Analog Q-learning Methods for Secure Multiparty Computation

Hirofumi Miyajima, Noritaka Shigei, Hiromi Miyajima, and Norio Shiratori

Abstract—**One problem in cloud computing system is how to conceal individual information. Although encryption technology is one of methods to solve the problem, the computation time required for encryption and decryption as the amount of data increases is a bottleneck. On the other hand, the secret processing of SMC, by reducing the amount of data processed by partitioning data, achieves confidentiality and speeding up. Compared with encryption technology, SMC can realize high-speed and secret processing, but in order to perform it many servers are required. Therefore, an easily method using SMC and simple encryption has been proposed. Several algorithms related to supervised, unsupervised and reinforcement learning have been proposed so far as the methods of SMC on machine learning. Since there is no learning data in reinforcement learning, the result is obtained on the client without informing the solution system (parameter) to any server. Algorithms for Q learning and PS learning in digital model have been proposed so far but no results on analog model have been obtained yet.**

In this paper, an algorithm of Q learning in analog model is proposed. Moreover, the effectiveness of the result is shown by numerical simulation. of this column.

Index Terms—**cloud computing, secure multiparty computation, Q-learning, analog model.**

I. Introduction

WITH increasing interest in Artificial Intelligence (AI), many studies have been made with Machine Learning (ML). With ML, the supervised, the unsupervised and Reinforcement Learning (RL) are well known. Recently, the importance of RL in AI is increasing with the advancement of learning. Due to respond to the increase in the amount of data or complex problems for ML, the use of cloud computing systems is spreading. The development of cloud computing allows the use such as big data analysis to analyze enormous information accumulated by the client, and to create market value of data [1]–[6]. On the other hand, the client of cloud computing system cannot escape from anxiety about the possibility of information being abused or leaked. In order to solve the problem, data processing methods can be considered such as cryptographic one [1], [2]. However, data encryption system requires both encryption and decryption for requests of client or user, so much time is required for transformation of data. Therefore, safe systems for distributed processing with secure data attract attention, and a lot of studies with them have been done. It is known that SMC's idea of distributing learning data among multiple servers is one method to realize this [3], [4]. As for SMC, many methods of learning by sharing learning data into subsets have been proposed [3]–[6]. Then, in the case where learning data does not exist explicitly like RL, how should it be done? We have proposed an applicable model also in this case. The idea is to divide not only learning data but also learning parameters, find partial solutions at each server, combine them and make it the solution of the system [8]. Based on this idea, some algorithms for RL of SMC are already proposed in previous papers [9], [10]. However, they were methods using the digital model. On the other hand, solutions to analog model are desired in the real world [11], [12], [15]. It seems that the latter gives a solution closer to the optimal solution than the former.

In this paper, SMC algorithms for Q-learning are proposed in the analog model and their effectiveness using numerical simulations is shown. The idea is that in the digital model, only one action is selected at each time, whereas in the analog model it is decided as a sum of weighted vectors of actions. In Section 2, cloud computing system, related works on SMC and how to share the data used in this paper is explained. Further, a Q-learning method in the analog model is introduced. In Section 3, two Q-learning methods in the analog model for SMC are proposed. In section 4, numerical simulations for a maze problem are performed to show the performance of proposed methods.

II. Preliminary

A. Cloud system and related works with SMC

The system used in this paper is composed of a client and m servers. Each data is divided into m pieces of numbers and is sent to each server (See Fig.1 for $m = 2$). Each server performs its computation and sends the computation result to the client. The client can get the result using them. If the result is not obtained by one processing, then the multiple processing is repeated. As for the cloud system, there are many methods of secure preserving, but it seems that SMC method using distributed processing is suitable for the system. In particular, three types of conventional methods for partitioning data to be securely shared are well known [3], [4]. They are known as horizontal, vertical and arbitrary partitioning methods. In the following, three methods are only explained easily by using a data example of students' marks shown in Table I. See Miyajima [8] about the detailed explanation. In Table I, a and b are original data (marks) and ID is the identifier of student. The number of servers is two. The assumed task is to calculate the average of data for each subject.

The horizontal partitioning method assigns the horizontally partitioned data to servers as follows :

Server 1 : data for ID = 1, 2
Server 2 : data for ID = 3, 4

H. Miyajima is with the Faculty of Informatics, Okayama University of Science, 1-1 Ridaicho, Kitaku, Okayama, 700-0005, Japan e-mail: miya@mis.ous.ac.jp

N. Shigei is with the Kagoshima University, Kagoshima, Japan e-mail: shigei@eee.kagoshima-u.ac.jp

H. Miyajima is with the Kagoshima University, Kagoshima, Japan e-mail: miya@eee.kagoshima-u.ac.jp

N. Shiratori is with the Research and Development Initiative, Chuo University, Tokyo, Japan e-mail: norio@shiratori.riec.tohoku.ac.jp

TABLE I
CONCEPT OF HORIZONTALLY AND VERTICALLY PARTITIONED METHODS COMPOSED OF ONE CLIENT AND TWO SERVERS.

	ID	Subject A a	Subject B b
Server 1	1	22	32
Server 1	2	24	37
Server 2	3	40	40
Server 2	4	13	45
	Average	24.75	38.5

Horizontally partitioned method

Vertically partitioned method

In the method, Server 1 computes two averages for subjects A and B as $(22+24)/2$ and $(32+37)/2$, respectively. Likewise, Server 2 computes two averages for subjects A and B as $(40+13)/2$ and $(40+45)/2$, respectively. Servers 1 and 2 send calculated averages to the client and the client obtains the averages of subjects A and B as 24.75 and 38.5, respectively. Since each server cannot know half of the dataset, the method preserves security (or privacy). Remark that each server also cannot know the result when we consider average values as unknown parameters.

The vertical partitioning method is one of processing data for each subject (See Table I). All the dataset are divided into two servers, Server 1 and 2, as follows:

Server 1: dataset for subject A,

Server 2: dataset for subject B.

In this case, two averages with subsets A and B for Server 1 are computed as $(22+24+40+13)/4$ and $(32+37+40+45)/4$, respectively. As a result, two averages for subset A and B are 24.75 and 38.5, respectively. Each server cannot know half of the dataset, so security preserving holds.

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, any 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 security. On the other hand, the method explained in the next section divides each item of data and seems to keep it even in the case of a small number of servers.

B. Secure divided data for SMC and their application to machine learning

Let us explain secure divided data for the proposed method using Table II [8]. Let a and b be two marks and $m = 2$ (See Fig.1). Assume that the addition form is used for dividing each item. For example, two marks a and b are divided into two real numbers as $a = a_1 + a_2$ and $b = b_1 + b_2$ as follows:

$a = a_1 + a_2 : a_1 = 1(r_1/10)$, $a_2 = a(1 - r_1/10)$

$b = b_1 + b_2 : b_1 = b(r_1/10)$ and $b_2 = b(1 - r_1/10)$

where r_1 is a real random number for $-9 \le r_1 \le 9$. If $r_1 = -1$, then $a_1 = 0.2$ and $a_2 = -2.2$ are obtained. Remark that Server 1 and Server 2 have all the data in column-wise of a_1 and b_1 and a_2 and b_2 for each ID as shown in Table II, respectively.

Let us explain how to compute the average for subject A using data a. Server 1 and Server 2 compute the average of

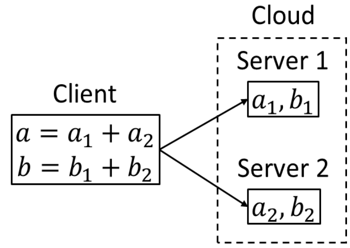


Fig. 1. An example of secure shared data for $m = 2$.

TABLE II
DATA FOR SERVER 1 AND SERVER 2.

ID	subject A	subject B	Additional form				
				a		b	
	a	b	r_1	a_1	a_2	b_1	b_2
1	-2	-6	-1	0.2	-2.2	0.6	-6.6
2	-8	2	-6	4.8	-12.8	-1.2	3.2
3	1	-9	5	0.5	0.5	-4.5	-4.5
Server 1				1.8		-1.7	
Server 2					-4.8		-2.6
average	-3	-4.3					

a_1 and a_2, respectively. In this case, each average in column-wise for a_1 and a_2 is 1.8 and -4.8, respectively. As a result, the average is obtained as -3 from $1.8 - 4.8$.

Remark that each data for server is randomized and the method does not need to use complicated computation as the encryption system.

Let us explain about the application of secure divided data to ML. In the conventional methods, the set of learning data for ML is shared into some 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 [8]. Then, how is the algorithm of RL for SMC? In this case, as there do not exist learning data explicitly, the solution is obtained by repeating trial and error. Since there is no data for learning, it seems that the conventional method using subset of learning data is difficult to use. On the other hand, several methods on security preserving for RL have been proposed, but these are almost all methods using encryption and homomorphic mapping [13], [14]. The proposed method attempts to realize SMC by simple secret computation processing which does not use such complicated cryptographic processing and homomorphic mapping. That is, the aim to reduce the computational complexity of client while keeping the secret of data (Q-value information in this case). In previous papers, we have already proposed Q-learning and PS methods for SMC in the digital model [9], [10]. In the next section, learning methods using secure divided data in the analog model are proposed.

C. Q-learning methods in the digital and analog models

First, let us explain about the Q-learning algorithm in the digital model. The Q-learning is one of RL techniques for environment-identity type [11]. 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 [11], [12], [15]. The 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. Boltzmann or ε-greedy method is used as the method selected data randomly as shown later. If a desired solution for the problem is not obtained, learning is iterated. In the Q-learning, the action is selected based on the Q-value function. If a solution is obtained, Q-values are updated based on the updated formula. By iterating these process, it is known that Q-value function is converges [11]. In the first part of learning, the action for the state is selected randomly and the action becomes decidable as learning steps proceed.

In learning step, Q-value function is updated as

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

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

where r, α and γ are the reward, learning constant and discount rate, respectively. The state $\boldsymbol{s}'$ is the next state selected for the state $\boldsymbol{s}$ and the action $\boldsymbol{a}$. The term $\max_{\boldsymbol{a}' \in A} Q(\boldsymbol{s}',\boldsymbol{a}')$ means the Q-value $Q(\boldsymbol{s}',\boldsymbol{a}'_0)$ for an action $\boldsymbol{a}'_0$ taking the maximum number of $Q(\boldsymbol{s}',\boldsymbol{a}')$. See the Ref. [9] about the example of Q-learning in the digital model.

Let us think about the analog model of Q-learning. Let us explain using an example of maze problem to make the story easy to understand. The problem is how the agent can find the shortest path to the goal from the start point. In the digital model, one of actions at each position is selected. Therefore, the path obtained as the result of learning becomes a zigzag path as behavior. However, in the real problem, it is desirable to get smooth path. An analog model is proposed as a model to realize all directions for behavior (action). Several methods have been proposed for Q-learning for the analog model. Here, we introduce a learning method that determines the action based on the distance between the current position and state (position) of the agent. That is, while hard selection of the action in the digital model is performed, the method of the analog model achieves a soft matching selection for the action. Let us explain how to select the action of Q-learning in the analog model using Fig.2. Assign n states (the center position $\boldsymbol{c}$) in the space where the agent moves. Let $|A| = M$.

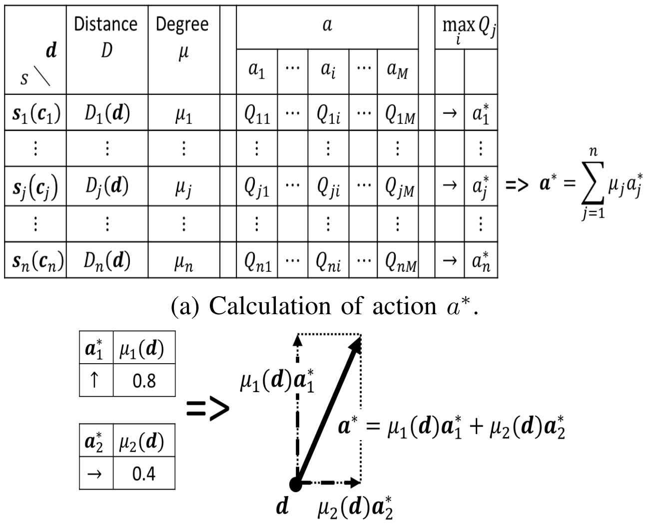
Let $\boldsymbol{d}$ be the current position of the agent. First, the distance $D_j(\boldsymbol{d})$ between $\boldsymbol{d}$ and the center position $\boldsymbol{c}_j$ of each state for $1{\leq}j{\leq}n$ is computed as follows:

$$D_j(\boldsymbol{d}) = \exp\left(-\frac{||\boldsymbol{d}-\boldsymbol{c}_j||^2}{2b_j^2}\right) \tag{3}$$

where $\exp(\cdot)$ means the Gauss function with the width b_j. Remark that the distance $D_j(\boldsymbol{d})$ is large if the distance between $\boldsymbol{d}$ and $\boldsymbol{c}_j$ is near.

Next, the action $\boldsymbol{a}_j^*$ for each state $\boldsymbol{s}_j$ is selected as the action with the maximum number of Q-value (See Fig.2(a)) and the degree μ_j of coincide between $\boldsymbol{d}$ and $\boldsymbol{s}_j$ is computed as follows:

$$\mu_j(\boldsymbol{d}) = \frac{D_j(\boldsymbol{d})}{\sum_{i=1}^n D_i(\boldsymbol{d})} \text{ for } 1{\leq}j{\leq}n \tag{4}$$



(a) Calculation of action a^*.

(b) An example of the action determination at the position $\boldsymbol{d}$ for $M = 2$.

Fig. 2. The method to determine the action in the analog model.

and

$$\sum_{j=1}^n \mu_j(\boldsymbol{d}) = 1 \tag{5}$$

Further, the action $\boldsymbol{a}^*$ for the agent is determined as the composition of vectors as follow:

$$\boldsymbol{a}^* = \sum_{j=1}^n \mu_j \boldsymbol{a}_j^* \tag{6}$$

For example, let us show an example with $M = 2$. Assume that $\boldsymbol{a}_1^*$ with $\mu_1(\boldsymbol{d}) = 0.8$ and $\boldsymbol{a}_2^*$ with $\mu_2(\boldsymbol{d}) = 0.4$ are selected as the upper and the right direction, respectively. Then, the result is shown as the composition of two vectors as Fig.2(b).

As a result, the agent at the position $\boldsymbol{d}$ moves in the direction of $\boldsymbol{a}^*$ and arrives at the new position $\boldsymbol{d}'$. In this case, the moving distance of the agent is selected randomly.

The conventional algorithm for Q-learning is shown as follows :

[Q-learning algorithm for the analog model]

$\boldsymbol{d}$: the current position.
$\boldsymbol{d}_0$: the initial position.
$Q(\boldsymbol{s},\boldsymbol{a})$: Q-value for the state $\boldsymbol{s}$ and the action $\boldsymbol{a}$.
S : The set of states.
A : The set of actions.
t_{max} : The maximum number of learning time.
T_{max} and T_{min}: Constants for Boltzmann selection.
$\boldsymbol{c}_j$: The center position of the state $\boldsymbol{s}_j$ for $1{\leq}j{\leq}n$.
$D_j(\boldsymbol{d})$: The distance between the state $\boldsymbol{s}_j$ and the place $\boldsymbol{d}$ for Eq.3.
ε : The probability for ε-greedy selection method.
p : The real number selected randomly from the interval $0{\leq}p{\leq}1$.

Step 1

Let R^+ and R^-, α and γ be plus and minus reward, learning constant and discount rate. Let S and A be defined. Let $Q(\boldsymbol{s},\boldsymbol{a}) = 0$ for $\boldsymbol{s}{\in}S$ and $\boldsymbol{a}{\in}A$. Let $t = 0$. (Let ε be defined for ε-greedy selection method.)

Step 2

Let $\boldsymbol{d} \leftarrow \boldsymbol{d}_0$ be the start position.

Step 3

The distance $D_j(\boldsymbol{d})$ and the degree μ_j for $1 \leq j \leq n$ are computed from Eqs.3 and 4.

Step 4 (In the case of Boltzmann selection method)

The action $\boldsymbol{a}_j^*$ at the state $\boldsymbol{s}_j$ is selected based on the value $B(\boldsymbol{s}_j, \boldsymbol{a})$ as follows :

$$B(\boldsymbol{s}_j, a) = \frac{\exp\left(Q(\boldsymbol{s}_j, \boldsymbol{a})/T(t)\right)}{\sum_{b \in A} \exp\left(Q(\boldsymbol{s}_j, \boldsymbol{b})/T(t)\right)} \quad (7)$$

$$T(t) = T_{max} \times \left(\frac{T_{min}}{T_{max}}\right)^{\frac{t}{T_{max}}} \quad (8)$$

Let $\boldsymbol{a}_j^*$ be the selected action using Eqs.7 and 8.

That is, the action $\boldsymbol{a}_j^*$ from the set A is selected with the probability $B(\boldsymbol{s}_j, \boldsymbol{a})$. It means that the selection becomes from random decidable as the time increase.

Step 5

The action $\boldsymbol{a}^*$ at the place $\boldsymbol{d}$ is computed using Eq.6.

Let $\boldsymbol{d}'$ be the next position determined from the action $\boldsymbol{a}^*$.

Step 6

Each value of $Q(\boldsymbol{s}_j, \boldsymbol{a}_j^*)$ for the state $\boldsymbol{s}_j$ and the action $\boldsymbol{a}_j^*$ is updated as follows :

$$Q(\boldsymbol{s}_j, \boldsymbol{a}_j^*) \leftarrow Q(\boldsymbol{s}_j, \boldsymbol{a}_j^*) + \mu_j \alpha (r + \triangle) \quad (9)$$

$$\triangle = \sum_{j=1}^{n} \{\gamma \mu_j' \max_{a_j' \in A} Q(\boldsymbol{s}_j', a_j') - \mu_j Q(\boldsymbol{s}_j, a_j^*)\} \quad (10)$$

, where μ_j', $\boldsymbol{s}_j'$ and a_j' are parameters for $\boldsymbol{d}'$ and if $\boldsymbol{d}'$ is in the goal area, then $r = R^+$ and go to Step 7 else if $\boldsymbol{d}'$ is not in the movable place(state), then $r = R^-$ and go to Step 3 else $r = 0$ and go to Step 3.

Step 7

If $t = t_{max}$ then the algorithm terminates else go to Step 3 with $t \leftarrow t + 1$.

If the ε-greedy method is used instead of Boltzmann selection, Step 4 is replaced with the following Step 4' :

Step 4'(In the case of the ε-greedy selection method)

Let p be the real number randomly selected. The action a_j^* for the state $\boldsymbol{s}_j$ is selected based on ε-greedy selection as follows :

$$\boldsymbol{a}_j^* = \begin{cases} \boldsymbol{a} \text{ s.t. } \max_{\boldsymbol{a} \in A} Q(\boldsymbol{s}_j, \boldsymbol{a}) \text{ for } p \geq \varepsilon & (a) \\ \text{randomly selected} \quad \text{for otherwise} & (b) \end{cases} \quad (11)$$

That is, if p is greater than or equal to ε, then the action $\boldsymbol{a}_j^*$ satisfying the condition of Eq.11(a) is selected otherwise the action $\boldsymbol{a}_j^*$ is randomly selected.

The method means that the selected method becomes randomly if ε is large and the selected method becomes decidable if ε is small.

III. Q-LEARNING FOR SECURE MULTIPARTY COMPUTATION IN THE ANALOG MODEL

In Q-learning for SMC on cloud system, Q-values are divided to each server in addition form . Each server updates divided 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 divided on each server. The divided representation of Q-value is given as follows :

$$Q(\boldsymbol{s}, \boldsymbol{a}) = \sum_{k=1}^{m} Q_k(\boldsymbol{s}, \boldsymbol{a}) \ for \ \boldsymbol{s} \in S \ and \ \boldsymbol{a} \in A \quad (12)$$

In the following, two learning methods are proposed.

The proposed methods can be easily applied to other Q-learning algorithms in the analog model.

A. Q-learning for SMC

The first algorithm using Boltzmann (ε-greedy) selection method is shown in Table III. The initial values of client and servers are set in Initializing Step. In Step 1, the initial position is set. In Step 2, the distance $D_j(\boldsymbol{d})$ and the degree μ_i for the current position $\boldsymbol{d}$ and each center position $\boldsymbol{c}_j$ of state $\boldsymbol{s}_j$ for $1 \leq j \leq n$ are computed. In Step 3, the action $\boldsymbol{a}_j^*$ for the state $\boldsymbol{s}_j$ is selected based on Boltzmann selection method. Further, the next action $\boldsymbol{a}^*$ is determined as the vector sum of $\boldsymbol{a}_j^*$ for $1 \leq j \leq n$ and the new position $\boldsymbol{d}'$ is obtained from the action $\boldsymbol{a}^*$. Furthermore, the degree $\mu_j{}'$ of coincide for the position $\boldsymbol{d}'$ is computed from $\boldsymbol{d}'$ and $\boldsymbol{s}_j$ ($1 \leq j \leq n$). In Step 4, the updating rate $\triangle_{\boldsymbol{b}}^k$ for $\boldsymbol{b} \in A$ and $1 \leq k \leq m$ is computed in each server. In Step 5, the updating rate $\triangle^*$ is computed using $\triangle_{\boldsymbol{b}}^k$ for $\boldsymbol{b} \in A$ and $1 \leq k \leq m$. Further, the updating rate ξ_k for each server is determined using $\beta_k(\boldsymbol{s}_j, \boldsymbol{a}_j^*)$ and $\triangle^*$. It means to divide $\triangle^*$ into m pieces of numbers. In Step 6, the Q-value $Q_k(\boldsymbol{s}_j, \boldsymbol{a}_j^*)$ for each server is updated. In Steps 7 and 8, it is checked if the final condition satisfied or not. If the final condition is not satisfied, then the next episode is iterated. The algorithm is called M1 method.

B. Q-learning with dummy updating

In M1 method, designated Q-values are updated based on μ_j for $1 \leq j \leq n$. Therefore, there is the possibility that the server knows information on which Q-value is important. In order not to inform the server of the update information, an improved method with dummy updating is proposed in which all the Q-values are updated.

The fundamental idea of Q-learning with dummy updating is that all Q-values are updated at each step. Therefore, it seems that each server cannot know which Q-value is important or not. Let us explain the improved M1 method. The number $p_k(\boldsymbol{s}, \boldsymbol{a})$ is randomly selected such that $|p_k(\boldsymbol{s}, \boldsymbol{a})| \leq 1$ and $\eta_k(\boldsymbol{s}, \boldsymbol{a})$ is calculated as follows :

$$\eta_k(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} \frac{p_k(\boldsymbol{s}, \boldsymbol{a})}{\sum_{l=1}^{m} p_l(\boldsymbol{s}, \boldsymbol{a})} \text{ for } \boldsymbol{s} = \boldsymbol{s}_j \text{ and } \boldsymbol{a} = \boldsymbol{a}_j^* \text{ (A)} \\ \qquad \text{for } 1 \leq j \leq n \\ \frac{p_k(\boldsymbol{s}, \boldsymbol{a})}{\sum_{l=1}^{m} p_l(\boldsymbol{s}, \boldsymbol{a})} - \frac{1}{m} \text{ for otherwise (B)} \end{cases} \quad (13)$$

TABLE III
M1 METHOD OF Q-LEARNING FOR SMC.

	Client	k-th Server ($1\leq k\leq m$)
Initialization	The numbers R^+, R^-, r, α and γ are given. Let $t=0$. Let $Q_k(s,a)=0$ for $s\in S$ and $a\in A$.	
Step 1	Let $\boldsymbol{d}\leftarrow\boldsymbol{d}_0$.	
Step 2	Calculate $D_j(\boldsymbol{d})$ and μ_j for $\boldsymbol{d}$ and $\boldsymbol{c}_j (1\leq j\leq n)$.	
Step 3		Send all Q-values $Q_k(\boldsymbol{s},\boldsymbol{a})$ for $\boldsymbol{s}\in S$ and $\boldsymbol{a}\in A$ to the client.
Step 4	Calculate $Q(\boldsymbol{s},\boldsymbol{a})=\sum_{k=1}^{m}Q_k(\boldsymbol{s},\boldsymbol{a})$ for $\boldsymbol{s}\in S$ and $\boldsymbol{a}\in A$. Select the action $\boldsymbol{s}_j^*$ for the state $\boldsymbol{s}_j$ based on Boltzmann selection of Eq.(7). Let $\boldsymbol{a}^*=\sum_{j=1}^{n}\mu_j\boldsymbol{a}_j^*$ be the vector sum of $\boldsymbol{a}_j^*(1\leq j\leq n)$. Let $\boldsymbol{d}'$ be the next position determined by $\boldsymbol{a}^*$. The degree μ_j' of coincide between $\boldsymbol{d}'$ and $\boldsymbol{s}_j$ for $1\leq j\leq n$ is computed.	
Step 5	If the position $\boldsymbol{d}'$ is possible(movable), then send the degree μ_j for $1\leq j\leq n$ to each server else go to Step 4.	
Step 6		Calculate $\triangle_b^k=r\mu_j'Q_k(\boldsymbol{s}_j',b)-\mu_jQ_k(\boldsymbol{s}_j,a_j^*)$ for $1\leq j\leq n$ and send them to client.
Step 7	Calculate $\triangle_b=\sum_{k=1}^{n}\triangle_b^k$ and $\triangle^*=r+\max_{b\in A}\triangle_b$ and send $r+\triangle^*$ to all servers, where $r=R^+$, R^- and 0 if $\boldsymbol{d}'$ is in goal, not in movable position and otherwise, respectively.	
Step 8	Select m pieces of random numbers $\beta_k(s,a)$ s.t. $\sum_{k=1}^{m}\beta_k(s,a)=1$. Let $\xi_k=\alpha\beta_k(s_j,a_j)\triangle^*$ for $1\leq k\leq m$. Send ξ_k for $1\leq k\leq m$ to each server.	
Step 9		The Q-value $Q_k(s_j,a_j^*)$ is updated as follows : $Q_k(s_j,a_j^*)\leftarrow Q_k(s_j,a_j^*)+\mu_j\xi_k$
Step 10	If $\boldsymbol{d}'$ is in goal state, then go to Step 11 else go to Step 2 with $\boldsymbol{d}\leftarrow\boldsymbol{d}'$.	
Step 11	It $t=t_{max}$ then the algorithm terminates else go to Step 1 with $t\leftarrow t+1$.	

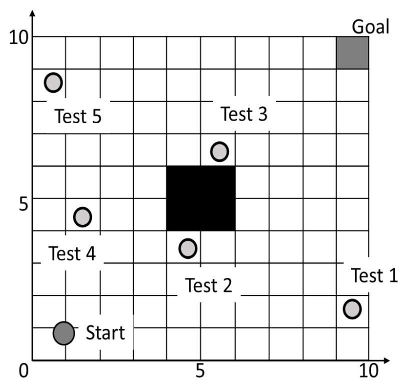


Fig. 3. The figure for the maze problem.

where $\boldsymbol{s}=\boldsymbol{s}_j$ and $\boldsymbol{a}=\boldsymbol{a}_j^*$ for $1\leq j\leq n$ are selected states and actions.

Remark that each case of A and B for Eq.13 holds $\sum_{k=1}^{m}\eta_k(\boldsymbol{s},\boldsymbol{a})=1$ and 0 for $s\in S$ and $\boldsymbol{a}\in A$, respectively. That is, all Q-values for states and actions are randomly updated using $p_k(\boldsymbol{s},\boldsymbol{a})$ and each server cannot know which Q-value is import or not at each step.

The second algorithm using Boltzmann(ε-greedy) selection method is shown in Table IV. The algorithm is called M2 method.

IV. Numerical Simulations for the Proposed Algorithms

In numerical simulations, the problem is to find the shortest path for the agent from the start position to goal area by Q-learning methods (See Fig.3). In Fig.3, the agent can go to any position except for black and outer areas. In order to find the shortest path, the agent iterates trial and error to move from the start to goal area based on each algorithm. The simulation conditions are as follows:

1) Let the start position $\boldsymbol{d}_0=(1,1)$, black(wall) area $B=\{(x_1,x_2)\in R^2|4\leq x_1,x_2\leq 6\}$ and the goal area $\{(x_1,x_2)\in R^2|9\leq x_1,x_2\leq 10\}$, where R is the set of all real numbers.
2) Let $A=\{0,\frac{\pi}{2},\pi,\frac{3\pi}{2}\}$ and the set of central positions $C(S)=\{(x_1,x_2)\in Z^2|1\leq x_1,x_2\leq 9\}-B$, where each of the set A means four directions up, down, right and left in Fig.3, and Z is the set of all integers. Let $n=|C(S)|$. The states are set as a lattice pattern at regular intervals.
3) Let $T_{max}=5.0$ and $T_{min}=0.03$ for Boltzman selection and $\varepsilon=0.1$ for ε-greedy selection.
4) The moving distance h for the agent at each position $\boldsymbol{d}$ is selected randomly for $0.01\leq h\leq 0.5$ and the action is determined by each algorithm.
5) If the agent selects to move to wall or outer area, the agent ignores the selection and reselects a new action, where $r=-1$ is given as the negative reward. It is not counted in the number of trials.
6) If the agent arrives at the goal area in the maximum number of learning time, then the agent starts the new trial, where the reward $r=1$ is added to Q-value as the positive reward.
7) Let $t_{max}=10000$, $r=10$, $\alpha=0.5$, $\gamma=0.92$, and $m=3$ and 10. Twenty trials for learning and test are performed for each algorithm.
8) In the test simulation, experiments are carried out with five positions as the starting points as shown in Fig.3. The result is evaluated as the rate of the number of successful trials and the average of moving distance from each starting point to goal area.

Figs.4 and 5 show the efficiency graphs. They represent the moving distance to the learning time. In Figs.4 and 5, the conventional (ε-greedy or Boltzmann), M1 for $m=3$ and 10, and, M2 for $m=3$ and 10 mean the conventional algorithm with ε-greedy or Boltzmann as selection method, proposed algorithms for M1 with $m=3$ and 10, and for M2 with $m=3$ and 10, respectively. All the results are almost

TABLE IV
M2 METHOD OF Q-LEARNING FOR SMC.

	Client	k-th Server ($1 \leq k \leq m$)
Initialization	The numbers R^{+}, R^{-}, r, α and γ are given. Let $t=0$. Let $Q_k(s,a)=0$ for $s \in S$ and $a \in A$.	
Step 1	Let $\boldsymbol{d} \leftarrow \boldsymbol{d}_0$.	
Step 2	Calculate $D_j(\boldsymbol{d})$ and μ_j for $\boldsymbol{d}$ and $\boldsymbol{c}_j (1 \leq j \leq n)$.	
Step 3		Send all Q-values $Q_k(\boldsymbol{s}, \boldsymbol{a})$ for $\boldsymbol{s} \in S$ and $\boldsymbol{a} \in A$ to the client.
Step 4	Calculate $Q(\boldsymbol{s}, \boldsymbol{a}) = \sum_{k=1}^{m} Q_k(\boldsymbol{s}, \boldsymbol{a})$ for $\boldsymbol{s} \in S$ and $\boldsymbol{a} \in A$. Select the action $\boldsymbol{s}_j^*$ for the state $\boldsymbol{s}_j$ based on Boltzmann selection of Eq.(7). Let $\boldsymbol{a}^* = \sum_{j=1}^{n} \mu_j \boldsymbol{a}_j^*$ be the vector sum of $\boldsymbol{a}_j^* (1 \leq j \leq n)$. Let $\boldsymbol{d}'$ be the next position determined by $\boldsymbol{a}^*$. The degree μ_j' of coincide between $\boldsymbol{d}'$ and $\boldsymbol{s}_j$ for $1 \leq j \leq n$ is computed.	
Step 5	If the position $\boldsymbol{d}'$ is possible(movable), then send the degree μ_j and μ_j for $1 \leq j \leq n$ to each server else go to Step 4.	
Step 6		Calculate $\triangle_b^k = r\mu_j' Q_k(\boldsymbol{s}_j', b) - \mu_j Q_k(\boldsymbol{s}_j, a_j^*)$ for $1 \leq j \leq n$ and send them to client.
Step 7	Calculate $\triangle_b = \sum_{k=1}^{n} \triangle_b^k$ and $\triangle^* = r + \max_{b \in A} \triangle_b$ and send $r + \triangle^*$ to all servers, where $r = R^{+}$, R^{-} and 0 if $\boldsymbol{d}'$ is in goal, not in movable position and otherwise, respectively.	
Step 8	Select m pieces of random numbers $p_k(\boldsymbol{s}, \boldsymbol{a})$ s.t. $\lvert p_k(\boldsymbol{s}, \boldsymbol{a}) \rvert \leq 1$. Calculate $\eta_k(\boldsymbol{s}, \boldsymbol{a})$ of Eq.(13) for $\boldsymbol{s} \in S$ and $\boldsymbol{a} \in A$. Let $\xi_k = \alpha \eta_k(\boldsymbol{s}, \boldsymbol{a}) \triangle^*$ for $1 \leq k \leq m$. Send ξ_k for $1 \leq k \leq m$ to each server.	
Step 9		The Q-value $Q_k(s_j, a_i)$ is updated as follows : $Q_k(s_j, a_i) \leftarrow Q_k(s_j, a_i) + \mu_j \xi_k$
Step 10	If $\boldsymbol{d}'$ is in goal state, then go to Step 11 else go to Step 2 with $\boldsymbol{d} \leftarrow \boldsymbol{d}'$.	
Step 11	It $t = t_{max}$ then the algorithm terminates else go to Step 1 with $t \leftarrow t+1$.	

the same as the conventional cases. Tables V and VI show the results of the test simulations, where Tests 1 to 5 mean the cases with different starting points as shown in Fig.3. In each Table, No. Suc. and M.D. mean the number of successful trials for twenty trials and the average of moving distance for successful trials. Further, the result on each server means one in the cases where the same trials are performed using only Q-values of each server. For example, 0.95 of No.success and 9.83 of M.D. for Test 1 of client in the conventional method mean $19/20$ and 9.83 steps as the average for 20 trials to the goal from the start point of Test 1 in Table V. All the results for client of M1 and M2 methods are almost the same as the conventional ones. As for servers, trials only using server's information are almost unsuccessful and a lot of times are needed even in the successful cases.

Finally, let us explain about the result of selection methods. As shown in Figs. 4 and 5, the ε-greedy method is superior in learning time to Boltzmann method. But, Boltzmann method is superior in accuracy of the test simulation to ε-greedy method as shown in Tables V and VI.

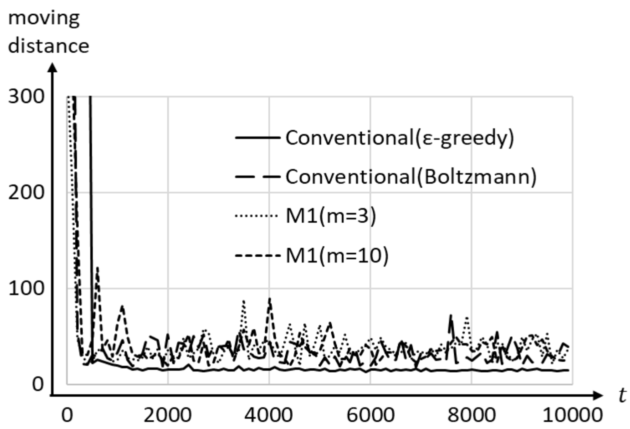


Fig. 4. The result of efficiency for M1 method of Q-learning for SMC.

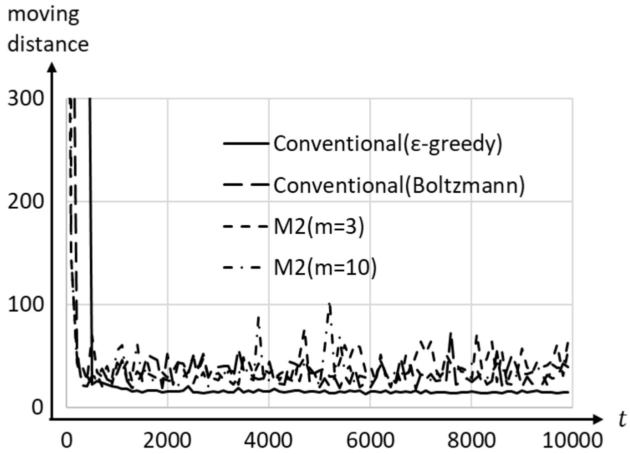


Fig. 5. The result of efficiency for M2 method of Q-learning for SMC.

V. CONCLUSION

In this paper, Q-learning algorithms in the analog model for SMC were proposed and the effectiveness of them were shown in numerical simulations. In Section 2, cloud computing system, related works on SMC and a secure data dividing mechanism used in this paper were explained. Further, Q-learning methods in the digital and analog models were introduced. In Section 3, Q-learning methods in the analog model for SMC were proposed. First, a Q-learning algorithm in the analog model was proposed using divided Q-values and the effectiveness was shown. The feature of Q-learning method for the analog model was that the action was selected as the weighted sum of plural actions. In

TABLE V
THE RESULT OF OPTIMALITY OF Q-LEARNING FOR BOLTZMANN SELECTION.

			Test 1		Test 2		Test 3		Test 4		Test 5	
			No.Suc.	M.D.	No.Suc.	M.D	No.Suc.	M.D	No.Suc.	M.D	No.Suc.	M.D
Conventional		Client	0.95	9.83	1.0	9.62	1.0	5.71	1.0	11.06	0.95	12.75
M1	$(m=3)$	Client	1.0	10.75	1.0	9.70	1.0	7.00	1.0	11.20	0.95	11.95
		Server 1∼10	0.05		0		0.13		0.10		0.05	
	$(m=10)$	Client	0.95	10.31	1.0	10.17	1.0	5.66	1.0	11.24	1.0	11.47
		Server 1∼10	0		0		0.05		0		0	
M2	$(m=3)$	Client	0.95	9.75	1.0	9.34	0.95	5.34	1.0	11.23	1.0	18.46
		Server 1∼10	0		0		0.05		0.05		0.05	
	$(m=10)$	Client	1.0	10.32	1.0	9.83	1.0	5.29	1.0	10.72	1.0	11.46
		Server 1∼10	0		0.05		0.025		0.05		0.05	

TABLE VI
THE RESULT OF OPTIMALITY OF Q-LEARNING FOR ε-GREEDY SELECTION.

			Test 1		Test 2		Test 3		Test 4		Test 5	
			No.Suc.	M.D.	No.Suc.	M.D	No.Suc.	M.D	No.Suc.	M.D	No.Suc.	M.D
Conventional		Client	0.65	8.89	0.95	11.39	0.95	5.39	0.95	10.33	0.70	19.93
M1	$(m=3)$	Client	0.75	9.80	1.0	10.09	1.0	5.88	1.0	11.97	0.80	18.07
		Server 1∼10	0.03		0		0.08		0.03		0	
	$(m=10)$	Client	0.80	9.49	0.9	11.33	1.0	5.94	0.95	11.41	0.85	20.40
		Server 1∼10	0.02		0.02		0.12		0.01		0.02	
M2	$(m=3)$	Client	0.80	9.43	0.85	11.10	0.95	5.27	1.0	10.79	0.75	27.80
		Server 1∼10	0		0		0		0		0	
	$(m=10)$	Client	0.85	10.78	0.95	11.36	1.0	5.21	0.8	10.31	0.75	11.05
		Server 1∼10	0		0		0		0		0	

the proposed algorithm, there was the possibility that some servers know secure computation. Therefore, an improved Q-learning algorithm in the analog model was proposed. It was the method with dummy updating and it seems that any server is difficult to know secure computation. In section 4, numerical simulations for a maze problem were performed to show the performance of proposed methods. In the future, we will reduce the computational complexity of the client and show the safety of algorithms for SMC in theoretical proof.

REFERENCES

[1] A. Shamir, "How to share a secret", Comm. ACM, Vol. 22, No. 11, pp. 612-613, 1979.

[2] S. Subashini, and V. Kavitha, "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. Ben-David, N. Nisan, B. Pinkas, "Fair play MP: a system for secure multi-party computation", Proceedings of the 15th ACM conference on Computer and communications security, pp.257-266, 2008.

[4] R. Cramer, I. Damgard, U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme", EUROCRYPT 2000, pp.316-334, 2000.

[5] J. Yuan, S. Yu, "Privacy Preserving Back-Propagation Neural Network Learning Made Practical with Cloud Computing", IEEE Trans. on Parallel and Distributed Systems, Vol.25, Issue 1, pp.212-221, 2013.

[6] N. Rajesh, K. Sujatha, A. A. Lawrence, "Survey on Privacy Preserving Data Mining Techniques using Recent Algorithms", International Journal of Computer Applications Vol.133, No.7, pp.30-33, 2016.

[7] 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, Sep. 2011 (in Japanese).

[8] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami, N. Shiratori, "New Privacy Preserving Back Propagation Learning for Secure Multiparty Computation", IAENG International Journal of Computer Science, Vol.43, No.3, pp.270-276, 2016.

[9] H. Miyajima, N. Shigei, S. Makino, H. Miyajima, Y. Miyanishi, S. Kitagami, N. Shiratori, "A proposal of privacy preserving reinforcement learning for secure multiparty computation", Artificial Intelligence Research, Vol.6, No.2, pp.57-68, 2017.

[10] H. Miyajima, N. Shigei, H. Miyajima, N. Shiratori, "A proposal of profit sharing method for secure multiparty computation", International Journal of Innovative Computing, Information and Control, Vol.14, No.2, pp.727-735, 2018.

[11] C. J. C. H. Watkins, P. Dayan, "Q-learning", Machine Learning 8, pp.55-68, 1992.

[12] R. S. Sutton, A. G. Barto, "Reinforcement Learning", MIT Press, 1998.

[13] C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC 2009, pp.169-178, 2009.

[14] J. Sakuma, S. Kobayashi, R. N. Wright. "Privacy-preserving reinforcement learning", In: W. W. Cohen, A. McCallum, S. T. Roweis, (eds.) ICML, vol. 307. ACM International Conference Proceeding Series, pp. 864-871, 2008.

[15] H. Miyajima, N. Shigei, H. Miyajima, N. Shiratori, "Privacy Preserving in the Analog Model for Secure Multiparty Computation," Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2018, 14-16 March, 2018, Hong Kong, pp.374-379.