The VLDB Journal (2011) 20:567–588
DOI 10.1007/s00778-010-0214-6

REGULAR PAPER

Anonymity meets game theory: secure data integration with malicious participants

Noman Mohammed · Benjamin C. M. Fung · Mourad Debbabi

Received: 22 November 2009 / Revised: 12 November 2010 / Accepted: 10 December 2010 / Published online: 29 December 2010

Abstract Data integration methods enable different data providers to flexibly integrate their expertise and deliver highly customizable services to their customers. Nonetheless, combining data from different sources could potentially reveal person-specific sensitive information. In VLDBJ 2006, Jiang and Clifton (Very Large Data Bases J (VLDBJ) 15(4):316–333, 2006) propose a secure Distributed k-Anonymity (DkA) framework for integrating two private data tables to a k-anonymous table in which each private table is a vertical partition on the same set of records. Their proposed DkA framework is not scalable to large data sets. Moreover, DkA is limited to a two-party scenario and the parties are assumed to be semi-honest. In this paper, we propose two algorithms to securely integrate private data from multiple parties (data providers). Our first algorithm achieves the k-anonymity privacy model in a *semi-honest* adversary model. Our second algorithm employs a game-theoretic approach to thwart *malicious* participants and to ensure fair and honest participation of multiple data providers in the data integration process. Moreover, we study and resolve a real-life privacy problem in data sharing for the financial industry in Sweden. Experiments on the real-life data demonstrate that our proposed algorithms can effectively retain the essential information in anonymous data for data analysis and are scalable for anonymizing large data sets.

N. Mohammed (✉) · B. C. M. Fung · M. Debbabi
Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H3G 1M8, Canada
e-mail: no_moham@ciise.concordia.ca

B. C. M. Fung
e-mail: fung@ciise.concordia.ca

M. Debbabi
e-mail: debbabi@ciise.concordia.ca

Keywords k-anonymity · Secure data integration · Privacy · Classification

1 Introduction

In the contemporary business environment, data sharing is an essential requirement for making better decisions and providing high-quality services. Often, multiple service providers need to collaborate and integrate their data and expertise to deliver highly customizable services to their customers. While data sharing can help their clients obtain the required information or explore new knowledge, it can also be misused by adversaries to reveal sensitive information that was not available before the data integration. In this paper, we study the privacy threats caused by data sharing and propose two algorithms to securely integrate person-specific sensitive data from multiple data providers, whereby the integrated data still retain the essential information for supporting general data exploration or a specific data mining task, such as classification analysis. The following *real-life* scenario illustrates the need for simultaneous information sharing and privacy preservation in the financial industry.

This research problem was discovered in a collaborative project with a financial industry, which is a provider of unsecured loans in Sweden. We generalize their problem as follows: A loan company A and a bank B observe different sets of attributes about the same set of individuals identified by the common identifier attribute (ID), e.g., $T_A(ID, Age, Balance)$ and $T_B(ID, Job, Salary)$. These companies want to integrate their data to support better decision making such as loan or credit limit approval, which is basically a data mining task on classification analysis. In addition to companies A and B, their partnered credit card

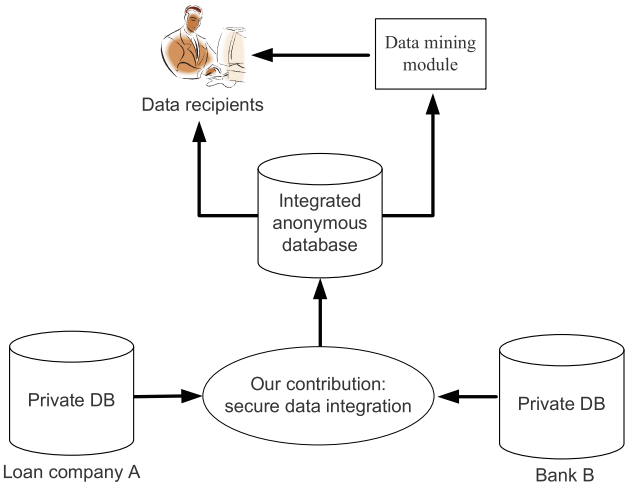


Fig. 1 Data flow of secure data integration model

Table 1 Raw tables

Shared		Party A		Party B		
ID	Class	Sex	...	Job	Salary (K)	...
1–3	0Y3N	Male		Janitor	30	
4–7	0Y4N	Male		Mover	32	
8–12	2Y3N	Male		Carpenter	35	
13–16	3Y1N	Female		Technician	37	
17–22	4Y2N	Female		Manager	42	
23–25	3Y0N	Female		Manager	44	
26–28	3Y0N	Male		Accountant	44	
29–31	3Y0N	Female		Accountant	44	
32–33	2Y0N	Male		Lawyer	44	
34	1Y0N	Female		Lawyer	44	

company C also has access to the integrated data, so all three companies A, B, and C are data recipients of the final integrated data. Figure 1 illustrates the data flow model of secure data integration generalized from the project. Companies A and B have two privacy concerns. First, simply joining T_A and T_B would reveal the sensitive information to the other party. Second, even if T_A and T_B individually do not contain person-specific or sensitive information, the integrated data can increase the possibility of identifying the record of an individual. The next example illustrates this point.

Example 1 Consider the data in Table 1 and taxonomy trees in Fig. 2 (ignore the dashed curve for now). Party A (the loan company) and Party B (the bank) own

$T_A(ID, Sex, \ldots, Class)$ and
$T_B(ID, Job, Salary, \ldots, Class)$,

respectively. Each row represents one or more raw records, and $Class$ contains the distribution of class labels Y and N, representing whether or not the loan has been approved. For example, the third row has five records of $\langle Male, Carpenter, 35\,\text{K}\rangle$, of which two records have class Y and three records have class N. Both parties want to integrate their data and use the integrated data to build a classifier on the $Class$ attribute. After integrating the two tables (by matching the ID field), the female lawyer on (Sex, Job) becomes unique and therefore vulnerable to be linked to sensitive information such as $Salary$. In other words, linking attack is possible on the fields Sex and Job. To prevent such linking, we can generalize *Accountant* and *Lawyer* to *Professional* so that this individual becomes one of many female professionals. No information is lost as far as classification is concerned because $Class$ does not depend on the distinction of *Accountant* and *Lawyer*.

In this paper, we consider the following *secure data integration* problem. Given multiple private tables for the same set of records on different sets of attributes (i.e., vertically partitioned tables), we want to efficiently produce an integrated table on all attributes for release to different parties. The integrated table must satisfy both the following privacy and information requirements:

Privacy requirement The integrated table has to satisfy k-anonymity [47,50]: A data table T satisfies k-anonymity if every combination of values on QID in T is shared by at least k records in T, where the *quasi-identifier* (QID) is a set of attributes in T that could potentially identify an individual in T, and k is a user-specified threshold. k-anonymity can be satisfied by generalizing domain values into higher level concepts. In addition, at any time in the procedure of generalization, no party should learn more detailed information about other parties other than the information in the final integrated table. For example, $Accountant$ and $Lawyer$ are more detailed than $Professional$. If the final table contains $Professaional$, then Party A should not able to determine whether the one is an $Accountant$ or a $Lawyer$.

Information requirement The generalized data have to be as useful as possible to classification analysis. One frequently raised question is: Why do not the data providers employ secure multiparty computing techniques [12,13,59] and simply release the statistical data or a classifier to the data recipients? In the project with the financial industry, the data recipients, such as the credit card company, want to have access to the financial data, not statistics, from the data providers because the data recipients want to have higher flexibility to perform the required classification analysis. It is impractical to continuously request the IT departments of the data providers to produce different types of classifiers with different parameters for various research purposes.

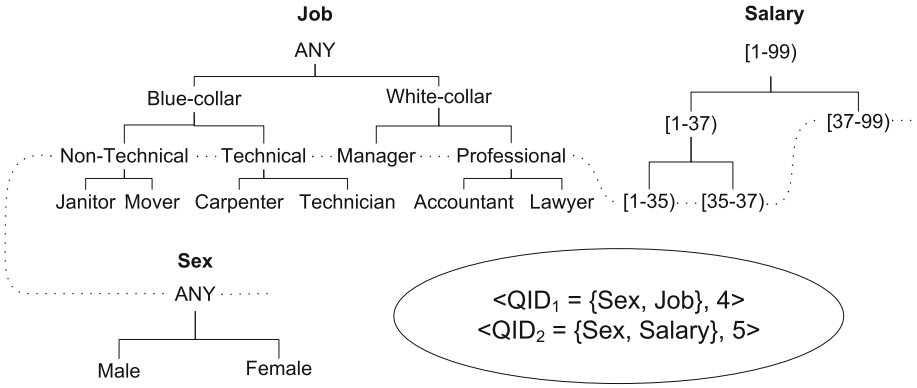


Fig. 2 Taxonomy trees and QIDs

Current techniques We briefly explain why the recently developed techniques are not applicable to proposed secure data integration problem.

- *Distributed k-Anonymity (DkA)*: Jiang and Clifton propose the DkA framework to securely integrate two distributed data tables satisfying k-anonymity requirement [24]. DkA framework requires a lot of encryption operations that are computationally expensive. The number of encryptions increases as the size of the data set increases. Therefore, DkA framework may not scale well to large real-life data sets. Furthermore, it is limited to only two parties and does not take into consideration the information requirement for classification analysis, which is the ultimate purpose of data sharing in our context. In Sect. 3, we provide an overview of DkA framework and discuss the differences.
- *Secure Multiparty Computation (SMC)*: Our problem is very different from the problem of secure multiparty computation (SMC) of classifiers [12,13,59], which allows "result sharing" (e.g., the classifier in our case) but completely prohibits data sharing. The goal of our proposed method is to allow data sharing for classification analysis in the presence of privacy concern.

There are two obvious yet incorrect approaches. The first one is "integrate-then-generalize": first integrate the local tables and then generalize the integrated table using some single table anonymization methods [5,17,22,33,38]. Unfortunately, this approach does not preserve privacy in the studied scenario because any party holding the integrated table will immediately know all private information of all parties. The second approach is "generalize-then-integrate": first generalize each table locally and then integrate the generalized tables. This approach does not work for a quasi-identifier that spans multiple tables. In Example 1, achieving k-anonymity on Sex and Job separately *does not* imply achieving k-anonymity on (Sex, Job) as a single QID.

Contributions The contributions of the paper are summarized as follows:

1. We identify a new privacy problem through a collaboration with the financial industry and generalize their requirements to formulate the secure data integration problem (Sect. 2).
2. We present two algorithms to securely integrate private data from multiple parties for two different adversary models. Our first algorithm assumes that parties are *semi-honest* (Sect. 5). In the semi-honest adversarial model, it is assumed that parties follow protocol but may try to deduce additional information. Our second algorithm further considers the presence of *malicious* parties (Sect. 6). We show that a party may deviate from the protocol for its own benefit. To overcome the malicious problem, we propose a game-theoretic approach to combine incentive compatible strategies with our anonymization algorithm.
3. We implement the proposed algorithms and evaluate the performance (Sect. 7). Experimental results on real-life data suggest that the algorithms can effectively achieve a privacy requirement without compromising data utility for classification. Moreover, the algorithms are scalable to handle large data sets and significantly outperform the DkA framework in terms of efficiency.
4. We further extend the proposed privacy-preserving methods to achieve other privacy models, such as ℓ-diversity [36], (α, k)-anonymity [56], and confidence bounding [54] (Sect. 8).

2 Problem definition

In this section, we define the anonymity requirement, state the basic requirements and assumption, and finally present the secure data integration problem for multiple parties.

2.1 Anonymity requirement

Consider a person-specific table $T(ID, D_1, D_2, \ldots, D_m, Class)$. ID is record identifier, such as SSN, which is removed before publishing the data. Each D_i is either a categorical or a continuous attribute. The $Class$ column contains class labels. Let $att(v)$ denote the attribute of a value v. The data provider wants to protect against linking an individual to a record in T through some subset of attributes called a *quasi-identifier*, or QID. A sensitive linking occurs if some value of the QID is shared by only a *small* number of records in T. This requirement is defined below.

Definition 1 (*Anonymity Requirement*) Consider p quasi-identifiers $QID_1, \ldots, QID_p$ on T. $a(qid_j)$ denotes the number of records in T that share the value qid_j on QID_j. The anonymity of QID_j, denoted by $A(QID_j)$, is the smallest $a(qid_j)$ for any value qid_j on QID_j. A table T satisfies the anonymity requirement $\{\langle QID_1, k_1\rangle, \ldots, \langle QID_p, k_p\rangle\}$ if $A(QID_j) \geq k_j$ for $1 \leq j \leq p$, where k_j is the anonymity threshold.

Definition 1 generalizes the traditional k-anonymity by allowing the data provider to specify multiple QIDs. More details on the motivation and specification of multiple QIDs can be found in [17]. Note that if QID_j is a subset of QID_i, where $i \neq j$, and if $k_j \leq k_i$, then $\langle QID_i, k_i\rangle$ *covers* $\langle QID_j, k_j\rangle$. $\langle QID_j, k_j\rangle$ is redundant and can be removed from the anonymity requirement because if a table T satisfies $\langle QID_i, k_i\rangle$, then T must also satisfy $\langle QID_j, k_j\rangle$.

Example 2 $\langle QID_1 = \{Sex, Job\}, 4\rangle$ states that every combination of qid on $\{Sex, Job\}$ in T must be shared by at least 4 records in T. In Table 1, the following qids violate this requirement: $\langle Male, Janitor\rangle$, $\langle Male, Accountant\rangle$, $\langle Female, Accountant\rangle$, $\langle Male, Lawyer\rangle$, and $\langle Female, Lawyer\rangle$. The example in Fig. 2 specifies a k-anonymity requirement with two QIDs.

2.2 Basic requirements and assumptions

We assume that there are n parties (data providers) such that each party y, where $1 \leq y \leq n$, owns a private table $T_y(ID, Attribs_y, Class)$ over the same set of records. Parties can identify the same set of records by executing a secure set intersection protocol (based on [3]) on the global unique identifiers (ID). The secure set intersection protocol of [3] uses commutation encryption. Commutative encryption has the property that when multiple parties encrypt a value successively by their keys, the result of the encryption is identical irrespective of the order of encryptions. Following, we briefly present the protocol for two parties which can be extended for n parties similarly.

Initially, both the parties encrypt the values of their global identifier and send $E_K(V)$ to the other party, where K is the secret key and V is the set of global identifier values. Each party then encrypts the received values by its own key and sends back the double-encrypted values along with the received values to the other party in the same order. For example, Party 1 receives $E_{K_2}(V_2)$ from Party 2 and sends back the pair $\langle E_{K_2}(V_2), E_{K_1}(E_{K_2}(V_2))\rangle$ to Party 2. Similarly, Party 1 receives the pair $\langle E_{K_1}(V_1), E_{K_2}(E_{K_1}(V_1))\rangle$ from Party 2. Now, both the parities can determine the common value set $V_1 \cap V_2$ by comparing the values in $E_{K_2}(E_{K_1}(V_1))$ and $E_{K_1}(E_{K_2}(V_2))$ and thus can identify the same set of records without disclosing the identifiers of the records that are not common between the parties. Note, parties also obtain an encrypted value $E_{K_1}(E_{K_2}(v))$ for each $v \in V_1 \cap V_2$ that uniquely identify the records. In the rest of the paper, we use ID values to refer these encrypted values that uniquely identify the records. These encrypted ID values are exchanged (see Sect. 5) to facilitate the anonymization process but removed (along with the real ID values) before publishing the data to third parties.

We assume that parties hold mutually exclusive set of attributes. That is, $Attribs_y \cap Attribs_z = \emptyset$ for any $1 \leq y, z \leq n$ (see Sect. 8 for further discussion on mutually exclusive set of attributes). ID and $Class$ are shared attributes among all parties.

Finally, we require that a *taxonomy tree* is specified for each categorical attribute in $\cup QID_j$. A leaf node represents a domain value, and a parent node represents a less specific value. For a continuous attribute in $\cup QID_j$, a taxonomy tree can be grown at runtime, where each node represents an interval, and each non-leaf node has two child nodes representing some optimal binary split of the parent interval that maximizes the information gain on the $Class$ attribute. Figure 2 shows a dynamically grown taxonomy tree for *Salary*.

2.3 Secure data integration

The secure data integration problem requires at least two parties, where parties agree to release "minimal information" to form an integrated table T for conducting a joint classification analysis. The notion of minimal information is specified by the *joint anonymity requirement* $\{\langle QID_1, k_1\rangle, \ldots, \langle QID_p, k_p\rangle\}$ on the integrated table. QID_j is *local* if it contains attributes from only one party, and *global* otherwise.

Definition 2 (*Secure Data Integration*) Given multiple private tables $T_1, \ldots, T_n$, a joint anonymity requirement $\{\langle QID_1, k_1\rangle, \ldots, \langle QID_p, k_p\rangle\}$, and a taxonomy tree for each categorical attribute in $\cup QID_j$, the problem of *secure data integration* is to efficiently produce a generalized integrated table T such that (1) T satisfies the joint anonymity require-

ment, (2) T contains as much information as possible for classification, and (3) each party learns nothing about the other party that is more specific than the information in the final generalized integrated table T.

For example, if a record in the final T has values $Female$ and *Professional* on Sex and Job, and if Party A learns that $Professional$ in the record comes from $Lawyer$, then condition (3) is violated. Our privacy model ensures the anonymity in the final integrated table as well as in any intermediate table during integration.

We consider two different adversarial models. In the first model, we assume that the data providers are semi-honest, meaning that they will follow the protocol but may attempt to derive additional information from the received data. This is the common security definition adopt in the SMC literature [24], and it is realistic in our problem scenario since different organizations are collaborating to share their data securely for mutual benefits. Hence, it is reasonable to assume that parties will not deviate from the defined protocol. However, they may be curious to learn additional information from the messages they received during the protocol execution. The algorithm presented in Sect. 5 ensures secure data integration for semi-honest adversary model.

In the second model, we consider that parties may deviate from the protocol for their own benefit. For example, Party A may not want to share its own data to form the integrated anonymous table if the protocol allows it to do so. Thus, parties might misbehave by acting selfishly and avoid sharing their own data while receiving others' data from the final integrated table. In Sect. 6, we elaborate how this attack can take place and propose a secure data integration algorithm for the malicious adversary model.

3 Previous approach: D*k*A

D*k*A (Distributed *k*-Anonymity) is a secure distributed framework for two parties to integrate their data satisfying *k*-anonymity privacy model [24]. The framework does not propose a specific anonymization algorithm, but can be easily incorporated with a *k*-anonymization algorithm to compute a global *k*-anonymous solution between two parties. D*k*A works in three steps: (1) Producing local *k*-anonymous data by the individual parties. (2) Verifying whether or not joining the local anonymous data ensures global *k*-anonymity. (3) Joining the local anonymous data using a global unique identifier. We elaborate these steps with an example followed by a discussion.

Consider the data in Table 2a and taxonomy trees in Fig. 2 (ignore the dashed curve for now). Party A and Party B own $T_A(ID, Job, Sex)$ and $T_B(ID, Salary)$, respectively. ID is a unique global identifier such as SSN.

Table 2 An example of D*k*A framework [24]

ID	Party A		Party B
	Job	Sex	Salary
(a)			
1	Janitor	Male	36 K
2	Mover	Male	33 K
3	Carpenter	Male	30 K
4	Technician	Male	36 K
(b)			
1	Non-Technical	Male	[35–37)
2	Non-Technical	Male	[1–35)
3	Technical	Male	[1–35)
4	Technical	Male	[35–37)
(c)			
1	Blue-collar	Male	[1–37)
2	Blue-collar	Male	[1–37)
3	Blue-collar	Male	[1–37)
4	Blue-collar	Male	[1–37)

Suppose the parties want to achieve 2-anonymity with $QID = \{Job, Sex, Salary\}$. First, both the parties generalize their data locally to satisfy 2-anonymity as shown in Table 2b. Based on the local *k*-anonymous data, each party partitions the IDs into disjoint subsets. Records corresponding to the ID from the same subset have the same value with respect to the QID. Let γ_c^i be the set of subsets, where $i \in \{A, B\}$ represents the party and $c \in integer$ represents the protocol round. From Table 2b, we get $\gamma_1^A = \{\{1, 2\}, \{3, 4\}\}$ and $\gamma_1^B = \{\{1, 4\}, \{2, 3\}\}$.

Then, the two parties compare γ_1^A and γ_1^B to check the global *k*-anonymity requirement. If there are no subsets p and q such that $0 < |\gamma_1^A[p] \cap \gamma_1^B[q]| < k$, then it is safe to join local data sets to generate the global *k*-anonymous data; otherwise, the parties further generalize their data until the condition satisfies. For example, γ_1^A and γ_1^B do not satisfy the condition because $|\{1, 2\} \in \gamma_1^A \cap \{1, 4\} \in \gamma_1^B| = 1 < k$, where $k = 2$. Accordingly, both the parties generalize their data one more step as shown in Table 2c and compare the new γ_c^is, $\gamma_2^A = \{\{1, 2, 3, 4\}\}$ and $\gamma_2^B = \{\{1, 2, 3, 4\}\}$. Here, we do not elaborate how the parties generalize their data. Any *k*-anonymization algorithm can be used to generate local *k*-anonymous data. In [24], Datafly [48,49] algorithm is used with D*k*A framework.

Finally, T_A and T_B are joined using the ID attribute. A solution based on commutative encryption is used to join the data without revealing the actual ID values. The encrypted unique ID values are used for the join process but removed from the integrated *k*-anonymous data before publishing. This step is similar to our secure set intersection protocol on the global identifier to identify the same set of records (See Sect. 2.2).

One of the core elements of DkA framework is a secure mechanism that allows parties to compare the γ_c^is. The mechanism enables parties to know only the result of the anonymity test. Parties do not learn the cardinality of the intersection of the failed subsets. This secure mechanism is based on secure set intersection (SSI) protocol and is quite computationally expensive. The number of encryption needed in each round is bounded by $O(|T|^2)$, where $|T|$ is the maximum value in the domain of ID. For the above example, $|T| = 4$.

DkA assumes that there is a unique global identifier (e.g. SSN), and data are vertically partitioned between two parties for the same set of records. Thus, each party knows the ID values of the shared records, though the parties never exchange the ID values directly. For example, to reduce the computational cost, a direct comparison of γ_c^is is not possible since parties should not learn the size of intersection that is smaller than k. Therefore, parties cannot share ID values directly.

DkA framework has a number of limitations making it inapplicable to the problem studied in this paper. First, the framework is not scalable to large data sets. Jiang and Clifton [24] report that DkA takes approximately 12.53 days to anonymize the *de facto* benchmark *Adult* [42] data set while our proposed technique takes less than 20 s to anonymize the same data set running on a slower machine. The scalability issue is further studied in Sect. 7.3. Second, DkA is limited to only two parties, while the proposed technique presented in this paper is applicable for multiple parties. Finally, DkA assumes that the parties are semi-honest, while this paper targets both semi-honest and malicious parties.

4 Anonymization technique

We can anonymize a single table T by a sequence of specializations starting from the topmost general state in which each attribute has the topmost value of its taxonomy tree. A *specialization*, written $v \rightarrow child(v)$, where $child(v)$ denotes the set of child values of v, replaces the parent value v with the child value that generalizes the domain value in a record. For example, *White-collar* $\rightarrow$ {*Manager*, *Professional*} replaces all instances of *White-collar* in a generalized table to either *Manager* or *Professional* depending on the raw value of Job in each record. A specialization is *valid* if the specialization results in a table that satisfies the anonymity requirement after the specialization. A specialization is *beneficial* if more than one class are involved in the records containing v. If not, then that specialization does not provide any helpful information for classification. Thus, a specialization is performed only if it is both valid and beneficial.

The specialization process can be viewed as pushing the "cut" of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute D_i, denoted buy Cut_i, contains exactly one value on each root-to-leaf path. A *solution cut* is $\cup Cut_i$ such that the generalized T represented by $\cup Cut_i$ satisfies the given anonymity requirement [17]. The specialization process starts from the topmost solution cut and pushes down the solution cut iteratively by specializing some value in the current solution cut until violating the anonymity requirement. Figure 2 shows a solution cut indicated by the dashed curve representing the anonymous Table 3. Each specialization tends to increase information because records are more distinguishable by specific values. One core step is computing the $Score$, which measures the "goodness" of a specialization on a value v with respect to the information requirement of classification analysis. Our selection criterion favors a specialization $v \rightarrow child(v)$ that has the maximum gain ratio [46]:

Table 3 Anonymous tables

Shared		Party A		Party B		
ID	Class	Sex	...	Job	Salary	...
1–7	0Y7N	Any		Non-technical	[1–35)	
8–16	5Y4N	Any		Technical	[35–37)	
17–25	7Y2N	Any		Manager	[37–99)	
26–34	9Y0N	Any		Professional	[37–99)	

$$Score(v) = GainRatio(v) = \frac{InfoGain(v)}{SplitInfo(v)}. \quad (1)$$

InfoGain(v): Let $T[x]$ denote the set of records in T generalized to the value x. Let $freq(T[x], cls)$ denote the number of records in $T[x]$ having the class cls. Note that $|T[v]| = \sum_c |T[c]|$, where $c \in child(v)$. We have

$$InfoGain(v) = I(T[v]) - \sum_c \frac{|T[c]|}{|T[v]|} I(T[c]), \quad (2)$$

where $I(T[x])$ is the *entropy* of $T[x]$:

$$I(T[x]) = -\sum_{cls} \frac{freq(T[x], cls)}{|T[x]|} \times \log_2 \frac{freq(T[x], cls)}{|T[x]|}. \quad (3)$$

Intuitively, $I(T[x])$ measures the mix of classes for the records in $T[x]$, and $InfoGain(v)$ is the reduction in the mix by specializing v.

SplitInfo(v): $InfoGain(v)$ is biased toward attributes with many child values. An attribute with large number of child values has higher $InfoGain(v)$ than other attributes. Thus, specializing such an attribute may not be useful for classification analysis. This bias can be avoided by dividing $InfoGain(v)$ by the split information:

$$SplitInfo(v) = -\sum_c \frac{|T[c]|}{|T[v]|} \times \log_2 \frac{|T[c]|}{|T[v]|}. \quad (4)$$

When all the records have the same child value, *SplitInfo* is undefined [46]. We handle this known problem by ignoring the value of *SplitInfo* and only consider the value of *InfoGain* as the *Score* of the candidate for this special case.

The value with the highest *Score* (gain ratio) is selected for specialization. If the *Score* of a value is 0, then this value will not be chosen as long as we have other values with a better *Score*. But if there is no better choice, then a value with zero score can be specialized, given that it is valid and beneficial. *Gain ratio* has another advantage over *information gain* in this distributed scenario because the $Score(x)$ value is used to measure the contribution of each party in the anonymization process. Gain ratio thus also ensures fairness since an attribute with a high $Score(x)$ possesses more information for classification analysis. We discuss more about contribution in Sect. 6.

Example 3 The specialization *ANY_Job* refines the 34 records into 16 records for *Blue-collar* and 18 records for *White-collar*. $Score(ANY\_Job)$ is calculated as follows.

$$I(ANY\_Job) = -\frac{21}{34} \times \log_2 \frac{21}{34} - \frac{13}{34} \times \log_2 \frac{13}{34} = 0.960$$

$$I(Blue\text{-}collar) = -\frac{5}{16} \times \log_2 \frac{5}{16} - \frac{11}{16} \times \log_2 \frac{11}{16} = 0.896$$

$$I(White\text{-}collar) = -\frac{16}{18} \times \log_2 \frac{16}{18} - \frac{2}{18} \times \log_2 \frac{2}{18} = 0.503$$

$$InfoGain(ANY\_Job) = I(ANY\_Job) - \left(\frac{16}{34} \times I(Blue\text{-}collar) + \frac{18}{34} \times I(White\text{-}collar)\right) = 0.272$$

$$SplitInfo(ANY\_Job) = -\frac{16}{34} \times \log_2 \frac{16}{34} - \frac{18}{34} \times \log_2 \frac{18}{34} = 0.998$$

$$GainRatio(ANY\_Job) = \frac{0.272}{0.998} = 0.272.$$

For a continuous attribute, the specialization of an interval refers to the optimal binary split that maximizes information gain with respect to the $Class$ attribute. We use information gain, instead of gain ratio, to determine the split of an interval because every possible split creates two child values, and therefore, information gain has no bias.

Example 4 For the continuous attribute *Salary*, the topmost value is the full-range interval of domain values, [*1–99*). To determine the split point of [*1–99*), we evaluate the information gain for the five possible split points for the values 30, 32, 35, 37, 42, and 44. The following is the calculation for the split point at 37:

$$InfoGain(37) = I([1\text{–}99)) - \left(\frac{12}{34} \times I([1\text{–}37) + \frac{22}{34} \times I([37\text{–}99)\right) = 0.9597 - \left(\frac{12}{34} \times 0.6500 + \frac{22}{34} \times 0.5746\right) = 0.3584.$$

As $InfoGain(37)$ is highest, we grow the taxonomy tree for *Salary* by adding two child intervals, [*1–37*) and [*37–99*), under the interval [*1–99*).

5 Algorithm for semi-honest parties

Fung et al. [17] propose a *top-down specialization (TDS)* approach to generalize a single table T. One non-privacy-preserving approach to the problem of data integration is to first join the multiple private tables into a single table T and then generalize T to satisfy a k-anonymity requirement using TDS. Though this approach does not satisfy the privacy requirement (3) in Definition 2 (because the party that generalizes the joint table knows all the details of the other parties), the produced table does satisfy requirements (1) and (2). Therefore, it is helpful to first have an overview of TDS.

Initially, in TDS, all values are generalized to the topmost value in its taxonomy tree and Cut_i contains the topmost value for each attribute D_i. At each iteration, TDS performs the best specialization that has the highest $Score$ among the *candidates* that are valid, beneficial specializations in $\cup Cut_i$, then updates the $Score$ and validity of the affected candidates. The algorithm terminates when there is no further valid and beneficial candidate in $\cup Cut_i$. In other words, the algorithm terminates if any further specialization would lead to a violation of the anonymity requirement. An important property of TDS is that the anonymity requirement is *anti-monotone* with respect to a specialization: If it is violated before a specialization, it remains violated after the specialization because an anonymity count $a(qid)$ does not increase with respect to a specialization.

Without loss of generality, we first present our solution in a scenario of two parties ($n = 2$) in the semi-honest model. Section 5.4 describes the extension to multiple parties ($n > 2$). Consider a table T that is given by two tables T_A and T_B with a common key ID, where Party A holds T_A and Party B holds T_B. At first glance, it seems that the change from one party to two parties is trivial because the change of $Score$ due to specializing on a single attribute depends only on that attribute and the $Class$ attribute, and each party knows about

Algorithm 1 Algorithm for Semi-Honest Parties

1: initialize T_g to include one record containing top most values;
2: initialize $\cup Cut_i$ to include only top most values;
3: **while** there exists some valid candidate in $\cup Cut_i$ **do**
4: find the local candidate x of highest $Score(x)$;
5: **if** the party has valid candidate **then**
6: communicate $Score(x)$ with Party B to find the winner;
7: **else**
8: send *Not-participate*;
9: **end if**
10: **if** the winner w is local **then**
11: specialize w on T_g;
12: instruct Party B to specialize w;
13: **else**
14: wait for the instruction from Party B;
15: specialize w on T_g using the instruction;
16: **end if**
17: replace w with $child(w)$ in the local copy of $\cup Cut_i$;
18: update $Score(x)$ and validity for candidates x in $\cup Cut_i$;
19: **end while**
20: **return** T_g and $\cup Cut_i$;

$Class$ and the attributes they have. This observation is wrong because parties will not be able to determine the validity of the candidate attributes in case the *QID* spans multiple tables.

To overcome this problem, each party keeps a copy of the current $\cup Cut_i$ and generalized T, denoted by T_g, in addition to the private T_A or T_B. The nature of the top-down approach implies that T_g is more general than the final answer and, therefore, does not violate the requirement (3) in Definition 2. At each iteration, the two parties cooperate to perform the same specialization as identified in TDS by communicating certain information that satisfies the requirement (3) in Definition 2. Algorithm 1 describes the algorithm at Party A (same for Party B).

First, Party A finds the local best candidate using the specialization criterion (Eq. 1) in Sect. 4 and communicates with Party B to identify the overall global winner candidate, denoted by w. To avoid disclosing the $Score$ to each other, the secure multiparty maximum protocol [60] can be employed. Suppose the winner w is local to Party A. Party A performs $w \rightarrow child(w)$ on its copy of $\cup Cut_i$ and T_g. This means specializing each record $t \in T_g$ containing w into more specialized records, $t'_1, \ldots, t'_z$ containing the child values of $child(w)$. Similarly, Party B updates its $\cup Cut_i$ and T_g and partitions $T_B[t]$ into $T_B[t'_1], \ldots, T_B[t'_z]$. Since Party B does not have the attribute for w, Party A needs to instruct Party B how to partition these records in terms of IDs. If the winner w is local to Party B, then the role of the two parties is exchanged in this discussion.

Example 5 Consider Table 1 and the joint anonymity requirement: $\{\langle QID_1 = \{Sex, Job\}, 4\rangle, \langle QID_2 = \{Sex, Salary\}, 5\rangle\}$. Initially, $T_g = \{\langle ANY\_Sex, ANY\_Job, [1\text{–}99)\rangle\}$ and $\cup Cut_i = \{ANY\_Sex, ANY\_Job, [1\text{–}99)\}$, and all specializations in $\cup Cut_i$ are candidates. To find the candidate, Party A computes $Score(ANY\_Sex)$ and Party B computes $Score(ANY\_Job)$ and $Score([1\text{–}99))$.

Below, we describe the key steps: find the winner candidate (Lines 4–8), perform the winning specialization (Lines 10–15), and update the score and status of candidates (Line 18). For Party A, a *local attribute* refers to an attribute from T_A and a *local specialization* refers to that of a local attribute.

5.1 Find the winner candidate

Party A first finds the local candidate x of highest $Score(x)$, by making use of computed $InfoGain(x)$, then communicates with Party B to find the winner candidate. If Party A has no valid candidate, then it sends *Not-participate*. This message indicates that the party has no attribute to specialize. $Score(x)$ comes from the update done in the previous iteration or the initialization prior to the first iteration. This step does not access data records. Updating $Score(x)$ is considered in Sect. 5.3.

5.2 Perform the winner candidate

Suppose that the winner candidate w is local at Party A (otherwise, replace Party A with Party B). For each record t in T_g containing w, Party A accesses the raw records in $T_A[t]$ to tell how to specialize t. To facilitate this operation, we represent T_g by the data structure called *Taxonomy Indexed PartitionS (TIPS)*.

Definition 3 (*TIPS*) TIPS is a tree structure. Each node represents a generalized record over $\cup QID_j$. Each child node represents a specialization of the parent node on exactly one attribute. A leaf node represents a generalized record t in T_g, and the *leaf partition* containing the raw records generalized to t, i.e., $T_A[t]$. For a candidate x in $\cup Cut_i$, P_x denotes a leaf partition whose generalized record contains x and $Link_x$ links up all P_xs.

With the TIPS, we can find all raw records generalized to x by following $Link_x$ for a candidate x in $\cup Cut_i$. To ensure that each party has access only to its own raw records, a leaf partition at Party A contains only raw records from T_A and a leaf partition at Party B contains only raw records from T_B. Initially, the TIPS has only the root node representing the most generalized record and all raw records. In each iteration, the two parties cooperate to perform the specialization w by refining the leaf partitions P_w on $Link_w$ in their own TIPS.

Example 6 Continue with Example 5. Initially, TIPS has the root node representing the most generalized record $\langle ANY\_Sex, ANY\_Job, [1\text{–}99)\rangle$, $T_A[root] = T_A$, and

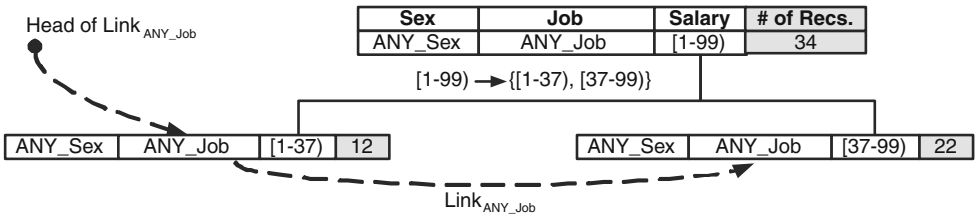


Fig. 3 The TIPS after the first specialization

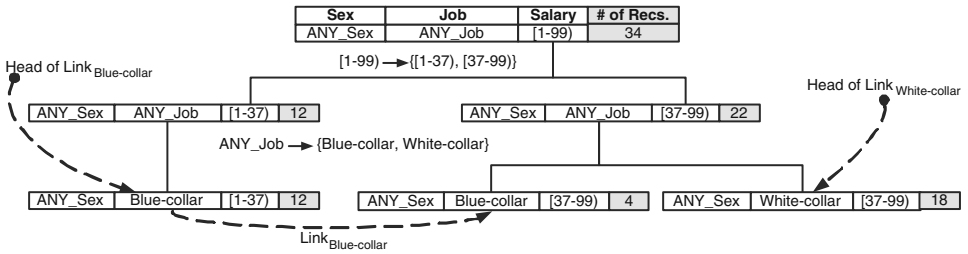


Fig. 4 The TIPS after the second specialization

$T_B[root] = T_B$. The root node is on $Link_{ANY\_Sex}$, $Link_{ANY\_Job}$, and $Link_{[1–99)}$. See the root node in Fig. 3. The shaded field contains the number of raw records generalized by a node. Suppose that the winning candidate w is

$[1–99) \rightarrow \{[1–37), [37–99)\}$ (on $Salary$).

Party B first creates two child nodes under the root node and partitions $T_B[root]$ between them. The root node is deleted from $Link_{ANY\_Sex}$, $Link_{ANY\_Job}$, and $Link_{[1–99)}$; the child nodes are added to $Link_{[1–37)}$ and $Link_{[37–99)}$, respectively, and both are added to $Link_{ANY\_Job}$ and $Link_{ANY\_Sex}$. Party B then sends the following instruction to Party A:
IDs 1–12 go to the node for *[1–37)*.
IDs 13–34 go to the node for *[37–99)*.
On receiving this instruction, Party A creates the two child nodes under the root node in its copy of TIPS and partitions $T_A[root]$ similarly. Suppose that the next winning candidate is

$ANY\_Job \rightarrow \{Blue\text{-}collar, White\text{-}collar\}$.

Similarly, the two parties cooperate to specialize each leaf node on $Link_{ANY\_Job}$, resulting in the TIPS in Fig. 4.

We summarize the operations at the two parties, assuming that the winner w is local at Party A.

Party *A* Refine each leaf partition P_w on $Link_w$ into child partitions P_c. $Link_c$ is created to link up the new P_cs for the same c. Mark c as beneficial if the records on $Link_c$ have more than one class. Also, add P_c to every $Link_x$ other than $Link_w$ to which P_w was previously linked. While scanning the records in P_w, Party A also collects the following information.

– *Instruction for Party B*. If a record in P_w is specialized to a child value c, collect the pair (id, c), where id is the ID of the record. This information will be sent to B to refine the corresponding leaf partitions there.

– *Count statistics*. The following information is collected for updating $Score$. (1) For each c in $child(w)$: $|T_A[c]|$, $|T_A[d]|$, $freq(T_A[c], cls)$, and $freq(T_A[d], cls)$, where $d \in child(c)$ and cls is a class label. Refer to Sect. 4 for these notations. $|T_A[c]|$ (similarly $|T_A[d]|$) is computed by $\sum |P_c|$ for P_c on $Link_c$. (2) For each P_c on $Link_c$: $|P_d|$, where P_d is a child partition under P_c *as if* c was specialized.

Party *B* On receiving the instruction from Party A, Party B creates child partitions P_c in its own TIPS. At Party B, P_cs contain raw records from T_B. P_cs are obtained by splitting P_w among P_cs according to the (id, c) pairs received.

We emphasize that updating TIPS is the only operation that accesses raw records. Subsequently, updating $Score(x)$ (in Sect. 5.3) makes use of the count statistics without accessing raw records anymore. The overhead of maintaining $Link_x$ is small. For each attribute in $\cup QID_j$ and each leaf partition on $Link_w$, there are at most $|child(w)|$ "relinkings." Therefore, there are at most $|\cup QID_j| \times |Link_w| \times |child(w)|$ "relinkings" for performing w. Moreover, TIPS has several useful properties. (1) Every data record appears in exactly one leaf partition. (2) Each leaf partition P_x has exactly one generalized qid_j on QID_j and contributes the count $|P_x|$ toward $a(qid_j)$. Later, we use the last property to extract $a(qid_j)$ from TIPS.

5.3 Update score and validity

This step updates $Score(x)$ and validity for candidates x in $\cup Cut_i$ to reflect the impact of the specialization on every iteration. The key to the scalability of our algorithm is updating $Score(x)$ using the count statistics maintained in Sect. 5.2 without accessing raw records again.

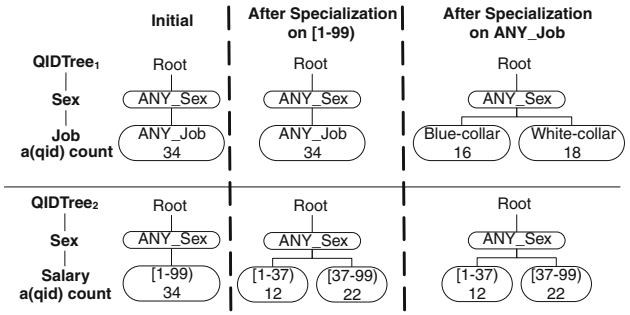


Fig. 5 The QIDTrees data structure

5.3.1 Updating $Score(x)$

An observation is that $Score(x)$ is not affected by $w \rightarrow child(w)$, except that we need to compute $InfoGain(c)$ and $SplitInfo(c)$ for each newly added value c in $child(w)$. The owner party of w can compute $Score(c)$ while collecting the count statistics for c in Sect. 5.2.

5.3.2 Validity check

A specialization $w \rightarrow child(w)$ may change the validity status of other candidates $x \in \cup Cut_i$ if $att(w)$ and $att(x)$ are contained in the same QID_j. Thus, in order to check the validity, we need to keep track of $A_x(QID_j)$, which is the smallest $a(qid_j)$ after specialization. The following $QIDTree_j$ data structure indexes $a(qid_j)$ by qid_j.

Definition 4 (*QIDTrees*) For each $QID_j = \{D_1, \ldots, D_q\}$, $QIDTree_j$ is a tree of q levels, where each level represents the generalized values for D_i. A root-to-leaf path represents an existing qid_j on QID_j in the generalized data T_g, with $a(qid_j)$ stored at the leaf node. A branch is trimmed if its $a(qid_j) = 0$. $A(QID_j)$ is the minimum $a(qid_j)$ in $QIDTree_j$.

$QIDTree_j$ is kept at a party if the party owns some attributes in QID_j. On specializing the winner w, a party updates its $QIDTree_j$ by creating the nodes for the new qid_js and computing $a(qid_j)$. We can obtain $a(qid_j)$ from the local TIPS: $a(qid_j) = \sum |P_c|$, where P_c is on $Link_c$ and qid_j is the generalized value on QID_j for P_c. Note that $|P_c|$ is given by the count statistics for w collected in Sect. 5.2.

Example 7 Continue with Example 6. Figure 5 shows the initial $QIDTree_1$ and $QIDTree_2$ for QID_1 and QID_2 on the left. On performing

$[1\text{–}99) \rightarrow \{[1\text{–}37), [37\text{–}99)\}$,

$\langle ANY\_Sex, [1\text{–}99)\rangle$ in $QIDTree_2$ is replaced with qids $\langle ANY\_Sex, [1\text{–}37)\rangle$ and $\langle ANY\_Sex, [37\text{–}99)\rangle$. $A(QID_2) = 12$. Next, on performing

$ANY\_Job \rightarrow \{Blue\text{-}collar, White\text{-}collar\}$,

$\langle ANY\_Sex, ANY\_Job\rangle$ in $QIDTree_1$ is replaced with new qids $\langle ANY\_Sex, Blue\text{-}collar\rangle$ and $\langle ANY\_Sex, White\text{-}collar\rangle$. To compute $a(vid)$ for these new qids, we need to add $|P_{Blue\text{-}collar}|$ on $Link_{Blue\text{-}collar}$ and $|P_{White\text{-}collar}|$ on $Link_{White\text{-}collar}$ (see Fig. 4):

$a(\langle ANY\_Sex, Blue - collar\rangle) = 0 + 12 + 4 = 16$ and
$a(\langle ANY\_Sex, White - collar\rangle) = 0 + 18 = 18$.

So $A_{ANY\_Job}(QID_1) = 16$.

For a local candidate x, a party needs to update $A_x(QID_j)$ in two cases. The first case is that x is a new candidate just added, i.e., $x \in child(w)$. The second case is that $att(x)$ and $att(w)$ are in the same QID_j. In both cases, the party owning x first computes $a(qid_j^x)$ for the new qid_j^xs created *as if* x was specialized. The procedure is similar to the above procedure of updating QID_j for specializing w, except that no actual update is performed on $QIDTree_j$ and TIPS. The new $a(qid_j^x)$s are then compared with $A(QID_j)$ to determine $A_x(QID_j)$. Finally, if the new $A_x(QID_j) \geq k_j$, we mark x as *valid* in $\cup Cut_i$.

5.4 Analysis

Generalization to multi-party case Algorithm 1 is extendable for multiple parties with minor changes; in Line 6, each party should communicate with all other parties for determining the winner. Similarly, in Line 12, the party holding the winner candidate should instruct the other parties, and in Line 14, a party should wait for instruction from the winner party.

Algorithmic correctness For the information requirement, our approach produces the same integrated table as the single party algorithm TDS [17] on a joint table and ensures that no party learns more detailed information about the other party other than what they agree to share. This claim follows from the fact that Algorithm 1 performs exactly the same sequence of specializations as in TDS in a distributed manner where T_A and T_B are kept locally at the sources.

For the privacy requirement, the only information revealed to each other is the $Score$ (Line 6) and the $instruction$ (Line 12) for specializing the winner candidate. The disclosure of the $Score$ does not breach privacy because $Score$ is calculated by the frequency of the class attribute. This value only indicates how good an attribute is for classification analysis and does not provide any information for a particular record. Although the $Score$ does not reveal any information for a particular record, the data providers can further enhance the protection and employ the secure max protocol [60] to securely determine the winner with the highest $Score$ without disclosing the $Score$ to other data providers. The *instruction* for specializing the winner candidate includes (id, c) pairs,

where id is the ID of the record and c is the child value of the winner candidate. This information is more general than the final integrated table that the two parties agree to share and hence does not violate privacy requirement.

Complexity analysis The cost of our proposed algorithm can be summarized as follows. Each iteration involves the following work: (1) Scan the records in $T_A[w]$ and $T_B[w]$ for updating TIPS and maintaining count statistics (Sect. 5.2). (2) Update $QIDTree_j$, $Score(x)$, and $A_x(QID_j)$ for affected candidates x (Sect. 5.3). (3) Send "instruction" to the remote party. Only the work in (1) involves accessing data records, which is in the order of $O(|T|)$; the work in (2) makes use of the count statistics without accessing data records and can be performed in constant time. This feature makes our approach scalable. Thus, for one iteration, the computation cost is $O(|T|)$. The total number of iterations is bounded by $O(\log|T|)$, resulting in the total computation cost to be $O(|T|\log|T|)$. For the communication cost (3), the instruction contains only IDs of the records in $T_A[w]$ or $T_B[w]$ and child values c in $child(w)$ and, therefore, is compact. The number of bits to be transmitted is proportional to the number of records in the database and thus in the order of $O(|T|)$. However, the instruction is sent only by the winner party. Assuming the availability of a broadcast channel, the maximum communication cost of a single party is bounded by $O(|T|\log|T|)$. If secure sum protocol is used, then there is an additional cost in every iteration. The running time of secure max protocol is bounded by $O(p(n))$, where $p(n)$ is the polynomial of n parties [60]. In other words, as the number of parties increases, the cost of the secure sum protocol also increases.

In contrast, the Distributed k-anonymity (DkA) algorithm [24] requires cryptographic technique to ensure security. The computation cost of the DkA is bounded by the number of encryption which is in the order of $O(|T|^2)$ for each iteration. The communication cost for each iteration is bounded by $O(|T|^2\log N)$, where N is the domain size from which the private–public key pairs are drawn. We will evaluate the scalability on real-life data in Sect. 7.

6 Algorithm for malicious parties

Algorithm 1 satisfies all the conditions of Definition 2 as long as parties follow the defined algorithm. However, a malicious party can deviate from the algorithm by under declaring its $Score(x)$ value (Line 6 of Algorithm 1) or sending *Not-participate* (Line 8 of Algorithm 1) in every round to avoid sharing its own data with others. For example, assume that Party A is malicious. During the anonymization process, Party A can either send a very low $Score(x)$ value or *Not-participate* to Party B for determining the global winner candidate. Hence, Party A indirectly forces Party B to specialize its attributes in every round. This can continue as long as Party B has a valid candidate. Thus, the malicious Party A successfully obtains the locally anonymized data of Party B while sharing no data of its own. Table 3 is an example of an integrated table in which Party A does not participate in the anonymization process. Moreover, this gives Party A a global data set, which is less anonymous than if it had cooperated with Party B.

We assume that a party exhibits its malicious behavior by reporting a low $Score(x)$ value or sending *Not-participate* message while it has valid candidate. However, a malicious party does not manipulate its input database to provide wrong data (Line 12 of Algorithm 1). Preventing malicious parties from sharing false data is difficult since the data are private and nonverifiable information. To prevent such malicious behavior, there can be an auditing mechanism where a trusted third party (e.g. judge) can verify the integrity of the data [2]. Further investigation is needed to thwart this kind of misbehavior without a trusted third party. In this regard, mechanism design [43] could be a potential tool to motivate parties to share their real data.

In Sect. 6.3, we provide a solution to prevent parties from deviating from the algorithm. Before that we first use rationality to understand the behavior of the participating parties and then review some basic concepts of game theory.

6.1 Rational participation

To generate the integrated anonymous table, each party specializes its own attributes, which can be considered as a contribution. The contribution can be measured by the attribute's $Score(x)$ value according to Eq. 1. Thus, the total contribution of Party A, denoted by μ_A, is the summation of all the $Score(x)$ values from its attribute specializations. We use φ_A to denote the contributions of all other parties excluding Party A. This is the ultimate value that each party wants to maximize from the integrated anonymous table. Since all the parties are rational, their actions are driven by self interest. Hence, they may want to deviate from the algorithm to maximize their φ while minimizing μ as much as possible.

An anonymous integrated table can be achieved in two ways. First, both parties ($n = 2$) can specialize their attributes. Certainly, there can be many different choices for attribute selection. Our proposed heuristic is one of them. Second, only one party can specialize its attributes (based on only local QIDs), while the other party's attributes are generalized to the topmost values. In this case, part of the table is completely locally anonymized and part of the table is completely generalized to the topmost values.

As an example, consider Parties A and B, having the same number of local attributes and being equally capable of contributing to the anonymization process. Let us assume that

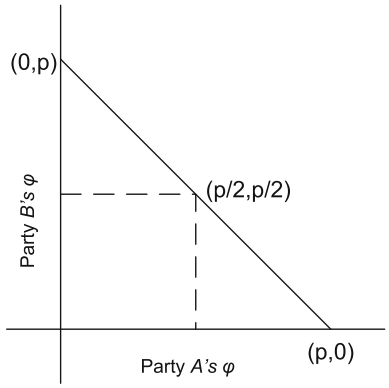


Fig. 6 Rational participation point

the integrated table cannot be more specialized after any s number of attribute specializations. Each specialization has the same $Score(x)$ value, and the sum of the $Score(x)$ value is p. Figure 6 shows the possible values of φ for both the parties. The line joining points $(0, p)$ and $(p, 0)$ shows different choices of φ values. Both the extreme points represent the participation of one party, while the points between are the different levels of contributions from the parties. Each party cannot increase its φ value without decreasing the other party's φ. Rationality suggests that the only point that can be accepted by both parties is $(p/2, p/2)$. We call it *rational participation point* where all the parties equally contribute in the anonymization process.

In reality, each party holds different attributes and some attributes are more informative than others with respect to classification analysis. Thus, all the parties are not equally capable of contributing to the anonymization process. Based on the contribution capability, parties can be divided into k different classes, where parties belonging to class 1 are the most capable and parties belonging to class k are the least capable to contribute. If there are a different number of parties from different classes, then, by extending the concept of rationality, we can conclude that the interaction between them will be dominated by the party of the least capable class. For example, if one party of class 1 and two parties of class 2 participate to form an integrated anonymous table, then the party of class 1 will behave as if it belongs to class 2. Because by contributing more than the class 2 parties, the party of class 1 will not receive any additional contribution from them.

We use game-theoretic concepts to develop a participation strategy that ensures (1) following the algorithm, parties will approach rational participation point (2) deviating from the algorithm will eventually decrease the value of φ. To make the paper self-contained, a background on game theory is provided in the following section. A more general overview of game theory can be found in [4,44].

6.2 Game theory

A game can be defined as an interaction model among players,[1] where each player has its own strategies and possible payoffs. A *strategic* game consists of a finite set of players $I = \{1, 2, \ldots, n\}$, a set of actions A_i for each player $i \in I$, and a set of outcomes O. Each player selects an action from the set of actions A_i. An action profile is the set of actions $a = \{a_1, a_2, \ldots, a_n\}$ chosen by the players, where $a_i \in A_i$. $a_{-i} = \{a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n\}$ denotes the actions of all the players except player i. Action profile determines the outcome, $o(a) \in O$, of the game. Each player has preferences over the outcomes of the game. Preferences over outcomes are represented through a utility function, u_i. The utility function of a player i can be considered as a transformation of the outcome to a real number. It is expressed as $u_i(a) : o(a_1, a_2, \ldots, a_n) \rightarrow \Re$. A player prefers outcome o_1, over outcome o_2, if $u_i(o_1) > u_i(o_2)$. A rational player always wants to maximize utility and, thus, chooses an action that will increase the utility given the preferences of the outcomes, the structure of the game, and the belief of others' actions.

In an *infinitely repeated game*, the strategic game is repeated infinitely, where players choose their actions simultaneously for every round. The utility of such an infinitely repeated game is computed by the discounted sum of the payoffs of each strategic game as follows:

$$u_i(s_i) = u_i^1(a^1) + \delta u_i^2(a^2) + \delta^2 u_i^3(a^3) + \cdots + \delta^{t-1} u_i^t(a^t), \quad (5)$$

where $0 < \delta < 1$ represents how the individual players value their future payoffs, a^t is the action profile of strategic game of round t, and s_i is the strategy of player i. A strategy s_i defines the actions of a player i for every round of a repeated game.

Game theory uses different techniques to determine the outcome of the game. These outcomes are the stable or equilibrium points of the game. The most well-known equilibrium concept is the *Nash equilibrium* [41]. It states that each player plays her best strategy to maximize the utility, given the strategies of the other players.

Definition 5 (*Nash Equilibrium*) [41] A strategy profile $s^* = \{s_1^*, s_2^*, \ldots, s_n^*\}$ is a Nash equilibrium if this strategy profile maximizes the utility of every player i. Formally,

$$\forall i \; u_i(o(s_i^*, s_{-i}^*)) \geq u_i(o(s_i', s_{-i}^*)), \quad \forall s_i' \quad (6)$$

Nash equilibrium is the point at which no player can take advantage of the other player's strategy to improve his own position.

[1] We use the word player to refer to a party.

Player A \ Player B	Cooperate	Defect
Cooperate	*(pw, (1-p)w)*	*(0,w)*
Defect	*(w, 0)*	*(0, 0)*

Fig. 7 Payoff matrix of information-sharing game

Nash equilibrium is the strategy profile from which no player has any incentive to deviate.

6.3 Participation strategy

Is it possible to devise a strategy for our secure data integration problem where following the algorithm constitutes the Nash equilibrium? This section presents such a strategy that guarantees that no party has any incentive to deviate from the algorithm since deviation does not lead to better utility. We first model our secure data integration problem as a information-sharing game and analyze the behavior of the parties. We then use game-theoretic techniques to devise a strategy where following the algorithm is the choice of rationality.

Our secure data integration problem can be modeled as a game where each player "cooperates" by following the algorithm or "defects" by deviating. Figure 7 presents the payoff matrix of our information-sharing game for two parties. In every round, only one party contributes. The party that contributes in the current round is considered as the loser and receives 0 utility because it does not gain any information from the other party. On the other hand, the party that does not contribute gets w utility and considered as the winner. Note, this is not a zero-sum game, where the loser has to lose the same amount what the winner gains. In our information-sharing game, the loser does not receive negative utility rather it receives zero utility because the gain of the other party does not harm the loser.

In Fig. 7, for the action profile $\{Defect, Defect\}$, the payoffs of both the parties are zero since they do not share information. For the action profiles $\{Defect, Cooperate\}$ and $\{Cooperate, Defect\}$, the cooperator gets 0 utility while the other party gains utility w. Finally, if both the parties cooperate by following the algorithm, then one of the parties contributes. Let p be the probability that the Party B contributes and $1-p$ be the probability that the Party A contributes. Thus, the payoffs of Party A and Party B for each round when both cooperates are pw and $(1-p)w$, respectively.

The payoff matrix for n parties can be extended similarly. If all the parties deviate, then they all receive zero utility. In all other cases, parties that defeat gain a utility of w, while the utilities of the cooperating parties depend on the probability of their contribution.

For each party i, defect is the best strategy since $\forall i \;\; u_i(o(Defect, s_{-i})) \;\geq\; u_i(o(Cooperate, s_{-i})), \forall s_{-i}$. A rational party has no incentive to cooperate since deviation from the algorithm brings more utility. Therefore, for one iteration of our information-sharing game, defect is the Nash equilibrium given the payoff matrix in Fig. 7. Since our information-sharing game continues for unknown number of iterations, the game can be modeled as an infinitely repeated game. According to Nash folk theorem [44], for any infinitely repeated game G, a strategy profile that guarantees a higher payoff than the players' minimax payoff constitutes a Nash equilibrium. Therefore, cooperation can emerge as the Nash equilibrium for our information-sharing game if cooperation ensures higher payoff than minimax payoff.

Minimax is the lowest payoff that can be forced upon any player by the other players. For our information-sharing game, the minimax payoff for each player is zero because

$$\min_{a_{-i} \in A_{-i}} \left(\max_{a_i \in A_i} u_i(a_i, a_{-i}) \right) = 0. \tag{7}$$

If all the players except player i defect, then player i receives almost zero payoff. Hence, if we devise a strategy, where each player's utility exceeds the minimax payoff, then that strategy profile constitutes Nash equilibrium. Any player i that deviates from the strategy gains only in that particular iteration but can be punished by others in the subsequent iterations by enforcing upon her the minimax payoff.

For our secure data integration problem, we device the following participation strategy that can be incorporated in Algorithm 1 to deter parties from deviating.

$$s_i = \begin{cases} Send\ Score(x) & \forall j\ \mu_i \leq \mu_j + \epsilon, \forall j \neq i \\ Send\ Not\text{-}participate & \text{otherwise} \end{cases}$$

Every party i keeps track of the total contributions $\{\mu_1, \ldots, \mu_n\}$ of every party including party i himself. In each iteration, each party decides whether or not to contribute based on these values. Here, ϵ is a small positive number. Thus, each party only cooperates by sharing its data if it has not contributed more than all the other parties. However, each party is generous in a sense that it participates if the difference is very small. Thus, the participation strategy helps parties to attain the rational participating point.

Finally, incorporating the participation strategy in Algorithm 1, we get Algorithm 2 for secure data integration for malicious parties. The new algorithm has a few differences. First, it has one additional condition in Line 5 to decide whether or not to participate. A party sends *Not-participate* when either others did not participate enough or the party himself has no valid attribute. Second, each party updates the value of μ_i (Lines 13 and 17) in every iteration. The use of the secure sum protocol does not prevent parties from

Algorithm 2 Algorithm for Malicious Parties

```
initialize T_g to include one record containing topmost values;
initialize ∪Cut_i to include only topmost values;
while there exists some valid candidate in ∪Cut_i do
  find the local candidate x of highest Score(x);
  if μ_A ≤ μ_B + ϵ, ∀B ≠ A and has valid candidate then
    communicate Score(x) with Party B to find the winner;
  else
    send Not-participate;
  end if
  if the winner w is local then
    specialize w on T_g;
    instruct Party B to specialize w;
    μ_A = μ_A + Score(x);
  else
    wait for the instruction from Party B;
    specialize w on T_g using the instruction;
    μ_B = μ_B + Score(x);
  end if
  replace w with child(w) in the local copy of ∪Cut_i;
  update Score(x) and validity for candidates x in ∪Cut_i;
end while
return T_g and ∪Cut_i;
```

calculating the contribution of the winner party because once the parties know how to specialize the winner candidate, they themselves can compute the *Score* of the winner candidate. Thus, secure sum protocol can also be employed in Algorithm2 to conceal the *Score* among the parties. Finally, if all the parties send *Not-participate* in any iteration, then the algorithm terminates.

Theorem 1 proves that a party cannot increase its utility by deviating from the participation strategy. Deviation may bring short-time benefit but eventually the total utility of a party cannot be increased. Therefore, parties have no incentive to deviate,and thus, the participation strategy profile constitutes Nash equilibrium.

Theorem 1 *Algorithm 2 guarantees that an adversary cannot increase its value of utility function by deviating from the algorithm for* $\delta^{n-1} \geq 1 - \frac{c}{w}$.

Proof We show that a party following the algorithm has a higher discounted average payoff than what she can obtain from deviating. If all parties employ Algorithm 2, then they obtain a stream of payoff, $\mathbf{U} = \langle u^1, u^2, \ldots \rangle$, where $u^t \in \{w, 0\}$. The total utility of the payoff stream can be computed by discounted sum, $S = \sum_{t=1}^{\infty} \delta^{t-1} u^t$, where $0 < \delta < 1$. A party is indifferent between $\mathbf{U}$ and any payoff stream $\mathbf{C}$ whose discounted sum is S. The discounted sum of a constant payoff stream $\mathbf{C} = \langle c, c, \ldots \rangle$ is $\frac{c}{1-\delta}$. Thus, a party is indifferent between the payoff streams if $c = (1-\delta) \sum_{t=1}^{\infty} \delta^{t-1} u^t$ and the value of c is the discounted average of the payoff stream $\mathbf{U} = \langle u^1, u^2, \ldots \rangle$.

However, if a party deviates (by not contributing) at any period $t = 1, \ldots, \infty$, then she at most obtains a payoff stream $\langle w, \ldots, w, 0, 0, \ldots \rangle$ in which $(n-1)$ w's are followed by a constant sequence of 0's. Because other parties punish her by deviating in the subsequent period after $t = n-1$. Thus, the discounted average payoff from deviation is

$$
\begin{aligned}
d &= (1-\delta)(\underbrace{w + w\delta + \cdots + w\delta^{n-2}}_{n-1} + 0 + 0 + \cdots) \\
&= (1-\delta)\left(\frac{w(1-\delta^{n-1})}{(1-\delta)} + 0\right) \\
&= w(1-\delta^{n-1})
\end{aligned}
$$

Hence, a party cannot increase its payoff by deviating if $w(1-\delta^{n-1}) \leq c$ or $\delta^{n-1} \geq 1 - \frac{c}{w}$. □

Remark The payoff of each iteration is the $Score(x)$ value of the winner attribute. Since the order of specializations for the same set of candidates has no impact for our data utility which is classification analysis, we can assume that the discount factor δ is close to 1 for all the parties. Hence, the constrain $\delta^{n-1} \geq 1 - \frac{c}{w}$ is not a problem for our information-sharing game.

6.4 Analysis

Algorithm 2 has some nice properties. First, each party requires to keep only n extra variables $\{\mu_1, \ldots, \mu_n\}$. This makes the decision algorithm scalable. Second, each party decides whether or not to participate based on the locally generated information. Thus, a party cannot be exploited by others in the decision-making process. Finally, the computational and communication costs of the algorithm remain the same as Algorithm 1.

Although the values of μ_i are not exactly the same for all parties when the algorithm terminates, the algorithm progresses by ensuring even contributions from all the parties. Since parties are unable to determine the last iteration of the algorithm, they will cooperate until the anonymization finishes.

In real life, different parties agree to share their data when they have mutual trust and benefits. However, if the parties do not want to share their data more than others, then Algorithm 2 is the appropriate solution. It is ultimately the participated parties who will decide whether to deploy Algorithm 1 or 2.

7 Experimental evaluation

We implemented the proposed algorithms in a distributed environment. Each party is running on an Intel Pentium IV 2.6 GHz PC with 1 GB RAM, connected to a LAN. The main objective of our empirical study is to evaluate the performance of our proposed algorithms in terms of data utility (the benefit of data integration) and scalability for handling

Table 4 Attributes for the *Adult* data set

Attribute	Type	Numerical range	
		# Leaves	# Levels
Age (Ag)	Continuous	17–90	
Education-num (En)	Continuous	1–16	
Final-weight (Fw)	Continuous	13492–1490400	
Relationship (Re)	Categorical	6	3
Race (Ra)	Categorical	5	3
Sex (Sx)	Categorical	2	2
Martial-status (Ms)	Categorical	7	4
Native-country (Nc)	Categorical	40	5
Education (Ed)	Categorical	16	5
Hours-per-week (Hw)	Continuous	1–99	
Capital-gain (Cg)	Continuous	0–99999	
Capital-loss (Cl)	Continuous	0–4356	
Work-class (Wc)	Categorical	8	5
Occupation (Oc)	Categorical	14	3

large data sets. We do not directly compare our methods with D*k*A [24] in terms of data utility since D*k*A measures the data utility by using the precision metric of [50], which calculates the distortion of the generalized data compared to the original data. In contrast, we measure the utility of the anonymous data by doing classification analysis, which is the information requirement of the financial industry studied in our problem. However, we do compare the scalability of D*k*A with our algorithms in Sect. 7.3.

Due to non-disclosure agreement, we cannot use the raw data of the financial industry, for the experiment, so we employ the *de facto* benchmark census data set *Adult* [42], which is also a real-life data set, to illustrate the performance of our proposed algorithms. The data set has 6 continuous attributes, 8 categorical attributes, and a binary *Class* column representing income levels ≤50 K or >50 K. Table 4 describes each attribute. After removing records with missing values, there are 30,162, and 15,060 records for the pre-split training and testing, respectively. For classification models, we use the well-known C4.5 classifier [46]. Unless stated otherwise, all 14 attributes are used for building classifiers, and the taxonomy trees for all categorical attributes are from [17].

For the same anonymity threshold k, a single QID is always more restrictive than one broken into multiple QIDs. We first show the results for a single QID. The single QID contains the top N attributes ranked by the C4.5 classifier; the top attribute is the attribute at the top of the C4.5 decision tree; then, we remove this attribute and repeat this process to determine the rank of other attributes. The top 9 attributes are Cg, Ag, Ms, En, Re, Hw, Sx, Ed, and Oc in that order. Top5, Top7, and Top9 represent the anonymity requirements in which the single QID contains the top 5, 7, and 9 attributes, respectively.

We collect several classification errors, all on the corresponding testing set. *Base error*, denoted by BE, is the error on the integrated data without generalization. *Upper bound error*, denoted by UE, is the error on the integrated data in which all attributes in the QID are generalized to the topmost ANY. This is equivalent to removing all attributes in the QID. *Integration error*, denoted by *IE*, is the error on the integrated data produced by the anonymization algorithms. We combine the training set and testing set into one set, generalize this set to satisfy a given anonymity requirement, and then build the classifier using the generalized training set. The error is measured on the generalized testing set. *Source error*, denoted by SE, is the error without data integration at all, i.e., the error of classifiers built from an individual raw private table. Each party has a SE. Thus, $SE - IE$ measures the benefit of data integration over an individual private table. $UE - IE$ measures the benefit of generalization compared to the brute removal of the attributes in the QID. $IE - BE$ measures the quality loss due to the generalization for achieving the anonymity requirement. $UE - BE$ measures the impact of the QID on classification. A larger $UE - BE$ means that the QID is more important to classification.

7.1 Results for semi-honest model

We model two private tables T_A and T_B as follows: T_A contains the first 9 attributes of Table 4, interesting to the Immigration Department, and T_B contains the remaining 5 attributes, interesting to the Taxation Department. A common key ID for joining the two tables is added to both tables. We would like to emphasize that the results of *IE* do not depend on the number of parties in the semi-honest model because the sequence of specializations performed does not depend on the decision of the participating parties.

7.1.1 Benefits of integration

Our first goal is evaluating the benefit of data integration over individual private table, measured by $SE - IE$. SE for T_A, denoted by $SE(A)$, is 17.7% and SE for T_B, denoted by $SE(B)$, is 17.9%. Figure 8 depicts the *IE* for Top5, Top7, and Top9, with the anonymity threshold k ranging from 20 to 1000.[2] For example, $IE = 14.8\%$ for Top5 for $k \leq 180$, suggesting that the benefit of integration, $SE - IE$, for each party is approximately 3%. For Top9, *IE* stays at above 17.6% when $k \geq 80$, suggesting that the benefit is less than 1%. This experiment demonstrates the benefit of data integration over a wide range of anonymity requirements. In practice, the

[2] In order to show the behavior for both small k and large k, the x-axis is not spaced linearly.

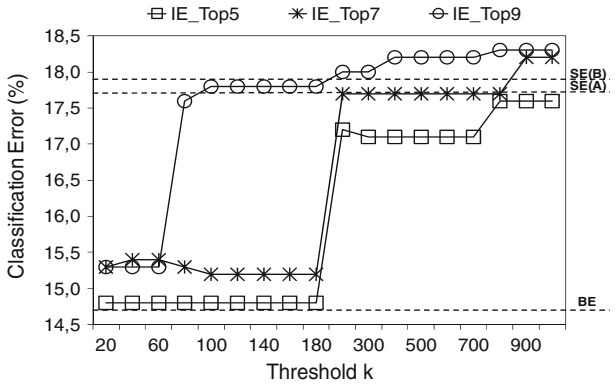


Fig. 8 Classification error for 2 parties in semi-honest model

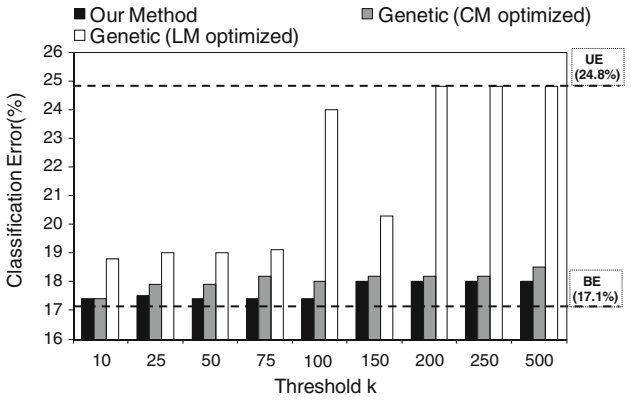


Fig. 9 Comparing with genetic algorithm

benefit is more than the accuracy consideration because our method allows the participating parties to share information for joint data analysis.

7.1.2 Impacts of generalization

Our second goal is evaluating the impact of generalization on data quality. *IE* generally increases as the anonymity threshold k or the QID size increases because the anonymity requirement becomes more stringent. $IE - BE$ measures the cost for achieving the anonymity requirement on the integrated table, which is the increase in error due to generalization. $UE - IE$ measures the benefit of our anonymization algorithm compared to the brute removal of the attributes in the QID. The ideal result is to have small $IE - BE$ (low cost) and large $UE - IE$ (high benefit). For the C4.5 classifier, $BE = 14.7\%$ and UEs are 20.4, 21.5, and 22.4% for Top5, Top7, and Top9, respectively.

Refer to Fig. 8. We use the result of Top7 to summarize the analysis. First, $IE - BE$ is less than 1% for $20 \leq k \leq 200$, and *IE* is much lower than $UE = 21.5\%$. This suggests that accurate classification and privacy protection can coexist. Typically, there are redundant classification structures in the data. Though generalization may eliminate some useful structures, other structures emerge to help the classification task. Interestingly, in some test cases, the data quality could even improve when k increases and when more generalization is performed. For example, *IE* drops as k increases from 60 to 100. This is because generalization could help eliminate noise, which in turn reduces the classification error.

7.1.3 Comparing with genetic algorithm

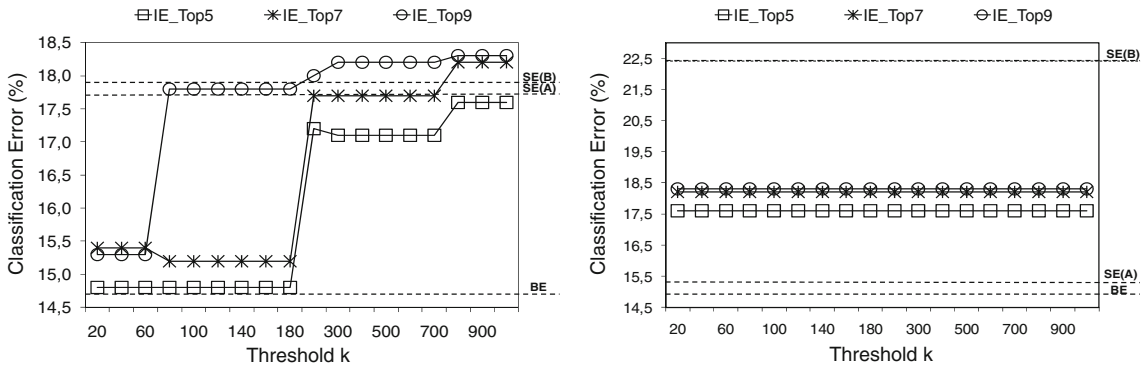
Iyengar [22] presented a genetic algorithm for generalizing a single table to achieve k-anonymity for classification analysis. One non-privacy-preserving approach is to apply this algorithm to the joint table of T_A and T_B. To ensure a fair comparison, both methods employ the same data set, the same $QID = \{Ag, Wc, Ed, Ms, Oc, Ra, Sx, Nc\}$, and the same taxonomy trees from [22]. For our method, T_A includes Ag, Ed, Ms, Ra, Sx, and Nc and T_B includes Wc and Oc. All errors in this experiment are based on the 10-fold cross validation. Results of the genetic algorithm are obtained from [22].

Figure 9 shows *IE* of Algorithm 1 and the errors for the two methods in [22]. Loss metric (LM) ignores the classification goal, and classification metric (CM) considers the classification goal. Algorithm 1 clearly generates lower error (better) than LM, suggesting that the classification quality can be improved by focusing on preserving the classification structures in the anonymous data and it is comparable to CM. However, our algorithm takes only 20 s to generalize the data. Iyengar reported that his method requires 18 h to transform this data on a Pentium III 1 GHz PC with 1 GB RAM. Of course, Iyengar's method does not address the secure integration requirement due to the joining of T_A and T_B before performing generalization.

7.2 Results for malicious model

To show the performance for our algorithm for malicious parties, we use two different attribute distributions. First, we use the same distribution as in Sect. 7.1, where T_A contains the first 9 attributes of Table 4 and T_B contains the remaining 5 attributes. This distribution gives both the parties (almost) equal capability to contribute since both the parties own top attributes. We call this distribution *equal* distribution. Figure 10a shows the classification error for Algorithm 2. Notice that with equal distribution, Algorithm 2 performs exactly like Algorithm 1. In other words, when both the parties are equal capability of contribution, the anonymization algorithm produces exactly the same integrated table as the single party method.

Next, we distribute the attributes among the parties *unequally*. We assign the first top 9 attributes to Party A and



(a) Equal distribution **(b)** Unequal distribution

Fig. 10 Classification error for 2 parties in malicious model

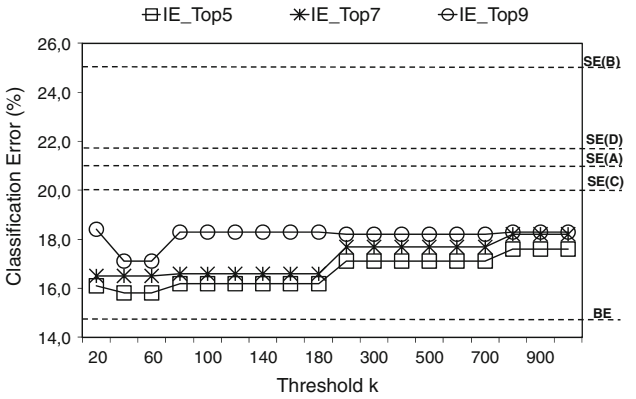


Fig. 11 Classification error for 4 parties in malicious model

the remaining 5 attributes to Party B. In this setting, Party A holds all the important attributes for classification analysis. This is also reflected in the source error, where $SE(A)$ is 15.3% and $SE(B)$ is 22.4%. Figure 10b shows the classification error for Algorithm 2 with unequal attribute distribution. The figure depicts that the *IE* for Top5, Top7, and Top9 is very high. Party A does not participate in the anonymization process because Party B is unable to contribute equally.

Finally, we evaluate the performance of Algorithm 2 for 4 parties. We divide the attributes as follows: T_A contains $\{Ag, Wc, Fw,$ and $En\}$, T_B contains $\{Ms, Re, Ra,$ and $Sx\}$, T_C contains $\{Cg, Hw,$ and $Nc\}$, and T_D contains $\{Ed, Oc,$ and $Cl\}$. The source errors are 21.2, 25.1, 20, and 21.7% for Party A, Party B, Party C, and Party D, respectively. Figure 11 depicts the *IE* for Top5, Top7, and Top9 with the anonymity threshold k ranging from 20 to 1000. *IE*s for Top5 and Top7 for $k \leq 180$ are around 16.2 and 16.6%, respectively, which is much lower than all the SEs. This suggests that all the parties gain by integrating their data, where $SE - IE$ is approximately between 3.5 to 8.5%. As the value of k increases, *IE* also increases. The highest *IE* is 18.3% for Top9 at $k \geq 800$, which is still less than the lowest $SE(C)$. The largest difference between *BE* and *SE* is around 10% due to the fact that some parties have less number of attributes and therefore higher SEs. Given that the (*SE–BE*) is large, it is more convincing to employ our proposed algorithm to conduct joint classification analysis on the integrated anonymous data. Thus, the result suggests that as the number of parties increases, the benefit of integration also increases in the scenario of equal distribution. For all the above cases, we fix the value of ϵ as 1%, meaning that parties participate in the anonymization process if the difference of the contribution is equal or less than 1%.

7.3 Scalability

For all previous experiments, our methods take at most 20 s to complete. Out of the 20 s, approximately 8 s are spent on initializing network sockets, reading data records from disk, and writing the generalized data to disk. The actual costs for data generalization and network communication are relatively low. In contrast, [24] report that their D*k*A method takes approximately 12.53 days to finish the anonymization process on an Intel Xeon 3 GHz processor with 1 GB RAM on a smaller *Adult* [42] data set with $|T| = 30,162$ for $k = 50$. Thus, our method significantly outperforms D*k*A in terms of efficiency.

Our other contribution is the scalability of handling large data sets by maintaining count statistics instead of scanning raw records. We evaluate this claim on an enlarged version of the *Adult* data set. We combine the training and testing sets, giving 45,222 records, and for each original record r in the combined set, we create $\alpha - 1$ variations of r, where $\alpha > 1$ is the blowup scale. Each variation has random values on some randomly selected attributes from $\cup QID_j$ and inherits the values of r on the remaining attributes. Together with original records, the enlarged data set has $\alpha \times 45,222$ records. For a precise comparison, the runtime reported in this section excludes the data loading time and result writing time with respect to disk, but includes the network communication time.

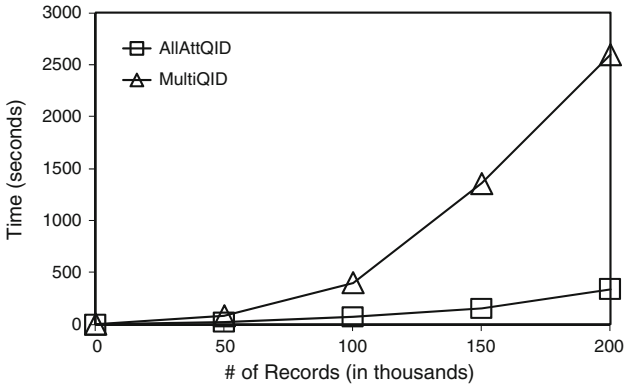


Fig. 12 Scalability ($k = 50$)

Figure 12 depicts the runtime of Algorithm 1 (Algorithm 2 achieves the same result) for 50–200 K data records based on two types of anonymity requirements. AllAttQID refers to the single QID having all 14 attributes. This is one of the most time-consuming settings because of the largest number of candidates to consider at each iteration. Moreover, the small anonymity threshold of $k = 50$ requires more iterations, and hence more runtime, to reach a solution than a larger threshold does. In this case, our algorithm takes approximately 340 s to transform 200 K records.

MultiQID refers to the average over the 30 random multi-QID anonymity requirements, generated as follows. For each requirement, we first determine the number of QIDs by uniformly and randomly drawing a number between 3 and 7, and the length of QIDs between 2 and 9. All QIDs in the same requirement have the same length and same threshold $k = 50$. For each QID, we randomly select attributes from the 14 attributes. A repeating QID is discarded. For example, a requirement of 3 QIDs and length 2 is $\{\langle\{Ag, En\}, k\rangle, \langle\{Ag, Re\}, k\rangle, \langle\{Sx, Hw\}, k\rangle\}$.

Comparing to AllAttQID, the algorithm becomes less efficient for MultiQID. There are two reasons. First, an anonymity requirement on multi-QIDs is less restrictive than the single QID anonymity requirement containing all attributes in the QIDs; therefore, the algorithm has to perform more specializations before violating the anonymity requirement. Moreover, a party needs to create one QIDTree for each related QID and maintains $a(qid)$ in QIDTrees. The time increase is roughly by a factor proportional to the number of QIDs in an anonymity requirement.

7.4 Summary

The experiments verified several claims about the anonymization algorithms. First, data integration does lead to improved data analysis. Second, the algorithms achieve a broad range of anonymity requirements without sacrificing significantly the usefulness of data to classification. The data quality of Algorithm 1 is identical or comparable to the result produced by the single party anonymization methods [17,22]. On the other hand, Algorithm 2 provides the same data quality when the parties have equal capability of contribution. Thus, it ensures fair participation and thwarts malicious behavior. Moreover, this study suggests that classification analysis has a high tolerance toward data generalization, thereby enabling data sharing across multiple data providers even in a broad range of anonymity requirements. Third, our algorithms are scalable for large data sets and different single QID anonymity requirements. They provide a practical solution to secure data integration where there is the dual need for information sharing and privacy protection.

8 Discussion

In this section, we provide answers to the following frequently raised questions: What changes do the algorithms require to accommodate other privacy models? How reasonable is it to assume that the parties hold a mutually exclusive set of attributes? What is the effect of specialization ordering on the information content of the integrated data set? Is it possible for malicious parties to collude to derive more information from others? Can the algorithms be easily modified to use local generalization or multi-dimensional generalization?

Privacy beyond k-anonymity

k-anonymity is an effective privacy model that prevents linking an individual to a record in a data table. However, if some sensitive values occur very frequently within a qid group, the attacker could still confidently infer the sensitive value of an individual by the qid value. This type of homogeneity attack was studied in [36,54]. The proposed approach in this paper can be extended to incorporate with other privacy models, such as ℓ-diversity [36], confidence bounding [54], and (α, k)-anonymity [56], to thwart homogeneity attacks.

To adopt these privacy models, we need 2 changes. First, the notion of valid specialization has to be redefined depending on the privacy model. Our anonymization algorithms guarantee that the identified solution is local optimal if the privacy measure holds the (anti-)monotonicity property with respect to specialization. ℓ-diversity [36], confidence bounding [54], and (α, k)-anonymity [56] hold such an (anti-)monotonicity property. Second, to check the validity of a candidate, the party holding the sensitive attributes has to first check the distribution of sensitive values in a qid group *before* actually performing the specialization. Suppose Party B holds a sensitive attribute S_B. Upon receiving a specialization instruction on value v from Party A, Party B has to first verify whether or not specializing v would violate the privacy requirement. If there is a violation, Party B rejects the specialization

request and both parties have to redetermine the next candidate; otherwise, the algorithm proceeds along the specialization as mentioned in Algorithms 1 and 2.

Mutually exclusive set of attributes

Our secure data integration algorithms require that parties hold mutually exclusive set of attributes. We assume that each party knows what attributes the other parties hold. Therefore, if there is a common attribute among the parties, there are two possible alternative solutions that parties can adopt before executing the secure data integration algorithms. First, if the attribute is common among all the parties, then they can exclude the common attributes from integration since parties already know the values of the attribute. Second, parties can make an agreement that outlines who will contribute the common attribute so that multiple parties do not contribute the same attribute. Hence, common attribute is not a problem since parties will communicate and agree on a setting that ensures that attributes are disjoint.

Effect of specialization ordering

Our proposed algorithms do not yield an optimal *solution cut* rather it is suboptimal. We take a greedy approach and choose an attribute with highest $Score$ in every iteration. Thus, it is possible that a different solution cut may provide better utility. However, it is important to note that maximizing the overall sum of the $Score$ for specializations in the training data does not guarantee having the lowest classification error in the testing data.

Colluding parties

Even if two or more malicious parties collude, it is not possible to derive more information from honest parties. Honest parties only participate and contribute when all the other parties equally contribute. Therefore, if a malicious party stops contributing when it should, then all the honest parties will no longer participate. However, other malicious parties may continue to participate. In fact, this does not harm but helps the honest parties.

Other anonymization techniques

Our algorithm performs the anonymization process by determining a good solution cut. The solution cut is obtained through specializing an attribute in every iteration based on its $Score$ value. In order to adopt local/multi-dimensional generalization, we need to modify the definition of cut and redesign the score function. Thus, these anonymization techniques cannot be implemented directly by our present algorithm.

Though local and multi-dimensional generalizations cause less data distortion, these techniques have a number of limitations. Local and multi-dimensional generalizations allow a value v to be independently generalized into different values. Mining classification rules from local/multi-dimensional recoded data may result in ambiguous classification rules, e.g., $White\text{-}collar \rightarrow Class\ A$ and $Lawyer \rightarrow Class\ B$ [18]. Furthermore, local and multi-dimensional recoded data cannot be directly analyzed by the off-the-shelf data analysis softwares (e.g., SPSS, Stata) due to the complex relationships among qid values [58].

9 Related work

Data privacy and security has been an active area of research in statistics, database, and security communities for the last three decades [1,18]. In this section, we briefly present various research works that are related to our problem of secure data integration.

9.1 Privacy-preserving techniques

Privacy-preserving techniques can be broadly classified into two frameworks: interactive and non-interactive. In interactive framework, users pose aggregate queries through a private mechanism and the data holder outputs macro-data (e.g., SUM, COUNT) in response. The proposed approaches can be roughly divided into two categories: restriction-based techniques and perturbation-based techniques. Restriction-based techniques ensure privacy by putting restriction on the query [7,10,16,19,37,51]. In the response to a query, the system determines whether or not the answer can be safely delivered without inference and thus controls the amount of information to be released. In perturbation-based approach, the system first computes the correct result and outputs a perturbed version of the result by adding noise [6,11,15]. All these works prohibit data publishing, which is the basic requirement of our problem. In non-interactive framework, the data provider publishes an anonymous/sanitized version of the data satisfying a privacy model. The privacy mechanism of this paper is based on non-interactive framework.

Different privacy models have been proposed to prevent an adversary from linking an individual with a sensitive attribute. Traditional k-anonymity [47,50], ℓ-diversity [36], and confidence bounding [54] are based on a predefined set of QID attributes. (α, k)-anonymity [56] requires every qid group to satisfy both k-anonymity and confidence bounding. t-closeness [34] requires the distribution of a sensitive attribute in any group to be close to the distribution of the attribute in the overall table. Xiao and Tao [57] propose the notion of *personalized privacy* to allow each record owner to specify her own privacy level. This model assumes that a

sensitive attribute has a taxonomy tree, and each record owner specifies a guarding node in the taxonomy tree. Dwork [14] proposes a privacy model called *differential privacy*, which ensures that the removal or addition of a single record does not significantly affect the overall privacy of the database. Differential privacy provides strong privacy guarantee compared to traditional privacy models. However, it limits the data utility. Perturbation-based techniques achieve differential privacy by adding noises. Perturbed data are useful at the aggregated level (such as average or sum), but not at the record level. Data recipients can no longer interpret the semantics of each individual record. Therefore, perturbation-based techniques do not satisfy the requirement of our data integration application for the financial industry.

9.2 Privacy-preserving distributed data mining

In privacy-preserving distributed data mining, multiple data providers want to compute a function based on their inputs without sharing their data with others. For example, multiple hospitals may want to build a data mining model (e.g. classifier for predicting disease based on patients' history) without sharing their data with one another. The usual assumption is that the data providers are semi-honest where they follow the protocol but may try to deduce additional information from the received messages. A number of works also propose secure protocols for malicious adversary model [25,27,29]. Kantarcioglu and Kardes [27] propose the malicious versions of the commonly used semi-honest protocols such as equality, dot product, and full domain set operations. Jiang et al. [25] propose an accountable computing (AC) framework that enables other parties to identify the adversary by verification. Extensive research has been conducted to design secure protocols for different data mining tasks, such as secure multiparty computation (SMC) of classifiers [12,13,59], association rules mining [52], clustering [53], and ID3 decision tree [35]. Refer to [8,45] for surveys on privacy-preserving distributed data mining.

Techniques based on SMC provide strong privacy guarantee and prevent any disclosure of sensitive information. But these methods are known to be very expensive in terms of computation cost and sometimes impractical in real-life scenarios. Moreover, comparing to data mining results sharing, data sharing offers greater flexibility to the data recipients to apply their own classifiers and parameters.

9.3 Game theory and information sharing

Recently, a body of research models the problem of information sharing as a game and employs incentive mechanisms to deter malicious behavior [2,28,30–32,61]. Kleinberg et al. [31] consider different information-exchange scenarios and use solution concepts from coalition games to quantify the value of each user's participation. Zhang and Zhao [61] address the problem of secure intersection computation and use non-cooperative games to determine an optimal countermeasure, where the defendant changes its input database to protect its private information from an adversary. Kantarcioglu et al. [28] define a game between the adversary and the data miner in the context of intrusion detection system. They present a technique to find the equilibrium point of the game so that the data miner can construct an effective classifier. Agrawal and Terzi [2] model information sharing as a game and propose an auditing device that checks the behavior of the participants. They use game theory to determine the lower bound on the auditing frequency that assures honestly.

The work closest to ours is by Layfield et al. [32]. They model information sharing among different organizations as a game, where organizations may provide false information. They adopt evolutionary game-theoretic framework and study the behavior of the agents (organizations). One fundamental difference with our approach is that they assume that information is verifiable. If an organization provides false information, then the other organizations can detect it given some additional verification cost. However, such verification is not possible in our application scenario. Moreover, all the proposed game-theoretic models do not address the problems of secure data integration, which is the primary contribution of this paper.

9.4 Privacy-preserving distributed data integration

Many techniques have been proposed for data integration in database research [9,55]. This literature typically assumes that all information in each database can be freely shared without considering the privacy concerns discussed in this paper. To minimize the information shared with other parties, Agrawal et al. [3] propose the principle of *minimal information sharing* for computing queries spanning private databases. Inan et al. [20,21] address the problem of private record linkage, where given two input data tables, the mechanism identifies similar records that represent the same real-world entity. Though closely related, these works do not address the problem of secure data integration.

Jurczyk and Xiong [26] propose an algorithm to securely integrate horizontally partitioned data from multiple data providers without disclosing data from one party to another. Mohammed et al. [40] propose a distributed algorithm to integrate horizontally partitioned high-dimensional health care data. Unlike the distributed anonymization problem for vertically partitioned data studied in this paper, all these methods [26,40] propose a distributed algorithm for horizontally partitioned data.

Jiang and Clifton [23] propose a method to integrate vertically partitioned data by maintaining k-anonymity among the

participating parties. However, this approach does not fulfill the security requirements of a semi-honest adversary model. To satisfy this requirement, [24] propose D*k*A (Distributed k-Anonymity) framework to securely integrate two distributed data tables satisfying *k*-anonymity requirement. To the best of our knowledge, Jiang and Clifton's works are the only ones that generate a *k*-anonymous table in a distributed setting for vertically partitioned data. Our proposed algorithms, as discussed already, outperform D*k*A framework in terms of algorithmic complexity and scalability for handling large data sets.

This paper is the extension of our previous work [39], where we proposed an anonymization algorithm for vertically partitioned data from multiple semi-honest data providers. In this paper, we propose an additional algorithm to integrate data with the consideration of malicious parties. We show that a party may deviate from the proposed protocol of [39] for its own benefit. To overcome the malicious problem, we propose a game-theoretic solution by combining incentive compatible strategies with our anonymization algorithm.

10 Conclusions and lesson learned

We solved a privacy-preserving data sharing problem for financial institutions in Sweden and generalized their privacy and information requirements to the problem of secure data integration for the purpose of joint classification analysis. We formalized this problem as achieving the *k*-anonymity on the integrated data without revealing more detailed information in the process. We presented two solutions based on two different adversary models and evaluated the benefits of data integration and the impacts of generalization. Our algorithms: (1) are scalable to large data sets, (2) can handle multi-party scenario, (3) address both semi-honest and malicious adversary model. Compared to classic secure multiparty computation, a unique feature is to allow data sharing instead of only result sharing. This feature is especially important for data analysis where the process is not performing an input/output black-box mapping, and user interaction and knowledge about the data often lead to superior results. Being able to share data would permit such exploratory data analysis and explanation of results.

We would like to share our experience in collaboration with the financial sector. In general, they prefer simple privacy requirements. Despite some criticisms on *k*-anonymity [36,54], the financial sector (and probably some other sectors) finds that *k*-anonymity is an ideal privacy requirement due to its intuitiveness. Their primary concern is whether they can still effectively perform the task of data analysis on the anonymous data. Therefore, solutions that solely satisfy some privacy requirement are insufficient for them. They demand anonymization methods that can preserve information for various data analysis tasks.

Acknowledgements We sincerely thank the reviewers for their insightful comments. The research is supported in part by the new researchers start-up program from Le Fonds québécois de la recherche sur la nature et les technologies (FQRNT), Discovery Grants (356065-2008), and Canada Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada.

References

1. Adam, N.R., Wortman, J.C.: Security control methods for statistical databases. ACM Comput. Surv. **21**(4), 515–556 (1989)
2. Agrawal, R., Terzi, E.: On honesty in sovereign information sharing. In: Proceedings of the EDBT (2006)
3. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of ACM SIGMOD, San Diego, CA (2003)
4. Axelrod, R.: The Evolution of Cooperation. Basic Books, New York (1984)
5. Bayardo, R.J., Agrawal, R.: Data privacy through optimal *k*-anonymization. In: ICDE (2005)
6. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the SuLQ framework. In: PODS (2005)
7. Brodsky, A., Farkas, C., Jajodia, S.: Secure databases: Constraints, inference channels, and monitoring disclosures. IEEE Trans. Knowl. Data Eng. **12**, 900–919 (2000)
8. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explor. Newsl. **4**(2), 28–34 (2002)
9. Dayal, U., Hwang, H.Y.: View definition and generalization for database integration in a multidatabase systems. IEEE Trans. Softw. Eng. **10**(6), 628–645 (1984)
10. Denning, D., Schlorer, J.: Inference controls for statistical databases. IEEE Comput. **16**(7), 69–82 (1983)
11. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: PODS (2003)
12. Du, W., Zhan, Z.: Building decision tree classifier on private data. In: Workshop on Privacy, Security, and Data Mining at the IEEE ICDM (2002)
13. Du, W., Han, Y.S., Chen, S.: Privacy-preserving multivariate statistical analysis: linear regression and classification. In: Proceedings of the SIAM International Conference on Data Mining (SDM), Florida (2004)
14. Dwork, C.: Differential privacy. In: ICALP (2006)
15. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC (2006)
16. Farkas, C., Jajodia, S.: The inference problem: A survey. ACM SIGKDD Explor. Newsl. **4**(2), 6–11 (2003)
17. Fung, B.C.M., Wang, K., Yu, P.S.: Anonymizing classification data for privacy preservation. IEEE TKDE **19**(5), 711–725 (2007)
18. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Comput. Surv. **42**(4), 14:1–14:53 (2010)
19. Hinke, T.: Inference aggregation detection in database management systems. In: IEEE S&P (1988)
20. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: Proceedings of the Int'l Conference on Data Engineering (2008)
21. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: Proceedings of the EDBT (2010)

22. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: SIGKDD (2002)
23. Jiang, W., Clifton, C.: Privacy-preserving distributed k-anonymity. In: BDSec (2005)
24. Jiang, W., Clifton, C.: A secure distributed framework for achieving k-anonymity. Very Large Data Bases J. (VLDBJ) **15**(4), 316–333 (2006)
25. Jiang, W., Clifton, C., Kantarcioglu, M.: Transforming semi-honest protocols to ensure accountability. Data Knowl. Eng. **65**(1), 57–74 (2008)
26. Jurczyk, P., Xiong, L.: Distributed anonymization: achieving privacy for both data subjects and data providers. In: DBSec (2009)
27. Kantarcioglu, M., Kardes, O.: Privacy-preserving data mining in the malicious model. Int. J. Inf. Comput. Secur. **2**(4), 353–375 (2008)
28. Kantarcioglu, M., Xi, B., Clifton, C.: A game theoretical model for adversarial learning. In: Proceedings of the NGDM Workshop (2007)
29. Kardes, O., Kantarcioglu, M.: Privacy-preserving data mining applications in malicious model. In: Proceedings of the PADM Workshop (2007)
30. Kargupta, H., Das, K., Liu, K.: A game theoretic approach toward multi-party privacy-preserving distributed data mining. In: Proceedings of the PKDD (2007)
31. Kleinberg, J., Papadimitriou, C., Raghavan, P.: On the value of private information. In: TARK (2001)
32. Layfield, R., Kantarcioglu, M., Thuraisingham, B.: Incentive and trust issues in assured information sharing. In: Proceedings of the CollaborateComm (2008)
33. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Workload-aware anonymization. In: SIGKDD (2006)
34. Li, N., Li, T., Venkatasubramanian, S. t-closeness: privacy beyond k-anonymity and ℓ-diversity. In: ICDE (2007)
35. Lindell, Y., Pinkas, B.: Privacy preserving data mining. J. Cryptol. **15**(3), 177–206 (2002)
36. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: privacy beyond k-anonymity. ACM TKDD **1**(1) (2007)
37. Malvestuto, F.M., Mezzini, M., Moscarini, M.: Auditing sum-queries to make a statistical database secure. ACM Trans. Inf. Syst. Secur. **9**(1), 31–60 (2006)
38. Mohammed, N., Fung, B.C.M., Hung, P.C.K., Lee, C.: Anonymizing healthcare data: a case study on the blood transfusion service. In: SIGKDD (2009a)
39. Mohammed, N., Fung, B.C.M., Wang, K., Hung, P.C.K.: Privacy-preserving data mashup. In: EDBT (2009b)
40. Mohammed, N., Fung, B.C.M., Hung, P.C.K., Lee, C. (2010) Centralized and distributed anonymization for high-dimensional healthcare data. ACM Trans. Knowl. Discov. Data (TKDD) **4**(4), 18:1–18:33
41. Nash, J.: Non-cooperative games. Ann. Math. **54**(2), 286–295 (1951)
42. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. http://archive.ics.uci.edu/ml/ (1998)
43. Nisan, N.: Algorithms for selfish agents. In: Proceedings of the STACS (1999)
44. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press, Cambridge, UK (1994)
45. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. ACM SIGKDD Explor. Newsl. **4**(2), 12–19 (2002)
46. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Los Altos (1993)
47. Samarati, P.: Protecting respondents' identities in microdata release. IEEE TKDE **13**(6), 1010–1027 (2001)
48. Sweeney, L.: Datafly: a system for providing anonymity in medical data. In: Proceedings of the DBSec (1998)
49. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(5), 571–588 (2002a)
50. Sweeney, L.: k-anonymity: a model for protecting privacy. In: International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (2002b)
51. Thuraisingham, B.M.: Security checking in relational database management systems augmented with inference engines. Comput. Secur. **6**(6), 479–492 (1987)
52. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of the ACM SIGKDD (2002)
53. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: Proceedings of the ACM SIGKDD (2003)
54. Wang, K., Fung, B.C.M., Yu, P.S.: Handicapping attacker's confidence: An alternative to k-anonymization. KAIS **11**(3), 345–368 (2007)
55. Wiederhold, G.: Intelligent integration of information. In: Proceedings of ACM SIGMOD, pp 434–437 (1993)
56. Wong, R.C.W., Li, J., Fu, A.W.C., Wang, K.: (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: SIGKDD (2006)
57. Xiao, X., Tao, Y.: Anatomy: simple and effective privacy preservation. In: VLDB (2006)
58. Xiao, X., Yi, K., Tao, Y. The hardness and approximation algorithms for l-diversity. In: EDBT (2010)
59. Yang, Z., Zhong, S., Wright, R.N.: Privacy-preserving classification of customer data without loss of accuracy. In: Proceedings of the SDM (2005)
60. Yao, A.C.: Protocols for secure computations. In: Proceedings of the IEEE FOCS (1982)
61. Zhang, N., Zhao, W.: Distributed privacy preserving information sharing. In: Proceedings of the VLDB (2005)