Privacy-Preserving Data Mashup

Noman Mohammed* Benjamin C. M. Fung* Ke Wang† Patrick C. K. Hung‡

*CIISE, Concordia University, Montreal, QC, Canada
†Simon Fraser University, Burnaby, BC, Canada
‡University of Ontario Institute of Technology, Oshawa, ON, Canada
{no_moham, fung}@ciise.concordia.ca wangk@cs.sfu.ca patrick.hung@uoit.ca

ABSTRACT

Mashup is a web technology that combines information from more than one source into a single web application. This technique provides a new platform for 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 this paper, we study and resolve a real-life privacy problem in a data mashup application for the financial industry in Sweden, and propose a privacy-preserving data mashup (PPMashup) algorithm to securely integrate private data from different data providers, whereas the integrated data still retains the essential information for supporting general data exploration or a specific data mining task, such as classification analysis. Experiments on real-life data suggest that our proposed method is effective for simultaneously preserving both privacy and information usefulness, and is scalable for handling large volume of data.

1. INTRODUCTION

Mashup is a web technology that combines information and services from more than one source into a single web application. It was first discussed in a 2005 issue of Business Week [17] on the topic of integrating real estate information into Google Maps. Since then, web giants like Amazon, Yahoo!, and Google have been actively developing mashup applications. Mashup has created a new horizon for service providers to integrate their data and expertise to deliver highly customizable services to their customers.

Data mashup is a special type of mashup application that aims at integrating data from multiple data providers depending on the user's service request. Figure 1 illustrates a typical architecture of the data mashup technology. A service request could be a general data exploration or a sophisticated data mining task such as classification analysis. Upon receiving a service request, the data mashup web application dynamically determines the data providers, collects information from them through their web service application programming interface (API),[1] and then integrates the collected information to fulfill the service request. Further computation and visualization can be performed at the user's site (e.g., a browser or an applet). This is very different from the traditional web portal which simply divides a web page or a website into independent sections for displaying information from different sources.

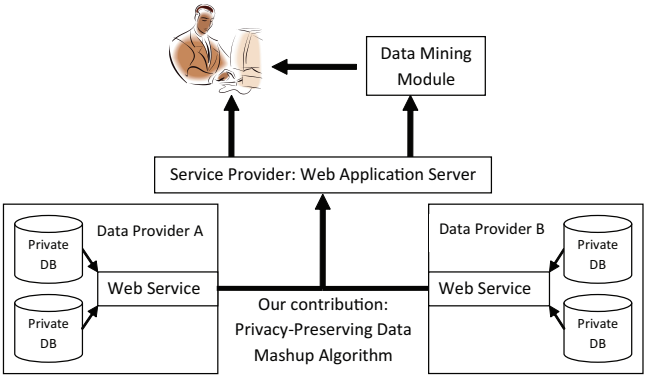

Figure 1: Architecture of (Privacy-Preserving) Data Mashup

A data mashup application can help ordinary users explore new knowledge. Nevertheless, it could 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 mashup and propose a privacy-preserving data mashup (PPMashup) algorithm to securely integrate person-specific sensitive data from different data providers, whereas the integrated data still retains 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 simultaneous need of information sharing and privacy preservation in the financial industry.

This research problem was discovered in a collaborative project with Nordax Finans AB, 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 key SSN,[2] e.g., $T_A(SSN, Age, Balance)$

[1]Authentication may be required to ensure that the user has access rights to the requested data.
[2]SSN is called "personnummer" in Sweden.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the ACM. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permissions from the publisher, ACM.
EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.
Copyright 2009 ACM 978-1-60558-422-5/09/0003 ...$5.00

Table 1: Raw tables

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

and $T_B(SSN, Job, Salary)$. These companies want to implement a data mashup application that integrates 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 additional to companies A and B, their partnered credit card company C also have access to the data mashup application, so all three companies A, B, and C are data recipients of the final integrated data. 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 Figure 2. Party A (the loan company) and Party B (the bank) own $T_A(SSN, Sex, \ldots, Class)$ and $T_B(SSN, Job, Salary, \ldots, Class)$, 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. After integrating the two tables (by matching the SSN field), the female lawyer on (Sex, Job) becomes unique, 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 *private data mashup* 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 releasing it 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: A data table T satisfies k-anonymity if every combination of values on QID 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

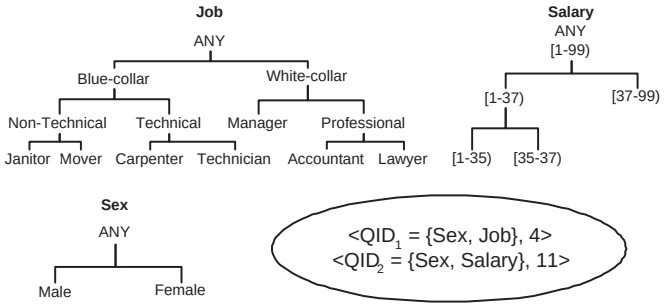

Figure 2: Taxonomy trees and QIDs

concepts. In addition, at any time in the procedure of generalization, no party should learn more detailed information about the other party other than those in the final integrated table. For example, *Lawyer* is more detailed than *Professional*. In other words, the generalization process must not leak more specific information other than the final integrated data,

Information Requirement: The generalized data is as useful as possible to classification analysis. Generally speaking, the privacy goal requires masking sensitive information that are *specific* enough to identify individuals, whereas the classification goal requires extracting trends and patterns that are *general* enough to predict new cases. If generalization is *carefully* performed, it is possible to mask identifying information while preserving patterns useful for classification.

There are two obvious yet incorrect approaches. The first one is "integrate-then-generalize": first integrate the two tables and then generalize the integrated table using some single table anonymization methods [4, 12, 13, 19, 24]. 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 both 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 the above example, the k-anonymity on (Sex,Job) cannot be achieved by the k-anonymity on each of *Sex* and *Job* separately.

In additional to the privacy and information requirements, the data mashup application is an online web application. The user dynamically specifies their requirement and the system is expected to be efficient and scalable to handle high volume of data.

This paper makes four contributions.

1. We identify a new privacy problem through a collaboration with the financial industry and generalize their requirements to formulate the private data mashup problem (Section 3). The goal is to allow data sharing for classification analysis in the presence of privacy concern. This problem is very different from secure multiparty computation [45], which allows "result sharing" (e.g., the classifier in our case) but completely prohibits data sharing. In many applications, data sharing gives greater flexibility than result sharing because data recipients can perform their required analysis and data exploration, such as, mine patterns in a specific group of records, visualize the transactions containing

a specific pattern, try different modeling methods and parameters.

2. We present a privacy-preserving data mashup (PP-Mashup) algorithm to securely integrate private data from multiple parties (Sections 4-5). Essentially, our algorithm produces the same final anonymous table as the integrate-then-generalize approach, but will only reveal local data that has satisfied a given k-anonymity requirement. This technique is simple but effective for privacy protection.

3. We implement the proposed method in the context of a data mashup web application and evaluate its performance (Section 6). Experimental results on real-life data suggest that the method can effectively achieve a privacy requirement without compromising the useful data for classification, and the method is scalable to handle large data set.

4. We further extend the proposed privacy-preserving method to achieve other privacy requirements, such as ℓ-diversity [25], (α,k)-anonymity [40], and confidence bounding [38] (Section 7).

2. RELATED WORK

Information integration has been an active area of database research [6, 39]. This literature typically assumes that all information in each database can be freely shared [2]. Secure multiparty computation (SMC), on the other hand, allows sharing of the computed result (e.g., a classifier), but completely prohibits sharing of data [46], which is a primary goal of our studied problem. An example is the secure multiparty computation of classifiers [5, 7, 8, 45].

Yang et al. [44] proposed several cryptographic solutions to collect information from a large number for data owners. Yang et al. [45] developed a cryptographic approach to learn classification rules from a large number of data owners while their sensitive attributes are protected. The problem can be viewed as a horizontally partitioned data table in which each transaction is owned by a different data owner. The model studied in this paper can be viewed as a vertically partitioned data table, which is completely different from [44, 45]. More importantly, the output of their method is a classifier, but the output of our method is an integrated anonymous data that supports classification analysis. Having accessing the data, the data recipient has the freedom to apply her own classifier and parameters.

Agrawal et al. [2] proposed the notion of minimal information sharing for computing queries spanning private databases. They considered computing intersection, intersection size, equijoin and equijoin size, assuming that certain metadata such as the cardinality of databases can be shared to both parties. Besides, there exists an extensive literature on inference control in multilevel secure databases [9, 16, 15, 14, 20]. All these works prohibit the sharing of databases.

The notion of k-anonymity was proposed in [33, 32], and generalization was used to achieve k-anonymity in Datafly system [34] and μ-Argus system [18]. Unlike generalization and suppression, Xiao and Tao [41] proposed an alternative approach, called *anatomy*, that does not modify the quasi-identifier or the sensitive attribute, but de-associates the relationship between the two. [42] proposed the notion of personalized privacy to allow each record owner to specify her own privacy level. This model assumes that Sensitive Attribute has a taxonomy tree and that each record owner specifies a guarding node in this taxonomy tree. Preserving k-anonymity for classification was studied in [4, 12, 13, 19, 24]. [11, 37] studied the privacy threats caused by publishing multiple releases. [43] proposed a new privacy notion called m-invariance and an anonymization method for continuous data publishing. All these works considered a single data source, therefore, data integration is not an issue. In the case of multiple private databases, joining all private databases and applying a single table method would violate the privacy requirement.

Jiang and Clifton [21, 22] proposed a cryptographic approach to securely integrate two distributed data tables to a k-anonymous table without considering a data mining task. First, each party determines a locally k-anonymous table. Then, determine the intersection of $RecID$'s for the QID groups in the two locally k-anonymous tables. If the intersection size of each pair of QID group is at least k, then the algorithm returns the join of the two locally k-anonymous tables which is globally k-anonymous; otherwise, perform further generalization on both tables and repeat the $RecID$ comparison procedure. To prevent the other party from learning more specific information than the final integrated table through $RecID$, they employ a commutative encryption scheme [30] to encrypt the $RecID$'s for comparison. This scheme ensures the equality of two values encrypted in different order on the same set of keys, i.e., $E_{Key1}(E_{Key2}(RecID)) = E_{Key2}(E_{Key1}(RecID))$. Moreover, Vaidya and Clifton proposed techniques to mine association rules [35] and to compute k-means clustering [36] over vertically partitioned data.

Miklau and Suciu [27] measured information disclosure of a view set V with respect to a secret view S. S is secure if publishing the answer to V does not alter the probability of inferring the answer to S. However, they only focus how to measure the information disclosure of exchange database views while our work removes privacy risks by anonymizing multiple private databases. There is a body of work on randomizing data for achieving privacy [3, 10, 23]. Randomized data are useful at the aggregated level (such as average or sum), but not at the record level. Instead of randomizing the data, we generalize the data to make information less precise while preserving the "truthfulness" of information (say, *Lawyer* generalized to *Professional*). Generalized data are meaningful at the record level, therefore, can be utilized by the human user to guide the search or interpret the result. Finally, these works do not consider integration of multiple data sources, which is a central topic in this paper.

Many synthetic data generation techniques were proposed in the literature of statistical disclosure control in the scenario of a single data publisher [1, 26]. Similar to randomization, the synthetic data usually preserve some important statistical properties including mean, variances, the covariance matrix, and the Pearson correlation matrix from the original data. Although the disclosure risk is shown to be lower than some simple masking methods such as additive noise [26], it again does not preserve the truthfulness of information at the record level. Therefore, both randomization and synthetic data generation do not satisfy the requirement of our data mashup application for the financial industry. Yet, they are still useful techniques if the applications do not require preserving data truthfulness at the record level.

3. PROBLEM DEFINITION

We first define k-anonymity on a single table and then extend it for private data mashup from multiple parties.

3.1 The k-Anonymity

Consider a person-specific table $T(ID, D_1, \ldots, D_m, Class)$. ID is record identifier, such as SSN, that we can ignore for now. Each D_i is either a categorical or a continuous attribute. The $Class$ column contains class labels or distribution. 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 3.1 (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 $A(QID_j)$, 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 $1 \leq i \leq p$, where k_j is the anonymity threshold on QID_j. ■

Definition 3.1 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 [12, 13]. 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 because if a table T satisfies $\langle QID_i, k_i\rangle$, then it must also satisfy $\langle QID_j, k_j\rangle$; therefore, $\langle QID_j, k_j\rangle$ can be removed from the anonymity requirement.

Example 2. $\langle QID_1 = \{Sex, Job\}, 4\rangle$ states that every qid on QID_1 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*$\rangle$,
$\langle$*Female*, *Lawyer*$\rangle$.
The example in Figure 2 specifies the k-anonymity requirement on two QIDs. ■

3.2 Private Data Mashup

Consider n data providers {Party 1,...,Party n}, where each Party y owns a private table $T_y(ID, Attribs_y, Class)$ over the same set of records. ID and $Class$ are shared attributes among all parties. $Attribs_y$ is a set of private attributes. $Attribs_y \cap Attribs_z = \emptyset$ for any $1 \leq y, z \leq n$. These parties agree to release "minimal information" to form an integrated table T (by matching the ID) 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 only attributes from one party, and *global* otherwise.

Definition 3.2 (Private Data Mashup). 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 *private data mashup* is to efficiently produce a generalized integrated table T such that (1) T satisfies the joint anonymity requirement, (2) T contains as much information as possible for classification, (3) each party learns nothing about the other party more specific than what is in the final generalized T. We assume that the data providers are semi-honest, meaning that they will follow the protocol but may attempt to derive sensitive information from the received data. ■

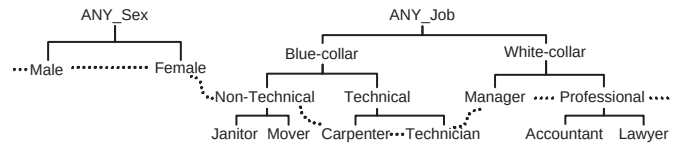

Figure 3: A solution cut for $QID_1 = \{Sex, Job\}$

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 this record comes from $Lawyer$, condition (3) is violated. Our privacy model ensures the anonymity in the final integrated table as well as in any intermediate table.

To ease the explanation, we present our solution in a scenario of two parties ($n = 2$). A discussion is given in Section 5.4 to describe the extension to multiple parties.

4. SPECIALIZATION CRITERIA

To generalize T, 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. Figure 2 shows a dynamically grown taxonomy tree for *Salary*. We generalize a table T by a sequence of specializations starting from the top most general state in which each attribute has the top most 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. A specialization is *valid* if the specialization results in a table satisfying 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 Cut_i, contains exactly one value on each root-to-leaf path. Figure 3 shows a solution cut indicated by the dashed curve. Our specialization starts from the top most solution cut and pushes down the solution cut iteratively by specializing some value in the current solution cut until violating the anonymity requirement. Each specialization tends to increase information and decrease anonymity because records are more distinguishable by specific values. The key is selecting a specialization at each step with both impacts considered.

One core step of this approach is computing *Score*, which measures the goodness of a specialization with respect to privacy preservation and information preservation. The effect of a specialization $v \rightarrow child(v)$ can be summarized by information gain, denoted $InfoGain(v)$, and anonymity loss, denoted $AnonyLoss(v)$, due to the specialization. Our selection criterion is to favor the specialization v that has the maximum information gain per unit of anonymity loss:

$$Score(v) = \frac{InfoGain(v)}{AnonyLoss(v) + 1}. \quad (1)$$

We add 1 to $AnonyLoss(v)$ to avoid division by zero.

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 $T[x]$ [31]:

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

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

AnonyLoss(v): This is the average loss of anonymity by specializing v over all QID_j that contain the attribute of v:

$$AnonyLoss(v) = avg\{A(QID_j) - A_v(QID_j)\}, \quad (4)$$

where $A(QID_j)$ and $A_v(QID_j)$ represents the anonymity before and after specializing v. Note that $AnonyLoss(v)$ not just depends on the attribute of v; it depends on all QID_j that contain the attribute of v. Hence, $avg\{A(QID_j) - A_v(QID_j)\}$ is the average loss of all QID_j that contain the attribute of v.

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.

$$I(R_{ANY_Job}) = -\frac{21}{34} \times log_2 \frac{21}{34} - \frac{13}{34} \times log_2 \frac{13}{34} = 0.9597$$
$$I(R_{Blue-collar}) = -\frac{5}{16} \times log_2 \frac{5}{16} - \frac{11}{16} \times log_2 \frac{11}{16} = 0.8960$$
$$I(R_{White-collar}) = -\frac{16}{18} \times log_2 \frac{16}{18} - \frac{2}{18} \times log_2 \frac{2}{18} = 0.5033$$
$$InfoGain(ANY_Job) = I(R_{ANY_Job}) - (\frac{16}{34} \times I(R_{Blue-collar}) + \frac{18}{34} \times I(R_{White-collar})) = 0.2716$$
$$AnonyLoss(ANY_Job) = avg\{A(QID_1) - A_{ANY_Job}(QID_1)\} = (34 - 16)/1 = 18$$
$$Score(ANY_Job) = \frac{0.2716}{18} = 0.0151. \blacksquare$$

For a continuous attribute, the specialization of an interval refers to the optimal binary split that maximizes information gain. We use information gain, instead of *Score*, to determine the split of an interval because anonymity is irrelevant to finding a split good for classification. This is similar to the situation that the taxonomy tree of a categorical attribute is specified independently of the anonymity issue. Among the specializations of different continuous attributes, we still use *Score* for selecting the best one, just like categorical attributes.

Example 4. For the continuous attribute *Salary*, the top most 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:

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

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)*. ■

The next example shows that $InfoGain$ alone may lead to a quick violation of the anonymity requirement, thereby, prohibiting specializing data to a lower granularity.

Table 2: Raw table for Example 5

Education	Sex	Work_Hrs	Class	# of Recs.
10th	M	40	20Y0N	20
10th	M	30	0Y4N	4
9th	M	30	0Y2N	2
9th	F	30	0Y4N	4
9th	F	40	0Y6N	6
8th	F	30	0Y2N	2
8th	F	40	0Y2N	2
			Total:	40

Table 3: Generalized table by *Score* for Example 5

Education	Sex	Work_Hrs	Class	# of Recs.
ANY_Edu	M	[40-99)	20Y0N	20
ANY_Edu	M	[1-40)	0Y6N	6
ANY_Edu	F	[40-99)	0Y8N	8
ANY_Edu	F	[1-40)	0Y6N	6

Example 5. Consider Table 2, an anonymity requirement $\langle QID = \{Education, Sex, Work_Hrs\}, 4\rangle$, and specializations:
ANY_Edu → {*8th, 9th, 10th*},
ANY_Sex → {*M, F*}, and
[1-99) → {*[1-40), [40-99)*}.
The class frequency for the specialized values is:
Education: 0Y4N (8th), 0Y12N (9th), 20Y4N (10th)
Sex: 20Y6N (M), 0Y14N (F)
Work_Hrs: 0Y12N ([1-40)), 20Y8N ([40-99))
Specializing *Education* best separates the classes, so is chosen by $InfoGain$. After that, the other specializations become invalid. Now, the two classes of the top 24 records become indistinguishable because they are all generalized into ⟨ *10th, ANY_Sex, [1-99)*⟩. In contrast, the *Score* criterion will first specialize *Sex* because of the highest *Score* due to a small *AnonyLoss*. Subsequently, specializing *Education* becomes invalid, and the next specialization is on *Work_Hrs*. The final generalized table is shown in Table 3 where the information for distinguishing the two classes is preserved. ■

5. OUR METHOD

In [12, 13], we proposed a *top-down specialization (TDS)* approach to generalize a **single table** T. One non-privacy-preserving approach to the problem of data mashup 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 3.2 (because the party that generalizes the joint table knows all the details of the other parties), the integrated table produced satisfies requirements (1) and (2). Therefore, it is helpful to first have an overview of TDS: Initially, all values are generalized to the top most value in its taxonomy tree, and Cut_i contains the top most value for each attribute D_i. At each iteration, TDS performs the best specialization, which has the highest *Score* among the *candidates* that are valid, beneficial specializations in $\cup Cut_i$, and then updates the *Score* of the affected candidates. The algorithm terminates when there is no more 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. This is because a specialization never increases the anonymity count $a(qid)$.

Now, we consider that the table T is given by two tables ($n = 2$) 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 a single attribute depends only on that attribute and *Class*, and each party knows about *Class* and the attributes they have. This observation is wrong because the change of *Score* involves the change of $A(QID_j)$ that depends on the combination of the attributes in QID_j. In PPMashup, each party keeps a copy of the current $\cup Cut_i$ and generalized T, denoted 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, therefore, does not violate the requirement (3) in Definition 3.2. At each iteration, the two parties cooperate to perform the same specialization as identified in TDS by communicating certain information in a way that satisfies the requirement (3) in Definition 3.2. Algorithm 1 describes the procedure at Party A (same for Party B).

First, Party A finds the local best candidate using the specialization criteria presented in Section 4 and communicates with Party B to identify the overall global winner candidate, say w. To protect the input score, the secure multiparty maximum protocol [46] can be used. Suppose that w is local to Party A (otherwise, the discussion below applies to Party B). 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 those $t'_1, \ldots, t'_z$ containing child values in $child(w)$. 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.

Example 6. Consider Table 1 and the joint anonymity requirement:

$\{\langle QID_1 = \{Sex, Job\}, 4\rangle,\ \langle QID_2 = \{Sex, Salary\}, 11\rangle\}$.

Initially,

$T_g = \{\langle ANY_Sex, ANY_Job, [1-99)\rangle\}$

and

$\cup Cut_i = \{ANY_Sex, ANY_Job, [1-99)\}$,

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 $Score([1\text{-}99))$. ■

Algorithm 1 PPMashup for Party A (same as Party B)

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 is some candidate in $\cup Cut_i$ **do**
4: find the local candidate x of highest $Score(x)$;
5: communicate $Score(x)$ with Party B to find the winner;
6: **if** the winner w is local **then**
7: specialize w on T_g;
8: instruct Party B to specialize w;
9: **else**
10: wait for the instruction from Party B;
11: specialize w on T_g using the instruction;
12: **end if**;
13: replace w with $child(w)$ in the local copy of $\cup Cut_i$;
14: update $Score(x)$, the beneficial/valid status for candidates x in $\cup Cut_i$;
15: **end while**;
16: output T_g and $\cup Cut_i$;

Below, we describe the key steps: find the winner candidate (Line 4-5), perform the winning specialization (Line 7-11), and update the score and status of candidates (Line 14). 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 $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$, and then communicates with Party B (using secure multiparty max algorithm in [46]) to find the winner candidate. $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$ come from the update done in the previous iteration or the initialization prior to the first iteration. This step does not access data records. Updating $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$ is considered in Section 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 5.1 (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_x's. ■

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 only access to its own raw records, a

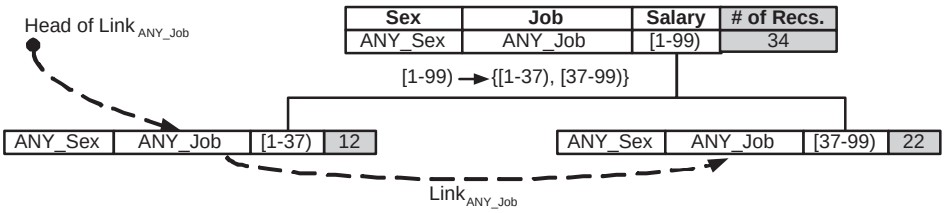

Figure 4: The TIPS after the first specialization

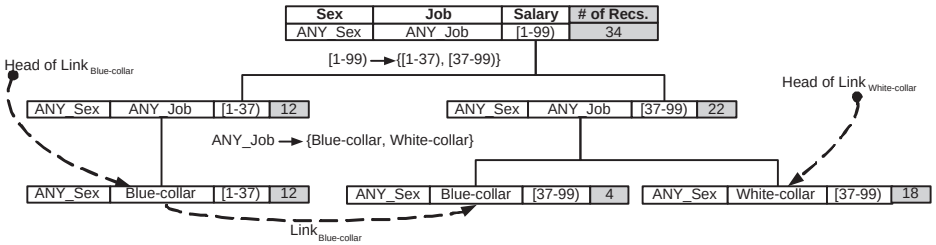

Figure 5: The TIPS after the second specialization

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 7. Continue with Example 6. Initially, TIPS has the root representing the most generalized record $\langle ANY_Sex, ANY_Job, [1-99)\rangle$, $T_A[root] = T_A$ and $T_B[root] = T_B$. The root is on $Link_{ANY_Sex}$, $Link_{ANY_Job}$, and $Link_{[1-99)}$. See the root in Figure 4. The shaded field contains the number of raw records generalized by a node. Suppose that the winning candidate w is

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

Party B first creates two child nodes under the root and partitions $T_B[root]$ between them. The root is deleted from all the $Link_x$, 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 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 Figure 5. ■

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_c's for the same c. Mark c as *beneficial* if the records on $Link_c$ has 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 Section 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_c's contain raw records from T_B. P_c's are obtained by splitting P_w among P_c's 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 Section 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.

5.3 Update the Score

The key to the scalability of our algorithm is updating $Score(x)$ using the count statistics maintained in Section 5.2 without accessing raw records again. $Score(x)$ depends on $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$. The updated $A(QID_j)$ is obtained from $A_w(QID_j)$, where w is the specialization just performed. Below, we consider updating $InfoGain(x)$ and $A_x(QID_j)$ separately.

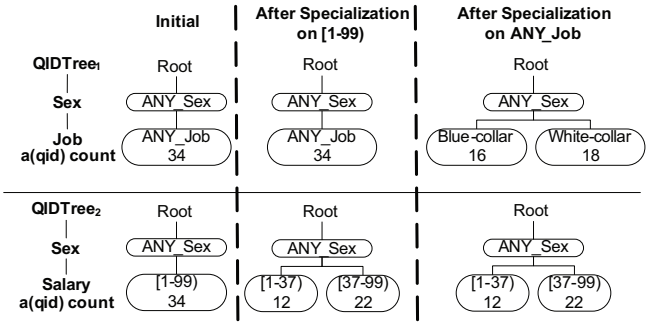

Figure 6: The QIDTrees data structure

5.3.1 Updating $InfoGain(x)$.

We need to compute $InfoGain(c)$ for the newly added c in $child(w)$. The owner party of w can compute $InfoGain(c)$ while collecting the count statistics for c in Section 5.2.

5.3.2 Updating $AnonyLoss(x)$.

Recall that $A_x(QID_j)$ is the minimum $a(qid_j)$ after specializing x. Therefore, if $att(x)$ and $att(w)$ both occur in some QID_j, the specialization on w might affect $A_x(QID_j)$, and we need to find the new minimum $a(qid_j)$. The following $QIDTree_j$ data structure indexes $a(qid_j)$ by qid_j.

Definition 5.2 (QIDTrees). For each $QID_j = \{D_1, \ldots, D_q\}$, $QIDTree_j$ is a tree of q levels, where level $i > 0$ represents 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 $QIDTree_j$'s that contain the attribute $att(w)$: creates the nodes for the new qid_j's and computes $a(qid_j)$. 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 Section 5.2.

Example 8. Continue with Example 7. Figure 6 shows the initial $QIDTree_1$ and $QIDTree_2$ for QID_1 and QID_2 on the left. On performing *[1-99)* → {*[1-37)*, *[37-99)*}, ⟨*ANY_Sex*, *[1-99)*⟩ in $QIDTree_2$ is replaced with qids ⟨ *ANY_Sex*, *[1-37)* ⟩ and ⟨ *ANY_Sex*, *[37-99)*⟩. $A(QID_2) = 12$.

Next, on performing *ANY_Job* → {*Blue-collar*, *White-collar*}, ⟨ *ANY_Sex, ANY_Job*⟩ in $QIDTree_1$ is replaced with new qids ⟨*ANY_Sex, Blue-collar*⟩ and ⟨*ANY_Sex, White-collar*⟩. 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 Figure 5): $a(\langle ANY_Sex, Blue\text{-}collar\rangle) = 0 + 12 + 4 = 16$, and $a(\langle ANY_Sex, White\text{-}collar\rangle) = 0 + 18 = 18$. So $A_{ANY_Job}(QID_1) = 16$. ■

Updating $A_x(QID_j)$. 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., 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^x's 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 then is compared with $A(QID_j)$ to determine $A_x(QID_j)$. If $A_x(QID_j) \geq k_j$, we mark x as *valid* in $\cup Cut_i$.

5.4 Analysis

Our approach produces the same integrated table as the single party algorithm TDS [12, 13] 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 PPMashup 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. The only information revealed to each other is those in $\cup Cut_j$ and T_g at each iteration. However, such information is more general than the final integrated table that the two parties agree to share.

PPMashup (Algorithm 1) is extendable for multiple parties with minor changes: In Line 5, each party should communicate with all the other parties for determining the winner. Similarly, in Line 8, the party holding the winner candidate should instruct all the other parties and in Line 10, a party should wait for instruction from the winner party.

Our algorithm is based on the assumption that all the parties are semi-honest. An interesting extension would be to consider the presence of malicious and selfish parties [29]. In such scenario, our developed algorithm has to be not only secure, but also incentive compatible at the same time.

The cost of PPMashup 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 (Section 5.2). (2) Update $QIDTree_j$, $InfoGain(x)$ and $A_x(QID_j)$ for affected candidates x (Section 5.3). (3) Send "instruction" to the remote party. The instruction contains only IDs of the records in $T_A[w]$ or $T_B[w]$ and child values c in $child(w)$, therefore, is compact. Only the work in (1) involves accessing data records; the work in (2) makes use of the count statistics without accessing data records and is restricted to only affected candidates. This feature makes our approach scalable. We will evaluate the scalability in the next section. For the communication cost (3), each party communicates (Line 5 of Algorithm 1) with others to determine the global best candidate. Thus, each party sends $n - 1$ messages, where n is the number of parties. Then, the winner party (Line 8) sends instruction to other parties. This communication process continues for at most s times, where s is the number of valid specializations which is bounded by the number of distinct values in $\cup QID_j$. Hence, for a given data set, the total communication cost is $s\{n(n-1) + (n-1)\} = s(n^2 - 1) \approx O(n^2)$. If $n = 2$, then the total communication cost is $3s$. In real-life data mashup application, such as the one developed for Nordax Finans AB, the number of parties is usually small.

6. EXPERIMENTAL EVALUATION

We implemented the proposed PPMashup in a distributed 2-party web service environment. Each party is running on an Intel Pentium IV 2.6GHz PC with 1GB RAM connected to a LAN. The objective is to evaluate the benefit of data integration for data analysis. PPMashup should produce exactly the same integrated table as the single party (non-

Table 4: Attributes for the *Adult* data set

Attribute	Type	Numerical Range	
		# Leaves	# Levels
Age (A)	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 (S)	categorical	2	2
Martial-status (M)	categorical	7	4
Native-country (N)	categorical	40	5
Education (E)	categorical	16	5
Hours-per-week (H)	continuous	1 - 99	
Capital-gain (Cg)	continuous	0 - 99999	
Capital-loss (Cl)	continuous	0 - 4356	
Work-class (W)	categorical	8	5
Occupation (O)	categorical	14	3

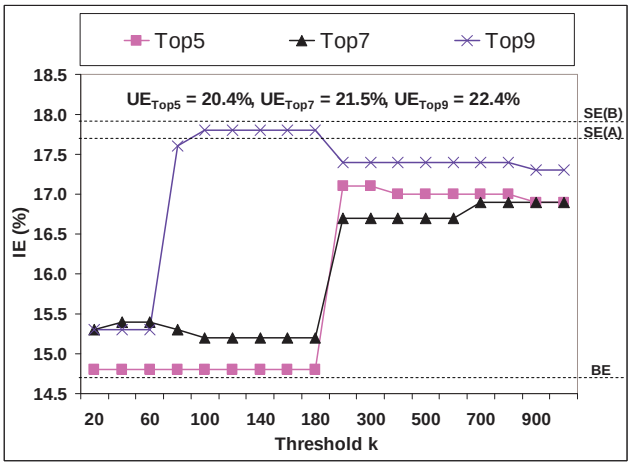

Figure 7: IE **for** Top5, Top7, **and** Top9

privacy-preserving) method that first joins T_A and T_B and then generalizes the joint table using the TDS approach.

Due to privacy agreement, we could not use the raw data of Nordax Finans AB for the experiment, so we employed the *de facto* benchmark census data set *Adult* [28], which is also a real-life dataset, to illustrate the performance of our proposed algorithm. The data set has 6 continuous attributes, 8 categorical attributes, and a binary *Class* column representing the income levels ≤50K or >50K. 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. We model two private tables T_A and T_B as follows: T_A contains the first 9 attributes 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 was added to both tables. For classification models, we used the well known C4.5 classifier [31]. Unless stated otherwise, all 14 attributes were used for building classifiers, and the taxonomy trees for all categorical attributes were from [13].

For the same anonymity threshold k, a single QID is always more restrictive than breaking it into multiple QIDs. We first show the results for 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 removed this attribute and repeated this process to determine the rank of other attributes. The top 9 attributes are $Cg, A, M, En, Re, H, S, E, O$ 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 collected 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 top most 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 our PPMashup algorithm. We combined the training set and testing set into one set, generalized this set to satisfy the given anonymity requirement, built the classifier using the generalized training set. The error is measured on the generalized testing set. *Source error*, denoted SE, is the error without data integration at all, i.e., the error of classifiers built from individual raw private table. Each party has a SE.

$SE - IE$ measures the benefit of data integration over 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.

6.1 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 7 depicts the IE for Top5, Top7, and Top9 with the anonymity threshold k ranging from 20 to 1000.[3] 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 17.2% when $k \geq 80$, suggesting that the benefit is less than 1%. In the data mashup application for Nordax Finans AB, the anonymity threshold k was set at between 20 and 50. This experiment demonstrates the benefit of data integration over a wide range of anonymity requirements. In practice, the benefit is more than the accuracy consideration because our method allows the participating parties to share information for joint data analysis.

6.2 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 of error due to generalization. For the C4.5 classifier, $BE = 14.7\%$. $UE - IE$ measures the benefit of our PPMashup 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 $UE - IE$

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

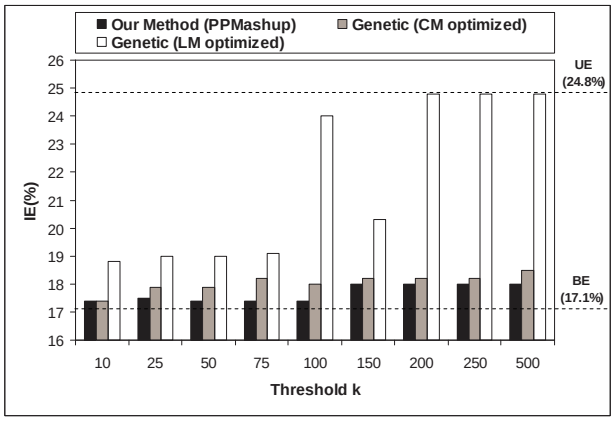

Figure 8: Comparing with genetic algorithm

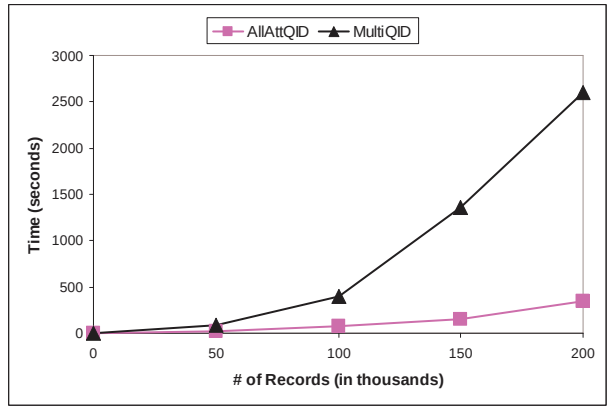

Figure 9: Scalability (k=50)

(high benefit).

Refer to Figure 7. We use the result of Top7 to summarize the analysis. First, $IE - BE$ is less than 2% for $20 \leq k \leq 600$, 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 reduce the classification error.

6.3 Comparing with Genetic Algorithm

Iyengar [19] 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 compare this method with PPMashup, we employed the data set and the single QID used in [19], both having the attributes A, W, E, M, O, Ra, S, N, and the taxonomy trees as in [19]. T_A includes A, E, M, Ra, S, N and T_B includes W, O. All errors in this experiment were based on the 10-fold cross validation. Results of the genetic algorithm were obtained from [19].

Figure 8 shows IE of PPMashup and the errors for the two methods in [19], Loss Metric (LM) ignores the classification goal. Classification Metric (CM) considers the classification goal. The error of PPMashup is clearly lower (better) than LM, suggesting that the classification quality can be improved by focusing on preserving the classification structures in the anonymous data. The error of PPMashup is at least comparable to CM. However, PPMashup took only 20 seconds to generalize the data, including reading data records from disk and writing the generalized data to disk, in a multiparty environment. Iyengar reported that his method requires 18 hours to transform this data on a Pentium III 1GHz PC with 1GB RAM. Of course, Iyengar's method does not address the secure integration requirement because of joining T_A and T_B before performing generalization.

6.4 Efficiency and Scalability

Our method took at most 20 seconds for all previous experiments. Out of the 20 seconds, approximately 8 seconds were 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.

Our other claim is the scalability of handling large data sets by maintaining count statistics instead of scanning raw records. We evaluated this claim on an enlarged version of the *Adult* data set. We combined the training and testing sets, giving 45,222 records, and for each original record r in the combined set, we created $\alpha - 1$ 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.

Figure 9 depicts the runtime of PPMashup for 50K to 200K 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. For PPMashup, the small anonymity threshold of $k = 50$ requires more iterations to reach a solution, hence more runtime, than a larger threshold. In this case, PPMashup took approximately 340 seconds to transform 200K records.

MultiQID refers to the average over the 30 random multi-QID anonymity requirements, generated as follows. For each requirement, we first determined 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 selected attributes from the 14 attributes. A repeating QID was discarded. For example, a requirement of 3 QIDs and length 2 is $\{\langle\{A, En\}, k\rangle, \langle\{A, R\}, k\rangle, \langle\{S, H\}, k\rangle\}$.

Compared to AllAttQID, PPMashup 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, PPMashup has to perform more specializations before violating the anonymity requirement. More-

over, a party needs to create one QIDTree for each related QID and maintains $a(vid)$ in QIDTrees. The time increase is roughly by a factor proportional to the number of QIDs in an anonymity requirement.

6.5 Summary

The experiments verified several claims about the PPMashup algorithm. First, data integration does lead to improved data analysis. Second, PPMashup achieves a broad range of anonymity requirements without sacrificing significantly the usefulness of data to classification. The data quality is identical or comparable to the result produced by the single party anonymization methods [12, 13, 19]. This study suggests that classification analysis has a high tolerance towards data generalization, thereby, enabling data mashup across multiple data providers even in a broad range of anonymity requirements. Third, PPMashup is scalable for large data sets and different single QID anonymity requirements. It provides a practical solution to data mashup where there is the dual need for information sharing and privacy protection.

7. PRIVACY BEYOND K-ANONYMITY

k-anonymity is an effective privacy requirement 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 his/her qid value. This type of homogeneity attack was studied in [25, 38]. The proposed approach in this paper can be extended to incorporate with other privacy requirements, such as ℓ-diversity [25], confidence bounding [38], and (α,k)-anonymity [40], to thwart homogeneity attacks.

To adopt these privacy requirements, we make 3 changes. First, the notion of valid specialization has to be redefined depending on the privacy requirement. Our PPMashup algorithm guarantees that the identified solution is local optimal if the privacy measure holds the (anti-)monotonicity property with respect to specialization. ℓ-diversity [25], confidence bounding [38], and (α,k)-anonymity [40] hold such (anti-)monotonicity property. Second, the $AnonyLoss(v)$ function in Section 4 has to be modified in order to reflect the loss of privacy with respect to a specialization on value v. We can, for example, adopt the $PrivLoss(v)$ function in [38] to capture the increase of confidence on inferring a sensitive value by a qid. Third, 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 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 the specialization as in Algorithm 1.

8. CONCLUSIONS AND LESSON LEARNED

We implemented a privacy-preserving data mashup application for some financial institutions in Sweden, and generalized their privacy and information requirements to the problem of private data mashup 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 this process. We presented a solution and evaluated the benefits of data integration and the impacts of generalization. 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 hardly 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 records 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 requirement. Despite some criticisms on k-anonymity [25, 38], 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 satisfying some privacy requirement are insufficient for them. They demand anonymization methods that can preserve information for various data analysis tasks.

9. ACKNOWLEDGEMENTS

The research is supported in part by the Discovery Grants (356065-2008) from the Natural Sciences and Engineering Research Council of Canada (NSERC).

10. REFERENCES

[1] J. M. Abowd and J. Lane. New approaches to confidentiality protection: Synthetic data, remote access and research data centers. In *Proc. of Privacy in Statistical Databases: CASC Project International Workshop (PSD 2004)*, pages 282–289, Barcelona, Spain, June 2004.

[2] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of the 2003 ACM SIGMOD*, 2003.

[3] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proc. of the 2000 ACM SIGMOD*, pages 439–450, Dallas, Texas, May 2000.

[4] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the 21st IEEE ICDE*, pages 217–228, Tokyo, Japan, 2005.

[5] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2), December 2002.

[6] U. Dayal and H. Y. Hwang. View definition and generalization for database integration in a multidatabase systems. *IEEE Transactions on Software Engineering*, 10(6):628–645, 1984.

[7] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proc. of the 4th SDM*, Florida, 2004.

[8] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Workshop on Privacy, Security, and Data Mining at the IEEE ICDM*, 2002.

[9] C. Farkas and S. Jajodia. The inference problem: A survey. *ACM SIGKDD Explorations Newsletter*, 4(2):6–11, 2003.

[10] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Official Statistics*, 9(2):383–406, 1993.

[11] B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Anonymity for continuous data publishing. In *Proc. of the 11th EDBT*, Nantes, France, March 2008.

[12] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proc. of the 21st IEEE ICDE*, Tokyo, Japan, April 2005.

[13] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(5):711–725, May 2007.

[14] J. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1984.

[15] T. Hinke. Inference aggregation detection in database management systems. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 96–107, Oakland, CA, April 1988.

[16] T. Hinke, H. Degulach, and A. Chandrasekhar. A fast algorithm for detecting second paths in database inference analysis. *Journal of Computer Security*, 1995.

[17] R. D. Hof. Mix, match, and mutate. *Business Week*, July 2005.

[18] A. Hundepool and L. Willenborg. μ- and τ-argus: Software for statistical disclosure control. In *Proc. of the 3rd International Seminar on Statistical Confidentiality*, 1996.

[19] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the 8th ACM SIGKDD*, pages 279–288, Edmonton, AB, Canada, July 2002.

[20] S. Jajodia and C. Meadows. Inference problems in multilevel database management systems. *IEEE Information Security: An Integrated Collection of Essays*, pages 570–584, 1995.

[21] W. Jiang and C. Clifton. Privacy-preserving distributed k-anonymity. In *Proc. of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 166–177, August 2005.

[22] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *Very Large Data Bases Journal (VLDBJ)*, 15(4):316–333, November 2006.

[23] J. Kim and W. Winkler. Masking microdata files. In *Proc. of the Section on Survey Research Methods*, pages 114–119, 1995.

[24] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *Proc. of the 12th ACM SIGKDD*, Philadelphia, PA, August 2006.

[25] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1), March 2007.

[26] J. M. Mateo-Sanz, A. Martínez-Ballesté, and J. Domingo-Ferrer. Fast generation of accurate synthetic microdata. In *Proceedings of Privacy in Statistical Databases: CASC Project International Workshop (PSD 2004)*, pages 298–306, Barcelona, Spain, June 2004.

[27] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *Proc. of ACM SIGMOD*, pages 575–586, Paris, France, 2004.

[28] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. http://ics.uci.edu/∼mlearn/MLRepository.html.

[29] N. Nisan. Algorithms for selfish agents. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, Trier, Germany, March 1999.

[30] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24:106–110, 1978.

[31] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[32] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge Engineering*, 13(6):1010–1027, 2001.

[33] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proc. of the 17th ACM PODS*, 1998.

[34] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[35] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the 8th ACM SIGKDD*, pages 639–644, Edmonton, AB, Canada, 2002.

[36] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proc. of the 9th ACM SIGKDD*, pages 206–215, 2003.

[37] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *Proc. of the 12th ACM SIGKDD*, pages 414–423, Philadelphia, PA, August 2006.

[38] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *KAIS*, 11(3):345–368, April 2007.

[39] G. Wiederhold. Intelligent integration of information. In *Proc. of the 1993 ACM SIGMOD*, 1993.

[40] R. C. W. Wong, J. Li., A. W. C. Fu, and K. Wang. (α,k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In *Proc. of the 12th ACM SIGKDD*, Philadelphia, PA, 2006.

[41] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proc. of the 32nd Very Large Data Bases (VLDB)*, Seoul, Korea, September 2006.

[42] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of ACM SIGMOD*, Chicago, IL, 2006.

[43] X. Xiao and Y. Tao. m-invariance: Towards privacy preserving re-publication of dynamic datasets. In *Proc. of ACM SIGMOD*, Beijing, China, June 2007.

[44] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *Proc. of the 11th ACM SIGKDD*, pages 334–343, 2005.

[45] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proc. of the 5th SDM*, pages 92–102, 2005.

[46] A. C. Yao. Protocols for secure computations. In *Proc. of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982.