Service-Oriented Architecture for Privacy-Preserving Data Mashup

Thomas Trojer
Research Group
Quality Engineering
University of Innsbruck
Innsbruck, Austria
thomas.trojer@student.uibk.ac.at

Benjamin C. M. Fung
Concordia Institute for
Information Systems Engineering
Concordia University
Montreal, QC, Canada
fung@ciise.concordia.ca

Patrick C. K. Hung
University of Ontario
Institute of Technology
Oshawa, ON, Canada
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. Therefore we propose a service-oriented architecture for privacy-preserving data mashup together with a multiparty protocol 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.

1 Introduction

Mashup is a web technology which has evolved from the strong need of integrating data from different sources. Mashup applications have been developed in recent years to support sophisticated knowledge representations in the service-oriented landscape. The idea was first presented and discussed in an issue of the Business Week [7] of 2005 on the topic of integrating real estate information into Google Maps. *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. 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 (mashup coordinator) dynamically determines the data providers, collects information from them through their web service interface (API), 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.

A data mashup application is designed to collect information accessible on arbitrary service providers. With the specification of common points for data integration, the integrated data can support clients of the mashup application to discover new knowledge for their purpose. However, there is a potential privacy risk because of the possibility of having sensitive information revealed which was impossible or not obvious before the integration. In this paper, we study the privacy threats caused by data mashup and propose a service-oriented architecture (SOA) for a privacy-preserving data mashup system together with a multiparty protocol, called *PPMashup*, to securely and efficiently 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. For illustration, we generalize their problem described 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 social security number (SSN), e.g., $T_A(SSN, Age, Balance)$ 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 classi-

978-0-7695-3709-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICWS.2009.123

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	

fication analysis. In addition to companies A and B, their partnered credit card company C also has 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.

Example 1 Consider the data in Table 1 and taxonomy trees in Figure 1. 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 $Salary$. 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*. ■

This *private data mashup* problem can be described as follows. 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 described as follows.

Privacy Requirement: The integrated table has to satisfy k-anonymity [15, 16] as follows. A data table T satisfies k-anonymity if every combination of values on a *quasi-identifier* QID is shared by at least k records in T, where the 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 the other party other than those in the final integrated table. For example, $Lawyer$ is more detailed than $Professional$.

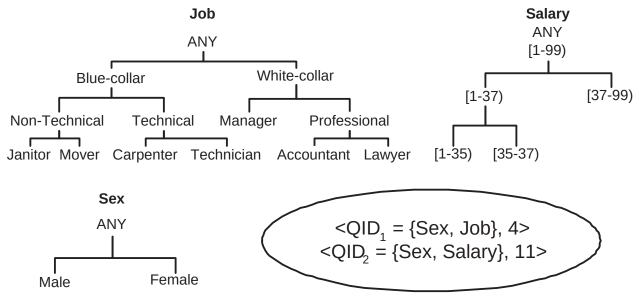

Figure 1. Taxonomy trees and QIDs

Information Requirement: The generalized data is as useful as possible to classification analysis. Generally speaking, the privacy goal requires masking 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 useful classification patterns.

Our contributions can be summarized as follows. First, 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 and Section 4). The goal is to allow data sharing and integration for classification analysis in the presence of privacy concern. Second, we propose a service-oriented architecture and a privacy-preserving protocol for multiparty data mashup (Section 5). Finally, we implement the proposed architecture and algorithm for the financial industry 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.

2 Related Work

Information integration has been an active area of database research [18]. This literature typically assumes that all information in each database can be freely shared [1]. Secure multiparty computation (SMC), on the other hand, allows sharing of the computed result (e.g., a classifier), but completely prohibits sharing of data [20], which is a primary goal of our studied problem. An example is the secure multiparty computation of classifiers [2, 3, 19].

Yang et al. [19] 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 [19]. 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.

The notion of k-anonymity was proposed in [15], and generalization was used to achieve k-anonymity in Datafly system [16] and μ-Argus system [8]. Preserving k-anonymity for classification was studied in [6, 11]. The research works [4, 17] studied the privacy threats caused by publishing multiple releases. 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. Furthermore, these works did not consider classification or a specific use of data, and used very simple heuristics to guide generalization.

Jiang and Clifton [9, 10] proposed a cryptographic approach to securely integrate two distributed data tables to a k-anonymous table without considering a data mining task.

3 Problem Definition

We first define k-anonymity on a single table and then extend it for private data mashup for 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 will be further discussed later. Each D_i is either a categorical or a continuous attribute. The $Class$ attribute 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 in 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 $1 \leq j \leq p$, 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 [5]. Note that if QID_j is a subset of QID_i, where $i \neq j$, and if $k_j \leq k_i$, then $\langle QID_j, k_j\rangle$ is redundant because if a table T satisfies $\langle QID_j, k_j\rangle$, then it must also satisfy $\langle QID_j, k_j\rangle$. $\langle QID_j, k_j\rangle$ can be removed from the anonymity requirement.

3.2 Private Data Mashup

Consider n data providers $\{$Party $1, \ldots,$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 data providers. $Attribs_y$ is a set of disjoint, private attributes. For any two data providers $y \neq z$, $Attribs_y \cap Attribs_z = \emptyset$. These data providers 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. ■

There are two obvious yet incorrect approaches. The first one is "integrate-then-generalize" which will first integrate the two tables and then generalize the integrated table using some single table anonymization methods [5, 11]. 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" which will first generalize each table locally and then integrate the generalized tables. This approach does not work for a quasi-identifier that spans multiple tables. Referring to the Example 1, the k-anonymity on (Sex,Job) cannot be achieved by the k-anonymity on each of Sex and Job separately.

4 Specialization Criteria

To generalize T, a *taxonomy tree* is specified for each categorical attribute in $\cup QID_j$. A leaf node represents a

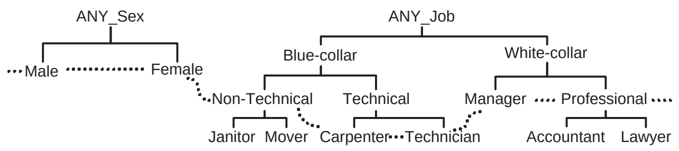

Figure 2. A solution cut for $QID_1 = \{Sex, Job\}$ indicating the most specific attribute values to use for *Sex* and *Job* to not violate the anonymity requirement.

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 1 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. 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 2 shows a solution cut indicated by the dashed curve. 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:

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

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

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

where $I(T[x])$ is the *entropy* of $T[x]$ [14]:

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

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:

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

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.

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 suitable for classification.

5 Proposed Architecture and Protocol

In this section, we first describe the proposed technical architecture shown in Figure 3 with the communication paths of all participating parties, and then followed by a privacy-preserving data mashup protocol. Referring to the architecture, the *mashup coordinator* plays the central role in initializing the protocol execution and presenting the final integrated dataset to the user. The architecture does *not* require that the mashup coordinator to be a trusted entity. This makes our architecture practical because a trusted party is often not available in real-life scenarios.

As the coordinator of the communication protocol, the mashup coordinator separates the architecture into two phases. In Phase I, the mashup coordinator receives requests from users, establishes connections with the data providers who contributes their data in a privacy-preserving manner. In Phase II, the mashup coordinator manages the privacy-preserving protocol among the data providers for a particular client request.

5.1 Phase I: Session Establishment

The objective of Phase I is to establish a common session context among the contributing data providers and the user.

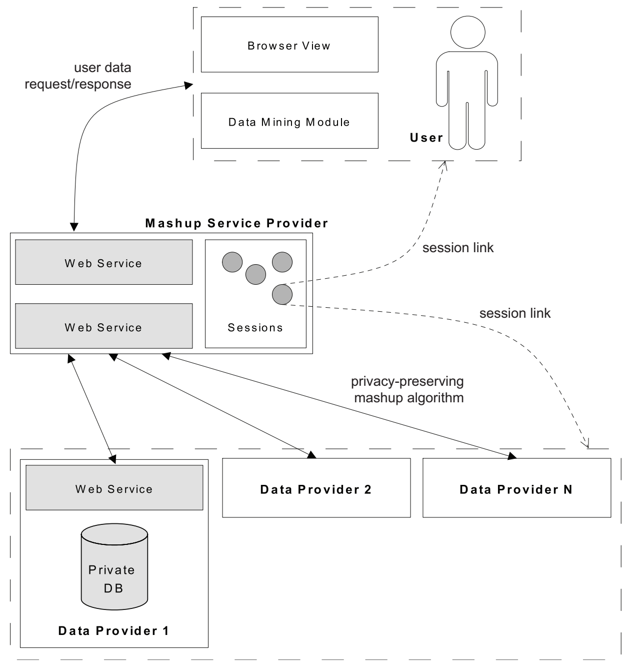

Figure 3. Service-Oriented Architecture for Privacy-Preserving Data Mashup

An operational context is successfully established by proceeding through the steps of *user authentication*, *contributing data providers identification*, *session initialization*, and *common requirements negotiation*.

Authenticate user: The mashup coordinator first authenticates a user to the requested service, generates a session token for the current user interaction, and then identifies the data providers *accessible* by the user. Some data providers are public and are accessible by any users.

Identify contributing data providers: Next, the mashup coordinator queries the data schema of the accessible data providers to identify the data providers that can contribute data for the requested service. To facilitate more efficient queries, the mashup coordinator could pre-fetch data schema from the data providers (i.e., the pull model), or the data providers could update their data schema periodically (i.e., the push model).

Initializing session context: Then, the mashup coordinator notifies all contributing data providers with the session identifier. All prospective data providers share a common session context, which represents a stateful presentation of information related to a specific execution of privacy-preserving mashup protocol called *PPMashup*, which will be discussed in Section 5.2. Due to the fact that multiple parties are involved and the flow of multiple protocol messages is needed in order to fulfill the data mashup, we propose the use of Web Service Resource Framework (WSRF) to keep stateful information along an initial service request. An established session context stored as a single web service resource contains several attributes to identify a PPMashup process, which are an unique session identifier (making use of end-point reference (EPR), which is built from service address and identifiers of the resource in use), the client address, the data provider addresses and their certificates, an authentication token (containing the user certificate), as well as additional status information.

Negotiating privacy and information requirements: The mashup coordinator is responsible to communicate the negotiation of privacy and information requirements among the data providers and the user. Specifically, this step involves negotiating the price, the anonymity requirement in Definition 3.1, and the expected information quality. For example, in the case of classification analysis, information quality can be estimated by classification error on some testing data.

5.2 Phase II: Privacy-Preserving Protocol

After a common session has been established among the data providers, the mashup coordinator initiates the privacy-preserving data mashup protocol (PPMashup) and stays back. Upon the completion of the protocol, the mashup coordinator will receive an integrated table that satisfies both, the information and anonymity requirements. There are two advantages that the mashup coordinator does not have to participate in the PPMashup protocol. First, the architecture does not require the mashup coordinator to be a trusted entity. The mashup coordinator only has access to the final integrated k-anonymous data. Second, this setup removes the computation burden from the mashup coordinator, and frees up the coordinator to handle other requests.

The rest of this section presents the PPMashup protocol for achieving both the anonymity and information requirements. One major contribution of this paper is to extend a single party anonymization algorithm, called *top-down specialization (TDS)* [5], to a multiparty privacy-preserving data mashup protocol.

The objective of TDS is to k-anonymize a single table T while preserving its information for classification analysis. 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 violates 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 $Score$

Algorithm 1 PPMashup for Party A (same for 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 $\cup Cut_i$;
```

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 violate 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 a specialization never increases the anonymity count $a(qid)$. This property guarantees that the identified k-anonymous solution must be local optimal .

Now, we consider the multiparty scenario. To ease the understanding of the protocol, we start by having two parties ($n = 2$): Party A holds private table T_A and Party B holds private table T_B, where T_A and T_B share a common key ID. At the end of this section, we generalize the protocol to multiparty with $n > 2$. Unlike the single party problem handled by TDS, this multiparty problem complicates the problem because specializing on a value of attribute D_x in one party will affect the $Score$ of other values of attribute D_y in another party, where some QID contains both D_x and D_y. In our proposed PPMashup protocol, 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, therefore, does not violate the requirement (3) of 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 protocol at Party A (same for Party B). For party i, a *local attribute* refers to an attribute in T_i, A *local specialization* refers to that of a local attribute.

Lines 4-5: Find 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 [20]) 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.

Lines 6-12: Perform winning specialization. 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)*. 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 efficiently identify 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 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.

We summarize the operations for the 2-party scenario, 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. P_c contains raw records from T_B. P_c is obtained by splitting P_w among P_c's according to the (id, c) pairs received.

Lines 13-14: Update Score and Status. The key to the scalability of our algorithm is updating $Score(x)$ using the count statistics maintained in the previous step 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 value specialized. If $A_x(QID_j) < k$ for all QID_j, then x is invalid and is removed from the $\cup Cut_i$.

Analysis: PPMashup 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.

We emphasize that updating TIPS is the only operation that accesses raw records. Subsequently, updating $Score(x)$ makes use of the count statistics without accessing raw records anymore. The TIPS data structure is the key of efficient anonymization in this protocol. In each iteration, 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 $s\{n(n-1)+(n-1)\} = s(n^2-1) \approx O(n^2)$.

6 Experimental Evaluation

To simulate the environment at Nordax Finans AB in Sweden, we implemented the proposed PPMashup in a distributed web service environment with 2 data providers and 1 mashup coordinator. To evaluate the benefit of data integration and the impacts of generalization to data analysis, we employed a real-life census data set, *Adult* [13]. The data set has 6 continuous attributes, 8 categorical attributes, and a binary $Class$ column representing the income levels $\leq$50K and $>$50K. After removing records with missing values, there are 30,162 and 15,060 records for the pre-split training and testing, respectively. T_A contains 9 attributes and T_B contains 5 attributes. They share a common key ID. For classification models, we used the well known C4.5 classifier [14]. We tested with a single QID because a single QID is always more restrictive than breaking it into multiple QIDs for the same anonymity threshold k. 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.

We collected several classification errors from the testing set. *Base error* (BE) is the error on the integrated data without generalization. *Upper bound error* (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* (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* (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.

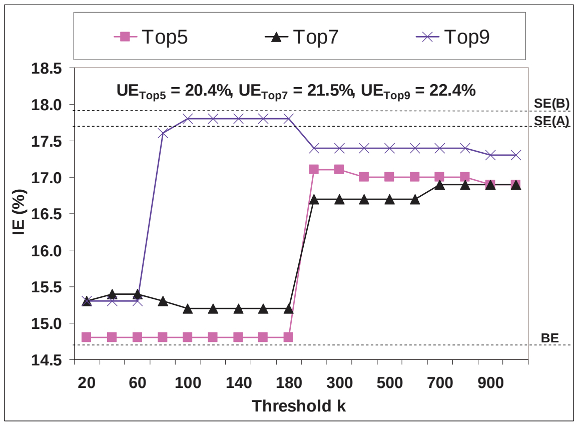

Figure 4. IE **for** Top5, Top7, **and** Top9

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 4 depicts the IE for Top5, Top7, and Top9 with the anonymity threshold k ranging from 20 to 1000.[1] 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.

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

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

$IE - BE$ (low cost) and large $UE - IE$ (high benefit).

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.

Our Architecture: Our third goal is evaluating the advantages of our proposed architecture. One first focus was to cleary seperate the requesting consumer of the mashup application from the backend process. Due to issues of convinience and control and also because a mashup coordinator represents a static point of connection between clients and providers with a high rate of availability. A mashup coordinator would also be able to cache frequently requested data tables during a period where they are valid. Requests are attached to a session token identifying a kind of contract between a consumer and several data providers and are maintained by the mashup coordinator, a generic service provider. Another benefit is that the mashup provider is able to handle and unify several service level agreements among different data providers and queues service requests according to the workload of single data providers.

7 Conclusions and Lesson Learned

We presented a real-life privacy problem faced by some financial institutes 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, and proposed a privacy-preserving data mashup architecture, together with a secure protocol, for integrating private data from multiple data providers. We evaluated the benefits of data integration and the impacts of generalization.

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 [12], 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.

References

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

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

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

[4] 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. ACM Press.

[5] 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.

[6] 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.

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

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

[9] 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, Storrs, CT, August 2005.

[10] 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.

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

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

[13] 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.

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

[15] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proc. of the 17th ACM PODS*, page 188, Seattle, WA, June 1998.

[16] 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.

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

[18] G. Wiederhold. Intelligent integration of information. In *Proc. of the 1993 ACM SIGMOD*, pages 434–437, 1993.

[19] 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, Newport Beach, CA, 2005.

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