Distributed Data Federation without Disclosure of User Existence

Takao Takenouchi[1,2], Takahiro Kawamura[2], and Akihiko Ohsuga[2]

[1] Knowledge Discovery Research Laboratories, NEC Corporation.
takenouchi@bu.jp.nec.com
[2] Graduate School of Information Systems, The University of Electro-Communications.

Abstract. Service providers collect user's personal information relevant to their businesses. Personal information stored by different service providers is expected to be combined to make new services. However, specific user records risk being identified from the combined personal information, and the user's sensitive information may be revealed. Also, personal information collected by a service provider must not be disclosed to other service providers because of security issues. Thus, several researchers have been investigating distributed anonymization protocols, which combine the personal information stored by the providers and sanitize it to ensure an anonymity policy with minimum disclosure. However, when providers have different sets of the users, there is a problem that the existence of users in either service provider may be revealed. This paper introduces a new notion, δ-*max-site-presence*, which indicates the probability of the existence of users being revealed in a distributed environment and a new distributed anonymization protocol for hiding the existence of users. Our evaluation results show that the proposed protocol can anonymize users in accordance with the policy of hiding their existence and user anonymity without too much information loss.

Keywords: Distributed Anonymization, Privacy Preserving Data Publishing, k-anonymity

1 Introduction

Service providers have recently started providing applications on cloud platforms, on which they collect vast amounts of users' personal information for their businesses. Personal information stored by different service providers is expected to be combined to make new services. For example, we expect a usage case in which an online video service (Provider A) and a finance company (Provider B) cooperate. In this case, Provider A has the information of the video titles rented by its customers and the times they watch them, and Provider B has the information of the customers' incomes. They then combine the three types of information and send it all to an advertising agency (Provider C) that performs segmentation analyses for targeting advertisements. In this case, Provider

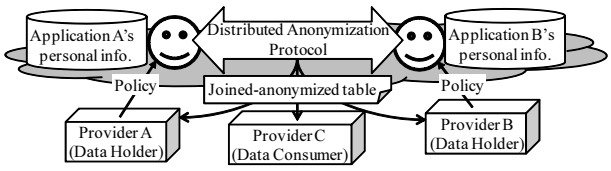

Fig. 1. Agents and distributed anonymization protocol for data federation

C will find three clusters: "daytime viewers", "high-income nighttime viewers", and "low-income nighttime viewers". However, if Provider C cannot obtain the income information, they will find only two clusters: "daytime viewers" and "nighttime viewers".

Although combining the personal information will generate new beneficial information, it may cause two problems. First, a specific user record could be identified from the combination of personal information, and the user's sensitive information may be revealed. Second, personal information stored by a service provider must not be disclosed to other service providers, because the personal information on cloud platforms is managed separately and protected by a security policy. Also, because the personal information is an asset of the service providers, the providers need to combine it with minimum disclosure.

Thus, several researchers have been investigating distributed anonymization protocols [4], which combine the personal information stored by different providers and sanitize it to ensure an anonymity policy with minimum disclosure. We expect that distributed anonymization protocols will be used for data federations between the applications of the service providers on a cloud platform. By using distributed anonymization protocols, the providers can create a joined-anonymized table, which is a combination of the tables of Providers A and B that is sanitized while ensuring the anonymity policy, and send it to other providers (Fig. 1).

However, when the service providers have different sets of users, there is a problem that the existence of a user in the data of each service provider may be revealed. The information of the existence of a user is sensitive information for that user. For example, if a user's information is stored by a financial service, then some people may guess that the user has debts.

In this paper, we consider the problem of revealing the existence of a user on each provider in distributed anonymization. This problem occurs when the attributes of the personal information are vertically partitioned, which means that the providers have different attributes of the user, and the providers have different sets of users. This paper consists of two main contributions. First, this paper propose a new notion, named δ-*max-site-presence*, which indicates the probability of the existence of the users being revealed in a distributed environment. This is an extension of an existing notion, named δ-*presence*, which indicates the probability of the existence of a user being revealed in a local environment.

Second, this paper introduces a new distributed anonymization protocol that hides the existence of a user in each database.

The rest of this paper is organized as follows. In Section 2, we discuss related works. Section 3 describes distributed anonymization and the problems of hiding the users' existence. Section 4 proposes the new notion, named *δ-max-site-presence*. In Section 5, we propose a new anonymization protocol that hides the existence. Next, we evaluate the utility of the proposed protocol in Section 6 and evaluate the security in Section 7. Then, in Section 8, we conclude the paper in Section 8.

2 Related Works

Anonymization is a method to sanitize personal information to prevent a specific user being identified. In this paper, we named a set of attributes that may uniquely identify a user *quasi-identifier*. Also, we named the attributes that users would not like to be revealed *sensitive attributes*. There is a well-known notion, named *k-anonymity* [12, 13], which indicates the anonymity of a user in a table. If a table satisfies the condition that the number of records identified by each value of the *quasi-identifier* is at least k, then the table satisfies *k-anonymity*.

Distributed anonymization is a method to join and anonymize the table stored by some providers [4]. Distributed anonymization can be categorized into two types: vertically partitioned data and horizontally partitioned data. Vertically partitioned data means that each provider has different types of attributes from the other provider. On the other hand, horizontally partitioned data means that each provider has different users from the other provider.

There are some distributed anonymization methods in vertically partitioned data [10, 14, 6]. Mohammed et al. [10, 14] used the top-down approach and some secure computation protocols [9, 15] in order to join the tables in multiple providers. The top-down approach is an algorithm to *specialize* a value *qid* of a *quasi-identifier* in the tables step by step. At the start, all *qid*s in the tables are generalized as top level, such as "*". The term *specialize* here means dividing a group identified by a *qid* into two groups on a *division point*. After the division, the set of the identifiers of the users of the two divided groups is sent to the other provider. Then, the providers continue dividing the tables as long as the tables satisfy *k-anonymity*. Finally, the providers join the divided tables in order to create the joined-anonymized table. Some secure computation protocols are used in order to calculate a heuristic function which decides the division point. By using secure computation protocols, the providers can calculate the heuristic function without sharing of the providers' local data. Jiang and Clifton [6] used the bottom-up approach to join the tables in multiple providers. In this approach, all providers anonymize their tables locally and join with each other by checking anonymity securely by using cryptographic technology.

Jurczyk and Xiong [7] proposed horizontal distributed anonymization. They mentioned a new privacy problem in which the location of a data holder is revealed by differences in providers' data types. They used some secure com-

putations and extended Mondrian[8], which is a well-known top-down approach algorithm. They also proposed a new notion, named *ℓ-site-diversity*.

Also, there is a method to hide the existence of a user in a local environment. Nergiz et al. [11] proposed *δ-presence* and the algorithm in order to satisfy this. However, this algorithm is not for a distributed environment. Therefore, we propose a new notion, named *δ-max-site-presence*, and a new distributed anonymization protocol for a distributed environment.

3 Problem of Distributed Anonymization Protocol

3.1 Distributed Anonymization

In this paper, we assumed that Providers A and B have vertically partitioned tables T_A and T_B and create joined-anonymized table T^*:

$$T_A(UID, QID_A)\,,\quad T_B(UID, QID_B, SA)\,,\quad T^*(QID_A, QID_B, SA)$$

where QID_A, QID_B is the *quasi-identifier*, SA is a *sensitive attribute*, and *UID* is a common identifier of a user in Providers A and B. T_A and T_B are joined by *UID* and anonymized into table T^*. Also, *UID* is a unique identifier of the record in T_A and T_B. In distributed anonymization, a joined-anonymized table should be created with minimum disclosure of personal information. This is because the providers in real-world businesses have difficulty fully trusting each other.

3.2 Problem of Revealing the Existence of a User

Existing distributed anonymization protocols assume that the sets of the users in Providers A and B are the same [4, 10, 7, 6]. In other words, each user who exists in Provider A also exists in Provider B. However, in the future, many providers are expected to federate with each other. Therefore, it is necessary to support the case in which the sets of the users in Providers A and B are not the same. However, if we use existing distributed anonymization protocols in these cases, the *joined-anonymized table problem* and the *UID sending problem* will occur.

In the *joined-anonymized table problem*, a provider can infer the existence of a user in the other provider by comparing the table of the first provider and the joined-anonymized table. For example, let us assume a case in which Provider A has Table 1(a) as T_A, Provider B has Table 1(b) as T_B, and the joined-anonymized table T^* is divided on *50K* of *income* like Table 1(c). In this case, the number of records where *income* $<$*50K* in T^* (Table 1(c)) is two, and also the number of records where *income* $<$*50K* in T_A is two: User 1 and User 2. Thus, Provider A can infer that Users 1 and 2 must exist in T^* (Table 1(c)). Furthermore, because T^* contains a common user who exists both on T_A and T_B, Provider A can certainly infer that Users 1 and 2 also exist in Provider B. In contrast, if T^* is divided on *60K* like Table 1(d), Provider A can infer only that two of Users 1, 2, and 3 exist in Provider B.

Table 1. Example of the joined-anonymized table problem

(a) Provider A (T_A)

UID	income
User 1	30K
User 2	40K
User 3	55K
User 5	60K
User 6	65K
User 8	70K

(b) Provider B (T_B)

UID	time	title
User 1	16:00	X movie
User 2	17:00	Y sports
User 4	17:30	X movie
User 5	16:30	Y sports
User 6	15:00	X movie
User 7	12:00	Y sports
User 9	14:00	Y sports
User 10	14:30	X movie

(c) Joined-anonymized table (T^*) with disclosure of the existence

income	time	title
<50K	16:00-	X movie
<50K	16:00-	Y sports
50K<=	-15:59	X movie
50K<=	-15:59	Y sports

(d) Joined-anonymized table (T^*) without disclosure of the existence

income	time	title
<60K	16:00-	X movie
<60K	16:00-	Y sports
60K<=	-15:59	X movie
60K<=	-15:59	Y sports

In the *UID sending problem*, the existence of a user in a provider is revealed to the other provider when *UID*s are sent. For example, if a provider sends the *UID*s that exist in the provider to the other provider, then the receiving provider can easily infer that the received *UID*s must exist in the sending provider. Also, if providers calculate common users beforehand, the existence of a user is revealed.

4 Proposed Notion: *δ-max-site-presence*

This section proposes a new notion that indicates the probability of the disclosure of the existence of a user in order to solve the *joined-anonymized table problem*. There is a existing notion, named *δ-presence*[11], which indicates the probability of the disclosure of the existence of a user by comparing two tables in a centralized environment. Thus, by applying *δ-presence* to a distributed environment, we propose a new notion, named *δ-max-site-presence*.

Let us assume that there are Tables T_1 and T_2, which is subset of records of T_1. Let $|T|$ be the number of records in Table T. Nergiz et al. [11] define the probability that a record in T_1 also exists in T_2 is $\frac{|T_2|}{|T_1|}$. We apply this definition of the probability of the existence of a user being disclosed to a distributed environment. For example, we assumed that Provider A's T_A is Table 1(a), Provider B's Table T_B is Table 1(b), and the joined-anonymized table is Table 1(d). In this case, the number of records of *income* $<60K$ in T_A is three: Users 1, 2, and 3. Moreover, the number of records of *income* $<60K$ in T^*(Table 1(d)) is two. According to the definition mentioned above, the probability that Users 1, 2, and 3 in T_A exist in T_B is $\frac{2}{3}$. Furthermore, because T^* is created from a common user in T_A and T_B, the probability that Users 1, 2, and 3 in Provider A also exist in Provider B is $\frac{2}{3}$.

We define a new notion that indicates the probability of the existence of a user in a provider as *δ-max-site-presence*.

Definition 1. *(δ-max-site-presence) Let T_A and T_B be the tables stored by Providers A and B. Note that T_A and T_B have different attributes. Also, let T^* be the joined-anonymized table, T^*_n be the table that consists of attributes of Provider*

$n \in \{A, B\}$, and $values_n$ be the set of values of any set of attributes in T_n^*. We represent $T[v]$ as the table that consists of records that are identified by value v in the table T. Then, we define that T^* satisfies δ-max-site-presence if the probability of the existence of all users in other providers being disclosed from the view of Providers A and B is less than δ as follows:

$$\frac{|T^*[v_{n,i}]|}{|T_n[v_{n,i}]|} \leq \delta \quad \forall v_{n,i} \in values_n \quad \forall n \in \{A, B\} \tag{1}$$

For example, in Table 1(d), the $values_A$ is the set of { *<60K*, *60K≤* }. In this example, let us consider *<60K*. When $v_{A,i}$ is *<60K*, the number of records of *income <60K* in Table 1(d) is two. This means $|T^*[v_{A,i}]| = 2$. Also, the number of records of *income <60K* in Table 1(a) is three. This means $|T_A[v_{A,i}]| = 3$. Also, let us consider $values_B$. The $values_B$ is the set of {(*16:00-*,*X movie*),(*16:00-*,*Y sports*),(*-15:59*,*X movie*),(*-15:59*,*Y sports*)}. When $v_{B,i}$ is (*16:00-*,*X movie*), the number of records of (*16:00-*,*X movie*) in Table 1(d) is two. This means $|T^*[v_{B,i}]| = 2$. Also, the number of records of (*16:00-*,*X movie*) in Table 1(a) is four. This means $|T_B[v_{B,i}]| = 4$. Thus, Table 1(d) satisfies $\frac{2}{3}$-*max-site-presence*.

5 Proposed Protocol: Dummy User Protocol

This section proposes a new distributed anonymization protocol named *Dummy user protocol*, which does not disclose the existence of a user. The proposed protocol is designed to create T^* that has as much detailed information as possible and satisfies the following requirements:

Requirement 1 T^* must satisfy k-*anonymity* and δ-*max-site-presence*.
Requirement 2 The disclosed information that is more detailed than T^* should be minimized.

In addition, every provider behaves semi-honestly. In this trust model, the provider who has received the messages of the protocols will analyze them to gain new knowledge, but the provider must follow the protocol. We assume that the providers partly trust each other in the real-world business. Therefore, we think this trust model is reasonable.

5.1 Dummy User Protocol

To solve the *UID sending problem* (Section 3.2), we introduce *dummy user* and propose *Dummy user protocol*. In this protocol, a provider treats the users who do not exist in the provider as if they actually did. In addition, we call the users who really exist in the provider *existing user*s. By using the dummy user, it will be difficult for a provider to distinguish whether the received *UID* is of an existing or a dummy user.

As the same as the work of Jurczyk and Xiong [7], Dummy user protocol is based on Mondrian [8], which is widely used as a top-down approach algorithm,

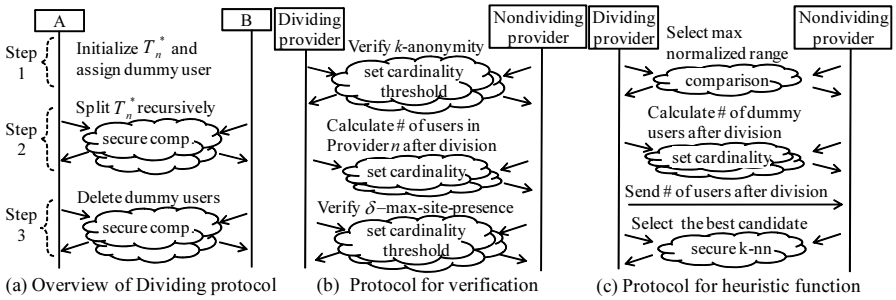

Fig. 2. Protocol sequences of Dividing protocol

and consists of two sub protocols: *Dividing protocol* and *Joining protocol*. First, Providers A and B execute Dividing protocol to create an internal anonymized table $T_n^*(n \in \{A, B\})$ locally (Fig. 2(a)). After that, Provider C executes Joining protocol to obtain the joined-anonymized table T^* by joining the T_n^* that Provider n has. Table 2 shows an example of T_n^* and T^*. The combination of income and watching time is a *quasi-identifier* and the title is a *sensitive attribute*.

Dividing Protocol: Step 1 At the first step, Providers A and B assign dummy users in their tables. We assume that each provider knows the parent population U beforehand. Here, let U_A be the set of the existing users in Provider A, U_B be the set of the existing users in Provider B, and U_O be the set of the users who do not exist in Providers A or B. Then, U can represent that $U = U_A \cup U_B \cup U_O$ $(U_O \neq \phi, U_A \cap U_B \neq \phi)$. This assumption is satisfied if Providers A and B use the same centralized authentication systems like Facebook Connect. In this case, all users registered in the authentication provider are the parent population U. Then, Provider A assigns $U - U_A$ as a dummy user of Provider A, and Provider B assigns $U - U_B$ as dummy user of Provider B.

Next, Providers A and B initialize T_n^* by generalizing all values in T_n^* into top state (Table 2(a)(b)(Initial)). The internal anonymized tables are T_A^* (*GID*, *userIDs*, QID_A), T_B^* (*GID*, *userIDs*, QID_B, $userCounts$). *GID* is a sequentially assigned identifier of a record of the T_n^*. *userIDs* is a set of *UID*s in a record of the T_n^*. $userCounts$ is the count of common users in *userIDs* for each value of the *sensitive attribute*. $userCounts$ is calculated after all dividing is finished.

Dividing Protocol: Step 2 Providers A and B execute a split function that divides T_A^* and T_B^* by communicating with each other (Fig. 3). First, Providers A and B assign the *qid* for a dummy user. We call these values *dummy values*. To make it difficult to distinguish between a dummy user and a existing user in a provider from the view of the other provider, the providers assign dummy values in accordance with the distribution of the existing users' *qid*s.

function split(U_p: a set of *UID*s including dummy user's *UID*s)
1: update the dummy values of all dummy users in U_p
2: *point* $\Leftarrow$ decide division point by the heuristic function
3: verify *k-anonymity* and *δ-max-site-presence* of the dividing on the *point*
4: **if** the verification is failed **then**
5: give up the split of U_p
6: **endif**
7: **if** the *point* is on my side **then**
8: divide my T_n^* on the *point*, and send the *UID*s to the nondividing provider
9: **else**
10: receive the *UID*s from the dividing provider, and divide my T_n^*
11: **endif**
12: $U_{hi}, U_{low} \Leftarrow$ *UID*s after the dividing, call split(U_{hi}), split(U_{low}) recursively

Fig. 3. Algorithm of Step 2 of Dividing protocol

Next, the providers decide the division point by using the heuristic function (Section 5.2). After that, the providers verify whether the T^* that is divided on the division point satisfies *k-anonymity* and *δ-max-site-presence*(Fig. 2(b)). In this verification, the providers use *cardinality* protocols and *cardinality threshold* protocols of *secure set intersection*[3]. The *cardinality* protocols can calculate the number of the intersections of two private sets, and the *cardinality threshold* protocols can compare the number of the intersections and a given number.

To verify the *k-anonymity*, the providers use a *cardinality threshold* protocol and check whether the number of the common users in the divided group is larger than k. The input parameters are two sets: the set of *UID*s of the existing users in the group after division by the *dividing provider*, who has an attribute of the dividing point, and the set of *UID*s of the existing users in the group before division by the other provider (*nondividing provider*).

To verify the *δ-max-site-presence*, the providers use a *cardinality* protocol and *cardinality threshold* protocol. By using these protocols, the providers check the following conditional formula for each Provider n ($n \in \{A, B\}$) :

$$|T^*[v_{n,i}]| \leq \delta * |T_n[v_{n,i}]| \quad \forall v_{n,i} \in values_n \tag{2}$$

When Provider n is a nondividing provider, Provider n cannot calculate $|T_n[v_{n,i}]|$ locally, because the nondividing provider does not know the groups after division. Therefore, the provider uses the *cardinality* protocol. The input parameters are the set of *UID*s of the group after division by the dividing provider and the set of the existing users' *UID*s of the group before division by the nondividing provider. Then, the nondividing provider obtains $|T_n[v_{n,i}]|$. Thus, the providers can calculate $\delta * |T_n[v_{n,i}]|$ locally. Then, the providers verify the conditional formula (2) by using the *cardinality threshold* protocol that has the same inputs as verifying *k-anonymity*.

After that, if T^* satisfies *k-anonymity* and *δ-max-site-presence*, the dividing providers divide T_n^*. Table 2(a)(b)(First division) list the results of the first division. In these divisions, the division point is *19:00* of *watching time* in *Provider*

Table 2. Internal anonymized table T_A^*,T_B^* and joined-anonymized table T^*

(a) Provider A's internal anonymized table (T_A^*)

	GID	userIDs	income
Initial	1	User 1-15	*
First division	2	User 1-10	*
	3	User 11-15	*
Second division	4	User 1-5	<40K
	5	User 6-10	40K<=
	3	User 11-15	*

(b) Provider B's internal anonymized table (T_B^*)

	GID	userIDs	time	userCounts
Initial	1	User 1-15	0:00-23:59	-
First division	2	User 1-10	0:00-18:59	-
	3	User 11-15	19:00-24:00	-
Second division	4	User 1-5	0:00-18:59	X movie: 1 Y sports: 1
	5	User 6-10	0:00-18:59	X movie: 1 Y sports: 1
	3	User 11-15	19:00-23:59	X movie: 1 Y sports: 1

(c) Joined-anonymized table (T^*)

income	time	title
<40K	0:00-18:59	X movie
<40K	0:00-18:59	Y sports
40K<=	0:00-18:59	X movie
40K<=	0:00-18:59	Y sports
*	19:00-23:59	X movie
*	19:00-23:59	Y sports

B. In this case, Provider B divides T_B^* on the division point locally and then sends the *UID*s of the groups before and after division to Provider A. Then, Provider A divides T_A^* along with the received *UID*s.

Finally, the providers call the split function recursively with the groups after division. Table 2(a)(b)(Second division) list the results of the second division. In this case, the second division point is *40K* of *income* in *Provider A*.

Dividing Protocol: Step 3 After finishing all dividing, Provider B calculates *userCounts* by using the *cardinality* protocol. Provider B obtains the number of common users for each sensitive value s of each record of T_B^*. The inputs parameters are the set of the existing users' *UID*s from Provider A and the set of *UID*s of the users who exist in and have s from Provider B. As an example of the records of Users 1-5 in Table 2(b)(Second division), the number of common users who watched *X movie* is one.

Joining Protocol Finally, Provider C, who wants to obtain a T^*, requests Providers n to obtain cleaned T_n^*. Before sending T_n^*, Provider n delete *userIDs* of T_n^*. Also, to prevent inference from sequential *GID*s, Provider A shuffles the *GID*s and sends the interaction of shuffled *GID*s to Provide B. Provide B updates the *GID*s in accordance with the interaction. After that, the providers send cleaned T_n^* to Provide C. Provider C joins the cleaned T_n^* by *GID* to acquire a joined-anonymized table T^* (Table 2(c)).

5.2 Heuristic Function for Dummy User Protocol

This section proposes a new heuristic function to decide a division point for Dummy user protocol. The heuristic function of the Mondrian algorithm [8]

selects the attribute that has the longest normalized range and selects the median of the selected attribute. In order that Dummy user protocol satisfies *δ-max-site-presence* additionally, we take an approach to extend the heuristic function of Mondrian. We consider that it is effective that the divided groups uniformly have dummy users. As an example of Table 1(c), which is the case of dividing on *50K* of *income*, the dummy users are not uniformly allocated across the divided groups. In contrast, in the case of dividing on *60K* of *income* (Table 1(d)), the dummy users are uniformly allocated.

Therefore, we introduce an entropy of dummy users (Dummy Entropy,DE) in Provider n across the divided groups:

$$DE(c,n) = -\sum_{U_i \in U_{hi}, U_{low}} \frac{|dummy(n,U_i)|}{|U_i|} log(\frac{|dummy(n,U_i)|}{|U_i|}) \tag{3}$$

where c is a candidate of dividing points that divide a group U_p into the upper group U_{hi} and the lower group U_{low}, and $dummy(n,U_i)$ is the set of *UID*s of the dummy users of Provider n in the group U_i.

By using this DE, we define the heuristic function for Dummy user protocol. First, the same as Mondrian, the function selects the attribute that has the longest normalized range. Then, for each candidate c_i of the selected attribute, the function calculates the following score S:

$$S(c_i) = \alpha(\frac{-L(c_i)}{\max_{x_j \in X}(L(x_j))}) + (1-\alpha)\frac{1}{2}\sum_{n \in \{A,B\}}(\frac{DE(c_i,n)}{\max_{x_j \in X}(DE(x_j,n))}) \tag{4}$$

$$L(c_i) = \sum_{x_j \in X} |x_j - c_i| \tag{5}$$

where $\alpha(0 \leq \alpha \leq 1)$ is the weight parameter to adjust the effect of DE, and L is the sum of the distance between the value of c_i and each x_i, which is a value of the selected attribute. Note that if we set α=1, then S will be maximized when c_i is median because the median is the value that minimizes the L. This means that the function selects the median the same way as Mondrian when α=1.

Then the function selects the c_i whose score S is the maximum as the division point. The division point is expected to divide a group into two groups across which the dummy users are allocated uniformly. As a result, the T_n^* may be divided many times.

Secure Computation The providers use three kinds of secure computations at some intermediate calculations in the function(Fig. 2(c)). First, the providers compare the normalized ranges that the providers calculate locally by *secure comparison*[15] protocol and decide a dividing provider.

Next, the providers calculate DE locally. However, nondividing provider cannot calculate $|dummy(n,U_i)|$ (the number of the dummy users in the group after division) because the provider does not know the groups after division of the candidate. Thus, the nondividing provider use the *cardinality* protocol of the *secure*

set intersection to obtain $|dummy(n, U_i)|$. Also, $|U_i|$ (the number of the *UID*s in the group after division) is necessary to calculate DE too, thus the dividing provider sends $|U_i|$ to the nondividing provider. As a result of these processes, the providers can calculate DE locally. Note that neither $|dummy(n, U_i)|$ nor $|U_i|$ contain the attribute value of the candidate of division point or *UID*s. For example, when Provider A is the provider of the candidate c_i, Provider B can obtain the number of the *UID*s of the group after dividing on candidate c_i but cannot obtain the attribute value or *UID*s of the c_i.

Finally, the providers use *secure k-nearest neighbor* protocol[16] in order to decide the c_i that maximizes the score S. The dividing provider obtains the attribute value of the division point. As mentioned above, the providers can decide the division point without disclosing the attribute values or the existence of a user.

6 Experimental Evaluation

We implemented a prototype of Dummy user protocol and evaluated it. The prototype was implemented in Java 1.6. It works in test architecture in which the providers are distributed virtually and communicate with each other virtually. We use an "Adult" data set in UCI Repository [2]. Furthermore, we divide the "Adult" data set the same way as Mohammed et al. [10].

Adult is almost 30K lines of data with 14 types of attributes and one class of data, income class (50K< or not). Also, we treat all records of "Adult" as a user set U and select the groups from the top: the users who exist in both ($U_A \cap U_B$), the users who exist only in Providers A or B (($U_A - U_A \cap U_B$), ($U_B - U_A \cap U_B$)), and the remaining users who exist in neither (U_O). In our experiment, we fixed the number of $U_A \cap U_B$ at 1200 and changed the number of $U_A - U_A \cap U_B$ and $U_B - U_A \cap U_B$ from 600 to 12000. The evaluation results are average values of the results of 20 experiments.

The same as the evaluation of δ-presence[11], we use the Discernibility Metric (DM)[1] as the indicator of our evaluation. DM measures information loss. Thus, the lower the loss, the better. By letting *qids* be the set of values of the *quasi-identifier* of T^*, DM can be calculated as follows:

$$DM = \sum_{q_i \in qids} |T^*[q_i]|^2 \tag{6}$$

For example, if 1200 records are divided into 150 8-record groups, $DM = 8^2 \times 150 = 9600$. Because data mining is the process of discovering rough patterns from data, we consider that even if a table is divided into 8-record groups, the table is useful for data mining.

6.1 Comparison with Mondrian Algorithm

δ-max-site-presence In order to evaluate Dummy user protocol, we compare Dummy user protocol and extended Mondrian, which is a simple distributed

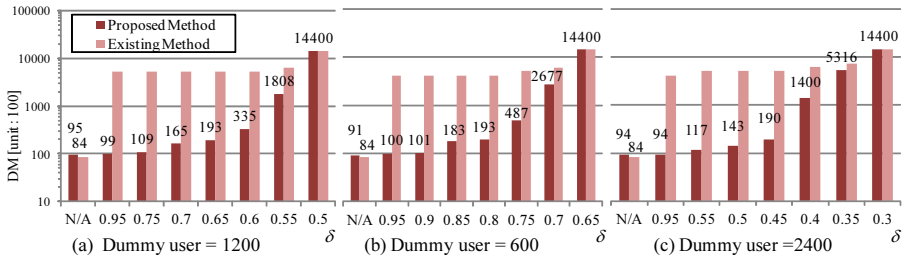

Fig. 4. Dummy user protocol vs Mondrian

anonymization protocol extended from Mondrian, in order to compare fairly. The extended Mondrian divides when satisfying not only *k-anonymity* but also *δ-max-site-presence* and outputs the table that contains the record of the common users.

First, we evaluated DM when the number of the dummy users existing only in Provider A or B is 2400, $k = 2$ and $\delta = \{0.95, \cdots, 0.5\}$. Fig. 4(a) shows DMs of the proposed method(Dummy user protocol) and existing method(extended Mondrian). Moreover, the weight α is set at 0.5 in order to make the effect of DE half.

These results show that when we make the method ignore δ-max-site-presence, the information loss of the existing method has slightly lower than that of the proposed method. On the other hand, when we set $\delta = \{0.95, \cdots, 0.5\}$ to hide the existence of a user, the information loss of the proposed method is lower than that of the existing method. When $\delta = \{0.95, \cdots, 0.75\}$, the results of proposed method are especially reasonable because the DM is almost $10,000$. This is because adding the dummy entropy into the heuristic function and the update of the dummy values enable the selection of a suitable division point that can hide the existence. As a result, the proposed method can reduce the information loss without disclosing the existence of users. However, when the δ is set at near 0.6, DM gets worse rapidly. This is because the proposed method cannot find a more effective division point.

Number of Dummy Users Next, we changed the number of dummy users and evaluated DM. Figure 4(b) and (c) show the results of the evaluation when the number of dummy users is decreased to 600 and increased to 2400.

The results show that when the number of the dummy users is increased to 2400, DM can be kept at a low value from $\delta = 0.95$ to $\delta = 0.5$. In contrast, when it is decreased to 600, DM gets worse at nearby $\delta = 0.75$. This is because the selected division point cannot satisfy *δ-max-site-presence* and the dividing is stopped. Note that, if the number of the dummy users increases to 12000, DM can be kept at a low value even if $\delta = 0.3$. According to the results above, increasing the number of the dummy users effectively hides user presence.

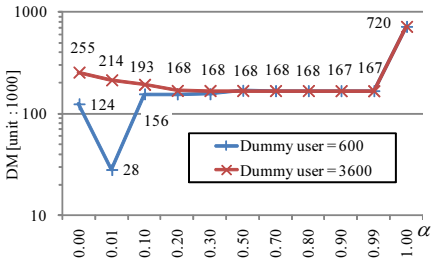

Fig. 5. DM in several α

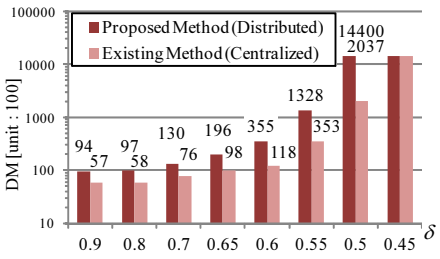

Fig. 6. Distributed vs. centralized

Weight α In order to find the best weight α, we changed α and evaluated DM. Fig. 5 shows the results of DM when the numbers of the dummy users are 600 and 3600. We set the δ at 0.75 and 0.3 along with the number of the dummy users. These results show that when there are many dummy users, it is better to set α at a larger value in order to weaken the effect of the DE. On the other hand, when there are few dummy users, it is better to set α at a smaller value. This is because when there are few dummy users, the density of the dummy users is likely to not be uniform and it's hard to satisfy δ-*max-site-presence*. Therefore, it is better to strengthen the effect of the DE. According to the above results, α should be set on the basis of the number of the dummy users.

6.2 Comparison with Centralized Algorithm

Also, we compared the proposed method for a distributed environment and the existing method for a centralized environment, named MPALM [11]. MPALM is an algorithm that is also extended from Mondrian to hide the existence of a user in an anonymized table. The big difference is that the existing method hides only one side of the existence, but the proposed method hides both sides. Thus, to evaluate them fairly, we create data in which the numbers of the dummy users who exist only in Provider {A,B} is {1200, 0} respectively.

Fig. 6 shows DM of the proposed method and the existing method in the case of varying δ. These results show that DM of the proposed method gets worse rapidly at nearby $\delta = 0.6$. On the other hand, DM of the existing method is kept lower. This is because the algorithm of the proposed method selects the division point by calculating the heuristic function. On the other hand, the algorithm of the existing method selects it by trying to divide the anonymized table and check the indicators of the users' existence. This means that existing methods can retry division many times, so the existing methods can make the information loss lower than the proposed method. If the existing algorithm is used in a distributed environment, the user's existence will be disclosed by knowing whether the table in the existing algorithm can be divided or not. However, when $\delta \geq 0.7$, the DM of the proposed method is almost the same as that of the existing method.

According to the results mentioned above, the proposed method can obtain the same utility as the existing centralized method when δ is not very small.

Finally, according to all results mentioned above, we can conclude that the proposed method can hide the existence of a user and anonymize personal information in distributed environments with little information loss.

7 Security Evaluation

In this section, we evaluate the security of the proposed protocol. If the providers cannot know more information than that we expect to be leaked, then we say that the protocol is secure. First, we will prove that Provider n ($n \in \{A, B\}$) cannot learn any more information than T_n^* and two types of the intermediate information. Next, we will show that even if these types of the information of a provider are known by the other provider, the privacy risk is low.

Our proof will show that, given T_n, T_n^* and the two types of the intermediate information, the simulator S can simulate all messages that Provider n receives during the execution of Dividing protocol in Dummy user protocol. Since the simulated messages do not contain any more information than that given, clearly Provider n cannot learn any more than the given information[7, 5]. Also, our proof will use *composition theorem*[5], because the proposed protocol uses some secure computations such as a *secure set intersection*. When a protocol F is composed of smaller secure functionalities $f_1 \dots f_n$, the composition theorem states that if the protocol F in *hybrid model* where the $f_1 \dots f_n$ are replaced with protocols that use a trusted third party (TTP) is secure, then the protocol F is secure. In our proof, we show that the simulator S can simulate the messages of Dividing protocol in the hybrid model. After that, we show that Dividing protocol is secure by using composition theorem.

Theorem 1. *Provider $n \in \{A, B\}$ cannot learn any more information than T_n^* and Intermediates 1 and 2 from the messages of Dividing protocol of Dummy user protocol.*

- ***Intermediate 1****: For the candidates of the other provider of Provider n, the number of the UIDs and the dummy users in the group after division (Note that UIDs are not revealed)*
- ***Intermediate 2****: For the canceled division points, the provider of the division point, the attribute value of the division point (only if Provider n is the dividing provider), the number of the existing users in the group after division in the nondividing provider (only if Provider n is the nondividing provider), and the reason for the cancellation (k-anonymity or δ-max-site-presence)*

Proof. We will show that simulator S_n can simulate the message received by Provider $n \in \{A, B\}$ from T_n, T_n^* and Intermediates 1 and 2. First, we show the messages received by Provider A can be simulated from T_A, T_A^* and Intermediates 1 and 2. S_A analyzes the dividing sequence by using *GID*, which is assigned sequentially, in T_A^*. (e.g. in Table 2(a)(Second division) group $GID = 2$

is divided into the groups $GID = 4$ and $GID = 5$.) Also, by comparing the two divided records, S_A can infer the dividing provider and *UID*s of the group after division. In addition, if the dividing provider is Provider A, S_A can easily infer the attribute value of the division point. (e.g. in Table 2(a)(Second division) Users 1-10 are divided on *40K* of *income* in *Provider A* into Users 1-5 and Users 6-10.) We call the result of this analysis an *analyzed sequence*.

Then, S_A starts the simulation by using the analyzed sequence. In Dividing protocol, the communication is performed in Steps 2 and 3 (Fig. 2(a)). In Step 2, the communication is performed during the calculation of the heuristic function(Fig. 2(c)), the verification(Fig. 2(b)), and sending of the *UID*s. First, S_A simulates the message at the first division. The messages of the calculation the heuristic function are three: (1) the provider of the dividing point (both providers received), (2) the number of the *UID*s and the dummy users in the group after division of the candidate (the nondividing provider received), and (3) the attribute value of the division point (the dividing provider received). S_A learns (1) and (3) from the analyzed sequence, and (2) from Intermediate 1. Thus, S_A can simulate the messages that Provider A receives. After that, S_A learns whether the division continues or not by checking the analyzed sequence. If the division continues, then S_A simulates the message of the verification as correct. Also, if the division is on Provider B, S_A simulates the sending of the *UID*s by using the analyzed sequence. After that, S_A performs the above function recursively in terms of the divided groups.

If the division does not continue, then S_A simulates the message of the validation as correct and simulates the messages of the calculation of the heuristic function the same way as described above by using Intermediates 1 and 2. Intermediate 2 contains all intermediate information of the cancelled division. After that, S_A simulates the message of the validation as incorrect by using Intermediate 2. As mentioned above, S_A performs these processes recursively, and the messages can be simulated.

Next, we show the message received by Provider B can be simulated from T_B, T_B^*, and Intermediates 1 and 2. In Step 2, S_B can simulate the same way as described above. In Step 3, Provider B receives $userCount$. Because $userCount$ is contained in T_B^*, S_B can simulate it.

As mentioned above, S_n can simulate all messages of Dividing protocol in the hybrid model. Also, by using composition theorem, if Dividing protocol is secure in the hybrid model, then the protocol is secure. Therefore, Provider $n \in \{A, B\}$ cannot learn any more information than T_n^* and Intermediates 1 and 2 from the message of Dividing protocol of Dummy user protocol. □

Because T_n^* has no attribute value of the provider other than Provider n, T_n^* is not very sensitive. Also, Intermediates 1 and 2 are statistic information, which does not include *UID*. Thus, it is difficult to reveal the *sensitive attributes* or the existence of a user from the T_n^* and Intermediates 1 and 2. Furthermore, the T_n^* and Intermediates 1 and 2 are known by Providers A and B but not C. We assumed there is some kind of contract between Providers A and B in the real-world business. Therefore, we consider that the privacy risk is low.

8 Conclusion and Future Works

We showed that there is a problem of the existence of a user in the data of different service providers being revealed when the providers have different sets of users. To solve the problem, we proposed *δ-max-site-presence*, which indicates the probability of the existence of a user being revealed in distributed environment, and Dummy user protocol. Our evaluation results show that the proposed protocol can anonymize users in accordance with the policy of hiding users' existence and user anonymity with little information loss.

In the future, we plan to calculate computation and communication costs. Also, we hope to evaluate the proposed protocol by using the real data. In addition, we plan to improve the heuristic function to make it more efficient and extend the protocol to support multiple sites.

References

1. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proc. ICDE'05. pp. 217–228. IEEE (2005)
2. Blake, C.L., Merz, C.J.: Uci repository of machine learning databases. (1998), `http://archive.ics.uci.edu/ml/`
3. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Proc. EUROCRYPT'04. pp. 1–19. Springer-Verlag (2004)
4. Fung, B., Wang, K., Fu, A., Yu, P.: Privacy-Preserving Data Publishing: Concepts and Techniques, chap. 11–12. CRC Press (2010)
5. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press (2004)
6. Jiang, W., Clifton, C.: Privacy-preserving distributed k-anonymity. In: Proc. DBSec'05. pp. 166–177. Springer (2005)
7. Jurczyk, P., Xiong, L.: Distributed anonymization: Achieving privacy for both data subjects and data providers. In: Proc. DBSec'09. pp. 191–207. Springer (2009)
8. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proc. ICDE'06. p. 25. IEEE (2006)
9. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality 1, 59–98 (2009)
10. Mohammed, N., Fung, B.C.M., Wang, K., Hung, P.C.K.: Privacy-preserving data mashup. In: Proc. EDBT'09. pp. 228–239. ACM (2009)
11. Nergiz, M.E., Atzori, M., Clifton, C.: Hiding the presence of individuals from shared databases. In: Proc. SIGMOD'07. pp. 665–676. ACM (2007)
12. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13, 1010–1027 (2001)
13. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10, 557–570 (2002)
14. Wang, K., Fung, B.C.M., Dong, G.: Integrating private databases for data analysis. In: Proc. of the 2005 IEEE International Conference on Intelligence and Security Informatics (ISI). vol. 3495, pp. 171–182 (2005)
15. Yao, A.C.: Protocols for secure computations. In: Proc. SFCS'82. pp. 160–164. IEEE Computer Society (1982)
16. Zhan, J., Chang, L., Matwin, S.: Privacy preserving k-nearest neighbor classification. International Journal of Network Security (2005)