ORIGINAL RESEARCH

A proposal of privacy preserving reinforcement learning for secure multiparty computation

Hirofumi Miyajima[1], Noritaka Shigei[2], Syunki Makino[2], Hiromi Miyajima*[2], Yohtaro Miyanishi[3], Shinji Kitagami[4], Norio Shiratori[5]

[1] *Graduate School of Biomedical Science, Nagasaki University, Nagasaki, Japan*
[2] *Graduate School of Science and Engineering, Kagoshima University, Kagoshima, Japan*
[3] *Information Systems Engineering and Management, Tokyo, Japan*
[4] *Graduate School of Global Information and Telecommunication Studies(GITS), Waseda University, Tokyo, Japan*
[5] *Research Development Initiative, Chuo unversity, Tokyo, Japan*

Received: March 28, 2017 **Accepted:** May 10, 2017 **Online Published:** May 23, 2017
DOI: 10.5430/air.v6n2p57 **URL:** https://doi.org/10.5430/air.v6n2p57

ABSTRACT

Many studies have been done with the security of cloud computing. Though data encryption is a typical approach, high computing complexity for encryption and decryption of data is needed. Therefore, safe system for distributed processing with secure data attracts attention, and a lot of studies have been done. Secure multiparty computation (SMC) is one of these methods. Specifically, two learning methods for machine learning (ML) with SMC are known. One is to divide learning data into several subsets and perform learning. The other is to divide each item of learning data and perform learning. So far, most of works for ML with SMC are ones with supervised and unsupervised learning such as BP and K-means methods. It seems that there does not exist any studies for reinforcement learning (RL) with SMC. This paper proposes learning methods with SMC for Q-learning which is one of typical methods for RL. The effectiveness of proposed methods is shown by numerical simulation for the maze problem.

Key Words: Cloud computing, Secure multiparty computation, Reinforcement learning, Maze problem

1. INTRODUCTION

With increasing interest in artificial intelligence (AI), many studies have been made with machine learning (ML). With ML, the supervised models such as multi-layer perceptron (MLP) for neural network (NN), the unsupervised models such as K-means and self organizing map (SOM) and the reinforcement learning (RL) without using learning data are well known. In response to the increase of data or complex problems for ML, the use of cloud computing systems is spreading. The spreading of cloud computing enables big data analysis, which analyzes enormous information accumulated by the client and creates a market value of data.[1–4] On the other hand, the client of cloud computing is concerned about abuse or leak of information. For this purpose, privacy preserving data processing can be achieved in various ways by use of randomization techniques, cryptographic algorithms, anonymization methods, *etc.*[2,5,6] Specifically, data encryption seems to be effective. However, data encryption system requires both encryption and decryption for requests of client or user, so its applications are limited. Therefore, safe system for distributed processing with secure data attracts attention, and a lot of studies have been done. SMC is one of these methods.[7–10] Most of the works in SMC are developed on applying the model of SMC on different data

* **Correspondence:** Hiromi Miyajima; Email: miya@eee.kagoshima-u-ac.jp; Address: Graduate School of Science and Engineering, Kagoshima University, 1-21-40 Korimoto, Kagoshima 890-0065, Japan.

distributions such as vertically, horizontally and arbitrarily partitioned data.[11–15] They are the methods that each server performs its processing for the subset of data. As shown later, the methods need a large number of servers in order to keep privacy and security. Therefore, SMC for shared data has been considered. In Miyajima, *et al.*,[17,22] learning methods for SMC of BP, fuzzy system and VQ methods have been proposed and the validity of them has been proved. On the other hand, though there are some studies on privacy preserving with RL,[18–21] they are ones of cryptogram algorithms. It seems that there do not exist any studies with SMC. The difference between BP and RL methods for SMC is that 1) learning data are not used explicitly for RL, and 2) BP estimates the actual input/output characteristic but RL estimates the desirable probabilistic behavior.

In this paper, RL methods for SMC will be proposed. Further, the performance of the proposed method is shown in numerical simulations. In Section 2, cloud computing system, related works on SMC and a secure data sharing mechanism used in this paper are explained. Further, RL method is introduced. A maze problem helps to understand the proposed algorithm for RL. In Section 3, RL methods for SMC are proposed. In Section 4, numerical simulations for a maze problem are performed to show the performance of proposed methods.

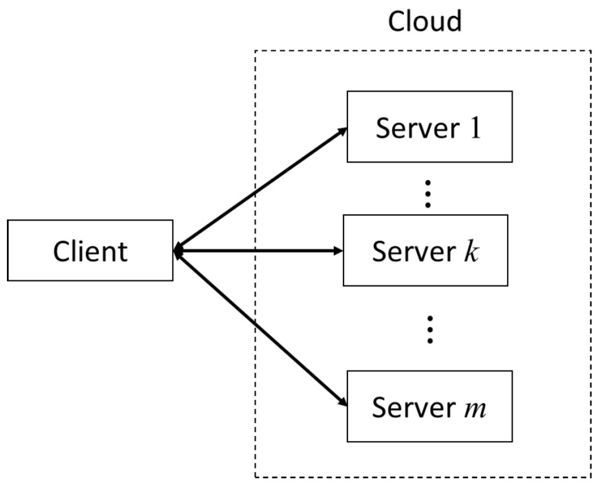

Figure 1. Cloud system

2. Preliminary

2.1 Cloud system and related works with SMC

The system shown in Figure 1 is used in this paper. It is composed of a client and m servers. Each data is divided into m pieces of numbers and is sent to each server. Each server performs own computation and sends computation results to the client. The client can get the result using them. If one processing does not obtain the result, then the plural processing is repeated.

Table 1. Concept of horizontally and vertically partitioned methods composed of one client and two servers

	ID	Subject A a	Subject B b
Server 1	1	34	46
	2	29	11
	3	14	27
Server 2	4	48	22
	5	42	45
	6	17	34
	Average	30.7	30.8

Horizontally partitioned method

Vertically partitioned method

Let us explain conventional works with them. Three types of methods for partitioning data to be securely shared are well known.[3,4,23,24] They are horizontal, vertical and arbitrary partitioning methods. In the following, the horizontal method is only explained by using a data example of students' marks shown in Table 1. See Miyajima, *et al.*[17] about vertical and arbitrary partitioning methods. In Table 1, a and b are original data (marks) and ID is the identifier of students. The assumed task is to calculate the average of the data. The horizontal partitioning method assigns the horizontally partitioned data to servers as follows:

Server 1: data for ID = 1, 2, 3.

Server 2: data for ID = 4, 5, 6.

In the method, Server 1 computes two averages for subjects A and B as (34 + 29 + 14)/3 and (46 + 11 + 27)/3, respectively. Likewise, Server 2 computes two averages for subjects A and B as (48 + 42 + 17)/3 and (22 + 45 + 34)/3, respectively. Servers 1 and 2 send the calculated averages to the client and the client obtains the averages of subjects A and B as 30.7 and 30.8, respectively. Since each server cannot know half of the dataset, the method preserves privacy (see Table 1). The second method, the vertical partitioning method can calculate two averages using data. The third method, the arbitrary partitioning method, splits horizontally and vertically the dataset into multiple parts, and the method assigns the split parts to the servers. For any of the above mentioned methods, if the number of servers is fewer, that is, the size of a partitioned data is larger, a server may more easily guess the feature of all the data from its own subset of data. Therefore, the methods need a large number of servers in order to keep privacy and security. On the other hand, the method explained in the next subsection shares each item of data and seems to keep them by a small number of servers.

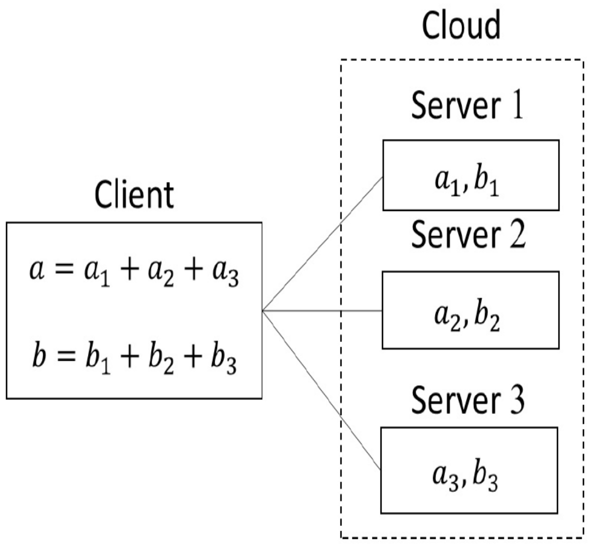

Figure 2. An example of secure shared data for $m = 3$

2.2 Secure shared data for SMC

Let us explain secure shared data for the proposed method using in Figure 2.[15,16] Let a and b be two marks and $m = 3$. Assume that the addition form is used for dividing each item. First, two marks a and b are shared into three real numbers as $a = a_1 + a_2 + a_3$ and $b = b_1 + b_2 + b_3$ in addition form. Then the following results hold:

1) $a + b = (a_1 + b_1) + (a_2 + b_2) + (a_3 + b_3)$

2) $a - b = (a_1 - b_1) + (a_2 - b_2) + (a_3 - b_3)$

It means that addition and subtraction are divided into three operations.[15,16]

Let us show a calculation example for shared data as follows (see Table 2):

$a = a_1 + a_2 + a_3$: $a_1 = a(r_1/10)$, $a_2 = a(r_2/10)$ and $a_3 = a(1 - r_1/10 - r_2/10)$, and $b = b_1 + b_2 + b_3$: $b_1 = b(r_1/10)$, $b_2 = b(r_2/10)$ and $b_3 = b(1 - r_1/10 - r_2/10)$, where r_1 and r_2 are real random numbers for $-9 \leq r_1 \leq 9$ and $-9 \leq r_2 \leq 9$ $(r_1 \neq 1, r_2 \neq 1)$, respectively. For example, a_1 and a_2 for ID = 1 are computed as $a_1 = 9 \times (7/10) = 6.3$, $a_2 = 9 \times (-2/10) = -1.8$ and $a_3 = 9 \times (1 - 7/10 + 2/10) = 4.5$, respectively. Note that Server 1, Server 2 and Server 3 have all the data in column-wise of a_1 and b_1, a_2 and b_2, and a_3 and b_3 for each ID as shown in Table 2, respectively.

Table 2. Data for Server 1, Server 2 and Server 3

ID	subject A	subject B	Addition form							
	a	b	r1	r2	a			b		
					a1	a2	a3	b1	b2	b3
1	9	2	7	-2	6.3	-1.8	4.5	1.4	9.2	-8.6
2	38	43	-9	6	-34.2	22.8	49.4	-38.7	37.4	44.3
3	13	77	-7	-5	-9.1	-6.5	28.6	-53.9	13.5	117.4
Sum	60	122			-37	14.5	82.5	-91.2	60.1	153.1
Average	20	40.7			-12.3	4.8	27.5	-30.4	20	51

Let us explain how to compute the sum and the average for subject A using data a. Server 1, Server 2 and Server 3 compute each sum of a_1, a_2 and a_3, respectively. In this case, each sum in column-wise for a_1, a_2 and a_3 is -37, 14.5 and 82.5, respectively. As a result, the total sum 60 is obtained from -37 + 14.5 + 82.5. Likewise, the average 20 is obtained from -12.3 + 4.8 + 27.5.

Remark that each data for server is randomized and the method does not need to use encrypted data.

In the next section, a learning method using secure shared data in addition form is proposed.

2.3 Q-learning method

Q-learning is a reinforcement learning technique for environment-identity type.[23,24] It can be used to find an optimal action-selection policy for a given Markov Decision Process (MDP). In solving problems using Q-learning, it is determined how the agent selects an action at any state. It is performed by learning an action-value (Q-value) function that gives the expected utility of taking the action for the current state. A Q-value function is defined as a function $Q : S \times A \rightarrow R$, where S, A and R are sets of states, actions and real numbers, respectively. First, let all Q-values be 0. Then each action for a state is selected randomly. If a solution for the problem is not obtained, learning is iterated. If a solution is obtained, Q-values are updated based on the updated formula. By iterating these processes, it is known that Q-value function is updated and converges.[24] In this case, there are some methods to select action randomly. In this paper, Boltzmann selection is used as shown later.

In the first part of learning, the action for the state is selected randomly and the action becomes decidable as learning steps proceed.

Further, Q-value function is updated as

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Delta \tag{1}$$

$$\Delta = r + \gamma \max_{a' \in A} Q(s', a') - Q(s,a) \tag{2}$$

where r, α and γ are the reward, learning constant and discount rate, respectively. The state s' is the next state selected for the state s and the action a. The term $\max_{a' \in A} Q(s', a')$ means the Q-value $Q(s', a'_0)$ for an action a'_0 taking the max value of $Q(s', a')$. The conventional algorithm for Q-learning is shown as follows:[23]

Q-learning algorithm

$s_{i_{l(t)}}$: the current state at learning step t; s_0: the initial state; s_f: the goal (target) state; $Q(s,a)$: Q-value for the state s and the action a; $t_{\max}$: The maximum number of learning time; $T_{\max}$ and $T_{\min}$: Constants for Boltzmann selection; S: The set of states; A: The set of actions.

Step 1: Let r, α and γ be reward, learning constant and discount rate. Let $Q(s,a) = 0$ for $s \in S$ and $a \in A$. Let $t = 0$.

Step 2: Let $i_{l(0)} = 0$, that is $s_{i_{l(t)}} = s_0$.

Step 3: The action a at the state $s_{i_{l(t)}}$ is selected based on the following $B(s_{i_{l(t)}}, a)$ (called Boltzmann selection) as follows:

$$B(s_{i_{l(t)}}, a) = \frac{\exp(Q(s_{i_{l(t)}}, a)/T)}{\sum_{b \in A}(Q(s_{i_{l(t)}}, a)/T)} \tag{3}$$

$$T = T_{\max} \times \left(\frac{T_{\min}}{T_{\max}}\right)^{\frac{t}{T_{\max}}} \tag{4}$$

Let a^* be the selected action based on Equations (3) and (4).

Step 4: Let $s_{i_{l(t)+1}}$ be the state after performing the action a^* at the state $s_{i_{l(t)}}$. If $s_{i_{l(t)+1}}$ is permissible(movable), then go to Step 5 else go to Step 3 to select another action.

Step 5: The Q-value $Q(s_{i_{l(t)}}, a^*)$ is updated as follows:

$$Q(s_{i_{l(t)}}, a^*) \leftarrow Q(s_{i_{l(t)}}, a^*) + \alpha\Delta \tag{5}$$

$$\Delta = r + \gamma \max_{a \in A} Q(s_{i_{l(t)+1}}, a) - Q(s_{i_{l(t)}}, a^*) \tag{6}$$

Step 6: If $s_{i_{l(t)+1}} = s_f$, then go to Step 7 else go to Step 3 with $l(t) \leftarrow l(t) + 1$.

Step 7: If $t = t_{\max}$, then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$.

To explain the algorithm easily, let us show an example of Q-learning for a problem to find the shortest path from the start state (0) to the goal state (15) as Figure 3.

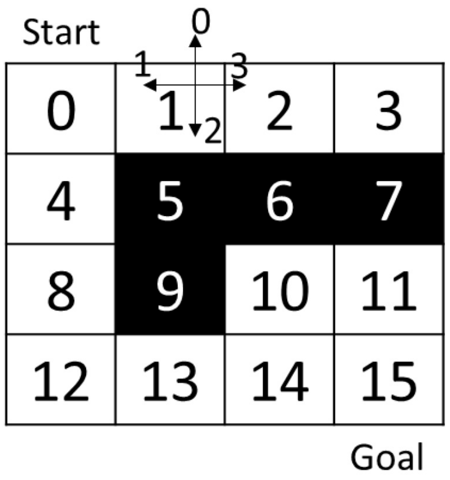

Figure 3. A maze problem for Example 1, where each number means the state of position

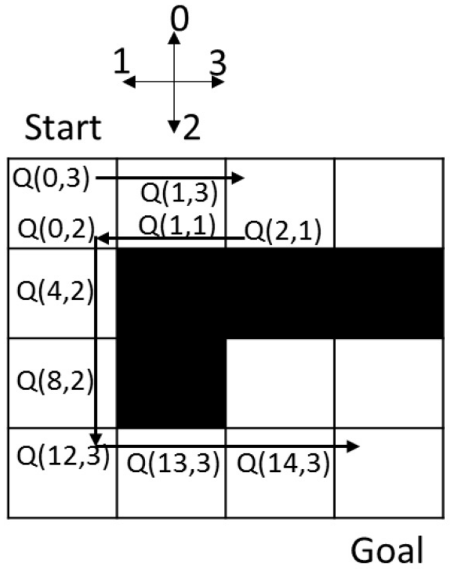

Figure 4. Example of the first route selected by Q-learning

Example 1

Let $S = \{0, 1, \cdots, 15\}$ and $A = \{0, 1, 2, 3\}$ for a maze problem as Figure 3, where black and outer areas mean prohibited ones and the agent does not go. Each of the set S and the set A means state number from 0 to 15 and action numbers to up (0), left (1), down (2) and right (3) direction, respectively. In the first learning process, assume that a route as Figure 4 is selected based on Equations (3) and (4). Then each of Q-values is updated based on Equations (5) and (6) as follows:

$$Q(0,3) \leftarrow Q(0,3) + 0.1 \times (0 + 0.9 \times 0 - 0) = 0 \quad (7)$$

$$\vdots$$

$$Q(13,3) \leftarrow Q(13,3) + 0.1 \times (0 + 0.9 \times 0 - 0) = 0 \quad (8)$$

$$Q(14,3) \leftarrow Q(14,3) + 0.1 \times (10 + 0.9 \times 0 - 0) = 1 \quad (9)$$

where $s_0 = s_{i_{l(0)}} = 0, s_{i_{l(0)+1}} = 1, s_{i_{l(0)+2}} = 2, s_{i_{l(0)+3}} = 1, \cdots$.

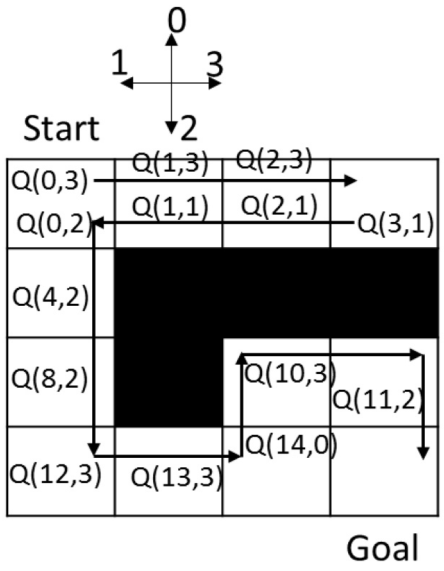

Figure 5. Example of the second route selected by Q-learning

In the second learning process (second episode), assume that a route as Figure 5 is selected based on Equations (3) and (4). Then each of Q-values is updated based on Equations (5) and (6) as follows:

$$Q(0,3) \leftarrow Q(0,3) + 0.1 \times (0 + 0.9 \times 0 - 0) = 0 \quad (10)$$

$$\vdots$$

$$Q(12,3) \leftarrow Q(12,3) + 0.1 \times (0 + 0.9 \times 0 - 0) = 0 \quad (11)$$

$$Q(13,3) \leftarrow Q(13,3) + 0.1 \times (0 + 0.9 \times 1.0 - 0) = 0.09 \quad (12)$$

$$Q(14,0) \leftarrow Q(14,0) + 0.1 \times (0 + 0.9 \times 0 - 0) = 0 \quad (13)$$

$$Q(10,3) \leftarrow Q(10,3) + 0.1 \times (0 + 0.9 \times 0 - 0) = 0 \quad (14)$$

$$Q(11,2) \leftarrow Q(11,2) + 0.1 \times (10 + 0.9 \times 0 - 0) = 0 \quad (15)$$

Finally, the shortest route is obtained as follows:

$Q(0,2) \rightarrow Q(4,2) \rightarrow \cdots \rightarrow Q(14,3) \rightarrow$ Goal.

3. Q-LEARNING FOR SECURE MULTIPARTY COMPUTATION

Let us consider a system shown in Figure 1. In RL on cloud system, Q-values are shared to each server in addition form. Each server updates shared Q-values and sends the result to the client. The client can get new Q-values by adding the results of m servers. The process is iterated until the evaluating value for the problem satisfies the final condition. The problem is how Q-values on the client are updated using Q-values shared on each server. The shared representation of Q-values is given as follows:

$$Q(s,a) = \sum_{k=1}^{m} Q_k(s,a) \text{ for } s \in S \text{ and } a \in A \qquad (16)$$

In the following, two types of learning methods were proposed.

Problem Statement: For SMC, the difference between the conventional methods and the proposed one was described in Section 2. Assuming that the shared data is learning data for ML, the conventional ones divide the learning data into subsets. On the other hand, the proposed one divides each item of the learning data into plural pieces and processes them. From the point of view, SMC algorithms for supervised learning such as BP method and unsupervised learning like K-means method were proposed.[17,22] So how is the algorithm of RL for SMC? In this case, as there is no data for learning, optimal behavior is found by repeating trial and error. Since there is no data for learning, it seems that the conventional method using subset of learning data as shown in Section 2 does not exist. On the other hand, several results on privacy preserving have been obtained, but these are all studies on theory and application using encryption and homomorphic mapping.[18–21] Our method attempts to realize SMC by simple secret computation processing which does not use such complicated cryptographic processing and homomorphic mapping. The aim is to reduce the computational complexity of client while keeping the secret of data (information of Q-values in this case). The proposed method here is a natural development to the RL of supervised and unsupervised learning algorithms (protocols) for the SMC in the previous paper.

In the following, the proposed method will be explained through solving the maze problem by Q-learning which is one of typical algorithms of RL.

Table 3. M1 method of Q-learning for SMC

	Client	k-th Server ($1 \le k \le m$)
Initialization	Let $t = 0$. Let $s_{i_{l(t)}}$ be the current state.	The numbers r, α and γ are given. Let $Q(s, a) = 0$ for $s \in S$ and $a \in A$.
Step 1	Let $i_{l(0)} = 0$.	
Step 2	Send the state $s_{i_{l(t)}}$ to each server.	
Step 3		Send all Q-values $Q_k(s, a)$'s for $a \in A$ to client.
Step 4	Calculate $Q\left(s_{i_{l(t)}}, a\right) = \sum_{k=1}^{m} Q_k(s_{i_{l(t)}}, a)$ for $a \in A$. Select the action a^* for the state $s_{i_{l(t)}}$ based on Boltzmann selection of Equations (3) and (4). Let $s_{i_{l(t)+1}}$ be the next state for the action a^* and the state $s_{i_{l(t)}}$.	
Step 5	If the state $s_{i_{l(t)+1}}$ is permissible (movable), then send the state $s_{i_{l(t)+1}}$ to each server else go to Step 4.	
Step 6		Calculate $\Delta_b^k = \gamma Q_k\left(s_{i_{l(t)+1}}, b\right) - Q_k(s_{i_{l(t)}}, a^*)$ for all Q values of $Q_k\left(s_{i_{l(t)+1}}, b\right)$ for $b \in A$ and send them to client.
Step 7	Calculate $\Delta_b = \sum_{k=1}^{m} \Delta_b^k$ and $\Delta^* = \max_{b \in A} \Delta_b$. Let $\Delta^* = \sum_{k=1}^{m} \Delta_k^*$. Send them to each server.	
Step 8		The Q-value $Q_k(s_{i_{l(t)}}, a^*)$ is updated as follows: $Q_k(s_{i_{l(t)}}, a^*) \leftarrow Q_k(s_{i_{l(t)}}, a^*) + \alpha(r + \Delta_k^*)$.
Step 9	Let $i_{l(t)} \leftarrow i_{l(t)+1}$. If $s_{i_{l(t)}} = s_f$, then go to Step 10 else go to Step 2.	
Step 10	If $t = t_{max}$, then the algorithm terminates else go to Step 1 with $t \leftarrow t + 1$.	

3.1 Reinforcement learning for SMC

In this section, two RL methods are proposed. The first algorithm is that updating of Q-values is performed in each server as shown in Table 3. The initial values of client and servers are set in Step 1 to Step 3. In Step 4, the current Q-value $Q(s_{i_{l(t)}}, a)$ is computed from Q-values of each server and the action a^* is selected based on Boltzmann selection. Further, the next state $s_{i_{l(t)+1}}$ is determined from $s_{l(t)}$ and a. In Step 5, if $s_{i_{l(t)+1}}$ is permissible, the information of $s_{i_{l(t)+1}}$ is sent to each server else new action and state is selected. In Step 6, each server computes the difference between the future and the current Q-values for all $a \in A$ of $Q_k(s_{i_{l(t)+1}}, a)$ and sends them to the client. In Step 7, the client computes all Q-values for $a \in A$ of $Q_k(s_{i_{l(t)+1}})$ and determines the max value of them (called Δ^*). The value Δ^* is divided as $\Delta^* = \sum_{k=1}^{m} \Delta_k^*$, where Δ_k^* is a real random number. In Step 8, Q-value, $Q(s_{i_{l(t)+1}}, a^*)$, is updated using Δ_k^*. In Steps 9 and 10, it is checked if the final condition is satisfied. If the final condition is not satisfied, the next episode is iterated.

The second RL method is that updating of Q-value is performed in one server selected randomly as shown in Table 4. The difference between first and second RL methods is that the number of servers is single or all. Then, how is one server only updated in the updating step of Q-value. In Step 1, the number p is selected randomly. It means that the p-th server is updated. In Steps 2 to 7, the same processes are defined as Table 3. In Step 8, Q-value $Q_p(s_{i_{l(t)+1}}, a^*)$ of the p-th server is only updated and Q-values of other servers are not updated. In Step 10, the number p is updated and Q-value of another server in the next episode is updated.

In the following, first and second RL methods are called M1 and M2 ones, respectively.

3.2 Reinforcement learning with dummy updating

The fundamental idea of RL with dummy updating is that all Q-values are updated at each step. Therefore, it seems that each server cannot know which Q-values are important or not.

Table 4. M2 method of Q-learning for SMC

	Client	**k-th Server ($1 \le k \le m$)**
Initialization	Let $t = 0$ and $j = 1$. Let $s_{i_{l(t)}}$ be the current state.	The numbers r, α and γ are given. Let $Q(s,\ a) = 0$ for $s \in S$ and $a \in A$.
Step 1	Let $i_{l(0)} = 0$ and $p = e_j$ is selected randomly.	
Step 2	Send the state $s_{i_{l(t)}}$ to each server.	
Step 3		Send all Q-values $Q_k(s,\ a)'s$ for $a \in A$ to client.
Step 4	Calculate $Q\left(s_{i_{l(t)}},\ a\right) = \sum_{k=1}^{m} Q_k(s_{i_{l(t)}},\ a)$ for $a \in A$. Select the action a^* for the state $s_{i_{l(t)}}$ based on Boltzmann selection of Equations (3) and (4). Let $s_{i_{l(t)+1}}$ be the next state for the action a^* and the state $s_{i_{l(t)}}$.	
Step 5	If the state $s_{i_{l(t)+1}}$ is permissible (movable), then send the state $s_{i_{l(t)+1}}$to each server else go to Step 4.	
Step 6		Calculate $\Delta_b^k = \gamma Q_k\left(s_{i_{l(t)+1}},\ b\right) - Q_k(s_{i_{l(t)}},\ a^*)$ for all Q values of $Q_k\left(s_{i_{l(t)+1}},\ b\right)$ for $b \in A$ and send them to client.
Step 7	Calculate $\Delta_b = \sum_{k=1}^{m} \Delta_b^k$ and $\Delta^* = \max_{b \in A} \Delta_b$. Send them to each server.	
Step 8		The Q-value $Q_k(s_{i_{l(t)}},\ a^*)$ is updated as follows: $Q_k(s_{i_{l(t)}},\ a^*) \leftarrow Q_k(s_{i_{l(t)}},\ a^*) + \alpha(r + \Delta^*)$ for $k = p$ or $Q_k(s_{i_{l(t)}},\ a^*) \leftarrow Q_k(s_{i_{l(t)}},\ a^*)$ for $k \neq p$, where $r = 0$ for $s_{i_{l(t)+1}} \neq s_f$.
Step 9	Let $i_{l(t)} \leftarrow i_{l(t)+1}$. If $s_{i_{l(t)}} = s_f$, then go to Step 10 else go to Step 2.	
Step 10	If $t = t_{max}$, then the algorithm terminates else go to Step 1 with $t \leftarrow t + 1$ and $j \leftarrow j + 1$.	

Table 5. M1 method with dummy updating of Q-learning for SMC

	Client	**k-th Server ($1 \le k \le m$)**
Initialization	Let $t = 0$. Let $s_{i_{l(t)}}$ be the current state.	The numbers r, α and γ are given. Let $Q(s,\ a) = 0$ for $s \in S$ and $a \in A$.
Step 1	Let $i_{l(0)} = 0$.	
Step 2	Send the state $s_{i_{l(t)}}$ to each server.	
Step 3		Send all Q-values $Q_k(s,\ a)'s$ for $a \in A$ to client.
Step 4	Calculate $Q\left(s_{i_{l(t)}},\ a\right) = \sum_{k=1}^{m} Q_k(s_{i_{l(t)}}, a)$ for $a \in A$. Select the action a^* for the state $s_{i_{l(t)}}$ based on Boltzmann selection of Equations (3) and (4). Let $s_{i_{l(t)+1}}$ be the next state for the action a^* and the state $s_{i_{l(t)}}$.	
Step 5	If the state $s_{i_{l(t)+1}}$ is permissible (movable), then send the state $s_{i_{l(t)+1}}$to each server else go to Step 4.	
Step 6		Calculate $\Delta_b^k = \gamma Q_k\left(s_{i_{l(t)+1}}, b\right) - Q_k(s_{i_{l(t)}}, a^*)$ for all Q values of $Q_k\left(s_{i_{l(t)+1}}, b\right)$ for $b \in A$ and send them to client.
Step 7	Calculate $\Delta_b = \sum_{k=1}^{m} \Delta_b^k$ and $\Delta^* = \max_{b \in A} \Delta_b$ and send $(r + \Delta^*)$ to all server, where $r = 0$ for $s_{i_{l(t)+1}} \neq s_f$. $D_k'(s,\ a)(\lvert D_k'(s,\ a)\rvert \le 1)$ is selected randomly and calculate $D_k\left(s_{i_{l(t)}},\ a^*\right) = \frac{D_k'\left(s_{i_{l(t)}},\ a^*\right)}{\sum_{k=1}^{m} D_k'\left(s_{i_{l(t)}},\ a^*\right)}$ or $D_k(s,\ a) = \frac{D_k'(s,\ a)}{\sum_{k=1}^{m} D_k'(s,\ a)} - \frac{1}{m}$ $(s \neq s_{i_{l(t)}}$ or $a \neq a^*)$. Let $\Delta^* = \sum_{k=1}^{m} \Delta_k^*$. Send them to each server.	
Step 8		The Q-value $Q_k(s_{i_{l(t)}},\ a^*)$ is updated as follows: $Q_k(s_{i_{l(t)}},\ a^*) \leftarrow Q_k(s_{i_{l(t)}},\ a^*) + \alpha D_k(s,\ a)(r + \Delta_k^*)$.
Step 9	Let $i_{l(t)} \leftarrow i_{l(t)+1}$. If $s_{i_{l(t)}} = s_f$, then go to Step 10 else go to Step 2.	
Step 10	If $t = t_{max}$, then the algorithm terminates else go to Step 1 with $t \leftarrow t + 1$.	

Let us explain the principle of improved M1 method. The numbers $D'_k(s,a)$ for $s \in S$ and $a \in A$ are randomly selected such that $|D'_k(s,a)| \leq 1$ and $D_k(s,a)$ for $s \in S$ and $a \in A$ are calculated as follows:

$$D_k(s,a) = \begin{cases} \frac{D'_k(s,a)}{\sum_{k=1}^{m} D'_k(s,a)} & \text{if } s = s^* \text{ and } a = a^* \ (A) \\ \frac{D'_k(s,a)}{\sum_{k=1}^{m} D'_k(s,a)} - \frac{1}{m} & \text{if } s \neq s^* \ or \ a \neq a^* \ (B) \end{cases} \tag{17}$$

where $s = s^*$ and $a = a^*$ are designated state and action. Remark that cases of A and B are $\sum_{k=1}^{m} D_k(s,a) = 1$ and 0 for $s \in S$ and $a \in A$, respectively. That is, the Q-value for designated state and action is only updated and each server cannot know designated state and action.

Let us explain Table 5. The processes from Step 1 to Step 6 are the same as Table 3. In Step 7, dummy processing for designated state $s_{i_{l(t)}}$ and action a^* is performed and $D_k(s,a)$ for $s \in S$ and $a \in A$ are defined. In Step 8, all Q-values for $s \in S$, $a \in A$ and $k \in Z_m$ are updated using $D_k(s,a)$. In Steps 9 and 10 it is checked if the algorithm terminates.

Table 6. M2 method with dummy updating of Q-learning for SMC

	Client	***k*-th Server (1 ≤ k ≤ m)**
Initialization	Let t = 0 and j = 1. Let $s_{i_{l(t)}}$ be the current state.	The numbers r, α and γ are given. Let $Q(s, a) = 0$ for $s \in S$ and $a \in A$.
Step 1	Let $i_{l(0)} = 0$ and $p = e_j$ is selected randomly.	
Step 2	Send the state $s_{i_{l(t)}}$ to each server.	
Step 3		Send all Q-values $Q_k(s, a)'s$ for $a \in A$ to client.
Step 4	Calculate $Q\left(s_{i_{l(t)}}, a\right) = \sum_{k=1}^{m} Q_k(s_{i_{l(t)}}, a)$ for $a \in A$. Select the action a^* for the state $s_{i_{l(t)}}$ based on Boltzmann selection of Equations (3) and (4). Let $s_{i_{l(t)+1}}$ be the next state for the action a^* and the state $s_{i_{l(t)}}$.	
Step 5	If the state $s_{i_{l(t)+1}}$ is permissible (movable), then send the state $s_{i_{l(t)+1}}$ to each server else go to Step 4.	
Step 6		Calculate $\Delta_b^k = \gamma Q_k\left(s_{i_{l(t)+1}}, b\right) - Q_k(s_{i_{l(t)}}, a^*)$ for all Q values of $Q_k\left(s_{i_{l(t)+1}}, b\right)$ for $b \in A$ and send them to client.
Step 7	Calculate $\Delta_b = \sum_{k=1}^{m} \Delta_b^k$ and $\Delta^* = \max_{b \in A} \Delta_b$. $D'_k(s, a)(\lvert D'_k(s, a)\rvert \leq 1)$ is selected randomly and calculate $D_k(s, a) = 1\ (k = p,\ s = s_{i_{l(t)}},$ $a = a^*)$ or $D_k(s, a) = \frac{D'_k(s, a)}{\sum_{k=1}^{m} D'_k(s, a)} - \frac{1}{m-1}\ (k \neq p,\ s = s_{i_{l(t)}},\ a = a^*)$ or $D_k(s, a) = \frac{D'_k(s, a)}{\sum_{k=1}^{m} D'_k(s, a)} - \frac{1}{m}$.	
Step 8		The Q-value $Q_k(s, a)$ is updated as follows: $Q_k(s, a) \leftarrow Q_k(s, a) + \alpha D_k(s, a)(r + \Delta^*)$.
Step 9	Let $i_{l(t)} \leftarrow i_{l(t)+1}$. If $s_{i_{l(t)}} = s_f$, then go to Step 10 else go to Step 2.	
Step 10	If $t = t_{max}$, then the algorithm terminates else go to Step 1 with t ← t + 1 and j ← j + 1.	

Let us explain the improved M2 method in Table 6. The numbers $D_k(s,a)$ are also defined as the case of Table 5. The numbers $D'_k(s,a)$ are randomly selected such that $|D'_k(s,a)| \leq 1$ and $D_k(s,a)$ are defined as follows:

$$D_k(s,a) = \begin{cases} 1 & \text{if } s = s^*, a = a^* \text{ and } k = p \ (A) \\ \frac{D'_k(s,a)}{\sum_{k=1}^{m} D'_k(s,a)} - \frac{1}{m-1} & \text{if } s = s^*, a = a^* \text{ and } k \neq p \ (B) \\ \frac{D'_k(s,a)}{\sum_{k=1}^{m} D'_k(s,a)} - \frac{1}{m} & \text{others} \qquad (C) \end{cases} \tag{18}$$

where s^*, a^* and p are designated state, action and the number. Remark that the case A, B and C are $\sum_{k=1}^{m} D_k(s,a) = 1, 0$ and 0, respectively. The processes from Steps 1 to 6 are same as Table 4. In Step 7, dummy updating for designated state $s_{l(t)}$ and action a^* is performed and $D_k(s,a)$ for $s \in S$ and $a \in A$ is defined. In Step 8, all Q-values for $s \in S$, $a \in A$ and $k \in Z_m$ are updated using $D_k(s,a)$. In Steps 9 and 10, it is checked if the algorithm terminates.

In the following, improved M1 and M2 methods are called M1 and M2 methods with dummy updating, respectively.

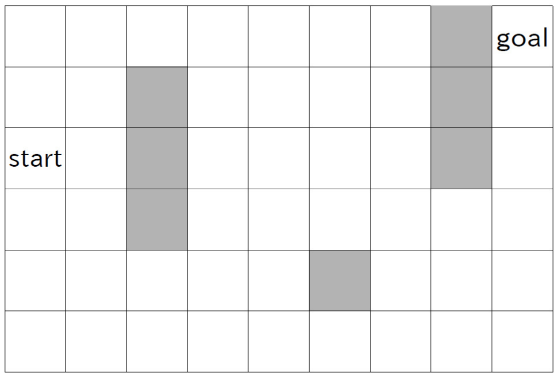

Figure 6. The figure for Sutton's maze problem

4. Numerical simulations for the proposed algorithm

The problem is to find the shortest path of Figure 6 known as Sutton's maze problem by RL.[24] In Figure 6, the agent cannot go to black and outer areas and the agent iterates to move from the start to goal area based on each algorithm. The simulation conditions are as follows:

(1) The agent can move to four direction at each area (state) excepting black areas.
(2) Let $T_{\max} = 5.0$ and $T_{\min} = 0.03$ for Boltzmann selection.
(3) If the agent selects to move to wall or outer area, the agent ignores the selection and reselects a new movement. It is not counted in the number of trials.
(4) If the agent arrives at the goal area in the maximum number of learning time, the agent starts the new trial.
(5) Let $t_{\max} = 10,000$, $r = 10$, $\alpha = 0.5$, $\gamma = 0.92$, and $m = 3$ and 10. Twenty trials for learning and test are performed for each algorithm.
(6) In the test simulation, experiments are carried out with all places except for black areas as the starting points. The result is evaluated as the number of success trials and the sum of movement distance from each starting point.

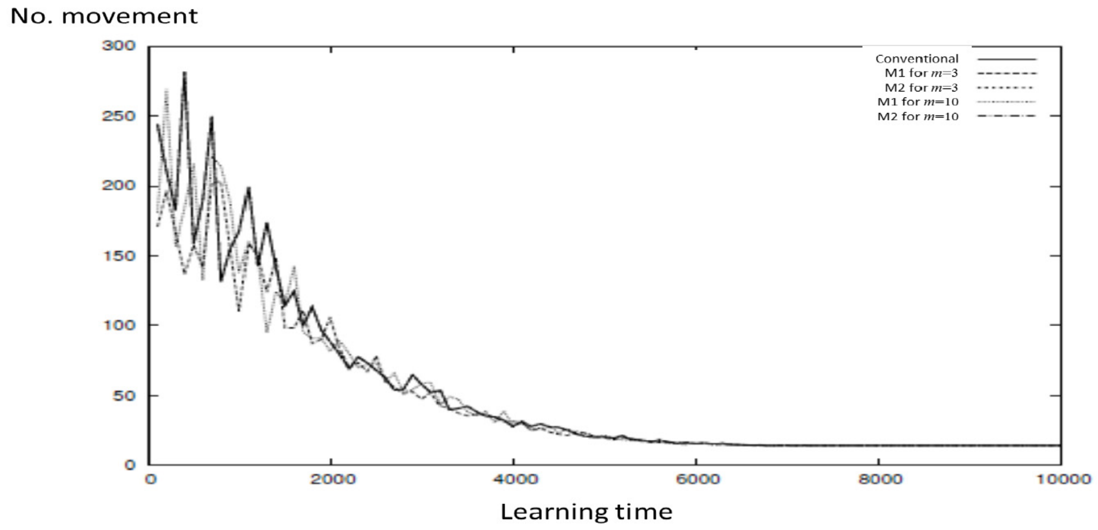

Figure 7. The result of efficiency of Q-learning for SMC

Figure 7 shows the efficiency graph. The graph represents the number of times of movement (abbreviated as No. movement) to the learning time. In Figure 7, the conventional, M1 for m = 3 and 10, and, M2 for m = 3 and 10 means the conventional algorithm with no server, proposed algorithms for M1 with m = 3 and 10, and for M2 with m = 3 and 10, respectively. All the results are almost the same as the conventional case. Table 7 shows the result of the test simulation. In Table 7, # Suc. and M.D. mean the number of success trials for twenty trials and the average of movement distance for success trials. Further, the result on each server means one of the case where the same trials are performed using only Q-values of each server. The symbol "-" in each column means that the agent cannot arrive at the goal state in the maximum number of learning time.

The client with all information of servers is only the same result as the conventional method and the number 404 is optimal one for the simulation. Likewise, Figure 8 shows the efficiency graph for M1 and M2 methods with dummy updating. All the results are almost same as the conventional case. Table 8 shows the result of the test simulation for them. The client with all information of servers is only the same result as the conventional method and the optimal results are obtained.

Table 7. The result of optimality of Q-learning for SMC

	Conventional		**Model 1 for $\Delta_k^* = \Delta^*/m$**				**Model 1**				**Model 2**			
			***m* = 3**		***m* = 10**		***m* = 3**		***m* = 10**		***m* = 3**		***m* = 10**	
	# Suc.	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**
Client	20	404.0	20	404.0	20	404.0	20	404.0	20	404.0	20	404.0	20	404.0
Server 1			20	404.0	20	457.4	0	-	0	-	0	-	0	-
Server 2			20	404.1	20	456.9	0	-	0	-	0	-	0	-
Server 3			20	404.1	20	458.2	0	-	0	-	0	-	0	-
Server 4					20	459.3			0	-			0	-
Server 5					20	459.8			0	-			0	-
Server 6					20	463.5			0	-			0	-
Server 7					20	461.9			0	-			0	-
Server 8					20	464.8			0	-			0	-
Server 9					20	461.9			0	-			0	-
Server 10					20	459.4			0	-			0	-

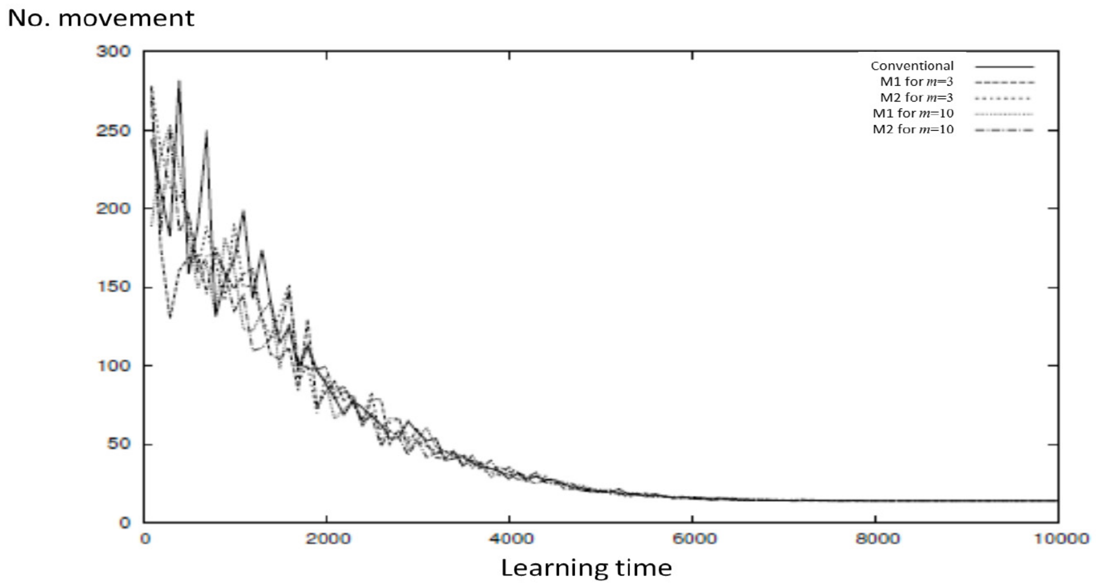

Figure 8. The result of efficiency of Q-learning for SMC (dummy)

Table 8. The result of optimality of Q-learning with dummy updating for SMC

	Conventional		**Model 1 for $m = 3$**		**Model 1 for $m = 10$**		**Model 2 for $m = 3$**		**Model 2 for $m = 10$**	
	# Suc.	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**	**# Suc.**	**M.D.**
Client	20	404.0	20	404.0	20	404.0	20	404.0	20	404.0
Server 1			0	-	0	-	0	-	0	-
Server 2			0	-	0	-	0	-	0	-
Server 3			0	-	0	-	0	-	0	-
Server 4					0	-			0	-
Server 5					0	-			0	-
Server 6					0	-			0	-
Server 7					0	-			0	-
Server 8					0	-			0	-
Server 9					0	-			0	-
Server 10					0	-			0	-

Let us explain the features of the proposed methods and the meaning of simulations. The proposed methods for Q-learning with SMC have the following three features:

(1) The addition form is used for dividing each item of (learning) data,
(2) The Q-value on each server does not become a solution to the problem and,
(3) In addition, the proposed methods with dummy updating hide the state of updated Q-values from the servers.

For each of the features, the meaning is discussed as follows:

(1) There are many forms to divide each item of (learning) data. The addition form, used in this paper, is the most standard method. In the addition form, as shown in Section 2.2, the whole calculation can be replaced by partial calculation. This also holds for the Q-function. On the other hand, in the multiplication form used in Miyajima, *et al.*,[17] it does not necessarily hold. As described above, there are various formats to divide each item of (learning) data, but the addition form is used in this paper.
(2) The results shown in Tables 7 and 8 are obtained by using Q-values on each server. Model 1 for $\Delta_k^* = \Delta^*/m$ shown in Table 7 is not our proposed method because Δ_k^* is constant. Our proposed methods use a real random number for Δ_k^*. For Model 1 for $\Delta_k^* = \Delta^*/m$ which is considered as a conventional method, the goal unwantedly has been reached. On the other hand, for all the proposed methods, the goal has never been reached. The results demonstrate one of the novel features of Q-learning with SMC.
(3) As mentioned in this paper, for RL such as Q-learning on SMC, if each server can know the problem (information on start and goal in maze problem), the server can perform Q-learning by itself. The server can solve the problem. Therefore, it is necessary to hide which state has been updated in learning steps of Q-values. The learning method using dummy updating is proposed in consideration of this point.

The proposed methods are shown only for Q-learning. However, the proposed methods show key points to be considered in the application of SMC to other RL algorithms and how to solve it. Of course, theoretical considerations and propositions and applications of other RL methods should be discussed in the future.

5. Conclusions

In this paper, RL algorithms for SMC were proposed and the effectiveness of them was shown in numerical simulation. Important points for RL algorithms are that explicit learning data are not used and which action is selected based on Q-values. First, two RL algorithms were proposed using shared data and the effectiveness was shown. When they performed RL for SMC, there was possibility that some servers know secure computation. Therefore, improved RL algorithms for SMC were proposed. They are the methods with dummy updating and it seems that any server cannot know secure computation. In today's AI technology, it is often used in combination with RL instead of BP alone. The obtained results suggest that SMC learning is possible in such cases.

In the future, Q-learning method in an analog model and another RL methods for SMC will be developed. Further, the safety of algorithms for SMC in theoretical proof will be also shown.

References

[1] Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. J. Network and Computer Applications. 2011; 34: 1-11. https://doi.org/10.1016/j.jnca.2010.07.006

[2] Shamir A. How to share a secret. Comm. ACM. 1979; 22(11): 612-3. https://doi.org/10.1145/359168.359176

[3] Rajesh N, Sujatha K, Lawrence AA. Survey on Privacy Preserving Data Mining Techniques using Recent Algorithms. International Journal of Computer Applications. 2016; 133(7): 30-3. https://doi.org/10.5120/ijca2016907917

[4] Rathna SS, Karthikeyan T. Survey on Recent Algorithms for Privacy Preserving Data mining. International Journal of Computer Science and Information Technologies. 2015; 6 (2): 1835-40.

[5] Gentry C. Fully Homomorphic Encryption Using Ideal Lattices. STOC2009. 2009: 169-78.

[6] HElib. An Implementation of homomorphic encryption. https://github.com/shaih/HElib

[7] Beimel A. Secret-sharing schemes: a survey. In Proc. of the Third international conference on Coding and cryptology (IWCC 11); 2011. https://doi.org/10.1007/978-3-642-20901-7_2

[8] Canetti R, Feige U, Goldreich O, et al. Adaptively secure multi-party computation. Annual Acm Symposium on Theory of Computing. 1996; 96: 639-48. https://doi.org/10.1145/237814.238015

[9] Cramer R, Damgård I, Maurer U, et al. General secure multi-party computation from any linear secret-sharing scheme. International Conference on Theory & Application of Cryptographic Techniques. 2000; 1807: 316-34. https://doi.org/10.1007/3-540-45539-6_22

[10] Ben-David A, Nisan N, Pinkas B, et al. Fairplay MP: a system for secure multi-party computation. Acm Conference on Computer & Communications Security. 2008: 257-66.

[11] Yuan J, Yu S. Privacy Preserving Back-Propagation Neural Network Learning Made Practical with Cloud Computing. IEEE Trans. On Parallel and Distributed Systems. 2013; 25(1): 212-21. https://doi.org/10.1109/TPDS.2013.18

[12] Schlitter N. A Protocol for Privacy Preserving Neural Network Learning on Horizontal Partitioned Data. Privacy Statistics in Database (PSD); 2008.

[13] Chen T, Zhong S. Privacy-Preserving Back Propagation Neural Network Learning. IEEE Trans. on NN. 2009; 20(10): 1554-64. PMid:19709975. https://doi.org/10.1109/TNN.2009.2026902

[14] Catak FO. Secure multi-party computation based privacy preserving extreme learning machine algorithm over vertically distributed data. International Data Mining & Cybersecurity Workshop. 2015; 9490: 337345.

[15] Chida K, Ikarashi D, Hamada K, et al. A Lightweight Three-Party Secure Function Evaluation with Error Detection and Its Experimental Result. IPSJ Journal. 2011 September; 52(9): 2674-85 (In Japanese).

[16] Miyanishi Y, Kanaoka A, Sato F, et al. New Methods to Ensure Security to Increase User's Sense of Safety in Cloud Services. Proc. of The 14th IEEE Int. Conference on Scalable Computing and Communications (ScalCom-2014). 2014 December: 859-65. https://doi.org/10.1109/uic-atc-scalcom.2014.37

[17] Miyajima H, Shigei N, Miyajima H, et al. New Privacy Preserving Back Propagation Learning for Secure Multiparty Computation. IAENG International Journal of Computer Science. 2016; 43(3): 270-6.

[18] Sakuma J, Kobayashi S, Wright RN. Privacy-preserving reinforcement learning. In: Cohen WW, McCallum A, Roweis ST. (eds.) ICML. ACM International Conference Proceeding Series. 2008; 307: 864871. https://doi.org/10.1145/1390156.1390265

[19] Ling MH, Yau KLA, Qadir J, et al. Application of reinforcement learning for security enhancement in cognitive radio networks. Appl. Soft Comput. 2015; 37: 809829. https://doi.org/10.1016/j.asoc.2015.09.017

[20] Poh GS, Yau KA. Preserving Privacy of Agents in Reinforcement Learning for Distributed Cognitive Radio Networks. Springer International Publishing; 2016. https://doi.org/10.1007/978-3-319-46687-3_61

[21] Koo J, Lin X, Bagchi S. RL-BLH: Learning-Based Battery Control for Cost Savings and Privacy Preservation for Smart Meters. 47th Annual IEEE/IFIP International Symposium on Dependable Systems and Networks (DSN). June 2017.

[22] Miyajima H, Shigei N, Miyajima H, et al. New Privacy Preserving Clustering Methods for Secure Multiparty Computation. Artificial Intelligence Research. 2017; 6(1): 27-36. http://doi.org/10.5430/air.v6n1p27

[23] Watkins CJCH, Dayan P. Q-learning. Machine Learning. 1992; 8: 55-68. https://doi.org/10.1007/BF00992698

[24] Sutton RS, Barto AG. Reinforcement Learning. MIT Press; 1998.