International Journal of Innovative Computing, Information and Control
Volume **14**, Number **2**, April **2018**

ICIC International ©2018 ISSN 1349-4198
pp. 727–735

A PROPOSAL OF PROFIT SHARING METHOD FOR SECURE MULTIPARTY COMPUTATION

Hirofumi Miyajima[1], Noritaka Shigei[2], Hiromi Miyajima[2] and Norio Shiratori[3]

[1]Faculty of Informatics
Okayama University of Science
1-1 Ridaicho, Kita-ku Okayama-shi, Okayama 700-0005, Japan
miya@mis.ous.ac.jp

[2]Graduate School of Science and Engineering
Kagoshima University
1-21-24 Korimoto, Kagoshima-shi, Kagoshima 890-0065, Japan
{ shigei; miya }@eee.kagoshima-u.ac.jp

[3]Research and Development Initiative
Chuo University
1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
norio@shiratori.riec.tohoku.ac.jp

Received July 2017; revised November 2017

Abstract. *Many studies for secure computation using shared data on the cloud system are made to avoid secure risks being abused or leaked and to reduce computing cost. The secure multiparty computation (SMC) is one of these methods. There are two methods for constructing a machine learning (ML) based on SMC. One is a method of sharing learning data into several subsets and learning at each server. The other method is to divide the learning data itself and learn by using each server. In the latter, we have proposed learning methods of BP, k-means and fuzzy inference about SMC so far. Further, we proposed a learning method of SMC on Q-learning which is one of reinforcement learning methods, and showed its effectiveness in the previous paper. Though Q-learning is a learning method with excellent generalization ability, it is known that it takes much time to learn. On the other hand, the profit sharing (PS) method is known to have a shorter learning time than Q-learning. Therefore, it is desired that PS learning method for SMC is superior in learning time to Q-learning method for SMC. In this paper, we propose PS learning method on SMC and show its effectiveness.*
Keywords: Cloud computing, Secure multiparty computation, Reinforcement learning

1. **Introduction.** Privacy preserving for machine learning and data mining on the cloud system can be achieved in various methods by use of randomization techniques, cryptographic algorithms, anonymization methods, etc. [1, 2, 3, 4, 5]. However, data encryption system requires both encryption and decryption for requests of client or user, so its complexity is very high. In the field of privacy preserving, the problem is the trade-off between the security and the complexity of computation. As one of these studies, secure multiparty computation (SMC) has been introduced [6, 7, 8, 9]. The purpose of SMC is to allow servers to carry out distributed computing tasks in secure way. Most of the works in SMC are developed on applying the model of SMC on different data distributions such as vertically, horizontally and arbitrarily partitioned data [6, 7]. They are methods that each server performs its processing for the subset of data. However, they need a large number of servers in order to keep privacy and security. Therefore, SMC systems sharing

data itself to each server attract attention and some studies with them have been done [10, 11]. In Miyajima et al. [12, 13, 14], learning methods for SMC of BP and VQ (Vector Quantization) methods have been proposed and the effectiveness of them has been shown. Further, Q-learning which is one of reinforcement learning methods for SMC has been proposed in the previous paper [14]. It is known that PS learning is faster in learning time than Q-learning [16, 17].

In this paper, a learning method for PS which is one of reinforcement learning methods for SMC is proposed and it is shown that the learning time and accuracy of the proposed method are almost the same as the conventional PS learning and the proposed method is superior in learning time than Q-learning method 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, the conventional PS method is introduced. In Section 3, PS method for SMC is proposed. In Section 4, numerical simulations for a maze problem are performed to show the performance of the proposed method. Finally, Section 5 concludes the paper.

2. **Preliminary.**

2.1. **Cloud system and related works with SMC.**

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

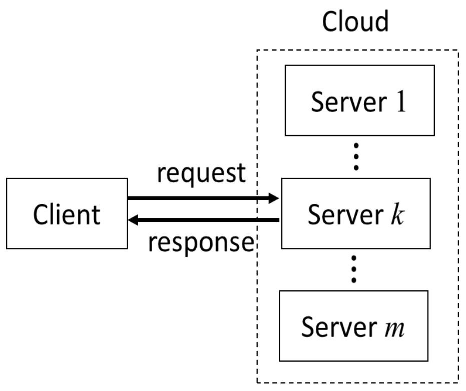
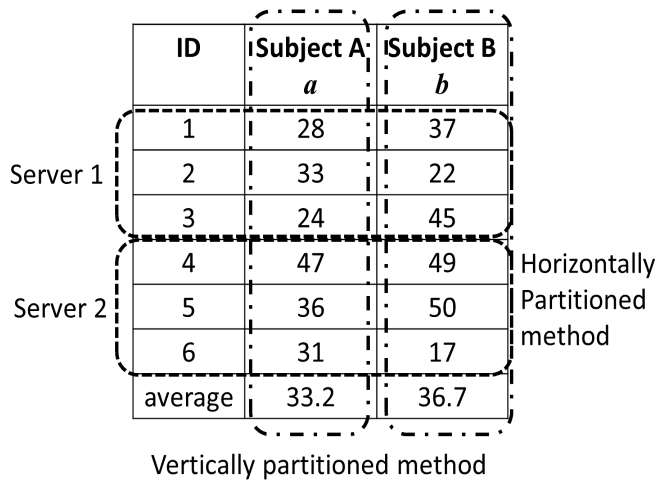


FIGURE 1. A cloud system

Let us explain about conventional works with them. Three types of methods for partitioning data to be securely shared are well known [5, 6, 7, 8, 9]. 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. 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 $(28+33+24)/3$ and $(37+22+45)/3$, respectively. Likewise, Server 2 computes two averages for subjects A and B as $(47+36+31)/3$ and $(49+50+17)/3$, respectively. Servers 1 and 2 send the

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



ID	Subject A a	Subject B b
1	28	37
2	33	22
3	24	45
4	47	49
5	36	50
6	31	17
average	33.2	36.7

Server 1
Server 2
Horizontally Partitioned method
Vertically partitioned method

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

ID	subject A	subject B	Additional form							
					a			b		
	a	b	r_1	r_2	a_1	a_2	a_3	b_1	b_2	b_3
1	8	14	5	4	4	3.2	0.8	7	5.6	1.4
2	25	12	5	3	12.5	7.5	5	6	3.6	2.4
3	39	29	−5	2	−19.5	7.8	50.7	−14.5	5.8	37.7
sum	72	55			−3	18.5	56.5	−1.5	15	41.5
average	24	18.3			−1	6.2	18.8	−0.5	3	13.8

calculated averages to the client and the client obtains the averages of subjects A and B as 33.2 and 36.7, 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 of subjects. The third method, the arbitrary partitioning method, horizontally and vertically splits the dataset into multiple parts, and the method assigns the split parts to the servers. See [14] about vertical and arbitrary partitioning methods. 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 to dividing (sharing) data itself seems to keep them by a small number of servers.

Let us explain secure shared data used in the proposed method [11, 12, 13, 14]. In Table 2, a, b and ID are original data (marks) and the identifier. The assumed task is to calculate the average of the data. Let $m = 3$. Two numbers 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, where $a_1 = a(r_1/10)$, $a_2 = a(r_2/10)$ and $a_3 = a(1 - r_1/10 - r_2/10)$, and $b_1 = b(r_1/10)$, $b_2 = b(r_2/10)$ and $b_3 = b(1 - r_1/10 - r_2/10)$, and $-9 \leq r_1, r_2 \leq 9$, $r_1 \neq 1$ and $r_2 \neq 1$. For example, a_1, a_2 and a_3 for ID = 1 are computed as $a_1 = 8 \times (5/10) = 4$, $a_2 = 8 \times (4/10) = 3.2$ and $a_3 = 8 \times (1 - 5/10 - 4/10) = 0.8$, 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.

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 -3, 18.5 and 56.5, respectively. As a result, the total sum 72 and the average 24 are obtained from $-3 + 18.5 + 56.5$ and $-1 + 6.2 + 18.8$, respectively. Remark that each data for secure randomized and the method does not need to use complex encrypted or decrypted data.

2.2. **Profit sharing method.** Profit sharing (PS) is a reinforcement learning technique for the exploitation type [15, 16, 17]. It can be used to find an optimal action-selection policy for a given problem. In solving problems using PS, it is determined how the agent selects an action at any state. It is performed by learning an action-value (PS-value) function that gives the expected utility of taking the action for the current state. A PS-value function is defined as a function PS: $S \times A \to R$, where S, A and R are sets of states, actions and real numbers, respectively. First, let all PS-values be random. Then each action for a state is selected randomly. If a solution for the problem is obtained (it is called an episode), PS-values are updated based on the updated formula. In this case, there are some methods to select action randomly. In this paper, the roulette selection is used as shown later.

In the first part of PS learning, the action for the state is selected randomly and the action becomes decidable as learning steps proceed. Let $Z_l = \{1, 2, \ldots, l\}$ and $Z_l^* = \{0, 1, \ldots, l\}$. In learning steps, PS-value function is updated as

$$PS\left(s_{i(l)}, a_{j(l)}\right) \leftarrow PS\left(s_{i(l)}, a_{j(l)}\right) + r\gamma^{l_f - l} \tag{1}$$

for $l \in Z_{l_f}^*$, where r and γ are the reward and discount rate, respectively. l is time step for the episode and $s_{i(l)}$ and $a_{j(l)}$ are the state and action on time step l, respectively. The time step l_f means the time when one episode finished. Equation (1) means that all PS-values passed during the episode are updated after an episode finished, that is, the agent reached the goal state from the start state. The conventional algorithm for profit sharing is shown as follows [16, 17].

[Profit Sharing Algorithm]

l: time step for one episode, t: learning time.

$s_{i(l)}$: the current state at learning step l for one episode.

s_0: the initial state, s_f: the goal (target) state.

$s_{i(l+1)}$: the state after performing the action $a_{j(l)}$ at the state $s_{i(l)}$

$PS(s, a)$: PS-value for the state $s \in S$ and the action $a \in A$.

$t_{\max}$: the maximum number of learning time.

Step 1 Let r and γ be reward and discount rate. All $PS(s, a)$ are set randomly for $s \in S$ and $a \in A$. Let $t = 0$.

Step 2 Let $l = 0$ and $i(l) = 0$, that is $s_{i(l)} = s_0$.

Step 3 The action a at the state $s_{i(l)}$ is selected based on the following $R\left(s_{i(l)}, a\right)$ (called the roulette selection) as follows:

$$R\left(s_{i(l)}, a\right) = \frac{PS\left(s_{i(l)}, a\right)}{\sum_{b \in A} PS\left(s_{i(l)}, b\right)} \tag{2}$$

Let $a_{j(l)}$ be the selected action based on Equation (2).

Step 4 Let $s_{i(l+1)}$ be the state after performing the action $a_{j(l)}$ at the state $s_{i(l)}$. If $s_{i(l+1)} \neq s_f$ and $s_{i(l+1)}$ is permissible (movable), then go to Step 5; else go to Step 3 to select another action.

Step 5 If $s_{i(l+1)} = s_f$, then go to Step 6; else go to Step 3 with $l \leftarrow l + 1$.

Step 6 Let $l_f \leftarrow l$. Each of PS-values $PS\left(s_{i(l)}, a_{j(l)}\right)$ for $l \in Z_{l_f}^*$ is updated as follows:

$$PS\left(s_{i(l)}, a_{j(l)}\right) \leftarrow PS\left(s_{i(l)}, a_{j(l)}\right) + r\gamma^{l_f - l} \tag{3}$$

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

3. **Profit Sharing for Secure Multiparty Computation.** In PS learning on cloud system, PS-values are shared to each server in addition form. Each server updates shared PS-values and sends the result to the client. The client can get new PS-values by adding data of m servers. The process is iterated until the evaluating value for the problem satisfies the final condition. The problem is how PS-values on the client are updated using PS-values shared on each server. The shared PS-values to each server are given as follows:

$$PS(s,a) = \sum_{k=1}^{m} PS^k(s,a) \text{ for } s \in S \text{ and } a \in A \tag{4}$$

See [14] about the detailed meaning of SMC for reinforcement learning.

Table 3. The method M1 of profit sharing for SMC

	Client	k-th Server ($1 \le k \le m$)
Initialization	Let $t = 0$.	The numbers r and γ are given.
Step 1	Let $l = 0$ and $s_{i(l)} = s_0$.	$PS^k(s_i, a_j)$ is set randomly.
Step 2	Send the state $s_{i(l)}$ to each server.	
Step 3		Send $PS^k\left(s_{i(l)}, a_j\right)$ for $a_j \in A$ to the client.
Step 4	Calculate $R\left(s_{i(l)}, a\right) = \dfrac{\sum_{k=1}^{m} PS^k\left(s_{i(l)}, a\right)}{\sum_{b \in A}\sum_{k=1}^{m} PS^k\left(s_{i(l)}, b\right)}$ for $a \in A$ and select $a_{j(l)}$ based on the roulette selection (Equation (2)). Let $s_{i(l+1)}$ be the next state for the action $a_{j(l)}$ and the state $s_{i(l)}$.	
Step 5	If $s_{i(l+1)} = s_f$ then go to Step 6 else if $s_{i(l+1)}$ is permissible (movable), then go to Step 2 with $l \leftarrow l + 1$ else go to Step 4 to select another action.	
Step 6	Let $l_f = l$. Let $L = \left\{(i(l), j(l)) \text{ for } s_{i(l)} \text{ and } a_{j(l)} \mid l \in Z^*_{l_f}\right\}$ and $\sum_{k=1}^{m} \varepsilon_k = 1$. Send the set L and ε_k for $k \in Z_m$ to each server.	
Step 7		Each of PS-values is updated as follows: $PS^k(s_i, a_j) \leftarrow PS^k(s_i, a_j) + \varepsilon_k r \gamma^{l_f - l}$ for $(i(l), j(l)) \in L$.
Step 8	If $t \neq t_{\max}$ else go to Step 1 with $t \leftarrow t + 1$ else the algorithm terminates and $PS(s_i, a_j) = \sum_{k=1}^{m} PS^k(s_i, a_j)$ for $s_i \in S$ and $a_j \in A$.	

The proposed method is that updating of PS-values is performed in each server as shown in Table 3. Each of initial values of client and servers is set in Steps 1, 2, and 3. In Step 4, the current PS-value $PS\left(s_{i(l)}, a\right)$ for the client is computed from PS-values of each server and the action $a_{j(l)}$ is selected based on the roulette selection. Further, the next state $s_{i(l+1)}$ is determined from $s_{i(l)}$ and $a_{j(l)}$. In Step 5, if $s_{i(l)} \neq s_f$ and $s_{i(l)}$ is permissible (movable), then Steps 2 to 4 are iterated; else new action and state are selected. In Step 6, the set of PS-values for actions of initial to goal states selected in an episode is defined as L and real numbers ε_k for $k \in Z_m$ are defined. In Step 7, each PS-value $PS^k(s_i, a_j)$ for the set L in each server is updated using L and ε_k. In Step 8, it is checked if the final condition is satisfied. If the final condition is not satisfied, the next episode is iterated.

The proposed method of Table 3 is called the method M1.

The problem of the method M1 is that there is the possibility that some servers know secure computation. In order to improve it, a method with dummy updating is proposed. In the method M1, Steps 6 and 7 of Table 3 are changed as follows.

Step 6 (Client) Let $l_f = l$. Let $L = \left\{(i(l), j(l))\ s_{i(l)} \text{ and } a_{j(l)} | l \in Z_{l_f}^*\right\}$. Let

$$\sum_{k=1}^{m} O_k(s, a) = \begin{cases} 1 & ((i(l), j(l)) \in L) \quad \text{(A)} \\ 0 & \text{(otherwise)} \quad \text{(B)} \end{cases} \tag{5}$$

and send them to each server.

Step 7 (k-th Server)

$$PS^k(s, a) \leftarrow PS^k(s, a) + O_k(s, a) r \gamma^{l_f - l} \tag{6}$$

for $s \in S$ and $a \in A$.

Remark that from two conditions of Equation (5), the PS-value for the set L is only updated and each server cannot know which PS-value is updated. The improved M1 method is called the method M2.

4. Numerical Simulations for the Proposed Algorithm.

The problem is to find the shortest path of Figure 2 known as Sutton's maze problem [15]. In Figure 2, the agent cannot go to black and outer areas and the agent iterates to move from the start to goal areas based on each algorithm. The simulation conditions are as follows.

1) The agent can move to four directions at each area (state) except black and border areas.

2) 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.

3) If the agent arrives at the goal area in the maximum number of learning time, the

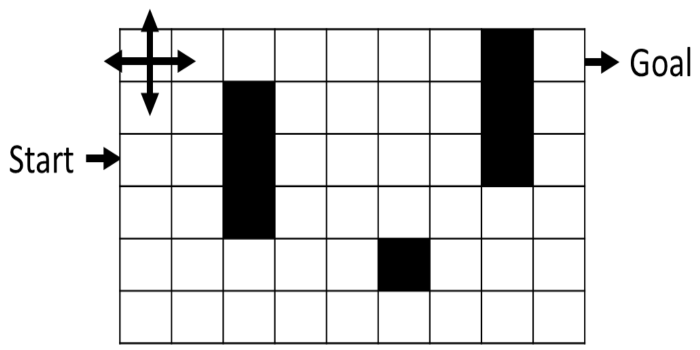


Figure 2. Sutton's maze problem

agent starts the new trial.
4) Let $t_{\max} = 10000$, $r = 10$, $\alpha = 0.5$, $\gamma = 0.92$, and $m = 10$. Twenty trials for learning and test are performed for each algorithm. These constants are selected as the optimum numbers in pre-simulations. The number $m = 10$ is selected as the large number of servers.
5) 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 3 shows the efficiency graph (of the learning result). The graph represents the number of times of movement (abbreviated as No. movement) to the learning time. In Figure 3, PS, PS-M1 ($m = 10$), PS-M2 ($m = 10$), QL-M1 ($m = 10$) and QL-M2 ($m = 10$) mean the conventional PS algorithm and the methods M1 and M2 with $m = 10$ for PS learning and Q-learning, respectively. See [14] about Q-learning methods for SMC. As shown in Figure 3(a), all the results of the proposed methods are almost the same as the conventional case. Further, as shown in Figure 3(b), it is also shown that the proposed methods M1 and M2 are faster than the methods M1 and M2 for SMC of Q-learning. Table 4 shows the result of the test simulation. In Table 4, M.D. means the average of movement distance from twenty trials. The symbol "−" in each columun means that the agent cannot arrive at the goal state in the maximum number of learning time. The proposed method shows almost the same results as the conventional one, where the optimal number is 404. The optimum value 404 is calculated as the sum of the number of steps of the shortest path to the goal with all states as the starting point. Further, the same test trial was performed if the agent can arrive at the goal using PS-values of each server. In this case, the agent could never arrive at the goal in the maximum number of learning time (See Table 4). Figure 3 and Table 4 mean that the proposed methods using shared data give almost the same results as the conventional case while keeping secret computation.

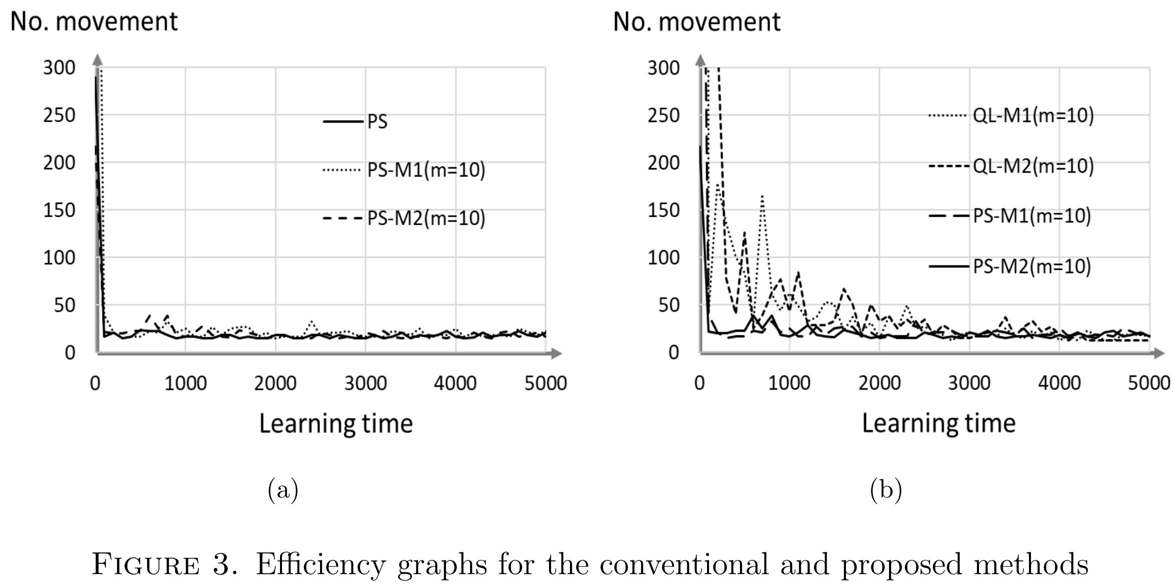


Figure 3. Efficiency graphs for the conventional and proposed methods

Let us explain the features of the proposed methods and the meaning of simulations. The proposed methods for PS learning with SMC have the following three features.
(1) The addition form is used for dividing each item of (learning) data.
(2) PS-values on each server do not become a solution to the problem.
(3) In addition, the proposed method with dummy updating can hide updated PS-values

Table 4. The result of optimality of profit sharing for SMC

	The conventional		M1 for $m = 10$		M2 for $m = 10$	
	No.Suc.	M.D.	No.Suc.	M.D.	No.Suc.	M.D.
The client	20	406.2	20	408.1	20	407.9
Server 1			0	–	0	–
Server 2			0	–	0	–
Server 3			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	–

from each server. As a result, the client can only get the solution for the problem, but each server cannot get it.

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 2.1, the whole calculation can be replaced by partial calculation. This also holds for the PS-value function. On the other hand, in the multiplication form used in [13], it does not necessarily hold.

(2) It is impossible to get the solution to the problem only using the PS values of each server, and only the client having all the information on PS-values can obtain the approximate solution (See Table 4).

(3) As mentioned in this paper, for RL such as PS learning on SMC, if each server can know the problem (information on start and goal in maze problem), the server can perform PS learning by itself. The server is possible to solve the problem. Therefore, it is necessary to hide which state has been updated in learning steps of PS-values. The proposed method with dummy updating is one method to realize it.

5. **Conclusion.** In this paper, PS learning method, one of RL for SMC on the cloud system, was proposed and its effectiveness was shown in numerical simulation. In Section 2, cloud computing system, related works on SMC and a secure data sharing mechanism used in this paper are explained. Further, the conventional PS method is introduced. In Section 3, PS method for SMC is proposed. In Section 4, numerical simulations for a maze problem are performed to show the performance of the proposed method. Important points for RL algorithms are that explicit learning data are not used and the action is selected based on PS-values. As a result, the results of the proposed methods were almost the same as the conventional PS learning. Further, it was also shown that the proposed methods M1 and M2 were faster in learning time than the methods M1 and M2 for SMC of Q-learning. In the future works, the proposed method in an analog model for SMC will be developed and will be applied to another problems.

REFERENCES

[1] A. Shamir, How to share a secret, *Communications of the ACM*, vol.22, no.11, pp.612-613, 1979.

[2] S. Subashini et al., A survey on security issues in service delivery models of cloud computing, *J. Network and Computer Applications*, vol.34, pp.1-11, 2011.

[3] A. Beimel, Secret-sharing schemes: A survey, *Proc. of the 3rd International Conference on Coding and Cryptology (IWCC 2011)*, 2011.

[4] HElib, *An Implementation of Homomorphic Encryption*, https://github.com/shaih/HElib.

[5] J. Yuan et al., Privacy preserving back-propagation neural network learning made practical with cloud computing, *IEEE Trans. PDS*, vol.25, no.1, pp.212-221, 2013.

[6] N. Schlitter, A protocol for privacy preserving neural network learning on horizontal partitioned data, *Privacy Statistics in Database (PSD)*, 2008.

[7] F. O. Catak, Secure multi-party computation based privacy preserving extreme learning machine algorithm over vertically distributed data, *International Conference on Neural Information Processing, LNCS*, vol.9490, pp.337-345, 2015.

[8] R. Canetti et al., Adaptively secure multi-party computation, *STOC'96*, pp.639-648, 1996.

[9] R. Cramer et al., General secure multi-party computation from any linear secret-sharing scheme, *Proc. of International Conference on the Theory and Application of Cryptographic Techniques*, 2000.

[10] K. Chida et al., A lightweight three-party secure function evaluation with error detection and its experimental result, *IPSJ Journal*, vol.52, no.9, pp.2674-2685, 2011 (in Japanese).

[11] Y. Miyanishi, A. Kanaoka, F. Sato, X. Han, S. Kitagami, Y. Urano and N. Shiratori, 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)*, pp.859-865, 2014.

[12] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami and N. Shiratori, *New Privacy Preserving Clustering Methods for Secure Multiparty Computation*, vol.6, no.1, 2017.

[13] H. Miyajima et al., New privacy preserving back propagation learning for secure multiparty computation, *IAENG IJCS*, vol.43, no.3, pp.270-276, 2016.

[14] H. Miyajima et al., A proposal of privacy preserving reinforcement learning for secure multiparty computation, *Artificial Intelligence Research*, vol.6, no.2, pp.57-68, 2017.

[15] R. S. Sutton et al., *Reinforcement Learning*, MIT Press, 1998.

[16] K. Miyazaki et al., On the rationality of profit sharing in multi-agent reinforcement learning, *Proc. of the 4th International Conference on Computational Intelligence and Multimedia Applications*, pp.123-127, 2001.

[17] K. Miyazaki et al., MarcoPolo: A reinforcement learning system considering trade off exploitation and exploration under Markovian environments, *The Japan Society for Artificial Intelligence*, vol.12, no.1, pp.78-89, 1997 (in Japanese).