be imple-
mented in the 3D CUBE algorithm because secret keys are not
exchanged. However, to implement this, studies on reinforce-
ment learning over AI technology-applied ANN are essential.
In addition, it can be used during real-time data transmission
in real network environments only when effective and effi-
cient learning algorithms are derived and applied.

Learning degeneration occurred during the operation of the
DeepCube algorithm, which was the core learning algorithm,
while performing simulations after implementing the 3D
CUBE algorithm. That is, the solution was converged to the
basic solution or completely eliminated, thereby incurring the
branch of the learning algorithm. To solve this problem, larger
training weights were assigned to samples that were closer to
the solved cube, whereas smaller weights were assigned to
sample groups that were farther from the solution candidates.
The action of branching in the DeepCube algorithm was
removed by setting and assigning a loss weight to $x_i$ and
$W(\frac{1}{D(x_i)})$, which are the samples of the DeepCube
algorithm.

In future, even the implementation of differentiated
encryption algorithms according to system importance and
resource limitations where the 3D CUBE algorithm is applied
can be possible by improving the possibility of application of
various sizes and shapes of CUBE through the advancement
of DNN optimization applied to the 3D CUBE algorithm.
The proposed method can also be developed into an encryption
system with excellent performance that can provide in-depth
security if the encryption system is implemented, even con-
sidering the coordinate values inside the CUBE.

REFERENCES
[1] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital
signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2,
pp. 120–126, Feb. 1978.
[2] D. Harkins and D. Carrel, The Internet Key Exchange (IKE),
document IETF RFC 2409, 2006.
[3] T. Dierks, The TLS Protocol Version 1.0, document RFC 2246, Jan. 1999.
[4] T. Dierks, The Transport Layer Security (TLS) Protocol Version 1.1,
document RFC 4346, Apr. 2006.
[5] T. Dierks, The Transport Layer Security (TLS) Protocol Version 1.2,
document RFC 5246, Aug. 2008.
[6] A. Freier, The Secure Sockets Layer (SSL) Protocol Version 3.0,
document RFC 6101, Aug. 2011.
[7] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms
and factoring,” in Proc. 35th Annu. Symp. Found. Comput. Sci., Dec. 2002,
pp. 124–134.
[8] C.-Y. Lu, E. Daniel Browne, T. Yang, and J.-W. Pan, “Demonstration of
a compiled version of Shor’s quantum factoring algorithm using photonic
qubits,” Amer. Phys. Soc., vol. 99, no. 25, pp. 25054(1)–25054(4), 2007.
[9] M. Abadi and G. D. Andersen, “Learning to protect communications with adversarial neural cryptography,” in Proc. ICLR, 2017, Oct. 2016.

[10] R. Coulom, “Efficient selectivity and backup operators in Monte-Carlo tree search,” in Proc. 5th Int. Conf. CG, Turin, Italy, May 2006, pp. 72–83.

[11] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in Proc. 17th Eur. Conf. Mach. Learn., Berlin, Germany, 2006, pp. 282–293.

[12] R. Coulom, “Efficient selectivity and backup operators in Monte-Carlo tree search,” in Computers and Games, H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, Eds. Berlin, Germany: Springer, 2007, pp. 72–83.

[13] P. J. Werbos, “Backpropagation through time: What it does and how to do it,” Proc. IEEE, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in Proc. 5th Int. Conf. CG, Turin, Italy, May 2006, pp. 72–83.

[15] G. Tesauro, “Temporal difference learning and TD-gammon,” Commun. ACM, vol. 35, no. 8, pp. 58–68, Jun. 1992.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, K. Simonyan, D. Hassabis, “Mastering the game of Go without human knowledge,” Nature, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “Mastering chess and Shogi by self-play with a general reinforcement learning algorithm,” 2017, arXiv:1712.01815. [Online]. Available: http://arxiv.org/abs/1712.01815

[19] R. Bellman, Dynamic Programming. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[20] R. S. Sutton, “Learning to predict by the methods of temporal differences,” Mach. Learn., vol. 3, no. 1, pp. 9–44, Aug. 1988.

[21] R. S. Sutton, “Learning to predict by the methods of temporal differences,” Mach. Learn., vol. 8, nos. 3–4, pp. 279–392, 1992.

[22] R. S. Sutton and A. Barto, Reinforcement learning: An introduction, vol. 1. Cambridge, MA, USA: MIT Press, 1998.

[23] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, p. 436, 2015.

[24] R. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in Proc. 5th Int. Conf. CG, Turin, Italy, May 2006, pp. 72–83.

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in Proc. 5th Int. Conf. CG, Turin, Italy, May 2006, pp. 72–83.

[26] R. S. Sutton and A. Barto, Reinforcement learning: An introduction, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[27] J. Daemen and V. Rijmen, AES Proposal: Rijndael, Version 2, NIST document, Mar. 1999.

[28] Specification for the Advanced Encryption Standard (AES), Federal Inf. Process. Standard 197, Nov. 2001.

[29] National Bureau of Standards, Data Encryption Standard, Standard FIPS-46. National Bureau of Standards, U.S. Department of Commerce, Washington, DC, USA, Jan. 1977.

[30] C. Paar and J. Pelzl, Understanding Cryptography. New York, NY, USA: Springer, 2010.

[31] I. Kokkinos and A. Yuille, “Inference and learning with hierarchical shape models,” Int. J. Comput. Vis., vol. 93, no. 2, pp. 201–225, Jun. 2011.

[32] C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” in Proc. 26th Int. Conf. Neural Inf. Process. Syst. (NIPS), vol. 2, Dec. 2013, pp. 2553–2561.

[33] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.