Received September 21, 2020, accepted November 1, 2020, date of publication November 9, 2020,
date of current version November 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3036899

Privacy-Preserving Reinforcement Learning Using Homomorphic Encryption in Cloud Computing Infrastructures

JAEHYOUNG PARK[1], (Graduate Student Member, IEEE),
DONG SEONG KIM[2], (Senior Member, IEEE),
AND HYUK LIM[3], (Member, IEEE)
[1]School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, South Korea
[2]School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD 4072, Australia
[3]AI Graduate School, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, South Korea

Corresponding author: Hyuk Lim (hlim@gist.ac.kr)

This work was supported in part by the National Research Foundation (NRF) through the Ministry of Science and ICT (MSIT) of Korean Government under Grant 2020R1F1A1060941, in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by MSIT, Resilient Cyber-Physical Systems Research, under Grant 2014-3-00065, and in part by the IITP funded by MSIT through the Artificial Intelligence Graduate School Program (GIST) under Grant 2019-0-01842.

ABSTRACT Reinforcement learning (RL) is a learning technique that enables state-dependent learning through feedback from an environment and makes an action decision for maximizing a reward without prior knowledge of the environment. If these RL techniques are used for data-centric services running on cloud computing, serious data privacy issues may occur because it is required to exchange privacy-related user data for RL-based services between the users and the cloud computing platform. We consider using homomorphic encryption (HE) scheme, which enables cloud computing platforms to perform arithmetic operations without decrypting ciphertexts. Using the HE scheme, users are allowed to deliver only ciphertexts to the cloud computing platform for using RL-based services. We propose a privacy-preserving reinforcement learning (PPRL) framework for the cloud computing platform. The proposed framework exploits a cryptosystem based on learning with errors (LWE) for fully homomorphic encryption (FHE). Performance analysis and evaluation for the proposed PPRL framework are conducted in a variety of cloud computing-based intelligent service scenarios.

INDEX TERMS Privacy preserving, homomorphic encryption, reinforcement learning, Q-learning.

I. INTRODUCTION

In recent years, artificial intelligence (AI) technologies have developed tremendously, and AI services have been actively utilized in our daily life. In order to provide AI services, a huge amount of computing and storage resources are required continuously and reliably. Because the computing and storage resources are limited at the customer's side, cloud computing is expected to play an important role in an AI service platform to solve the problem of lack of the computing and storage resources. Cloud computing is a technology that enables customers' personal data and processes to be stored and executed remotely using virtualization technologies such as virtual machines (VMs) and containers running on a data center connected to the Internet. Due to the advantages in terms of deployment and use costs, system reliability, and ease of use and management, it greatly contributes to the development and production of AI services and applications [1].

Reinforcement learning (RL) is one of the most important learning strategies that are being actively researched with aim of providing artificial intelligence in various application areas such as energy, healthcare, finance and so on [2], [3]. RL algorithms develop a policy that finds behaviors that maximize a reward using state and performance observations in a given environment. In a complex and dynamic environment, a number of training iterations for RL algorithms are required to reach a competent performance. Therefore, the cloud computing infrastructure is quite useful to implement high-quality RL-based services, especially when many users need to share

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

and process a vast amount of shared data. However, users who want to use RL services must send their sensitive personal information such as health care and financial data to cloud servers in the cloud computing infrastructure, and thus serious data privacy issues may be raised [4].

To solve data privacy issues in the cloud computing based AI service environment, homomorphic encryption (HE) [5] can be considered as a promising approach because it enables a third party to perform arithmetic operations on ciphertexts without decrypting them. Using HE scheme, users with sensitive personal data send only ciphertexts to cloud computing platform without providing decryption key information for using AI services. HE scheme can be roughly divided into partially homomorphic encryption (PHE), which has only one addition or multiplication homomorphism, and somewhat homomorphic encryption (SHE), which has both addition and multiplication homomorphism. In PHE schemes, since only one of homomorphic addition or multiplication operations is available, it is difficult to implement various functions for implementating complex AI algorithms. On the other hand, although SHE schemes have both addition and multiplication homomorphism, the number of homomorphic operations that can be performed continuously without decryption is limited because random errors inserted in ciphertexts for improving security level are accumulated as the number of successive homomorphic operations increases.

Several privacy-preserving reinforcement learning (PPRL) algorithms have been proposed to provide RL-based services without leaking user's sensitive information by using HE schemes [6], [7]. However, there are still critical problems to be solved for the implementation of PPRL algorithms using HE schemes in a cloud computing environment. First of all, the efficiency issue of PPRL algorithms is the most challenging because HE scheme significantly incurs the computational and communication overhead for implementing PPRL algorithms. Liu *et al.* proposed a secure multiparty computation (SMC)-based PPRL algorithm using additive homomorphic encryption (AHE) [6]. The communication overhead of the distributed cryptosystem used in their proposed scheme is quite high because it requires a lot of multi-party communication. When multiple parties have to cooperate with each other to process data in the SMC-based cloud systems, data needs to be synchronized, and it may result in a longer service latency. Second, in the SHE schemes, the number of homomorphic operations that can be performed continuously without decryption is limited because the random errors are accumulated. To overcome the limited number of homomorphic operations, Gentry designed a fully homomorphic encryption (FHE) scheme. The proposed scheme suppressed the increasing error by developing a bootstrapping algorithm [8]. Although the bootstrapping algorithm can suppress the growth of errors, the computational complexity of the entire algorithm increases as it requires a periodic execution of the bootstrapping with a high computational complexity to refresh accumulated errors. Third, in order to use the PPRL algorithm in a cloud computing environment, it is necessary to safely collect data from multiple users and perform learning using the collected data. However, in the HE scheme, since it is impossible to perform homomorphic operation between ciphertexts encrypted with different keys, only data from users using the same key can be shared in a cloud computing. If the same key is used among multiple users, it is difficult to preserve data privacy among them.

To solve the aforementioned problems, we propose the secure centralized computation PPRL (SCC-PPRL) scheme for a secure Q-learning using FHE scheme in a cloud computing environment. The proposed SCC-PPRL scheme stores and processes the data in a single cloud server without any support of third-parties to reduce communication overhead and avoid the synchronization problem. The proposed algorithm can restrict the growth of errors in the FHE scheme without a bootstrapping algorithm by enabling the error growth to be cancelled out during the iterative Q-value computation. In addition, the proposed algorithm exchanges the state and action information in a binary-encoded form to simplify the encrypted Q-table updating operations. The proposed algorithm also provides confidentiality to the users that share the same FHE key in a cloud system by applying a public key encryption based on Rivest–Shamir–Adleman (RSA) algorithm to the exchange of FHE data between the cloud server and users encrypted by the shared FHE key. The main contributions of this work are summarized as follows:

- We propose the SCC-PPRL algorithm using FHE scheme in a cloud computing environment. Compared to the PPRL algorithm using SMC in a cloud computing environment, the proposed algorithm is more efficient in terms of communication overhead because it stores and processes the data in a single cloud server.
- In the proposed algorithm, the growth of errors in the FHE scheme is restricted without the bootstrapping algorithm by enabling the error growth to be cancelled out during the iterative Q-value computation. We numerically evaluate the error suppression performance of the proposed scheme.
- The confidentiality among users who share the same FHE key is guaranteed by applying RSA encryption to the data exchange encrypted with the shared FHE key in multiple users cloud environment.
- We perform theoretical analysis of the proposed PPRL algorithm in term of computational and communication overhead, and simulation studies of the proposed scheme in a variety of scenarios.

The remainder of this paper is organized as follows. Section II presents the related works on HE and privacy preserving machine learning in a cloud computing environment. In Section III, we describe the preliminaries for understanding the RL algorithm and FHE scheme. Section IV describes the system model for the proposed PPRL algorithm in a cloud computing environment, and Section V explains the PPRL algorithm using FHE algorithm. Section VI and Section VII present theoretical analysis and simulation results, respectively, to verify the performance of the proposed PPRL scheme. Finally, Section VIII summarizes the study and concludes the paper.

II. RELATED WORK

A. HOMOMORPHIC ENCRYPTION

Data privacy issues have been steadily and extensively raised. For example, unreliable service providers can access decrypted content for operations to deliver services and decryption key in cryptosystem can be stolen by attackers. HE technologies have been actively researched as a technology that can protect data privacy by allowing third parties to perform arithmetic operations on ciphertexts directly without decryption [9]. After Rivest *et al.* proposed the privacy homomorphism, which is the basic concept of HE [5], a significant amount of research has been conducted to implement feasible HE schemes. Gentry first proposed FHE that can arbitrarily compute on encrypted data by using ideal lattices, and developed bootstrapping and squashing algorithms to refresh the ciphertext [8]. As Gentry demonstrated the feasibility of HE techniques, many studies have been actively conducted to improve the efficiency of FHE scheme. Van Dijk *et al.* developed an integer-based FHE that uses simple integer arithmetic operations. They improved the efficiency of the FHE scheme while maintaining the security level of the existing FHE scheme [10]. FHE schemes based on ring learning with errors (RLWE) and learning with errors (LWE) were developed in [11] and [12], respectively. Brakerski *et al.* used RLWE assumption to simplify the lattice analysis, making the FHE scheme simpler and more efficient [11]. Brakerski and Vaikuntanathan also used LWE assumption to propose efficient FHE techniques and introduced a new dimension-modulus reduction technique to shorten the ciphertext and reduce the complexity [12]. Until recently, many studies have been conducted to improve FHE efficiency by utilizing the previous studies [13], [14].

B. PRIVACY PRESERVING MACHINE LEARNING USING HOMOMORPHIC ENCRYPTION IN A CLOUD COMPUTING ENVIRONMENT

Research on machine running using cloud computing has been actively conducted, as the need for fast and efficient processing of massive data generated by rapid growth in hardware, software, and communications technologies [1], [15]–[17]. As a lot of data from many users has been concentrated on cloud servers, data protection concerns have been raised [4]. In order to overcome privacy issues of machine learning algorithms in cloud computing environments, privacy-preserving machine learning algorithms were developed using HE scheme [6], [18], [19]. For the purpose of preserving user's data privacy, Ibtihal *et al.* [18] utilized HE for outsourced images in cloud computing environments, and Sun *et al.* [19] developed a privacy-preserving machine learning classification algorithm using FHE scheme.

Furthermore, to preserve data privacy in RL algorithms, PPRL algorithms were developed. Sakuma *et al.* [7] proposed the privacy-preserving distributed RL that can preserve data-privacy between distributed agents by using AHE scheme developed by Paillier [20]. In the cloud computing environment, Liu *et al.* proposed an SMC based PPRL framework for dynamic treatment regimes by using the property of the AHE [6]. However, the SMC based algorithm developed in [6] has a high communication overhead and data synchronization issues because it is implemented using the distributed cryptosystem. However, research on PPRL algorithms that do not rely on third-party agents in a cloud computing environment has not been carried out yet.

III. PRELIMINARY

A. REINFORCEMENT LEARNING

RL is a learning algorithm concerned with how agents defined in an environment recognize the current state and then select an action or sequence of actions to maximize cumulative reward. The basic RL model is based on Markov decision process, and it consists of agent state space S, action space A, and reward space R. At every discrete time step t, the agent has a current state $s_t \in S$ and possible action space $A(s_t)$. The agent takes an action $a_t \in A(s_t)$ and receives a new state s_{t+1} and reward r_{t+1} from the environment. Agents in RL develop a policy $\pi : S \to A$ on the basis of these interactions with the environment to maximize the cumulative reward.

In this paper, RL is implemented by using the Q-learning algorithm [21]. In the Q-learning algorithm, the optimal policy is learned by updating the Q-function, which estimates the reward to be obtained by taking action on the current state. The Q-function initially has a constant value. The agent takes action a_t on the state s_t at time t, and then the state transitions to a new state s_{t+1}. At this time, reward r_{t+1} is obtained from the environment, and the Q-function is updated as follows:

$$Q(s_t, a_t) \leftarrow (1-\alpha)\cdot Q(s_t, a_t) + \alpha \left\{ r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) \right\}, \quad (1)$$

where α is the learning rate, and γ is the discount factor. The agent can obtain the optimal action on the given state by selecting the maximum Q-value through the procedure of Q-function update. However, if the agent always takes the maximum Q-value, the agent may retrieve the local maximum value. To improve the performance of the Q-learning algorithm, an ϵ-greedy policy is adopted, in which an action is randomly chosen with probability ϵ, where ϵ is a positive small number between 0 and 1 [22]. Therefore, the probability of selecting action a_t on current state $P_\pi(s_t, a_t)$ can be represented as

$$P_\pi(s_t, a_t) = \begin{cases} 1-\epsilon+\dfrac{\epsilon}{|A|}, & \text{if } a_t = a_t{}^* \\ \dfrac{\epsilon}{|A|}, & \text{if } a_t \neq a_t{}^* \end{cases}$$

where $|A|$ is the cardinality of the action set A and $a_t{}^* = \max_{a_t \in A} Q(s_t, a_t)$. Thus, the frequency of exploration is determined by adjusting the ϵ value. Since there is tradeoff issue between exploitation and exploration, the problem of determining ϵ is important in the RL algorithm [22].

Every Q-value is determined by a pair of a state and an action, and they are stored in a (the number of states) $\times$ (the number of actions) table form called Q-table.

B. HOMOMORPHIC ENCRYPTION BASED ON LEARNING WITH ERRORS

HE allows arithmetic operations to be performed directly on ciphertexts without decrypting them. In this paper, a FHE scheme based on LWE-based cryptosystem is utilized for the implementation of PPRL algorithm. In the LWE-based cryptosystem, we select all the modulus of the encryption scheme as the powers of 10 to represent a decimal system. Let M denote plaintext space, $m_i \in M$ denote the i-th element of M, and $p \in \mathbb{N}$ denote the cardinality of M. The set of plaintexts is bounded by $\left\{m_i \in \mathbb{Z} : -\frac{p}{2} \leq m_i \leq \frac{p}{2}\right\}$. In addition, $\mathbb{Z}_q$ represents a set of integers modulo $q = L \cdot p$, where L is a parameter for LWE encryption and is used as a boundary of errors inserted into the ciphertext to enhance security. The secret key sk is selected as an integer vector of size N from the uniform distribution, such that $sk \in \mathbb{Z}_q^N$. Then, the ciphertext $\eta_A(m_i)$ of the plaintext m_i is obtained by

$$\eta_A(m_i) = [(-X_i \cdot sk + Lm_i + e_i) \bmod q,\ X_i^T] \in \boldsymbol{C} = \mathbb{Z}_q^{N+1}, \tag{2}$$

where $X_i \in \mathbb{Z}_q^N$ is a random vector sampled from the uniform distribution, e_i is an injected error randomly sampled from $\{1, 2, \ldots, r\}$ as a uniform probability, r is a positive integer less than $L/2$, and $\boldsymbol{C}$ is the ciphertext space. Hereafter, the ciphertext encrypted by (2) is represented as $\overline{m_i}$ instead of $\eta_A(m_i)$ for readability. Note that the random error e_i is injected into a ciphertext in the encryption process. Even if a large number of pairs $\{m_i, \overline{m_i}\}$ are given for solving sk, the problem is extremely difficult, since the security of the cryptosystem is based on the worst-case hardness of lattice problem. It is called a learning with error problem [23]. In the decryption process, let $sv := [1, sk^T]^T$ denote the secret key vector and $\|\cdot\|$ denote the rounding operation; then the ciphertext $\overline{m_i}$ is decrypted as follows:

$$\left\| \frac{(\overline{m_i} \cdot sv) \bmod q}{L} \right\| = \left\| \frac{Lm_i + e_i}{L} \right\| \rightarrow m_i. \tag{3}$$

Since e_i is less than $L/2$, the decryption result is obtained successfully. Note that a plaintext is bounded by $\left\{-\frac{p}{2} \leq m_i \leq \frac{p}{2}\right\}$. To represent negative integers in this system, if the decrypted result from (3) is greater than $p/2$, we subtract p from the result to make it negative.

Let Enc_A and Dec denote encryption using (2) and decryption functions, respectively, and $+_C$ is an addition operation on the ciphertext space. Homomorphic addition operation is represented as follows:

$$\begin{aligned} &\text{Enc}_A(m_1) +_C \text{Enc}_A(m_2) \\ &\quad = \overline{m_1} + \overline{m_2} \\ &\quad = [(-X_{\text{add}} \cdot sk + Lm_{\text{add}} + e_{\text{add}}) \bmod q,\ X_{\text{add}}], \end{aligned} \tag{4}$$

where X_{add} is (X_1+X_2), m_{add} is (m_1+m_2), and e_{add} is (e_1+e_2). Then, the result of homomorphic addition in (4) is decrypted using the decryption function (3) as follows:

$$\left\| \frac{L(m_1 + m_2) + e_1 + e_2}{L} \right\| \rightarrow m_1 + m_2 \tag{5}$$

as long as $m_1 + m_2 \in M$ and $(e_1 + e_2)$ is less than $L/2$. As shown in (5), since e_i is a positive value, the magnitude of the error in the ciphertext accumulates as the number of homomorphic addition operations performed in succession without decryption increases. As a result of many homomorphic addition operations, a decryption error can occur in case that the sum of errors $\sum e_i$ is greater than $L/2$. Note that the maximum value of injected random errors is r. The upper bound of accumulated error for each homomorphic addition can be calculated by $B_{add} = 2 \cdot r$. In the case of homomorphic subtraction operation, its procedure for encryption and decryption is the same as that of the homomorphic addition operation, and the homomorphic subtraction operation is denoted by $-_C$.

In the FHE scheme, homomorphic multiplication is a more complicated operation than homomorphic addition. Moreover, since the magnitude of error growth is larger in homomorphic multiplication, an effective homomorphic multiplication operation is required to reduce the magnitude of error growth within the error boundary in the LWE-based cryptosystem. For the implementation of homomorphic multiplication, the multiplicand is encrypted using (2), whereas the encryption function for the multiplier is changed. Let R be $[10^0, 10^1, \ldots, 10^{\log q - 1}] \otimes I_{N+1}$, where the Kronecker product is denoted by $\otimes$ and I_{N+1} is denoted by the identity matrix of size $N+1$. Then, the encryption function for the multiplier is represented as follows [24]:

$$\eta_B(m_i) = m_i \cdot R + \text{Enc}_A(0_{\log q(N+1)\times 1}) \in \mathbb{Z}_q^{\log q(N+1)\times(N+1)}, \tag{6}$$

where $0_{\log q(N+1)\times 1}$ is a zero vector of size $\log q(N+1) \times 1$. Hereafter, the ciphertext encrypted by (6) is represented as $\widetilde{m_i}$ instead of $\eta_B(m_i)$ for readability. Let $\boldsymbol{c_i}$ denote an element of ciphertext $\overline{m_i}$; it can be represented as $\boldsymbol{c_i} = \sum_{k=0}^{\log q - 1} c_i^k \cdot 10^k$, where c_i^k is the $(k+1)$-th single digit of $\boldsymbol{c_i}$. Then, the function that decomposes the argument is defined as follows:

$$D(\overline{m_i}) = [c_1^0, c_2^0, \ldots, c_{N+1}^0, \ldots, c_1^{\log q - 1}, c_2^{\log q - 1}, \ldots, c_{N+1}^{\log q - 1}]. \tag{7}$$

With this function, $\overline{m_i}$ can be obtained by the arithmetic operation of the formula $D(\overline{m_i}) \cdot R = \overline{m_i}$ for any $\overline{m_i} \in \mathbb{Z}_q^{1\times(N+1)}$. Let $\times_C$ denote multiplication on the ciphertext space. The homomorphic multiplication is represented with the secret vector sv as follows:

$$\begin{aligned} (\overline{m_1} \times_C \widetilde{m_2}) \cdot sv &= D(\overline{m_1}) \cdot (m_2 R + \boldsymbol{O}) \cdot sv \\ &= m_2 \overline{m_1} \cdot sv + D(\overline{m_1}) \cdot e_{m_2}, \end{aligned} \tag{8}$$

where $\boldsymbol{O}$ is $\mathrm{Enc}(0_{\log q \cdot (N+1) \times 1})$ and e_{m_2} is the error vector inside ciphertext $\boldsymbol{O}$. Then, the result of homomorphic multiplication is decrypted as follows:

$$\left\| \frac{m_2(Lm_1 + e_1) + D(\overline{m_1}) \cdot e_{m_2}}{L} \right\| \rightarrow m_2 m_1 \qquad (9)$$

as long as $m_1\ m_2 \in M$ and $|m_2\ e_1 + D(\overline{m_1}) \cdot e_{m_2}|$ is less than $L/2$. The upper bound of accumulated error for each homomorphic multiplication can be calculated by $B_{mul} = r \cdot p/2 + 9 \cdot r \cdot \log q \cdot (N+1)$. Note that the elements of $D(\overline{m_1})$ range from 0 to 9.

IV. PROPOSED RL SERVICE MODEL FOR PRIVACY PRESERVATION

In this paper, we consider a RL scenario that operates in a cloud system using centralized storage and computation. As shown in Fig. 1, users collect data through interactions with a given environment, the collected data is delivered to the cloud server, and the data is exploited to provide RL services in the cloud. Here, Q-learning algorithm is adopted for RL service, and the cloud platform (CP) manages the Q-table for the Q-learning, which is updated using the shared data from multiple users.

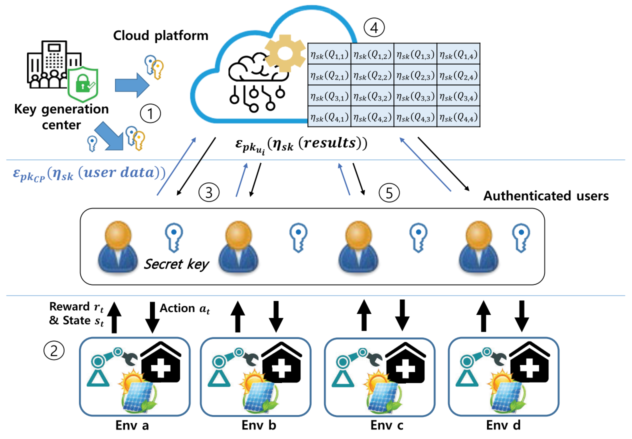

Cloud platform
Key generation center
①
④
$\eta_{sk}(Q_{1,1})$ $\eta_{sk}(Q_{1,2})$ $\eta_{sk}(Q_{1,3})$ $\eta_{sk}(Q_{1,4})$
$\eta_{sk}(Q_{2,1})$ $\eta_{sk}(Q_{2,2})$ $\eta_{sk}(Q_{2,3})$ $\eta_{sk}(Q_{2,4})$
$\eta_{sk}(Q_{3,1})$ $\eta_{sk}(Q_{3,2})$ $\eta_{sk}(Q_{3,3})$ $\eta_{sk}(Q_{3,4})$
$\eta_{sk}(Q_{4,1})$ $\eta_{sk}(Q_{4,2})$ $\eta_{sk}(Q_{4,3})$ $\eta_{sk}(Q_{4,4})$
$\varepsilon_{pk_{u_i}}(\eta_{sk}\ (results))$
$\varepsilon_{pk_{CP}}(\eta_{sk}\ (user\ data))$
③
⑤
Authenticated users
Secret key
Reward r_t & State s_t
Action a_t
②
Env a
Env b
Env c
Env d

FIGURE 1. System model for privacy-preserving reinforcement learning with a huge number of states.

In this scenario, we propose a cryptosystem that can preserve users' data privacy in the cloud environment where the privacy sensitive data from multiple users is collected, shared, and processed. The proposed cryptosystem utilizes the LWE-based FHE scheme to protect data privacy between the CP and users and RSA based public key scheme to protect the privacy among multiple users in the cloud. The users in Fig. 1 share the same FHE key for performing homomorphic arithmetic operations using the shared Q-table. In the FHE scheme in a cloud computing environment, the CP cannot perform homomorphic arithmetic operations using ciphertexts encrypted if each user uses a different key, i.e., $\overline{m_1}^{u_a} + \overline{m_2}^{u_b} \neq \overline{(m_1 + m_2)}^{u_i}$, where $\overline{m_i}^{u_i}$ is a ciphetext of m_i using HE function with a secret key of user u_i for i, a, $b \in \{1, 2, \ldots, N_u\}$, and N_u is the number of users sharing the Q-table.

However, if the same key is used among multiple users, it is difficult to preserve data privacy among them. Note that the proposed cryptosystem uses a symmetrical secret for HE scheme. In order to provide confidentiality to users that share the same FHE key, RSA algorithm based encryption is applied to the exchange of data between the cloud server and users. The privacy sensitive data encrypted with the shared FHE key is encrypted again by RSA encryption in our cryptosystem. In other words, the proposed cryptosystem utilizes a double encryption (DE) technology that encrypts values encrypted by FHE using RSA encryption for preserving privacy among the users. When a user requesting the service sends user information to the CP, the encrypted value with HE is encrypted by the CP's public key for RSA algorithm, so that only the CP can access the homomorphic encrypted value. Then, when the CP delivers the result of the homomorphic operation, the result is encrypted by the public key of the user again so that only the user who requested the service can access the data using the user's private key. The ciphertexts encrypted with public key of CP and public key of i-th user can be denoted by $\varepsilon_{pk_{CP}}$ and $\varepsilon_{pk_{u_i}}$, respectively.

The proposed system consists of multiple users, a CP, and key generation center (KGC), and each user interacts with a given environment. The procedure of the PPRL algorithm using the DE scheme is as follows:

- The KGC is assumed to be a trusted agency in this paper. Users requesting the same RL service are grouped through the authentication process in the KGC, A symmetrical secret key is generated by using the LWE based FHE scheme, and then the secret key is delivered to the authenticated users belonging to the group through trusted channels. It also creates public and private key pairs based on the RSA algorithm and delivers a private key for each user and CP through trusted channels. Then, the public key of the CP is delivered to the users, and the public key of the users is delivered to the CP. This process is annotated by ① in Fig. 1.
- The users deliver action a_t to maximize the reward, and the environment reports the reward r_t and state s_t to the user after executing the action a_t as shown in ②.
- In Fig. 1, ③ showes the user-side procedure for PPRL algorithm. The secret key for FHE scheme is kept only on the users' side and is not shared over the network, and the homomorphic encryption and decryption operations using the symmetrical secret key can be performed only on the users' side. For performing the Q-learning algorithm, the user encodes the state and action information into binary vectors. The encoding procedure will be explained in detail in Section V. The user also encrypts reward-related data for Q-table update. To preserve data privacy between users, the values encrypted with the secret key using HE scheme are encrypted again with the RSA public key of the CP, and then the double-encrypted values are transmitted to the CP.

- After receiving the double-encrypted value delivered by the user, the CP performs RSA decryption using its own RSA private key to obtain the encrypted value by HE scheme. By using the homomorphic encrypted value, the CP performs the Q-table updating and generates a Q-vector for the action selection at the user side in ④. For privacy-preserving purpose among authenticated users, the CP encrypts the Q-vector using RSA public key of the user requesting the RL-based service, and then it transmits the double-encrypted values to the user.
- After receiving the double-encrypted Q-vector, the user decrypts the vector using its own RSA private key and the secret key for the HE scheme, and it selects an action to maximize the cumulative reward by finding the maximum value of the vector in a given environment as shown in ⑤. The procedures in ② – ⑤ are repeated.

V. PRIVACY-PRESERVING REINFORCEMENT LEARNING

This section explains the operations of a CP and users for the privacy-preserving RL service described in the previous section in detail. The secure action selection algorithm performs the task to find the action that can maximize cumulative rewards and is described in Section V-A. The secure Q-table updating algorithm performs the task of updating the Q-table and is described in Section V-C.

A. SECURE ACTION SELECTION ALGORITHM THAT MAXIMIZES CUMULATIVE REWARD

We propose a secure action selection algorithm to find an action that maximizes cumulative reward in a given environment by using only encrypted data. The proposed secure action selection algorithm selects a vector containing Q-values that correspond to the current state and finds the one element with the maximum Q-value among the Q-values in the Q-vector.

First, we assume that the Q-table initialized with a constant value is stored on the CP side, and all values of the Q-table are encrypted with HE function using (2) at the system configuration stage for the PPRL algorithm. Let the number of states and actions for the Q-learning algorithm be N_s and N_a, respectively, and then the set of states and actions that the user can have at time t be $S_t = \{s_t^1, s_t^2, \ldots, s_t^{N_s}\}$ and $A_t = \{a_t^1, a_t^2, \ldots, a_t^{N_a}\}$, respectively. The state information is transmitted in a binary vector form. Let $V_t = [v_t^1, v_t^2, \cdots, v_t^{N_s}]$ denote a binary vector that corresponds to the current state. If the current state s_t is equal to s_t^i at time t, the i-th element v_t^i is set to 1, whereas the other elements of the binary vector are set to 0, and then the vector encrypted with the FHE secret key. Even though $(N_s - 1)$ elements of V_t have a value of zero, they have different encrypted values after applying HE scheme. The other element with a value of one also has a random value for each time it is encrypted. Therefore, the CP cannot know the user's state information when it receives the encrypted V_t from the users.

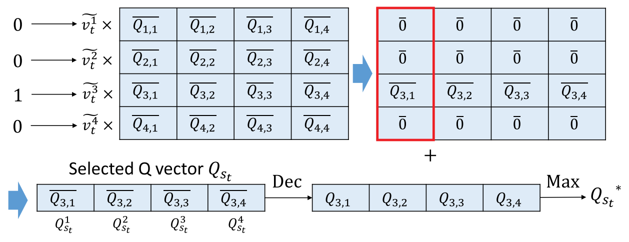

0
0
1
0
$\widetilde{v_t^1}$ ×
$\widetilde{v_t^2}$ ×
$\widetilde{v_t^3}$ ×
$\widetilde{v_t^4}$ ×
$\overline{Q_{1,1}}$ $\overline{Q_{1,2}}$ $\overline{Q_{1,3}}$ $\overline{Q_{1,4}}$
$\overline{Q_{2,1}}$ $\overline{Q_{2,2}}$ $\overline{Q_{2,3}}$ $\overline{Q_{2,4}}$
$\overline{Q_{3,1}}$ $\overline{Q_{3,2}}$ $\overline{Q_{3,3}}$ $\overline{Q_{3,4}}$
$\overline{Q_{4,1}}$ $\overline{Q_{4,2}}$ $\overline{Q_{4,3}}$ $\overline{Q_{4,4}}$
$\overline{0}$
+
Selected Q vector Q_{s_t}
$Q_{s_t}^1$ $Q_{s_t}^2$ $Q_{s_t}^3$ $Q_{s_t}^4$
Dec
$Q_{3,1}$ $Q_{3,2}$ $Q_{3,3}$ $Q_{3,4}$
Max
$Q_{s_t}{}^*$

FIGURE 2. **Example of secure optimal action selection algorithm for $N_s = 4$ and $N_a = 4$.**

Fig. 2 shows how the proposed secure action selection algorithm works for a toy example with $N_s = 4$, $N_a = 4$, and $s_t = s_t^3$. Note that the ciphertexts encrypted with (2) and (6) are represented by $\overline{Q_{i,j}}$ and $\widetilde{v_t^i}$, respectively in Fig. 2. To select an action that maximizes cumulative reward, the user sends the encrypted binary vector representing s_t^3 to the CP. In the CP's side, homomorphic multiplication operations between the i-th element of the binary vector and each element in the i-th row of the Q-table are performed using the binary vector, i.e., $\overline{Q_{i,j}} = \overline{Q_{i,j}} \times_C \widetilde{v_t^i}$ for $j = \{1, \ldots, 4\}$, and then homomorphic addition operations are performed for obtaining sum of each column, i.e., $\overline{Q_{s_t}^j} = \overline{Q_{1,j}} + \ldots + \overline{Q_{4,j}}$ for $j = \{1, \ldots, 4\}$. As shown in Fig. 2, the CP can obtain the selected Q-vector corresponding to s_t^3 by using the homomorphic arithmetic operations, and returns it to the user. After performing decryption, the user can obtain the Q-vector that corresponds to the current state when the current state is s_t^3. Finally, it can decide the action for the state by selecting one with the maximum Q-value.

Algorithm 1 shows the procedure of the proposed secure action selection algorithm. In the beginning of Algorithm 1, a binary vector V_t is determined by the current state of the user. In lines 2-3, the user encrypts the binary vector V_t with the FHE secret key by using (6) for data privacy between the CP and users, and then encrypts again $\widetilde{V_t}$ using RSA public key of the CP for data privacy among multiple users. The user sends it to the CP for obtaining a Q-vector that corresponds to the current state. The CP receives the encrypted binary vector and decrypts it with its own RSA private key to obtain $\widetilde{V_t}$. Since all elements of the vector are binary values, after the for-loop of lines 6-8 is performed, the decrypted value of $\overline{Q_{s_t}^{i,j}}$ will be 0 except for the Q-value corresponding to the current state. In line 9, the CP performs homomorphic addition operations using the calculation results of line 7, and then the results obtained from the operation of line 9 are also equal to the Q-values corresponding to the current state. In lines 11-12, the CP obtains the selected Q-vector Q_{s_t} that stores the Q-values corresponding to the current state after the for-loop of lines 5-10 is performed. Then, the CP encrypts the vector using public key of user u_i requesting the RL service and sends the encrypted vector to the user. The user double-decrypts the received Q-vector using the FHE secret key and its own RSA private key in line 13. In the proposed algorithm, an ϵ-greedy policy can be adopted

Algorithm 1 Secure action selection algorithm

```
1:  Input: $V_t = [v_t^1, v_t^2, \cdots, v_t^{N_s}]$.
2:  (@User) Double-encrypt binary vector for current state
3:  and send $\varepsilon_{pk_{CP}}(\widetilde{V_t})$ to the CP.
4:  (@CP) Decrypts the vector using the private key.
5:  for $j \leq N_a$ do
6:      for $i \leq N_s$ do
7:          (@CP) $\overline{Q_{s_t}^{i,j}} \leftarrow \overline{Q_{i,j}} \times_C \widetilde{v_t^i}$.
8:      end for
9:      (@CP) $\overline{Q_{s_t}^{j}} \leftarrow \overline{Q_{s_t}^{1,j}} + ... + \overline{Q_{s_t}^{N_s,j}}$.
10: end for
11: (@CP) Encrypt Q-vector $\overline{Q_{s_t}} = [\overline{Q_{s_t}^1}, ..., \overline{Q_{s_t}^{N_a}}]$ using
12:  public key of user $u_i$ and send the vector to the user.
13: (@User) Double-decrypt the Q-vector.
14: (@User) Generate a random number $n_r$ in [0, 1]
15: if $n_r < \epsilon$ then
16:     (@User) Select randomly a Q-value $Q_{s_t}^*$ in $Q_{s_t}$.
17: else
18:     (@User) Select a Q-value using max function
19:     $Q_{s_t}^* \leftarrow \max [Q_{s_t}^1, Q_{s_t}^2, ..., Q_{s_t}^{N_a}]$.
20: end if
21: (@User) Find $a_t^x$ by using index $x$ that
22: satisfies $Q_{s_t}^* = Q_{s_t}^x$.
23: Output: Q-value $Q_{s_t}^*$ and action $a_t^x$.
```

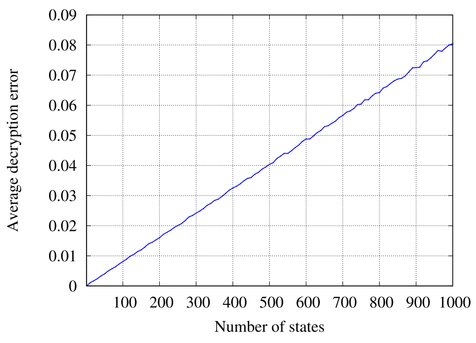
to improve performance of privacy preserving Q-learning algorithm. By using the ϵ-greedy policy, the user selects a Q-value $Q_{s_t}^*$ to find an action among all elements of the decrypted Q-vector in lines 14-20. The user randomly selects one element among all elements of the decrypted Q-vector with the probability ϵ for exploration in line 16. On the other hand, the user finds the maximum value among all elements of the decrypted Q-vector and assigns the maximum value to $Q_{s_t}^*$ with the probability $(1-\epsilon)$ in lines 18-19. Finally, the user selects an action from the set of available actions A_t in the environment by using the index x that satisfies $Q_{s_t}^* = Q_{s_t}^x$ in lines 21-22.

B. SUPPRESSING CUMULATIVE ERROR DUE TO SUCCESSIVE HOMOMORPHIC OPERATIONS

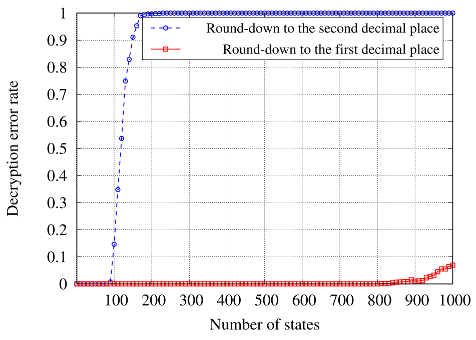
One point that is worthy of mentioning is that in an LWE-based cryptosystem, randomly sampled errors are injected to enhance security. As the number of homomorphic arithmetic operations performed in succession without decryption increases, the magnitude of cumulative error increases. If the magnitude of accumulated error exceeds a certain threshold, it will not be completely removed and the remaining error is added to the plaintext value. In the propose scheme, the number of homomorphic addition operations increases as the number of states increases in line 9 of Algorithm 1. Thus, if the number of states is very large, the accumulated error may increase significantly, resulting in an increase of the probability of decryption error in the LWE-based cryptosystem. It implies that the number of states should be restricted. The remaining error after decryption should be successfully eliminated. Otherwise, it is not possible to increase the number of homomorphic operations that can be performed continuously without decryption error.

We propose a scaling-and-discarding method to reduce the impact of the remaining error on the data. Usually, cryptosystem operations can deal with only integer numbers. A floating-point number used in the Q-learning algorithm should be represented as integers by multiplying a large number. Each time a user sends numeric data to the CP, the user multiplies the data by 10^d and rounds it. When a user receives data from the CP, the user divides the data by 10^d to get the corresponding floating-point value. Note that d is fixed and is known to the CP and users in advance. We propose to protect data in the ciphertext by temporarily increasing d to reduce the impact of the remaining error on the data by using a guard digit g. In the proposed scheme, whenever the user transmits data to the CP, the numeric data is additionally multiplied by 10^g before encryption. That is, a value v is converted to $\mathsf{round}(v \times 10^d) \times 10^g$ and is encrypted. The CP performs a number of homomorphic operations and sends the result to the user. After the user receives and decrypts it, the user divides the results by 10^g and then rounds it down. That is, a decrypted value u is converted to $\mathsf{round\text{-}down}(u \times 10^{-g}) \times 10^{-d}$. At this point, if the error due to the homomorphic operations is less than 10^g, the error can be successfully eliminated. For example, suppose that the guard digit g is 1 and d is 2. When the CP returns the result value of 5220 to the user, the user divides 10 and truncates it below the decimal point to obtain the value of 522. The user divides 522 by 10^2, and finally obtain the desired value of 5.22. Consider the case that the user receives the value of 5222 instead of 5220 due to the remaining error. Even in this case, the users can still obtain 5.22. This is because the remaining error of 2 is discarded using the guard digit through round-down operation. Therefore, even if the remaining error exists after the decryption process, the remaining error below a certain level can be successfully eliminated in the proposed scheme.

Fig. 3 shows the results of decryption errors caused by homomorphic arithmetic operations. Fig. 3(a) shows the average decryption errors with respect to the number of states in case of $d = 3$. The average decryption errors are calculated by averaging the discrepancy between the values with and without homomorphic operations. For each case, the simulation is performed 1000 times for averaging. As shown in Fig. 3(a), the average decryption error increases monotonically as the number of states increases because of the error growth problem in the LWE-based cryptosystem. Fig. 3(b) shows the decryption error rates for the proposed scaling-and-discarding method. The blue line and the red line show the decryption error rates for $d = 2$ and $g = 1$, and those for $d = 1$ and $g = 2$, respectively. The error rates of the blue line and the red line are zero for the number of states below 80 and below 800, respectively. Thus, the proposed scheme can successfully eliminate the remaining error if the



(a) Average error caused by homomorphic operations.



(b) Error rate of decryption.

FIGURE 3. Results related with decryption errors caused by homomorphic arithmetic operations of Algorithm 1 with respect to the number of states in an LWE-based cryptosystem with parameters L = 10^4, p = 10^4, r = 10, N = 4.

number of states is kept below a certain level. However, if the number of state becomes extremely large, the errors can still remain even after applying the proposed scaling-and-discarding method.

To further reduce the decryption errors, we propose to divide a long series of homomorphic operations into multiple pieces if the number of homomorphic operations is greater than the number of homomorphic operations that the system can accommodate. Namely, if a number of homomorphic operations have to be executed, we calculate n_f intermediate results of smaller pieces and send them one by one instead of sending the final result of the homomorphic operations. Let N_w denote the number of homomorphic operations that can be performed without decryption error. Then, n_f is given by round-up(N_s/N_w). In Algorithm 1, homomorphic additions are performed N_s times in line 9. The intermediate results are obtained for every N_w homomorphic operations. Using the upper-bounds of accumulated error for homomorphic addition and multiplication B_{mul} and B_{mul} in Section III-B, the upper-bound of accumulated error for N_w homomorphic additions is given by $N_w \cdot B_{mul}$ because the homomorphic additions are performed N_w times using the results of the homomorphic multiplication in line 7 of Algorithm 1. To perform the homomorphic operations without decryption errors, the upper-bound of accumulated error have to be less than the error bound that the system can accommodate. Note that the error bound that the system can accommodate is given by 10^g. Then, the value of N_w is determined as the maximum value that satisfies the following inequality:

$$\frac{N_w \cdot B_{mul}}{L} \leq 10^g. \tag{10}$$

Using the largest value of N_w in (10), a long series of homomorphic operations are divided into n_f pieces. The CP calculates n_f intermediate results using the divided pieces of homomorphic operations and sends them to the user. After receiving the intermediate results, the user decrypts them without decryption errors. Then, the user adds the decrypted intermediate results to obtain the result of the original entire homomorphic operations. The user can obtain the desired result without decryption error regardless of the number of operations.

C. UPDATING Q-TABLE OF CP WITH ENCRYPTED STATE AND ACTION VALUES

1) PROPOSED Q-TABLE UPDATING ALGORITHM

We propose a Q-table updating algorithm for CP when the state and action values are encrypted and the CP does not decrypt for privacy-preserving purpose. Algorithm 1, users calculate the updated Q-value Q_u using the maximum Q-value and the reward value according to the standard Q-function in (1). The Q-value calculated by the users needs to be updated at the Q table that CP manages. However, if the CP is able to know which element of the Q-table needs to be updated, it comes that the information about the users' state and action is revealed to the CP. The challenge is to develop a Q-table updating algorithm to replace the former Q-value with a new Q-value using only ciphertext without letting the CP know which one is updated among $N_s \times N_a$ elements of the Q-table.

In the proposed PPRL, a user sends the following encrypted values to CP:

- $\overline{Q_u}$ - encrypted Q-value to be updated.
- $\widetilde{W_t}$ - encrypted binary vector representing action.
- $\widetilde{V_t}$ - encrypted binary vector representing state.

Here, the action a_t^* is encoded in $W_t = [w_t^1, w_t^2, \cdots, w_t^{N_a}]$ as done for the states V_t in (V-A). If the i-th action is selected at time t, the i-th element of W_t is set to 1, and all the other elements are set to 0. Even though $(N_a - 1)$ elements of W_t are 0, the ciphertext for 0 is not the same, and thus CP cannot recognize which action is selected among N_a actions. Note that all elements of the binary vectors are encrypted using (6). Using (V-C1), we propose a Q-table updating algorithm for CP as follows:

$$\overline{Q_{i,j}} \leftarrow \overline{Q_{i,j}} + \overline{Q_u} \times_C \widetilde{v_t^i} \times_C \widetilde{w_t^j} - \overline{Q_{i,j}} \times_C \widetilde{v_t^i} \times_C \widetilde{w_t^j}. \tag{11}$$

The multiplication $v_t^i \times w_t^j$ becomes 1 only if the position of $Q_{i,j}$ is equal to the position to be updated because only one element in each binary vector is represented by 1. In other cases, the results of the multiplication become zero. Therefore, with elements of binary vectors, the CP can update the Q-table without knowing a specific position of Q-table using the homomorphic arithmetic operations in (11). Fig. 4 shows a toy example of update procedure for $s_t = s_t^2$, $a_t = a_t^2$, $N_s = 2$ and $N_a = 2$. After the homomorphic operations in (11), the Q-value $Q_{2,2}$ is updated to Q_u, and the rest of the values in the Q-table remain unchanged because only the homomorphic multiplication of v_t^2 and w_t^2 is 1 as shown in Fig. 4.

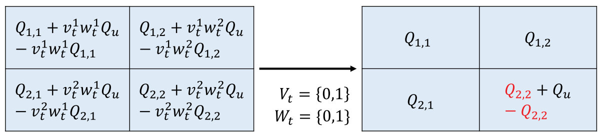

$Q_{1,1} + v_t^1 w_t^1 Q_u - v_t^1 w_t^1 Q_{1,1}$
$Q_{1,2} + v_t^1 w_t^2 Q_u - v_t^1 w_t^2 Q_{1,2}$
$Q_{2,1} + v_t^2 w_t^1 Q_u - v_t^2 w_t^1 Q_{2,1}$
$Q_{2,2} + v_t^2 w_t^2 Q_u - v_t^2 w_t^2 Q_{2,2}$
$V_t = \{0,1\}$
$W_t = \{0,1\}$
$Q_{1,1}$
$Q_{1,2}$
$Q_{2,1}$
$Q_{2,2} + Q_u - Q_{2,2}$

FIGURE 4. **Example of Q-function update algorithm for $N_s = 2$ and $N_a = 2$.**

Algorithm 2 Secure Q-Table Updating Algorithm

```
1: Input: $\alpha,\ \gamma,\ r_t,\ Q_s^*,\ W_t$.
2: (@User) Calculate $r_\alpha = \alpha \times r_t,\ Q_\alpha^* = \alpha \times \gamma \times Q_s^*$,
3: $Q_{1-\alpha}^* = (1-\alpha) \times Q_s^*$.
4: if $r_\alpha'$ and $Q_{1-\alpha}'$ are not empty then
5:     (@User) $Q_u \leftarrow Q_{1-\alpha}' + Q_\alpha^* + r_\alpha'$.
6:     (@User) $r_\alpha' \leftarrow r_\alpha,\ Q_{1-\alpha}' \leftarrow Q_{1-\alpha}^*$.
7:     (@User) Double-encrypt $Q_u$ and $W_t$.
8:     (@User) Send $\varepsilon_{pk_{CP}}(\overline{Q_u}),\ \varepsilon_{pk_{CP}}(\widetilde{W_t})$ to the CP.
9:     (@CP) Decrypts the values using the private key.
10:    for $i \le N_s$ do
11:        for $j \le N_a$ do
12:            (@CP) $\overline{Q_{i,j}} \leftarrow \overline{Q_{i,j}} + \overline{Q_u} \times_C \widetilde{v_i'} \times_C \widetilde{w_j'}$
13:            $-\overline{Q_{i,j}} \times_C \widetilde{v_i'} \times_C \widetilde{w_j'}$.
14:        end for
15:    end for
16:    (@CP) $\widetilde{W_t}' \leftarrow \widetilde{W_t}$
17: else
18:    (@User) $r_\alpha' \leftarrow r_\alpha,\ Q_{1-\alpha}' \leftarrow Q_{1-\alpha}^*$.
19:    (@User) Double-encrypt $W_t$ and send $\varepsilon_{pk_{CP}}(\widetilde{W_t})$.
20:    (@CP) Decrypts the value using the private key.
21:    (@CP) $\widetilde{W_t}' \leftarrow \widetilde{W_t}$
22: end if
23: Output: Updated Q-table.
```

Algorithm 2 describes the procedure of the proposed secure Q-table updating algorithm. The inputs to Algorithm 2 are learning rate α, discount factor γ, reward from the environment r_t, maximum Q-value obtained by Algorithm 1 Q_s^*, and the binary vector $W_t = [w_t^1, w_t^2, \cdots, w_t^{N_a}]$. The user calculates the multiplications in lines 2-3 using floating-point numbers α and γ to obtain a newly updated Q-value. At first iteration, r_α and $Q_{1-\alpha}^*$ are stored for the next updating iteration in line 18. On the other hand, the user calculates the updated Q-value Q_u with r_α' and $Q_{1-\alpha}'$ stored in the previous iteration for updating the Q-table, and then current information r_α and $Q_{1-\alpha}^*$ are stored for the next updating iteration as shown in lines 5-6. The user encrypts the updated Q-value Q_u with HE function (2) and binary vector W_t with HE function (6). Then, the user encrypts agian the ciphertexts $\overline{Q_u}$ and $\widetilde{W_t}$ using the RSA public key of the CP, and sends them to the CP. For the first iteration, the user encrypts only the binary vector W_t and sends it to the CP in line 19. After receiving the encrypted values sent by the user, the CP decryps them using its own RSA private key in line 9 and 20. In the case of the first iteration, only the binary vector $\widetilde{W_t}$ is stored in line 21. Otherwise, the CP performs the homomorphic arithmetic operations for updating the Q-table, and then $\widetilde{W_t}$ is stored as shown in lines 10-16. The binary vector representing the state previously sent for Algorithm 1 $V' = [v_1', v_2', \cdots, v_{N_s}']$ is used to find the position to be updated with the binary vector representing the action previously sent by the user $W' = [w_1', w_2', \cdots, w_{N_a}']$. Therefore, we can successfully update the encrypted Q-table using only the ciphertexts by using (11) in lines 12-13.

2) ERROR ANALYSIS OF THE PROPOSED Q-TABLE UPDATING ALGORITHM

In the proposed scheme, the LWE-based cryptosystem is used for FHE. The probability of decryption error in the LWE-based cryptosystem increases as the number of successive homomorphic operations increases because injected random errors for enhancing the security level of the cryptosystem accumulates after performing successive homomorphic operations without decryption. In the existing scheme, the bootstrapping algorithm is used to suppress the increase of errors. However, the algorithm has a high computational complexity and has to be executed periodically whenever errors accumulate above a certain level. The proposed Q-table updating algorithm can control the increases of errors without bootstrapping algorithm. This is because the magnitude of the error converges even though the number of iterations to update encrypted Q-values keeps increasing.

Proposition 1: The cumulative error converges as the number of iterations to update encrypted Q-value using (11) increases.

Proof: In order to prove the convergence of the accumulated error, we analyze how the error changes as the number of iterations for updating Q-table on the CP side increases. Let $\Delta_{i,j}$ denote the error change for the Q-table updating algorithm in (11). Then, $\Delta_{i,j}$ is given by

$$\begin{aligned}\Delta_{i,j} &= v_i' \cdot (w_j' e_u + \sum_{k=1}^{Q} c_u^k e_w^k) + \sum_{k=1}^{Q} c_{wu}^k e_v^k \\ &\quad - v_i' \cdot (w_j' e_f + \sum_{k=1}^{Q} c_f^k e_w^k) - \sum_{k=1}^{Q} c_{wf}^k e_v^k \\ &= v_i' \cdot w_j' (e_u - e_f) + v_i' \sum_{k=1}^{Q} \{c_u^k e_w^k - c_f^k e_w^k\} \\ &\quad + \sum_{k=1}^{Q} \{c_{wu}^k e_v^k - c_{wf}^k e_v^k\}, \qquad (12)\end{aligned}$$

where e_f and e_u are the errors injected in the HE function using (2) for the former Q-value and the updated Q-value, respectively, e_v^k and e_w^k are the k-th element of error vector injected in the HE function using (6) for binary vector representing state and action, respectively, c_u^k, $c_{\widetilde{wu}}^k$, c_f^k, and c_{wf}^k are the k-th element of $D(\overline{Q_u})$, $D(\overline{Q_u} \times_C \widetilde{w_j'})$, $D(\overline{Q_{i,j}})$, and $D(\overline{Q_{i,j}} \times_C \widetilde{w_j'})$, respectively, and Q is $\log q \cdot (N+1)$.

TABLE 1. Converged error for each binary value.

	$w_j^{'} = 0$	$w_j^{'} = 1$
$v_i^{'} = 0$	$\sum(c_{wu}^i e_v^i - c_{wf}^i e_v^i)$	$\sum(c_{wu}^i e_v^i - c_{wf}^i e_v^i)$
$v_i^{'} = 1$	$\sum(c_u^i e_w^i + c_{wu}^i e_v^i - c_f^i e_w^i - c_{wf}^i e_v^i)$	$e_u - e_f + \sum(c_u^i e_w^i + c_{wu}^i e_v^i - c_f^i e_w^i - c_{wf}^i e_v^i)$

There are four cases of v_i' and w_j' shown in Table 1. The summation $\sum\{c_u^k e_w^k - c_f^k e_w^k\}$ and $\sum\{c_{wu}^k e_v^k - c_{wf}^k e_v^k\}$ converge to zero as the number of iterations increases sufficiently because the value of the elements of the decomposed vectors is in the range of integer values from 0 to 9 and follows an uniform distribution as shown in (7), and the errors e_v^k and e_w^k are sampled from the same uniform distribution. Then, the error change $\Delta_{i,j}$ can be simply represented as $\Delta_{i,j} = v_i' w_j' (e_u - e_f)$. Let e_u^l and e_f^l denote the error of the updated Q-value and the former Q-value at the l-th iteration, respectively. For $v_i' = 1$ and $w_j' = 1$, the error change at the l-th iteration can be represented as $\Delta_{i,j}(l) = (e_u^l - e_f^l) = (e_u^l - e_u^{l-1})$ because e_f^l is equal to e_u^{l-1}. Namely, the summation of the error change $\sum_{l=1}^{N_l} \Delta_{i,j}(l)$ can be represented as $e_u^{N_l} - e_f^1$. Therefore, because $\sum \Delta_{i,j}$ converges for all cases of v_i' and w_j', the error does not accumulate even if the number of iterations to update the encrypted Q-value without decryption using (11) increases. $\square$

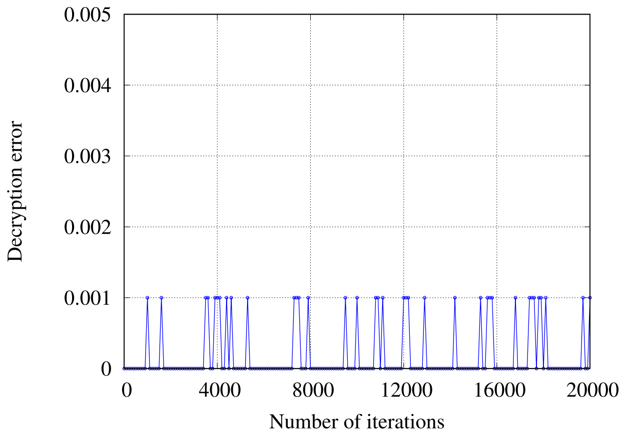
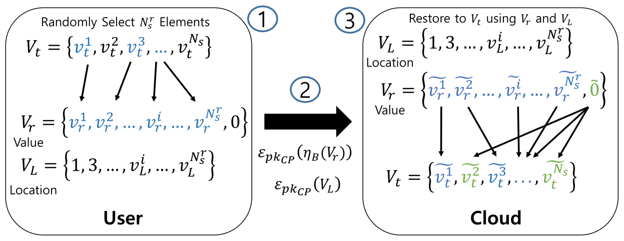


FIGURE 5. Error caused by homomorphic arithmetic operations in Algorithm 2 with respect to the number of iterations.

Fig. 5 shows the decryption error caused by homomorphic arithmetic operations in (11) with respect to the number of iterations to update the encrypted Q-values. In this simulation, the number of states and actions is 12 and 4, respectively, and d is set to 3. The other parameters are the same as in the environment of Fig. 3. The decryption error in Fig. 5 fluctuates between 0 and 0.001 without increasing to more than 0.001 even if the number of iterations for updating the Q-table increases. As a result, the accumulated error does not continue to grow in the proposed Q-table updating algorithm even if the number of successive homomorphic arithmetic operations in (11) without decryption increases.

D. REDUCING THE DIMENSION OF THE BINARY VECTORS FOR THE STATE AND ACTION REPRESENTATION WITH A RANDOM SELECTION METHOD

In the proposed PPRL algorithm, the number of states and actions is determined according to a given environment. If the number of states and actions is very large, the exchange of states and actions may incur significant communication overhead between the CP and users. The binary vectors that a user transmits for the states and actions include a lot of redundant information because every element of the binary vectors representing the state and action has a value of zero except one element. When the number of states and actions is large, the binary vector needs to be represented as a reduced form with less redundant information to reduce the communication overhead.



Randomly Select N_s^r Elements
$V_t = \{v_t^1, v_t^2, v_t^3, \dots, v_t^{N_s}\}$
$V_r = \{v_r^1, v_r^2, \dots, v_r^i, \dots, v_r^{N_s^r}, 0\}$
Value
$V_L = \{1, 3, \dots, v_L^i, \dots, v_L^{N_s^r}\}$
Location
User
1
2
$\varepsilon_{pk_{CP}}(\eta_B(V_r))$
$\varepsilon_{pk_{CP}}(V_L)$
3
Restore to V_t using V_r and V_L
$V_L = \{1, 3, \dots, v_L^i, \dots, v_L^{N_s^r}\}$
Location
$V_r = \{\widetilde{v_r^1}, \widetilde{v_r^2}, \dots, \widetilde{v_r^i}, \dots, \widetilde{v_r^{N_s^r}}, \tilde{0}\}$
Value
$V_t = \{\widetilde{v_t^1}, \widetilde{v_t^2}, \widetilde{v_t^3}, \dots, \widetilde{v_t^{N_s}}\}$
Cloud

FIGURE 6. Procedure for reducing the dimension of the binary vector for the state representation.

Instead of sending a full binary vector, a user randomly chooses N_s^r elements including the element of '1' among N_s elements of the binary vector and sends the N_s^r elements with their indexes. Then, the CP knows that the selected state or action is one of the N_s^r elements but cannot specify which one is the exactly selected state or action. When the user sends N_s^r elements, it has to send one single element of zero, and the CP uses $\tilde{0}$ for the other $(N_s - N_s^r)$ elements that are not received. Fig. 6 illustrates the procedure for reducing communication overhead when a user sends the binary vector for the state representation to the CP. At first, instead of the binary vector $V_t = [v_t^1, v_t^2, \cdots, v_t^{N_s}]$, the user creates a vector $V_r = [v_r^1, v_r^2, \cdots, v_r^{N_s^r}, 0]$ by choosing N_s^r elements among N_s elements of the binary vector and a vector representing their indexes $V_L = [v_L^1, v_L^2, \dots, v_L^{N_s^r}]$. Here, the element of '1' should be included in V_r. Second, the user encrypts V_r with the HE function in (6), and encrypts again $\widetilde{V_r}$ and V_L with the RSA public key for the CP, and then sends the RSA encrypted vectors $\varepsilon_{pk_{CP}}(\widetilde{V_r})$ and $\varepsilon_{pk_{CP}}(V_L)$ to the CP. After receiving the encrypted vectors, the CP decrypts them with its own RSA private key. Even though the number of elements sent by the user is reduced, the CP still unable to recognize which state is selected among N_s^r elements because the ciphertext for each elements is not the same. Finally, the CP places the element of V_r at the appropriate location of V_t using the index number stored in V_L, and fills the remaining locations of V_t with $\widetilde{0}$. The number of elements to transmit is then reduced from N_s to $(2\,N_s^r + 1)$ when the user sends the encrypted binary vector for the state representation. Therefore, we can suppress the communication overhead below a certain level by adjusting the dimension of the binary vector even if the number of states is very large.

VI. OVERHEAD ANALYSIS

This section describes the theoretical analysis of the proposed privacy-preserving Q-learning algorithm in terms of computational and communication overhead.

A. COMPUTATIONAL OVERHEAD

The propsed SCC-PPRL algorithm is theoretically analyzed in terms of computational complexity. The computational complexity of the proposed algorithm is influenced by the number of states and actions, as well as the parameters of the LWE-based cryptosystem. Note that p is the cardinality of the plaintext set, L is the parameter representing the limitation of random errors injected into the ciphertext for enhancing the security level, and N is the vector size of ciphertexts. In addition, the proposed algorithm includes the RSA algorithm for the DE scheme. The computational complexity of RSA encryption and decryption is affected by the size of the RSA public key and private key. Let R_n be the multiplication of two large prime numbers $p' \cdot q'$. According to a key generation procedure in the RSA algorithm, the integer numbers R_e and R_d can be determined using p' and q' for RSA public key and private key, respectively, and then the RSA public key and RSA private key are represented by (R_n, R_e) and (R_n, R_d), respectively. The computational complexity of RSA encryption and decryption can be represented by $\mathcal{O}(R_e)$ and $\mathcal{O}(R_d)$, respectively, because the exponentiation operation in RSA encryption and decryption requires R_e and R_d multiplications, respectively.

TABLE 2. Computational complexity of secure algorithms.

	Action selection algorithm	Q-table updating algorithm
User	$\mathcal{O}(N_s \cdot N \log q \cdot (N + R_e) + N_a \cdot N \cdot R_d)$	$\mathcal{O}(N_a N \log q \cdot (N + R_e))$
CP	$\mathcal{O}(N_s \cdot N_a \cdot N^2 \log q + N \cdot (N_s \log q R_d + N_a R_e))$	$\mathcal{O}(N_s \cdot N_a \cdot N^2 \log q + N \cdot N_a \log q \cdot R_d)$

We analyze the computational complexity of the secure action selection algorithm and secure Q-table updating algorithm in the user and CP, respectively. Table 2 summarizes the computational complexity of the secure action selection algorithm and secure Q-table updating algorithm for the user and CP, respectively.

1) OVERHEAD OF ACTION SELECTION ALGORITHM AT USER SIDE

For the secure action selection algorithm, the DE scheme using the HE and RSA algorithms, and the function to find the maximum value are required at the user's side. The computational complexity of the HE function using (6) is given by $\mathcal{O}(N^2 \log q)$ because the computational complexity of the multiplication between m_i and R, and the computational complexity of encrypting the zero vector using the HE function (2) are both $\mathcal{O}(N^2 \log q)$ as shown in (6). In addition, the computational complexity of the homomorphic decryption function and the function that finds the maximum value for selecting action are $\mathcal{O}(N)$ and $\mathcal{O}(N_a)$, respectively. The HE function using (6) is executed N_s times, the homomorphic decryption function is executed N_a times, the function to find the maximum value is executed once, and the RSA encryption and decryption functions for the DE scheme are executed $N_s \cdot N \log q$ and $N_a \cdot N$ times in the user's side, respectively. Thus, the computational complexity of the secure action selection algorithm for the user is given by $\mathcal{O}(N_s \cdot N \log q \cdot (N + R_e) + N_a \cdot N \cdot R_d)$.

2) OVERHEAD OF ACTION SELECTION ALGORITHM AT CP SIDE

To perform the secure action selection algorithm in the CP's side, the homomorphic multiplication, homomorphic addition, RSA encryption and decryption operations are required. The computational complexity of homomorphic multiplication can be represented as $\mathcal{O}(N^2 \log q)$. This is because the computational complexity of decomposition function (7) and matrix multiplication between the decomposed vector and a ciphertext encrypted with (6) are $\mathcal{O}(N \log q)$ and $\mathcal{O}(N^2 \log q)$, respectively, as shown in (8). The computational complexity of homomorphic addition is given by $\mathcal{O}(N)$ in (4). Therefore, since the homomorphic multiplication and homomorphic addition are performed $N_s \cdot N_a$ times, and the RSA encryption and decryption are executed $N_a \cdot N$ and $N_s \cdot N \log q$ times, respectively, the computational complexity of the secure action selection algorithm in the CP's side is given by $\mathcal{O}(N_s \cdot N_a \cdot N^2 \log q + N \cdot (N_s \log q \cdot R_d + N_a \cdot R_e))$.

3) OVERHEAD OF Q-TABLE UPDATING ALGORITHM AT USER SIDE

For the implementation of the secure Q-table updating algorithm, the computational complexity in the user's side is given by $\mathcal{O}(N_a N \log q \cdot (N + R_e))$. This is because the HE functions using (6) and (2) are performed N_a times and once, respectively, and the RSA encryption function is performed $N_a \cdot N \log q$ times.

4) OVERHEAD OF Q-TABLE UPDATING ALGORITHM AT CP SIDE

The computational complexity of the secure Q-table updating algorithm in the CP's side is $\mathcal{O}(N_s \cdot N_a \cdot N^2$

$\log q + N \cdot N_a \log q \cdot R_d$) because the homomorphic multiplication and homomorphic addition are performed $4N_sN_a$ and $2N_sN_a$ times, respectively, and the decryption function of RSA algorithm is performed $N_a \cdot N \log q$ times.

B. COMMUNICATION OVERHEAD

To preserve data privacy of users, the LWE-based FHE algorithm and RSA algorithm are utilized in the proposed algorithm. In the LWE-based FHE scheme, the encrypted values have a matrix form, and each element of that matrix is round-up$(\log q/\log 2)$ bits. For the secure action selection algorithm, the user sends N_s matrices of size $\log q \cdot (N+1) \times (N+1)$ to represent state information and receives N_a matrices of size $n_f \times (N+1)$ to select an action. Note that the number of partitions n_f is obtained by round-up(N_s/N_w) as described in Section V-A. For the secure Q-table updating algorithm, the user sends N_a matrices of size $\log q \cdot (N+1) \times (N+1)$ and a matrix of size $1 \times (N+1)$ to the CP. Thus, communication overhead is affected by the parameters of the LWE-based cryptosystem, the number of states, and the number of actions. In addition, if the values encrypted with the HE function are encrypted again with the RSA encryption algorithm for the DE scheme, the elements of each matrix become R_n bits. Thus, a fully double encryption (FDE) scheme that double-encrypts all elements of the matrices can significantly increase communication overhead if R_n is much greater than round-up$(\log q/\log 2)$. However, even if only some elements of the matrix are encrypted with the RSA algorithm, the security level of the proposed scheme is still high enough because all values of the matrix are required to decrypt the value encrypted with the HE algorithm. Therefore, to reduce the communication overhead of the DE scheme, it can be implemented by encrypting only some elements of the matrix instead of applying the FDE scheme.

The number of states and actions is determined by a given environment, while the parameters for cryptosystem are selected in the system configuration process. Thus, in case that the number of states and actions is very large due to the complexity of a given environment, communication overhead can be significantly increased. Reducing the dimension of the binary vectors with a random selection method in Section V-D can decrease the communication overhead because it reduces redundant data exchanged between the user and CP. In case that the proposed scheme is adopted, the communication overhead is then reduced from $N_s \cdot \log Lp \cdot (N+1) \times (N+1)$ to $(N_s^r+1) \cdot \log Lp \cdot (N+1) \times (N+1) + N_s^r \cdot 1 \times (N+1)$, when the user sends encrypted state information. Therefore, the communication overhead can be significantly reduced because N_s^r can be selected with a much smaller number than N_s.

VII. PERFORMANCE EVALUATION

This section presents the simulation studies to evaluate the performance of the proposed algorithms.

A. ACCURACY OF PRIVACY-PRESERVING Q-LEARNING

In this section, we applied the proposed privacy-preserving Q-learning in a simple frozen lake problem to verify the feasibility of the SCC-PPRL algorithm in the cloud computing environment. Fig. 7 shows examples of frozen lake environment. S represents the starting point for the agent and G represents the goal. White and gray boxes represent a safe surface and a hole, respectively. When the agent reaches the hole, the agent fails the mission and has to go back to the starting point. The agent can find optimal route to reach the goal by avoiding holes through Q-learning algorithms. The frozen lake environment is implemented using MATLAB for the simulation. The parameters of the LWE cryptosystem p, L, r, and N are 10^4, 10^4, 10, and 4, respectively. Discount factor γ and learning rate α are 0.9 and 0.1, respectively, and ϵ for ϵ-greedy policy is 0.1 in this simulation environment. The number of actions is set to 4.

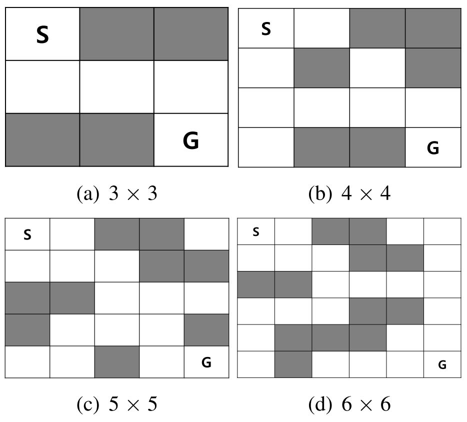


FIGURE 7. An example of frozen lake environment.

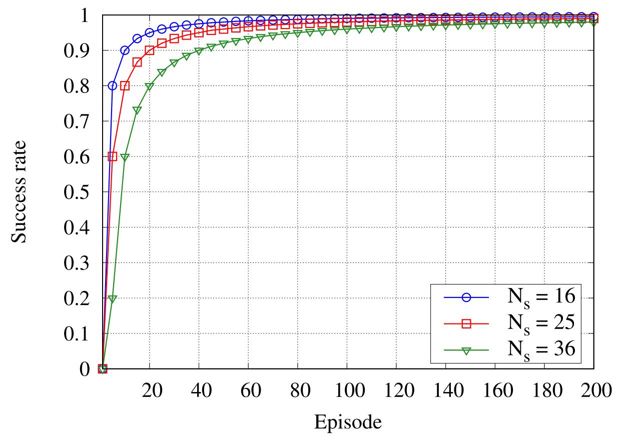


FIGURE 8. Success rate of privacy-preserving Q-learning with respect to the number of episodes in a frozen lake problem.

Fig. 8 shows the success rate of the proposed SCC-PPRL scheme for different numbers of states in the frozen lake environment with respect to the number of episodes. As the number of states increases, since the complexity of the problem increases, convergence speed decreases as shown in Fig. 8. All three cases in the simulation have success rates of more than 0.9 before 40 episodes, and the rates converge to near 1 as the number of episodes increases. This shows that the proposed privacy-preserving Q-learning algorithm works successfully even if the user only provides ciphertexts to the CP. Therefore, the user can receive RL-based services from the CP without personal information leakage.

B. OVERHEAD IN PRIVACY-PRESERVING Q-LEARNING

The effectiveness of the proposed SCC-PPRL algorithm is verified by comparing with the SMC-based algorithm developed in [6] in terms of computational and communication overhead. In the distributed public key cryptosystem using the AHE scheme for the SMC-based PPRL algorithm, the key size is selected as 1024 bits, and the cloud computing-based service provider consists of one CP and one health service provider (HSP) for the distributed cryptosystem in the simulation environment. The private key is split into two partial private keys. Then, the CP has a partial private key and the HSP has the remaining partial private key. On the other hand, in the cryptosystem using the DE scheme for the proposed SCC-PPRL algorithm, the key size used for RSA algorithm is selected as 1024 bits, and the parameters of the LWE-based FHE scheme p, L, r, and N are selected 10^4, 10^4, 10, and 10. Furthermore, in the proposed SCC-based PPRL algorithm using DE scheme, since data privacy between users can be preserved even if the DE scheme is applied to only some elements of matrices as described in Section VI-B, the DE scheme is applied to elements corresponding to 10 percent of the matrices for the simulation studies.

1) COMPUTATIONAL OVERHEAD

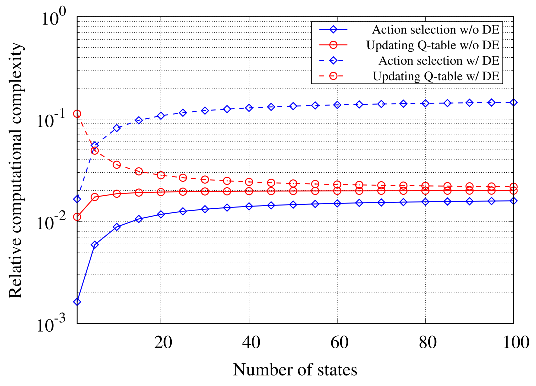
To compare the computational overhead of the proposed SCC-PPRL scheme with the compuational overhead of the SMC-based PPRL scheme developed in [6], the relative computational complexity (RCC) is calculated. The RCC is defined as follows:

$$RCC = \frac{\text{computational complexity of the proposed PPRL}}{\text{computational complexity of SMC-based PPRL}}.$$

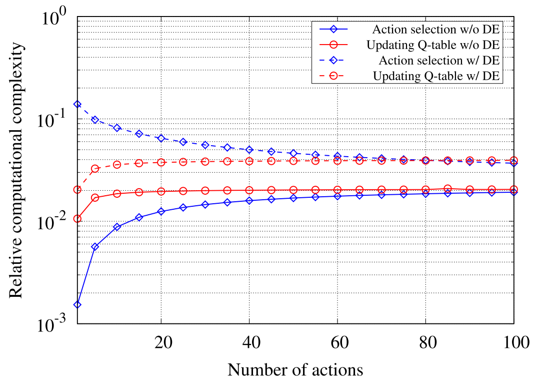
Fig. 9 shows the RCC of the proposed SCC-PPRL and the SMC-based PPRL algorithms with respect to the number of states and actions. The number of actions is set to 10 in Fig. 9(a), and the number of states is set to 10 in Fig. 9(b). If the RCC is smaller than 1, the computational overhead of the proposed SCC-PPRL algorithm is less than that of the SMC-based algorithm.

Fig. 9(a) and Fig. 9(b) show the RCCs in the secure action selection and secure Q-table updating algorithms as the number of states and actions increases, respectively. As shown in Fig. 9(a) and Fig. 9(b), the RCCs converge to a certain value as the number of states and actions increases because the computational complexity of both the SMC and SCC-PPRL algorithms increases linearly for N_s and N_a. Furthermore, since the value of RCC is always smaller than 1, it can be seen that the computational overhead of the SCC-PPRL algorithm is less than that of the SMC-based algorithm. In the SMC-based algorithm, the encryption and decryption operations through the cooperation of distributed computing servers have a dominant effect on computational complexity because the exponentiation operation in the encryption and decryption procedures requires much more computation than the other operations such as addition and multiplication. For the PPRL-based services, the SMC-based algorithm using the AHE-based distributed cryptosystem requires many encryption and decryption operations because each basic homomorphic operation such as multiplication, comparison, and equality test requires multiple encryption and decryption operations. On the other hand, in the proposed SCC-PPRL algorithm, the encryption and decryption operations are performed only in a data exchange procedure.

In the case where the DE scheme is applied to the proposed SCC-PPRL algorithms, the RCCs of the action selection and Q-table updating algorithms are greater than those without DE scheme as shown in Fig. 9(a) and Fig. 9(b). The reason that the algorithms using the DE scheme have higher computational complexity than those without the DE scheme is that additional RSA encryption and decryption operations are required. In Fig. 9(a), as the number of states

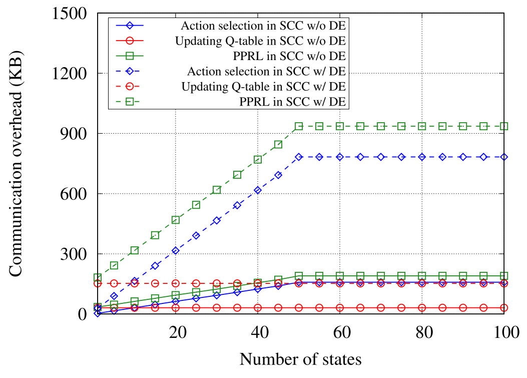


(a) Relative computational complexity with respect to the number of states.

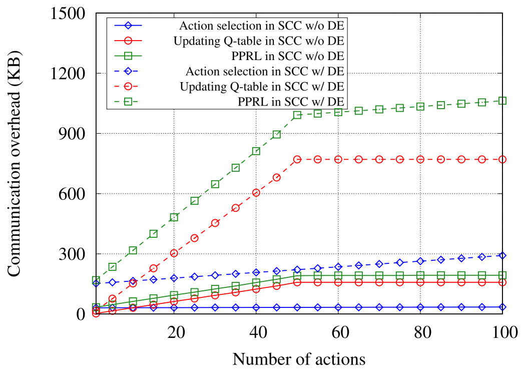


(b) Relative computational overhead with respect to the number of actions.

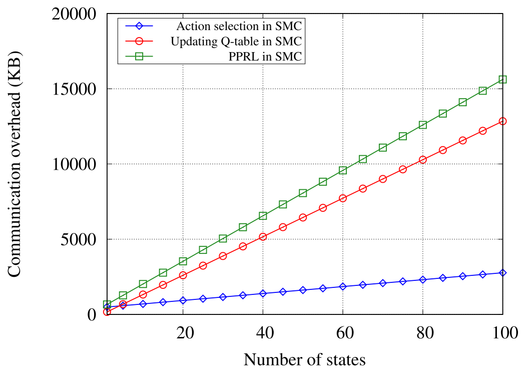
FIGURE 9. Relative computational complexity of SMC-based and proposed SCC-PPRL for algorithms with and without double encryption.



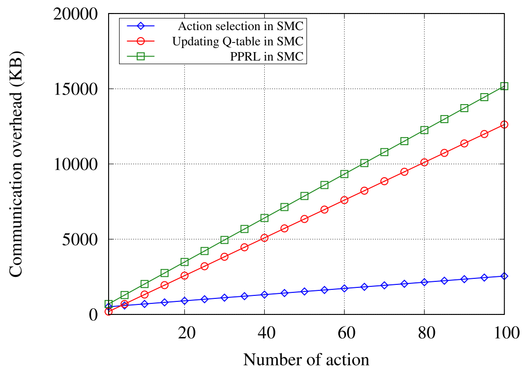
(a) Communication overhead of the SCC-PPRL algorithm with respect to the number of states.



(b) Communication overhead of the SCC-PPRL algorithm with respect to the number of actions.



(c) Communication overhead of the SMC-based PPRL algorithm with respect to the number of states.



(d) Communication overhead of the SMC-based PPRL algorithm with respect to the number of actions.

FIGURE 10. Communication overhead of the SCC-PPRL and SMC-based PPRL algorithms with respect to the number of states and actions.

increases, the RCCs of the action selection algorithm with and without DE scheme increase. This is because the number of elements of the binary vector representing the states to be encrypted using the DE scheme increases as the number of states increases in the action selection algorithm. The RCC of the Q-table updating algorithm with DE scheme becomes close to that without DE scheme because the amount of information exchange between the user and CP is almost the same regardless of the number of states. In Fig. 9(b), as the number of actions increases, the RCC of the Q-table updating algorithm with DE scheme increases because the number of elements to be encrypted using the DE scheme is increased as the number of elements of the binary vector representing the actions is increased in the Q-table updating algorithm. In the action selection algorithm, as the number of states increases, the number of elements of the selected Q-vector is increased. However, the effect of the increase in computational complexity by the DE scheme is less than that of Q-table updating algorithm because the size of the selected Q-vector encrypted with (2) is smaller than the size of the binary vector representing the actions encrypted by (6).

2) COMMUNICATION OVERHEAD

To compare the efficiency of the proposed SCC-PPRL algorithm with the SMC-based PPRL algorithm [6] in terms of the communication overhead, the amount of communication data generated while each PPRL algorithm is running is measured. In the simulation environment, the technique for reducing the dimension of the binary vectors in Section V-D is enabled with $N_s^r = 50$ if the number of states and actions is equal to or greater than 50.

Fig. 10(a) and Fig. 10(b) show the communication overhead for the action selection and Q-table updating in the proposed SCC-PPRL algorithm with respect to number of states and actions. In the figures, we plot the overhead of PPRL algorithm, which is the summation of the communication overhead for the action selection and Q-table updating algorithms. The number of actions is set to 10 in Fig. 10(a), and the number of states is set to 10 in Fig. 10(b). In Fig. 10(a), the communication overhead of the action selection algorithm increases as the number of states increases up to 50, and it levels off when the number of states exceeds 50. This is because the number of elements of binary vector representing the states for the action selection algorithm increases as the number of states increases up to 50. If the number of states is greater than 50, the technique for reducing the dimension of the binary vectors is applied to the action selection algorithm, and it successfully suppresses the increase in communication overhead. On the other hand, it is observed that the communication overhead of the Q-table updating algorithm is almost constant regardless of the number of states because communication data between the user and CP in the Q-table updating algorithm does not change as the number of states increases. In addition, the communication overhead of the algorithms with DE scheme is higher than that of the algorithms without DE scheme because the length of each element of the matrix to be transmitted becomes longer if RSA encryption is applied to the elements for DE scheme.

In Fig. 10(b), the communication overhead of the Q-table updating algorithm increases as the number of actions increases up to 50, and it levels off when the number of actions exceeds 50. This is because the number of binary vectors representing the actions for the Q-table updating algorithm increases as the number of actions increases up to 50. If the number of actions is greater than 50, the technique for reducing the dimension of the binary vectors is applied to the Q-table updating algorithm, and it successfully suppresses the increase in communication overhead. The communication overhead in the action selection algorithm also increases as the number of actions increases because the number of elements in the selected Q-vector returned by the CP increases as the number of actions increases. The communication overhead in the Q-table updating algorithm increases much faster than that of the action selection algorithm because the size of the matrix encrypted by (6) for the binary vector is greater than that of the matrix encrypted by (2) for the selected Q-vector. Moreover, for the same reason in Fig. 10(a), it is observed that the communication overhead increases if DE scheme is applied in Fig. 10(b) as well.

For comparison purpose, we plot the communication overhead for the action selection and Q-table updating in the SMC-based PPRL algorithm [6] with respect to the number of states and actions in Fig. 10(c) and Fig. 10(d). In the figures, the overhead of PPRL algorithm is the summation of the communication overhead for the action selection and Q-table updating algorithms. As the number of states and actions increases, the communication overhead of the action selection and Q-table updating algorithms constantly increases. Moreover, it is seen that the communication overhead of the SMC-based algorithm increases linearly and becomes very large in comparison with the communication overhead of the proposed SCC-PPRL algorithms. This reason is that the SMC-based PPRL algorithm using the distributed cryptosystem requires a number of data exchanges between CP and HSP for performing the HE basic functions such as multiplication, comparison, and equality test for the implementation of PPRL algorithm. On the contrary, because the proposed SCC-PPRL algorithm stores and processes the data in a single cloud server without any support of third-parties, the communication overhead for PPRL algorithm is significantly reduced. Therefore, the communication overhead of the proposed SCC-PPRL algorithm is significantly less than that of the SMC-based PPRL algorithm.

VIII. CONCLUSION

This paper has proposed the SCC-PPRL algorithm using FHE scheme for reinforcement learning with the aim of preserving user's data privacy in a cloud computing infrastructure. We have also proposed a double encryption scheme that uses both FHE and RSA algorithms to provide further data confidentiality to users who shares the same FHE key in a multi-user cloud environment. The proposed SCC-PPRL algorithm consists of a secure action selection algorithm and secure Q-table updating algorithm for secure reinforcement learning. To solve the error growth problem in the LWE-based FHE scheme, we have proposed a scaling-and-discarding method that divides a long series of homomorphic operations into multiple pieces. Theoretical analysis and simulation studies showed that the proposed PPRL algorithm has low computation and communication overhead in comparison with conventional PPRL algorithms.

For future research efforts, we will expand the proposed PPRL to other RL algorithms such as deep RL and multi-agent RL algorithms and evaluate the performance of the PPRL algorithms in more complicated real-world reinforcement learning problems.

REFERENCES

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of things data analysis: A survey," *Digit. Commun. Netw.*, vol. 4, no. 3, pp. 161–175, Aug. 2018.

[2] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," 2019, *arXiv:1908.08796*. [Online]. Available: http://arxiv.org/abs/1908.08796

[3] T. G. Fischer, "Reinforcement learning in financial markets–A survey," Discuss. Papers Econ., Univ. Erlangen-Nürnberg, Erlangen, Germany, Tech. Rep. 12, 2018.

[4] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 843–859, May 2013.

[5] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[6] X. Liu, R. Deng, K.-K.-R. Choo, and Y. Yang, "Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes," *IEEE Trans. Emerg. Topics Comput.*, early access, Jan. 30, 2019, doi: 10.1109/TETC.2019.2896325.

[7] J. Sakuma, S. Kobayashi, and R. N. Wright, "Privacy-preserving reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 864–871.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, vol. 9, 2009, pp. 169–178.

[9] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–35, 2018.

[10] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2010, pp. 24–43.

[11] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proc. Annu. Cryptol. Conf.*, 2011, pp. 505–524.

[12] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM J. Comput.*, vol. 43, no. 2, pp. 831–871, 2014.

[13] C. Peikert and S. Shiehian, "Multi-key FHE from LWE, revisited," in *Proc. Theory Cryptogr. Conf.*, 2016, pp. 217–238.

[14] Y. Doröz, J. Hoffstein, J. Pipher, J. H. Silverman, B. Sunar, W. Whyte, and Z. Zhang, "Fully homomorphic encryption from the finite field isomorphism problem," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2018, pp. 125–155.

[15] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[16] X. Fei, N. Shah, N. Verba, K.-M. Chao, V. Sanchez-Anguix, J. Lewandowski, A. James, and Z. Usman, "CPS data streams analytics based on machine learning for cloud and fog computing: A survey," *Future Gener. Comput. Syst.*, vol. 90, pp. 435–450, Jan. 2019.

[17] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 67, 2016.
[18] M. Ibtihal and N. Hassan, "Homomorphic encryption as a service for outsourced images in mobile cloud computing environment," in *Cryptography: Breakthroughs in Research and Practice*. Hershey, PA, USA: IGI Global, 2020, pp. 316–330.
[19] X. Sun, P. Zhang, J. K. Liu, J. Yu, and W. Xie, "Private machine learning classification based on fully homomorphic encryption," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 352–364, Jun. 2020.
[20] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, 1999, pp. 223–238.
[21] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[22] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 209–232, 2002.
[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, p. 34, 2009.
[24] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. 33rd Annu. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, 2013, pp. 75–92.

JAEHYOUNG PARK (Graduate Student Member, IEEE) received the B.S. degree from the Information and Computer Engineering, Ajou University, Suwon, South Korea, in 2015, and the M.S. degree from the School of Electrical Engineering and Computer Science (EECS), Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea, in 2017, where he is currently pursuing the Ph.D. degree. His research interest includes data security technology using homomorphic encryption for privacy-preserving artificial intelligence.

DONG SEONG KIM (Senior Member, IEEE) received the Ph.D. degree from Korea Aerospace University, in February 2008. He has been an Associate Professor of cybersecurity with the University of Queensland, Australia, since January 2019. He was a Senior Lecturer/Lecturer of cybersecurity with the University of Canterbury, from August 2011 to December 2018. He was a Visiting Scholar with the University of Maryland, College Park, in 2007. From June 2008 to July 2011, he was a Postdoctoral Researcher with Duke University. His research interests are in automated cybersecurity modeling and analysis for the Internet of Things, cloud computing, and moving target defense. He was the General Co-Chair of ACISP2019 and the General Chair of IEEE PRDC 2017. He served as a Program Co-Chair of IEEE TrustCom2019, IEEE ICIOT2019, ATIS2017, GraMsec2015, and IEEE DASC2015, and a Program Committee Member of international conferences, including IFIP/IEEE DSN, ISSRE, SRDS, and ICC CISS.

HYUK LIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, South Korea, in 1996, 1998, and 2003, respectively. From 2003 to 2006, he was a Postdoctoral Research Associate with the Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL, USA. He is currently a Full Professor with the AI Graduate School and the School of Electrical Engineering and Computer Science (EECS), Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea. His research interests include network protocol design, optimization, and the performance evaluation of computer and communication networking systems.