Service-Oriented Architecture for High-Dimensional Private Data Mashup

Benjamin C.M. Fung, *Member*, *IEEE*, Thomas Trojer, *Member*, *IEEE*, Patrick C.K. Hung, *Member*, *IEEE*, Li Xiong, *Member*, *IEEE*, Khalil Al-Hussaeni, *Member*, *IEEE*, and Rachida Dssouli, *Member*, *IEEE*

Abstract—*Mashup* is a web technology that allows different service providers to flexibly integrate their expertise and 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 request. However, integrating data from multiple sources brings about three challenges: 1) Simply joining multiple private data sets together would reveal the sensitive information to the other data providers. 2) The integrated (mashup) data could potentially sharpen the identification of individuals and, therefore, reveal their person-specific sensitive information that was not available before the mashup. 3) The mashup data from multiple sources often contain many data attributes. When enforcing a traditional privacy model, such as K-anonymity, the high-dimensional data would suffer from the problem known as *the curse of high dimensionality*, resulting in useless data for further data analysis. In this paper, we study and resolve a privacy problem in a real-life mashup application for the online advertising industry in social networks, and propose a service-oriented architecture along with a privacy-preserving data mashup algorithm to address the aforementioned challenges. Experiments on real-life data suggest that our proposed architecture and algorithm is effective for simultaneously preserving both privacy and information utility on the mashup data. To the best of our knowledge, this is the first work that integrates high-dimensional data for mashup service.

Index Terms—Privacy protection, anonymity, data mashup, data integration, service-oriented architecture, high dimensionality.

1 Introduction

Mashup service is a web technology that combines information from multiple sources into a single web application. An example of a successful mashup application is the integration of real estate information into Google Maps [1], which allows users to browse on the map for properties that satisfy their specified requirements. In this paper, we focus on *data mashup*, a special type of mashup application that aims at integrating data from multiple data providers depending on the service request from a user (a data recipient). An information service request could be a general count statistic task 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, and then integrates the collected information to fulfill the service request. Further computation and visualization can be performed at the user's site or on the web application server. This is very different from a traditional web portal that simply divides a webpage or a website into independent sections for displaying information from different sources.

A data mashup application can help ordinary users explore new knowledge; it could also be misused by adversaries to reveal sensitive information that was not available before the mashup. In this paper, we study the privacy threats caused by data mashup and propose a service-oriented architecture and a privacy-preserving data mashup algorithm to securely integrate person-specific sensitive data from different data providers, wherein the integrated data still retains the essential information for supporting general data exploration or a specific data mining task.

1.1 The Challenges

The research problem presented in this paper was discovered in a collaborative project with a social network company, which focuses on the gay and lesbian community in North America. The problem can be generalized as follows: social network companies A and B observe different sets of attributes about the same set of individuals (members) identified by the common User ID, e.g., $T_A(UID, Gender, Salary)$ and $T_B(UID, Job, Age)$. Every time a social network member visits another member's webpage, an advertisement is chosen to be displayed. Companies A and B want to

- *B.C.M. Fung is with CIISE (EV7.640), Concordia University, 1455 De Maisonneuve Blvd. West, Montreal, QC H3G 1M8, Canada. E-mail: fung@ciise.concordia.ca.*
- *T. Trojer is with University of Innsbruck, Room 2M01, ICT Building, Technikerstrasse 21a, Innsbruck 6020, Austria. E-mail: thomas.trojer@uibk.ac.at.*
- *P.C.K. Hung is with University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada. E-mail: patrick.hung@uoit.ca.*
- *L. Xiong is with Emory University, 400 Dowman DR, Atlanta, GA 30322. E-mail: lxiong@mathcs.emory.edu.*
- *K. Al-Hussaeni is with the Department of Electrical and Computer Engineering, Concordia University, 1455 De Maisonneuve Blvd. West, Montreal, QC H3G 1M8, Canada. E-mail: k_alhus@encs.concordia.ca.*
- *R. Dssouli is with CIISE (EV7.640), Concordia University, 1455 De Maisonneuve Blvd. West, Montreal, QC H3G 1M8, Canada, and Research Cluster Hire, FIT, United Arab Emirate University. E-mail: dssouli@ciise.concordia.ca.*

Manuscript received 3 Apr. 2010; revised 3 Dec. 2010; accepted 28 Jan. 2011; published online 17 Feb. 2011.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2010-04-0029.
Digital Object Identifier no. 10.1109/TSC.2011.13.

TABLE 1
Raw Data

Shared		Data Provider A		Data Provider B	
UID	Class	Sensitive	Gender	Job	Age
1	Y	s1	M	Janitor	34
2	N	s2	M	Doctor	58
3	Y	s1	M	Mover	34
4	N	s2	M	Lawyer	24
5	N	s2	M	Mover	58
6	Y	s2	M	Janitor	44
7	N	s2	M	Doctor	24
8	N	s2	F	Lawyer	58
9	N	s2	F	Doctor	44
10	Y	s2	F	Carpenter	63
11	Y	s2	F	Technician	63

implement a data mashup application that integrates their membership data, with the goal of improving their advertisement selection strategy. The analysis includes gathering general count statistics and building classification models [2]. In addition to companies A and B, other partnered advertising companies need access to the final mashup data. The solution presented in this paper is not limited only to the social networks sector but is also applicable to other similar data mashup scenarios. The challenges of developing the data mashup application are summarized as follows.

Challenge #1: Privacy concerns. The members are willing to submit their personal data to a social network company because they consider the company and its developed system to be trustworthy. Yet, trust to one party may not necessarily be transitive to a third party. Many agencies and companies believe that privacy protection means simply removing explicit identifying information from the released data, such as name, social security number, address, and telephone number. However, many previous works [3], [4] show that removing explicit identifying information is insufficient. An individual can be re-identified by matching other attributes called *quasi-identifiers* (QID). The following example illustrates potential privacy threats.

Example 1. Consider the membership data in Table 1. Data Provider A and Data Provider B own data tables $T_A(UID, Class, Sensitive, Gender)$ and $T_B(UID, Class, Job, Age)$, respectively. Each row (record) represents a member's information. The two parties want to develop a data mashup service to integrate their membership data in order to perform classification analysis on the shared $Class$ attribute with two class labels Y and N, representing whether or not the member has previously bought any items after following the advertisements on the social network websites. Let $QID = \{Job, Gender, Age\}$. After integrating the two tables (by matching the shared UID field), there are two types of privacy threats:

Record linkage. If a record in the table is so specific that not many members match it, releasing the data may lead to linking the member's record and his/her sensitive value. Let $s1$ be a sensitive value in Table 1. Suppose that the adversary knows the target member is a $Mover$ and his age is 34. Hence, record $\#3$, together with his sensitive value ($s1$ in this case), can be uniquely identified since he is the only $Mover$ who is 34 years old.

Attribute linkage. If a sensitive value occurs frequently along with some QID attributes, then the sensitive information can be inferred from such attributes, even though the exact record of the member cannot be identified. Suppose the adversary knows that the member is a male (M) of age 34. In such case, even though there are two such records (#1 and #3), the adversary can infer that the member has sensitive value $s1$ with 100 percent confidence since both records contain $s1$.

Many privacy models, such as K-anonymity [3], [4], ℓ-diversity [5], and confidence bounding [6], have been proposed to thwart privacy threats caused by record and attribute linkages in the context of relational databases owned by a *single data provider*. The traditional approach is to generalize the records into equivalence groups so that each group contains at least K records sharing the same qid value on the QID, and so that each group contains sensitive values that are diversified enough to disorient confident inferences. The privacy models can be achieved by generalizing domain values into higher level concepts and, therefore, more abstract concepts.

The data mashup problem further complicates the privacy issue because the data are owned by *multiple parties*. In addition to satisfying a given privacy requirement in the final mashup data, at any time during the process of generalization no data provider should learn more detailed information about any other data provider other than the data in the final mashup table. In other words, the generalization process must not leak more specific information other than the final mashup data. For example, if the final table discloses that a member is a $Professional$, then no other data providers should learn whether she is a $Lawyer$ or an $Accountant$. There are two obvious yet incorrect approaches. The first one is *mashup-then-generalize*: first integrate the two tables and then generalize the mashup table using some single table anonymization methods [7], [8], [9], [10]. This approach does not preserve privacy in the studied scenario because any data provider holding the mashup table will immediately know all private information of both data providers. The second approach is *generalize-then-mashup*: first generalize each table locally and then integrate the generalized tables. This approach fails to guarantee the privacy for a quasi-identifier that spans multiple tables. In the above example, the K-anonymity on ($Gender$, Job) cannot be achieved by the K-anonymity on each of $Gender$ and Job separately.

Challenge #2: High dimensionality. The mashup data from multiple data providers usually contain many attributes. Enforcing traditional privacy models on high-dimensional data would result in significant information loss. As the number of attributes increases, more generalization is required in order to achieve K-anonymity even if K is small, thereby resulting in data useless for further analysis. This challenge, known as *the curse of high dimensionality on K-anonymity*, is confirmed by [8], [11], [12]. To overcome this bottleneck, we exploit one of the limitations of the adversary: in real-life privacy attacks, it is very difficult for an adversary to acquire *all* the information of a target victim because it requires nontrivial effort to gather each piece. Thus, it is

TABLE 2
Anonymous Mashup Data ($L = 2$, $K = 2$, $C = 0.5$)

Shared		Data Provider A		Data Provider B	
UID	Class	Sensitive	Gender	Job	Age
1	Y	s1	M	Non-Technical	[30 − 60)
2	N	s2	M	Professional	[30 − 60)
3	Y	s1	M	Non-Technical	[30 − 60)
4	N	s2	M	Professional	[1 − 30)
5	N	s2	M	Non-Technical	[30 − 60)
6	Y	s2	M	Non-Technical	[30 − 60)
7	N	s2	M	Professional	[1 − 30)
8	N	s2	F	Professional	[30 − 60)
9	N	s2	F	Professional	[30 − 60)
10	Y	s2	F	Technical	[60 − 99)
11	Y	s2	F	Technical	[60 − 99)

reasonable to assume that the adversary's prior knowledge is bounded by at most L values of the QID attributes. Based on this assumption, in this paper we extend the privacy model called *LKC-privacy* [13], originally proposed for a *single party* scenario, to apply to a *multiparty* data mashup scenario.

The general intuition of LKC-privacy is to ensure that every combination of values in $QID_j \subseteq QID$ with maximum length L in the data table T is shared by at least K records, and the confidence of inferring any sensitive values in S is not greater than C, where L, K, C are thresholds and S is a set of sensitive values specified by the data provider. LKC-privacy limits the probability of a successful record linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$, provided that the adversary's prior knowledge does not exceed L.

Table 2 shows an example of an anonymous table that satisfies $(2, 2, 50\%)$-privacy by generalizing the values from Table 1 according to the taxonomies in Fig. 1. (The dashed curve can be ignored for now.) Every possible value of QID_j with maximum length 2 in Table 2 (namely, $QID_1 = \{Job, Gender\}$, $QID_2 = \{Job, Age\}$, and $QID_3 = \{Gender, Age\}$) is shared by at least two records, and the confidence of inferring the sensitive value $s1$ is not greater than 50 percent. In contrast, enforcing traditional 2-anonymity with respect to $QID = \{Gender, Job, Age\}$ will require further generalization. For example, in order to make $\langle Professional, M, [30\text{-}60)\rangle$ satisfy traditional 2-anonymity, we may further generalize [1-30) and [30-60) to [1-60), resulting in much higher information utility loss.

Challenge#3: Information requirements. The data recipients want to obtain general count statistics from the mashup membership information. Also, they want to use the mashup data as training data for building a classification model on the $Class$ attribute, with the goal of predicting the behavior of future members. One frequently raised question is: to avoid privacy concerns, why doesn't the data provider release the statistical data or a classifier to the data recipients? In many real-life scenarios, releasing data is preferable to releasing statistics for several reasons. First, the data providers may not have in-house experts to perform data mining. They just want to share the data with their partners. Second, having access to the data, data recipients are flexible to perform the required data analysis. It is impractical to continuously request data providers to produce different types of statistical information or to fine-tune the data mining results for research purposes for the data recipients.

1.2 Contributions

This paper is the first work that addresses all the aforementioned challenges in the context of mashup service. The contributions are summarized as follows.

Contribution #1. We identify a new privacy problem through a collaboration with the social networks industry and generalize the industry's requirements to formulate the privacy-preserving high-dimensional data mashup problem (Section 3). The problem is to dynamically integrate data from different sources for joint data analysis in the presence of privacy concerns.

Contribution #2. We present a service-oriented architecture (Section 4) for privacy-preserving data mashup in order to securely integrate private data from multiple parties. The generalized data have to be as useful as possible to data analysis. Generally speaking, the privacy goal requires anonymizing identifying information that is *specific* enough to pinpoint individuals, whereas the data analysis goal requires extracting *general* trends and patterns. If generalization is *carefully* performed, it is possible to anonymize identifying information while preserving useful patterns.

Contribution #3. Data mashup often involves a large volume of data from multiple data sources. Thus, scalability plays a key role in a data mashup system. After receiving a request from a data recipient, the system dynamically identifies the data providers and performs the data mashup. Experimental results (Section 5) on real-life data suggest that our method can effectively achieve a privacy requirement without compromising the information utility, and the proposed architecture is scalable to large data sets.

2 Related Work

Information integration has been an active area of database research [15], [16]. This literature typically assumes that all information in each database can be freely shared [17]. *Secure multiparty computation* (*SMC*) [18], [19], [20], on the other hand, allows sharing of the computed result (e.g., a classifier), but completely prohibits sharing of data. An

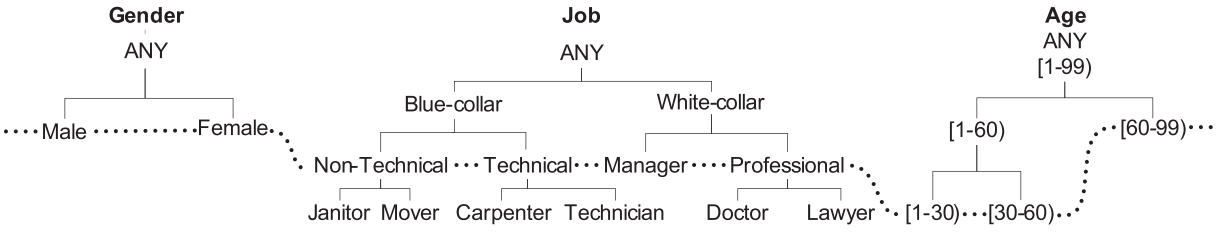


Fig. 1. Taxonomy trees and QIDs.

example is the secure multiparty computation of classifiers [21], [22], [23]. In contrast, the privacy-preserving data mashup problem studied in this paper allows data providers to share data, not only the data mining results. In many applications, data sharing gives greater flexibility than result sharing because the data recipients can perform their required analysis and data exploration [8].

Samarati and Sweeney [24] propose the notion of K-anonymity. Datafly system [4] and μ-Argus system [25] use generalization to achieve K-anonymity. Preserving classification information in K-anonymous data is studied in [8] and [10]. Mohammed et al. [13] extend the work to address the problem of high-dimensional anonymization for the healthcare sector using LKC-privacy. All these works consider a single data source; therefore, data mashup is not an issue. Joining all private databases from multiple sources and applying a single table anonymization method fails to guarantee privacy if a QID spans across multiple private tables. Recently, Mohammed et al. [14] propose an algorithm to address the horizontal integration problem, while our paper addresses the vertical integration problem.

Jiang and Clifton [26], [27] propose a cryptographic approach and Mohammed et al. [28] propose a top-down specialization algorithm to securely integrate two vertically partitioned distributed data tables to a K-anonymous table, and further consider the participation of malicious parties in [29]. Trojer et al. [30] present a service-oriented architecture for achieving K-anonymity in the privacy-preserving data mashup scenario. Our paper is different from these previous works [26], [27], [28], [29], [30] in two aspects. First, our LKC-privacy model provides a stronger privacy guarantee than K-anonymity because K-anonymity does not address the privacy attacks caused by attribute linkages, as discussed in Section 1. Second, our method can better preserve information utility in *high-dimensional* mashup data. High dimensionality is a critical obstacle for achieving effective data mashup because the integrated data from multiple parties usually contain many attributes. Enforcing traditional K-anonymity on high-dimensional data will result in significant information loss. Our privacy model resolves the problem of high dimensionality. This claim is also supported by our experimental results.

Yang et al. [23] develop a cryptographic approach to learn classification rules from a large number of data providers while 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 provider. The output of their method is a classifier, but the output of our method is an anonymous mashup data that supports general data analysis or classification analysis. Jurczyk and Xiong [31], [32] present a privacy-preserving distributed data publishing for horizontally partitioned databases. The mashup model studied in this paper can be viewed as a vertically partitioned data table, which is very different from the model studied in [23], [31], and [32].

Jackson and Wang [33] present a secure communication mechanism that enables cross-domain network requests and client-side communication with the goal of protecting the mashup controller from malicious code through web services. In contrast, this paper aims to preserve the privacy and information utility of the mashup data.

3 Problem Definition

We first define the LKC-privacy model [13] and the information utility measure on a single data table, then extend it for privacy-preserving high-dimensional data mashup from multiple parties.

3.1 Privacy Measure

Consider a relational data table $T(UID, D_1, \ldots, D_m, S_1, \ldots, S_e, Class)$ (e.g., Table 1). UID is an explicit identifier, such as User ID or SSN. In practice, it should be replaced by a pseudo identifier, such as a record ID, before publication. We use UID to ease the discussion only. Each D_i is either a categorical or numerical attribute. Each S_j is a categorical sensitive attribute. A record has the form $\langle v_1, \ldots, v_m, s_1, \ldots, s_e, cls\rangle$, where v_i is a domain value in D_i, s_j is a sensitive value in S_j, and cls is a class value in $Class$. The data provider wants to protect against linking an individual to a record or some sensitive value in T through some subset of attributes called a *quasi-identifier* $QID \subseteq \{D_1, \ldots, D_m\}$.

One data recipient, who is an adversary, seeks to identify the record or sensitive values of some target victim V in T. As explained in Section 1, we assume that the adversary knows at most L values of QID attributes of the victim. We use qid to denote such prior known values, where $|qid| \leq L$. Based on the prior knowledge qid, the adversary could identify a group of records, denoted by $T[qid]$, that contains qid. $|T[qid]|$ denotes the number of records in $T[qid]$. The adversary could launch two types of privacy attacks based on $T[qid]$.

- *Record linkage.* Given prior knowledge qid, $T[qid]$ is a set of candidate records that contains the victim Vs record. If the group size of $T[qid]$, denoted by $|T[qid]|$, is small, then the adversary may identify Vs record from $T[qid]$ and, therefore, Vs sensitive value. For example, if $qid = \langle Mover, 34\rangle$ in Table 1, $T[qid] = \{UID\#3\}$ and $|T[qid]| = 1$. Thus, the adversary can easily infer that V has sensitive value $s1$.
- *Attribute linkage.* Given prior knowledge qid, the adversary can identify $T[qid]$ and infer that V has sensitive value s with confidence $P(s|qid) = \frac{|T[qid \wedge s]|}{|T[qid]|}$, where $T[qid \wedge s]$ denotes the set of records containing both qid and s. $P(s|qid)$ is the percentage of the records in $T[qid]$ containing s. The privacy of V is at risk if $P(s|qid)$ is high. For example, given $qid = \langle M, 34\rangle$ in Table 1, $T[qid \wedge s1] = \{UID\#1, 3\}$ and $T[qid] = \{UID\#1, 3\}$, hence $P(s1|qid) = 2/2 = 100\%$.

To thwart the record and attribute linkages on *any* individual in the table T, we require every qid with a maximum length L in the anonymous table to be shared by at least a certain number of records, and the percentage of sensitive value(s) in every group cannot be too high. The privacy model, LKC-privacy [13], reflects this intuition.

Definition 3.1 (LKC-Privacy). *Let L be the maximum number of values of the adversary's prior knowledge. Let $S \subseteq \cup S_j$ be a set of sensitive values. A data table T satisfies LKC-privacy if and only if for any qid with $|qid| \leq L$,*

1. *$|T[qid]| \geq K$, where $K > 0$ is an integer representing the anonymity threshold, and*
2. *$P(s|qid) \leq C$ for any $s \in S$, where $0C \leq 1$ is a real number representing the confidence threshold.*

The data provider specifies the thresholds L, K, and C. The maximum length L reflects the assumption of the adversary's power. LKC-privacy guarantees the probability of a successful record linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$. Sometimes, we write C in percentage. LKC-privacy has several nice properties that make it suitable for anonymizing high-dimensional data. First, it only requires a subset of QID attributes to be shared by at least K records. This is a major relaxation from traditional K-anonymity, based on a very reasonable assumption that the adversary has limited power. Second, LKC-privacy generalizes several traditional privacy models. K-anonymity [3], [4] is a special case of LKC-privacy with $L = |QID|$ and $C = 100\%$, where $|QID|$ is the number of QID attributes in the data table. Confidence bounding [6] is also a special case of LKC-privacy with $L = |QID|$ and $K = 1$. (α, k)-anonymity [34] is a special case of LKC-privacy with $L = |QID|$, $K = k$, and $C = \alpha$. One instantiation of ℓ-diversity is also a special case of LKC-privacy with $L = |QID|$, $K = 1$, and $C = 1/\ell$. Thus, the data provider can still achieve the traditional models.

3.2 Utility Measure

The measure of information utility varies depending on the user's specified information service request and the data analysis task to be performed on the mashup data. Based on the information requirements specified by the social network data providers, we define two utility measures. First, we aim at preserving the maximal information for classification analysis. Second, we aim at minimizing the overall data distortion when the data analysis task is unknown.

In this paper, the general idea in anonymizing a table is to perform a sequence of specializations starting from the topmost general state in which each attribute has the topmost value of its taxonomy tree. We assume that a *taxonomy tree* is specified for each categorical attribute in QID. A leaf node represents a domain value and a parent node represents a less specific value. For numerical attributes in QID, taxonomy trees can be grown at runtime, where each node represents an interval, and each nonleaf node has two child nodes representing some optimal binary split of the parent interval [2]. Fig. 1 shows a dynamically grown taxonomy tree for *Age*.

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 resulting table still satisfies the specified LKC-privacy requirement after the specialization. A specialization is performed only if it is valid. The specialization process can be viewed as pushing the "cut" of each taxonomy tree downward. A *cut* of the taxonomy tree for an attribute D_i, denoted by Cut_i, contains exactly one value on each root-to-leaf path. Fig. 1 shows a solution cut indicated by the dashed curve representing the LKC-privacy preserved Table 2. The specialization procedures start from the topmost cut and iteratively pushes down the cut by specializing some value in the current cut until violating the LKC-privacy requirement. In other words, the specialization process pushes the cut downward until no valid specialization is possible. Each specialization tends to increase information utility and decrease privacy because records are more distinguishable by specific values. We define two utility measures (scores) to evaluate the "goodness" of a specialization depending on the information service request requirement.

3.2.1 Utility Measure for Classification Analysis

For the requirement of classification analysis, we use *information gain* [2] to measure the *goodness* of a specialization. Let $T[x]$ denote the set of records in table T generalized to the value x. Let $|T[x \wedge cls]|$ denote the number of records in $T[x]$ having the class cls. Note that $|T[v]| = \sum_c |T[c]|$, where $c \in child(v)$. Our selection criterion, $Score(v)$, is to favor the specialization $v \rightarrow child(v)$ that has the maximum information gain

$$Score(v) = E(T[v]) - \sum_c \frac{|T[c]|}{|T[v]|} E(T[c]), \quad (1)$$

where $E(T[x])$ is the *entropy* [35] of $T[x]$ and

$$E(T[x]) = -\sum_{cls} \frac{|T[x \wedge cls]|}{|T[x]|} \times log_2 \frac{|T[x \wedge cls]|}{|T[x]|}. \quad (2)$$

Intuitively, $E(T[x])$ measures the mix of classes for the records in $T[x]$, and the information gain of v (or $Score(v)$ in this case) is the reduction of the mix by specializing v into $c \in child(v)$.

For a numerical attribute, the specialization of an interval refers to the optimal binary split that maximizes information gain on the $Class$ attribute. See [2] for details.

3.2.2 Utility Measure for General Data Analysis

Sometimes, the mashup data are shared without a specific task. In this case of general data analysis, we use *discernibility cost* [36] to measure the data distortion in the anonymous data. The discernibility cost charges a penalty to each record for being indistinguishable from other records. For each record in an equivalence group qid, the penalty is $|T[qid]|$. Thus, the penalty on a group is $|T[qid]|^2$. To minimize the discernibility cost, we choose the specialization $vchild(v)$ that maximizes the value of

$$Score(v) = \sum_{qid_v} |T[qid_v]|^2, \quad (3)$$

over all qid containing v. Example 3 shows the computation of $Score(v)$ in more details.

3.3 Privacy-Preserving High-Dimensional Data Mashup

Consider n data providers $\{\text{Provider } 1, \ldots, \text{Provider } n\}$, where each Provider y owns a private table $T_y(UID, QID_y, S_y, Class)$ over the same set of records. UID and $Class$ are shared attributes among all data providers. QID_y is a set of quasi-identifying attributes and S_y is a set of sensitive values owned by provider y. $QID_y \cap QID_z = \emptyset$ and $S_y \cap S_z = \emptyset$ for any $1 \leq y, z \leq n$. These providers agree to release "minimal information" to form a mashup table T (by matching the UID) for conducting general data analysis or a joint classification analysis. The notion of minimal information is specified by an LKC-privacy requirement on the mashup table. A QID_j is *local* if all attributes in QID_j are owned by one provider; otherwise, it is *global*.

Definition 3.2 (Privacy-Preserving High-Dimensional Data Mashup). *Given multiple private tables* $T_1, \ldots, T_n$,

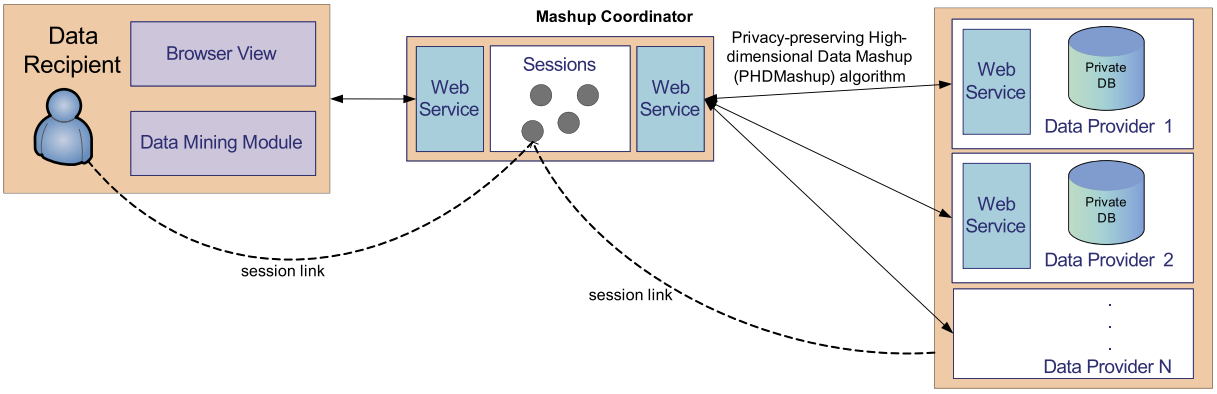


Fig. 2. Service-oriented architecture for privacy-preserving data mashup.

an LKC-privacy requirement, $QID = \cup QID_y$, $S \subseteq \cup S_y$, and a taxonomy tree for each categorical attribute in QID, the problem of privacy-preserving high-dimensional data mashup is to efficiently produce a generalized integrated (mashup) table T such that 1) T satisfies the LKC-privacy requirement, 2) T contains as much information as possible for general data analysis or classification analysis, and 3) each data provider learns nothing about the other providers' data more specific than what is in the final mashup table T. We assume that the data providers are semihonest [18], [19], [20], meaning that they will follow the algorithm but may attempt to derive sensitive information from the received data.

We use an example to explain condition 3. If a record in the final table T has values F and *Professional* on $Gender$ and Job, respectively. Condition 3 is violated if Provider A can learn that $Professional$ in this record comes from $Lawyer$. Our privacy model ensures the privacy protection in the final mashup table as well as in any intermediate tables. To ease the explanation, we present our solution in a scenario of two parties ($n = 2$). A discussion is given in Section 4.3 to describe the extension to multiple parties.

Given a QID, there are $\binom{|QID|}{L}$ combinations of decomposed QID_j with maximum size L. For any value of K and C, each combination of QID_j in LKC-privacy is an instance of the (α, k)-anonymity problem with $\alpha = C$ and $k = K$. Wong et al. [34] have proven that computing the optimal (α, k)-anonymous solution is NP-hard; therefore, computing the optimal LKC-privacy solution is also NP-hard.

4 SOA for Privacy-Preserving Data Mashup

We first present a service-oriented architecture (SOA) that describes the communication paths of all participating parties, followed by a privacy-preserving high-dimensional data mashup algorithm that can efficiently identify a suboptimal solution for the problem described in Definition 3.

SOA is an architectural paradigm for developing and integrating heterogeneous information systems with strict message-driven communication paradigm. Following the SOA design principles, the resulting system has several desirable properties including *interoperability* and *loosely coupling*. Interoperability refers to the capability of allowing platform-independent design of the system components based on a common understanding of service component interfaces. Loosely coupling refers to the capability of minimizing dependencies among the system components and, therefore, improving the overall flexibility, scalability, and fault tolerance of a system [37]. In the mashup system described in this paper, data sources can be dynamically composed to serve new mashup requests depending on the data analysis tasks and privacy requirements. SOA with the capabilities of interoperability and loosely coupling has become a natural choice to tackle the heterogeneity of different potential data providers.

Referring to the architecture shown in Fig. 2, the data mashup process can be divided into two phases. In Phase I, the mashup coordinator receives an information service request from the data recipient and establishes connections with the data providers who can contribute their data to fulfill the request. In Phase II, the mashup coordinator executes the privacy-preserving algorithm to integrate the private data from multiple data providers and to deliver the final mashup data to the data recipient. Note that our proposed solution does *not* require the mashup coordinator to be a trusted party. Though the mashup coordinator manages the entire mashup service, our solution guarantees that the mashup coordinator does not gain more information than the final mashup data, thereby protecting the data privacy of every participant. The mashup coordinator can be any one of the data providers or an independent party. This makes our architecture practical because a trusted party is not always available in real-life mashup scenarios.

4.1 Phase I: Session Establishment

The objective of Phase I is to establish a common session context between the data recipient and the contributing data providers. An operational context is successfully established by proceeding through the steps of data recipient authentication, contributing data providers identification, session context initialization, and common requirements negotiation.

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

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 prefetch 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).

Initialize session context. Next, the mashup coordinator notifies all contributing data providers with the session identifier. All prospective data providers share a common session context that represents a stateful presentation of information related to a specific execution of the privacy-preserving mashup algorithm called *PHDMashup*. This algorithm will be discussed in Section 4.2. An established session context contains several attributes to identify a PHDMashup process, including the data recipient's address; the data providers' addresses and certificates; an authentication token that contains the data recipient's certificate; and a unique session identifier that uses an *end-point reference* (*EPR*) composed of the service address, a PHDMashup process identifier and runtime status information about the executed PHDMashup algorithm.

Negotiate privacy and information requirements. The mashup coordinator is responsible to communicate the negotiation of privacy and information requirements among the data providers and the data recipient. Specifically, this step involves negotiating cost, LKC-privacy requirement, sensitive information, and expected information quality. For example, in the case of classification analysis, information quality can be estimated by classification error on some testing data.

4.2 Phase II: Privacy-Preserving High-Dimensional Data Mashup

The objective of Phase II is to integrate the high-dimensional data from multiple data providers such that the final mashup data satisfies a given LKC-privacy requirement and preserves as much information as possible for the specified information requirement. Recall that Definition 3.1 specifies three requirements. Requirements 1 and 2 specify the properties of the final mashup data. Requirement 3 states that no data provider should learn more detailed information than the final mashup data *during* the process of integration. To satisfy requirement 3, we propose a top-down specialization approach called *Privacy-preserving High-dimensional Data Mashup* (*PHDMashup*). We first present an overview of the algorithm followed by the details of each step.

To ease the discussion, we present the algorithm in a scenario of two data providers. An extension to multiple (>2) data providers will be given in Section 4.3. Consider two private tables T_A and T_B with a common key UID, where Provider A holds T_A and Provider B holds T_B. Initially, every data provider generalizes all of its own attribute values to the topmost value according to the taxonomy trees, and maintains a cut Cut_i that contains the topmost value for each attribute D_i in QID. The union cut $\cup Cut_i$ on all attributes represents a generalized table T, denoted by T_g. $\cup Cut_i$ also contains the set of candidates for specialization. A specialization $vchild(v)$ is *valid*, written as $IsValid(v)$, if the table T_g still satisfies the LKC-privacy requirement *after* the specialization on v. At each iteration, PHDMashup identifies the winner specialization, i.e., the valid candidate that has the highest $Score$, among all the candidates, performs the winner specialization, and updates the $Score$ and the $IsValid$ status of the new and existing candidates in the cut. PHDMashup terminates when there are no valid candidates in the cut.

Note, there is no need to further specialize a value once it becomes invalid because any further specializations also must be invalid. This *antimonotonic* [12] property of LKC-privacy with respect to a specialization significantly reduces the search space and ensures that the resulting solution is suboptimal.

Algorithm 1 describes the procedure of Provider A (same as Provider B). Provider A finds the local winner specialization using the utility measure discussed in Section 3.2, and communicates with Provider B to identify the global winner candidate, denoted by w. Suppose that w is local to Provider A (otherwise, the discussion below applies to Provider B). Provider A performs $wchild(w)$ 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, t'_z containing child values in $child(w)$. Similarly, Provider B updates its $\cup Cut_i$ and T_g, and partitions $T_B[t]$ into $T_B[t'_1], T_B[t'_z]$. Since Provider B does not have the attribute for w, Provider A needs to instruct Provider B how to partition these records in terms of UIDs.

Algorithm 1. PHDMashup for Provider A (same as Provider B)

1: initialize T_g to include one record containing topmost values;
2: initialize $\cup Cut_i$ to include only topmost values and update $IsValid(v)$ for every $v \in \cup Cut_i$;
3: **while** $\exists v \in \cup Cut_i$ s.t. $IsValid(v)$ **do**
4: find the local winner α that has the highest $Score(\alpha)$;
5: communicate $Score(\alpha)$ with Provider B to determine the global winner w;
6: **if** the winner w is local **then**
7: specialize w on T_g;
8: instruct Provider B to specialize w;
9: **else**
10: wait for the instruction from Provider 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 $Score(x)$ and $IsValid(x)$ for every candidate $x \in \cup Cut_i$;
15: **end while**
16: **return** T_g and $\cup Cut_i$

The nature of the top-down approach implies that T_g is *always* more general than the final mashup table and, therefore, does not violate requirement 3 in Definition 3.1. At each iteration, the data providers cooperate to perform the same identified specialization by communicating some count statistics information that satisfies requirement 3 in Definition 3.1. Below, we describe the key steps: find the winner candidate (Lines 4-5), perform the winner specialization (Lines 7-11), and update the score and status of candidates (Line 14). For Provider A, a *local attribute* refers to an attribute from T_A.

Example 2. Consider Table 1 and the LKC-privacy requirement with $L = 2$, $K = 2$, $C = 50\%$, $QID = \{Gender, Job, Age\}$, and $S = \{s_1\}$. Initially,

$$T_g = \langle ANY\_Gender, ANY\_Job, [1\text{-}99) \rangle$$

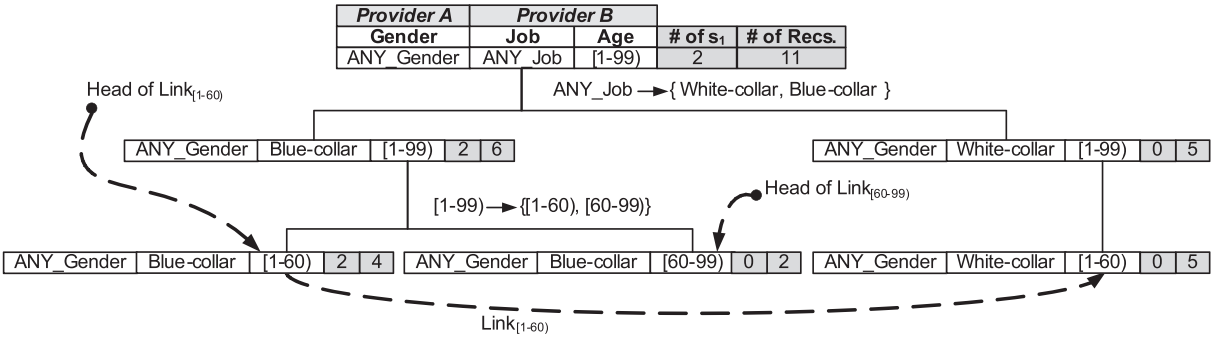


Fig. 3. The TIPS data structure.

and

$$\cup Cut_i = \{ANY\_Gender, ANY\_Job, [1\text{-}99)\},$$

and all specializations in $\cup Cut_i$ are candidates. To find the winner candidate w, Provider A computes $Score(ANY\_Gender)$ and Provider B computes $Score(ANY\_Job)$ and $Score([1\text{-}99))$.

4.2.1 Find the Winner Specialization

Provider A first finds a valid candidate α that has the highest $Score(\alpha)$ among all the local attribute values in $\cup Cut_i$. Then, Provider A communicates $Score(\alpha)$ with Provider B to determine the global winner specialization, denoted by w, that has the highest $Score(w)$ among all candidates in $\cup Cut_i$. $Score(x)$ and $IsValid(x)$ for every $x \in \cup Cut_i$ are updated from Line 14 in the previous iteration or the initialization prior to the first iteration. The updating procedure is discussed in Section 4.2.3. In case all local candidates are invalid, Provider A sends a default $Score = -1$ to Provider B to indicate its status. To avoid sharing the $Score$ between the data providers, PHDMashup employs the secure multiparty maximum protocol [20] to determine the highest $Score$ among the participating data providers.

4.2.2 Perform the Winner Specialization

Suppose that the winner specialization w is local at Provider A (otherwise, replace Provider A with Provider B in the procedure). For each record t in T_g containing w, Provider 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)*. The idea is to group the raw records in T_A according to their generalized records t in T_g.

Definition 4.1 (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 partitions P_x.*

With TIPS, we can efficiently find all raw records generalized to x by following $Link_x$ for any candidate x in $\cup Cut_i$. To ensure that every data provider has only access to its own raw records, a leaf partition at Provider A contains only raw records from T_A and a leaf partition at Provider B contains only raw records from T_B. Each provider maintains a local copy of TIPS, representing the current state of the integrated table. Initially, the TIPS has only the root node representing all data records with the most general values. In each iteration, the two data providers cooperate to perform the specialization w by refining the leaf partitions P_w on $Link_w$ in their own copy of TIPS. As both providers perform exactly the same specialization, both copies of TIPS are always identical after every specialization.

Example 3. Continue with Example 2. Initially, TIPS has only one partition containing all data records and representing the generalized record $\langle ANY\_Gender, ANY\_Job, [1\text{-}99)\rangle$, $T_A[root] = T_A$ and $T_B[root] = T_B$. The root is on $Link_{ANY\_Gender}$, $Link_{ANY\_Job}$, and $Link_{[1\text{-}99)}$. See Fig. 3. Suppose the first winner specialization is $ANY\_Job\{White\text{-}collar, Blue\text{-}collar\}$. We create two new partitions under the root partition as shown in the figure, and split data records between them. Both partitions are on $Link_{ANY\_Gender}$ and $Link_{[1-99)}$. $\cup Cut_i$ is updated into $\{ANY\_Gender, White\text{-}collar, Blue\text{-}collar, [1\text{-}99)\}$. Suppose that the second winner specialization is $[1\text{-}99)\{[1\text{-}60), [60\text{-}99)\}$, which specializes the two partitions on $Link_{[1-99)}$, resulting in the leaf partitions in Fig. 3.

We summarize the operations at the two data providers, assuming that the winner w is local at Provider A.

Provider A. Provider A refines 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. 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, Provider A also collects the following information:

- *uids for Provider B*. If a record in P_w is specialized to a child value c, collect the pair (uid, c), where uid is the UID value of the record. Then, Provider A sends these (uid, c) pairs to Provider B for specializing the corresponding leaf partitions there.

- *Count statistics for updating Score*. 1) For each c in $child(w)$: $|T_A[c]|$, $|T_A[d]|$, $|T_A[c \wedge cls]|$, $|T_A[d \wedge cls]|$, $|T_A[c \wedge s]|$, and $|T_A[d \wedge s]|$ for every sensitive value $s \in S$, where $d \in child(c)$ and cls is a class label. Refer to Section 3 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 is specialized.

Provider B. On receiving the instruction from Provider A, Provider B creates child partitions P_c in its own TIPS. At Provider Bs site, the child partitions P_c contain raw records from T_B. The child partitions P_c are obtained by splitting P_w among P_c according to the (uid, c) pairs received.

Example 4. Let us revisit the first specialization in Example 3. Provider B performs the first specialization $ANY\_Job \rightarrow \{White\text{-}collar, Blue\text{-}collar\}$ on its own TIPS and then sends the following instruction to Provider A:

$uid\#1, 3, 5, 6, 10, 11$ go to the node with *Blue-collar*.

$uid\#2, 4, 7, 8, 9$ go to the node with *White-collar*.

On receiving this instruction, Provider A creates the two child nodes under the root in its copy of TIPS and partitions $T_A[root]$. In general, the data providers update their own TIPS data structures either by specializing the winning candidate on their own local attributes or by receiving specialization instructions from the other data provider.

Updating TIPS is the only operation that accesses raw records in our algorithm. Subsequently, updating $Score(x)$ (in Section 4.2.3) makes use of the count statistics without further accessing raw records. 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.

4.2.3 Update the Score

The key to the scalability of our algorithm is updating $Score(x)$ and $IsValid(x)$ using the count statistics maintained in Section 4.2.2 without accessing raw records again. For any valid candidate $x \in \cup Cut_i$, its $Score(x)$ and $IsValid(x)$ need to be updated if some qid_x containing x also contains the winner candidate w in the current specialization. Updating $Score(x)$ depends on $InfoGain(x)$ or $|T[qid_x]|$. Updating $IsValid(x)$ depends on $|T[qid_x]|$ and $|T[qid_x \wedge s]|$ for every sensitive value $s \in S$. We consider both updates below.

Updating $InfoGain(x)$**.** We need to compute $InfoGain(c)$ the newly added c in $child(w)$. The owner provider of w can compute $InfoGain(c)$ while collecting the count statistics for c in Section 4.2.2.

Example 5. Let us revisit the first specialization in Example 3. We show the computation of $Score(ANY\_Job)$ for the specialization $ANY\_Job\{Blue\text{-}collar, White\text{-}collar\}$. For general data analysis, $Score(ANY\_Job) = 6^2 + 5^2 = 61$. For classification analysis,

$$E(T[ANY\_Job]) = -\frac{6}{11} \times log_2 \frac{6}{11} - \frac{5}{11} \times log_2 \frac{5}{11} = 0.994,$$

$$E(T[Blue\text{-}collar]) = -\frac{2}{6} \times log_2 \frac{2}{6} - \frac{4}{6} \times log_2 \frac{4}{6} = 0.918,$$

$$E(T[White\text{-}collar]) = -\frac{5}{5} \times log_2 \frac{5}{5} - \frac{0}{5} \times log_2 \frac{0}{5} = 0.0,$$

$$InfoGain(ANY\_Job) = E(T[ANY\_Job]) - \left(\frac{6}{11} \times E(T[Blue\text{-}collar]) + \frac{5}{11} \times E(T[White\text{-}collar])\right) = 0.493,$$

$$InfoGain(ANY\_Job) = 0.493.$$

Updating $IsValid(x)$ **and** $|T[qid_x]|$**.** Given an LKC-privacy requirement, a specialization on value x is valid (i.e., $IsValid(x) = true$) if $|T[qid_j]| \geq K$ and $P(s|qid_j) = \frac{|T[qid_j \wedge s]|}{|T[qid_j]|} \leq C$ for any qid_j with $|qid_j| \leq L$ containing x and for any $s \in S$. The following data structure, called *Quasi-Identifier Tree* (QIT), can efficiently maintain these counts.

Definition 4.2 (QIT). *For each $QID_j = \{D_1, \ldots, D_q\}$, QIT_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. The leaf nodes store the $|T[qid_j]|$ and $|T[qid_j \wedge s]|$ for every sensitive value $s \in S$. A branch is trimmed if its $|T[qid_j]| = 0$.*

QIT_j is kept at a data provider if the data provider owns some attributes in QID_j. On specializing the winner w, a data provider updates its QIT_js that contain the attribute of w: creates the nodes for the new qid_js and computes $|T[qid_j]|$ and $|T[qid_j \wedge s]|$ for every sensitive value $s \in S$. We can obtain $|T[qid_j]|$ and $|T[qid_j \wedge s]|$ from the local copy of TIPS: $|T[qid_j]| = \sum |P_c|$, where P_c is on $Link_c$ and qid_j is the generalized value on QID_j for P_c. $|T[qid_j \wedge s]|$ can be computed similarly. Note that $|P_c|$ is given by the count statistics for w collected in Section 4.2.2.

4.3 Analysis

Our approach produces the same integrated table as the *single data provider* anonymization algorithm [13] on a joint table, and ensures that no data provider learns more detailed information about any other provider other than what they agreed to share. This claim follows from the fact that PHDMashup performs exactly the same sequence of specializations as in [13] but in a distributed manner where T_A and T_B are kept locally at the sources. The only information revealed to each other is $\cup Cut_j$ and T_g at each iteration. However, such information is more general than the final mashup table that the two data providers have agreed to share.

PHDMashup (Algorithm 1) is extendable for multiple (more than two) data providers with minor changes: in Line 5, each data provider should communicate with all the other data providers for determining the winner. In Line 8, the data provider holding the winner specialization should instruct all the other data providers. In Line 10, a data provider should wait for instruction from the winner data provider. Our algorithm is based on the assumption that all the data providers are semihonest. An interesting extension would be to consider the presence of malicious and selfish data providers [29], [38], [39], [40]. In such a scenario, the algorithm has to be both secure and incentive compatible.

The overhead cost of session establishment in Phase I involves creating a session link between a mashup coordinator and a data recipient, and a session link between a mashup coordinator and each contributing data provider. A session link is simply an integer value (e.g., a Job ID) and is either used to actively keep the data recipient connected or to allow for asynchronous polling for the status of a current job. The data structure size for maintaining a session of a mashup request is linear to the number of contributing data providers; therefore, the overhead is limited. In case the number of data recipients or the number

TABLE 3
Attributes for the *Adult* Data Set

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

of mashup requests is large, the load of the mashup coordinator can be balanced across multiple servers.

The computational cost of PHDMashup in Phase II 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 4.2.2). 2) Update QIT_j, $InfoGain(x)$, and $IsValid(x)$ for affected candidates x (Section 4.2.3). 3) Send "instruction" to the remote data providers. The instruction contains only uids 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 of the algorithm on real-life data in the next section.

For the communication cost 3, each data provider communicates (Line 5 in Algorithm 1) with others to determine the global winner candidate. Thus, each data provider sends $n-1$ messages, where n is the number of data providers. Then, the data provider of the winner specialization (Line 8) sends instruction to other data providers. This communication process continues for at most s times, where s is the number of valid specializations bounded by the number of distinct values in $\cup QID_j$. Hence, for a given data set, the total communication cost is $s \times [n(n-1)+(n-1)] = s \times (n^2-1) \approx O(n^2)$. If $n = 2$, then the total communication cost is $3s$. In real-life data mashup application, the number of contributing data providers for an information request is usually small.

5 Empirical Study

We implement the proposed PHDMashup in a distributed web service environment. Each data provider is running on an Intel Core2 Quad Q6600 2.4 GHz PC with 2 GB RAM connected to a LAN. The objectives of the empirical study are to evaluate the benefit of data mashup for joint data analysis, and the impacts of anonymization and dimensionality on the data quality with respect to the information requirements. We first describe the data set and the settings, followed by the results.

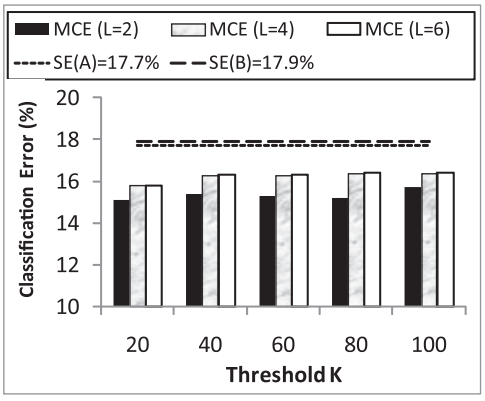


Fig. 4. Benefits of mashup ($C = 20\%$).

Due to the privacy agreement, we cannot use the raw data from the social network companies for experiments, so we employ the *de facto* benchmark census data set *Adult* [41], which is also a real-life data set, to illustrate the performance of our proposed architecture and algorithm. The *Adult* data set has six numerical attributes, eight categorical attributes, and a binary $Class$ attribute representing two income levels ≤ 50 K or >50 K. Table 3 describes each attribute. It contains 45,222 records after removing records with missing values. We model a 2-data provider scenario with two private tables T_A and T_B as follows: T_A contains the first nine attributes, and T_B contains the remaining five attributes. We consider *Divorced* and *Separated* in the attribute *Marital-status* as sensitive, and the remaining 13 attributes as QID. A common UID is added to both tables for joining. The taxonomy trees for all categorical attributes are from [8].

5.1 Benefits of Mashup

A trivial yet incorrect solution to avoid privacy concerns is to not integrate the data; each data provider simply performs the classification analysis on its own attributes and releases the data mining result, such as the classifier, to the data recipient. Our first goal is to illustrate the benefit of data mashup over this trivial solution with respect to the classification requirement.

To evaluate the impact on classification quality (Case 1 in Section 3.2.1), we use all records for anonymization, build a C4.5 classifier [2] on 2/3 of the anonymized records as the training set (30,162 records), and measure the *classification error* on 1/3 of the anonymized records as the testing set (15,060 records). Both the training and testing steps use all 14 attributes. Lower classification error means better data quality. We collect two types of classification errors from the testing set: *Mashup Classification Error* (MCE) is the error on the mashup data produced by our PHDMashup algorithm. *Source error* (SE) is the error on individual raw data table without generalization. SE for T_A, denoted by $SE(A)$, is 17.7 percent and SE for T_B, denoted by $SE(B)$, is 17.9 percent. $SE - MCE$ measures the benefit of data mashup over individual private table.

Fig. 4 depicts the MCE for the adversary's prior knowledge $L = 2$, $L = 4$, and $L = 6$ with confidence threshold $C = 20\%$ and anonymity threshold K ranging from 20 to 100. For example, $MCE = 16.3\%$ for $L = 4$ and $K = 60$, suggesting that the benefit of mashup, $SE - MCE$, is

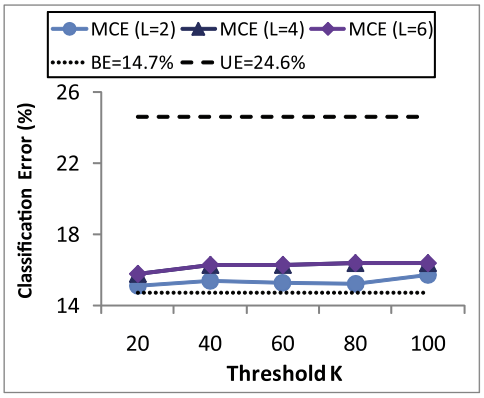


Fig. 5. Impacts on classification analysis ($C = 20\%$).

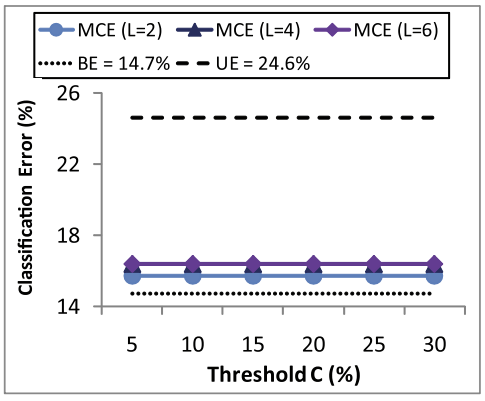


Fig. 6. Impacts on classification analysis ($K = 100$).

approximately 1.5 percent. This experiment demonstrates the benefit of data mashup over a wide range of privacy requirements. The benefit for all test cases illustrated in Fig. 4 spans from 1.3 to 2.1 percent. The benefit decreases as L increases because more generalization is required in order to thwart the linkage attacks. In practice, the benefit is more than the accuracy consideration because our method allows the participating data providers to share *data* for joint data analysis, rather than sharing a *classifier* from each provider.

5.2 Impacts of Anonymization

Our second goal is to illustrate the impacts for achieving LKC-privacy with respect to classification analysis and general data analysis.

To evaluate the impacts on classification quality (Case 1 in Section 3.2.1), we collect several classification errors, in addition to MCE, from the testing set: *Baseline Error* (BE) is the error measured on all 14 raw data attributes without generalization. $BE - MCE$ represents the cost in terms of classification quality for achieving a given LKC-privacy requirement. A naive method to avoid record and attributes linkages is to simply remove all QID attributes. Thus, we also measure *Upper bound Error* (UE), which is the error on the raw data with all QID attributes removed. $UE - MCE$ represents the benefit of our method over the naive approach. The experimental results below suggest that our method PHDMashup can yield a small $MCE - BE$ (low cost) and a large $UE - MCE$ (high benefit).

Fig. 5 depicts the MCE for the adversary's prior knowledge $L = 2$, $L = 4$, and $L = 6$ with confidence threshold $C = 20\%$ and anonymity threshold K ranging from 20 to 100. For example, at $L = 4$, $K = 60$, and $C = 20$, $MCE = 16.3\%$. The cost is $MCE - BE = 1.6\%$, where $BE = 14.7\%$. The benefit is $UE - MCE = 8.3\%$, where $UE = 24.6\%$. For all test cases in Fig. 5, the cost $MCE - BE$ spans from 0.4 percent to 1.7 percent and the benefit $UE - MCE$ spans from 8.2 to 9.5 percent. This result illustrates that the cost of anonymization is low and the benefit of anonymization is high, suggesting that accurate classification and privacy protection can coexist even for a wide range of anonymity threshold K. Typically, there are redundant classification patterns in the data. Though generalization may eliminate some useful patterns, other patterns emerge to help the classification task.

Fig. 6 depicts the MCE for the adversary's prior knowledge $L = 2$, $L = 4$, and $L = 6$ with $K = 100$ and confidence threshold C ranging from 5 to 30 percent. MCE stays flat at 15.7 percent for $L = 2$, at 16.4 percent for $L = 4$ and $L = 6$. For all test cases in Fig. 6, the cost $MCE - BE$ spans from 1 to 1.7 percent and the benefit $UE - MCE$ spans from 8.2 to 8.9 percent. This result suggests that the MCE is insensitive to the change of the confidence threshold C, implying that it does not cost much to thwart attribute linkages.

To evaluate the costs on general analysis quality (Case 2 in Section 3.2.2), we use all records for generalization and measure the *mashup discernibility ratio* (MDR) on the final mashup data

$$MDR(T) = \frac{\sum_{qid} |T[qid]|^2}{|T|^2}. \quad (4)$$

MDR is the normalized discernibility cost [36], with $0 \leq MDR \leq 1$. The lower MDR indicates the higher data quality.

Fig. 7 depicts the MDR for the adversary's prior knowledge $L = 2$, $L = 4$, and $L = 6$ with confidence threshold $C = 20\%$ and anonymity threshold K ranging from 20 to 100. For all test cases in Fig. 7, MDR spans from 0.02 to 4.92 percent, suggesting that it costs very little to achieve a given LKC-privacy requirement even for a wide range of adversary's prior knowledge L and anonymity threshold K. Similar to the measure on classification error, MDR is insensitive to the change of the confidence threshold C.

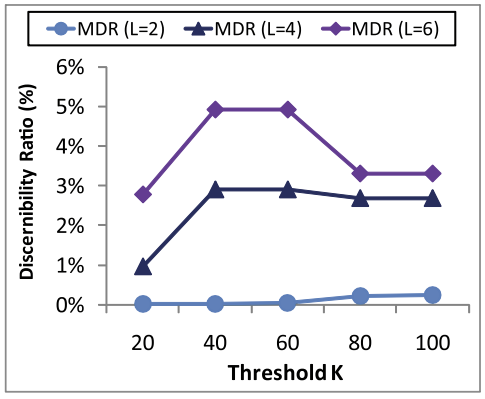


Fig. 7. Impacts on general data analysis ($C = 20\%$).

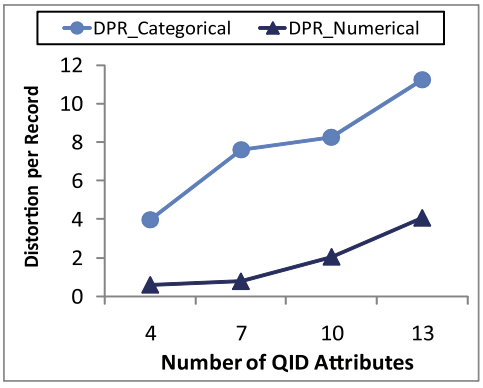


Fig. 8. Impacts of dimensionality ($L = 4$, $K = 60$, and $C = 20\%$).

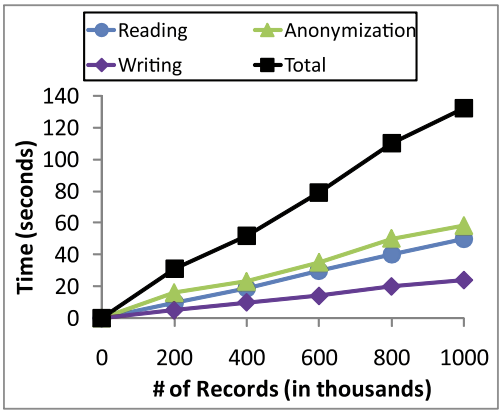


Fig. 9. Scalability ($L = 4$, $K = 20$, and $C = 100\%$).

The result in Fig. 7 also shows that MDR does not increase monotonically with respect to the increase K because PHDMashup employs a greedy approach for selecting a specialization at each iteration. The greedy approach can guarantee the identified solution is suboptimal but not necessarily global optimal. Although a global optimal solution is desirable, finding the optimal solution significantly degrades the efficiency and scalability of the method, which are also important requirements for real-life data mashup.

5.3 Impacts of Dimensionality

Our third goal is to evaluate the impact of dimensionality, i.e., the number of QID attributes, on the data quality with respect to the distortion metric proposed in [42]. Each time a categorical value is generalized to the parent value in a record, there is one unit of distortion. For a numerical attribute, if a value v is generalized to an interval $[a, b)$, there is $(b - a)/(f2 - f1)$ unit of distortion for a record containing v, where $[f1, f2)$ is the full range of the numerical attribute. The distortion is normalized by the number of records. The *distortion per record* (DPR) is separately computed for categorical attributes and numerical attributes, denoted by $DPR\_Categorical$ and $DPR\_Numerical$, respectively.

Fig. 8 depicts the $DPR\_Categorical$ and $DPR\_Numerical$ for the adversary's prior knowledge $L = 4$ with confidence threshold $C = 20\%$ and anonymity threshold $K = 60$ for 4, 7, 10, and 13 QID attributes. $DPR\_Categorical$ spans from 3.98 to 11.24 and $DPR\_Numerical$ spans from 0.62 to 4.05. This result illustrates that the distortion per record generally increases as the number of QID attributes increases because more generalizations are required in order to achieve the same LKC-privacy requirement.

5.4 Efficiency and Scalability

Our method takes at most 20 seconds for every previous experiment. Out of the 20 seconds, approximately 8 seconds is spent on initializing network sockets, reading data records from disk, and writing the generalized data to disk. The actual costs for data anonymization 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 evaluate this claim on an enlarged version of the *Adult* data set. We combine the training and testing sets, giving 45,222 records, and for each original record r in the combined set, we create $\alpha - 1$ variations of r, where $\alpha > 1$ is the blowup scale. Together with original records, the enlarged data set has $\alpha \times 45{,}222$ records.

Fig. 9 depicts the runtime from 200,000 to 1 million records for $L = 4$, $K = 20$, $C = 100\%$. The total runtime for anonymizing 1 million records is 132 seconds, where 50 seconds are spent on reading raw data, 58 seconds are spent on anonymization, and 24 seconds are spent on writing the anonymous data. Our algorithm is scalable due to the fact that we use the count statistics to update the $Score$, and thus it only takes one scan of data per iteration to anonymize the data. As the number of records increases, the total runtime increases linearly.

6 Summary

The experiments verified several claims about the PHDMashup algorithm. First, data mashup leads to improved information utility compared to the information utility separately available on each private table. Second, PHDMashup achieves a broad range of LKC-privacy requirements without significantly sacrificing the information utility. The cost for anonymization is low, and the benefit is significant. Third, our proposed architecture and method are scalable for large data sets. Our work provides a practical solution to the problem of high-dimensional data mashup with the dual goals of information sharing and privacy protection.

7 Conclusion and Lesson Learned

We implement a data mashup application for the online advertising industry in social networks, and generalize their privacy and information requirements to the problem of privacy-preserving data mashup for the purpose of joint data analysis on the high-dimensional data. We formalize this problem as achieving the LKC-privacy on the mashup data without revealing more detailed information in the process. We present a solution and evaluate the benefits of data mashup and the impacts of generalization. Compared to classic secure multiparty computation, a unique feature of our method is to allow data sharing instead of only result sharing. This feature is especially important for data analysis that requires user interaction. Being able to share data records would permit such exploratory data analysis and explanation of results.

Finally, we would like to share our experience of collaboration with industrial practitioners. In general, industrial practitioners prefer a simple privacy model that is intuitive to understand and to explain to their clients, such as LKC-privacy. Often, their primary concern is whether or not the anonymous data are still effective for data analysis; solutions that solely satisfy some privacy requirement are insufficient. The industry demands anonymization methods that can preserve information for various data analysis tasks.

Acknowledgments

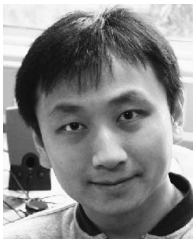
This research was supported in part by the Discovery Grant (356065-2008) and Strategic Project Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), Établissement de nouveaux chercheurs from Le Fonds québécois de la recherche sur la nature et les technologies (FQRNT), and the Cisco Research Award. The authors also thank Dr. Noman Mohammed for his insight on the analysis of the communication costs.

References

[1] R.D. Hof, "Mix, Match, and Mutate," *Business Week,* July 2005.
[2] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[3] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 6, pp. 1010-1027, Nov. 2001.
[4] L. Sweeney, "Achieving k-Anonymity Privacy Protection Using Generalization and Suppression," *Int'l J. Uncertainty, Fuzziness, and Knowledge-Based Systems,* vol. 10, no. 5, pp. 571-588, 2002.
[5] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "ℓ-Diversity: Privacy Beyond k-Anonymity," *ACM Trans. Knowledge Discovery from Data,* vol. 1, no. 1, Mar. 2007.
[6] K. Wang, B.C.M. Fung, and P.S. Yu, "Handicapping Attacker's Confidence: An Alternative to k-Anonymization," *Knowledge and Information Systems,* vol. 11, no. 3, pp. 345-368, Apr. 2007.
[7] R.J. Bayardo and R. Agrawal, "Data Privacy through Optimal K-Anonymization," *Proc. IEEE 21st Int'l Conf. Data Eng. (ICDE),* pp. 217-228, 2005.
[8] B.C.M. Fung, K. Wang, and P.S. Yu, "Anonymizing Classification Data for Privacy Preservation," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 5, pp. 711-725, May 2007.
[9] V.S. Iyengar, "Transforming Data to Satisfy Privacy Constraints," *Proc. Eighth ACM Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD),* pp. 279-288, July 2002.
[10] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, "Workload-Aware Anonymization," *Proc. 12th ACM Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD),* Aug. 2006.
[11] C.C. Aggarwal, "On k-Anonymity and the Curse of Dimensionality," *Proc. 31st Very Large Data Bases,* pp. 901-909, 2005.
[12] B.C.M. Fung, K. Wang, R. Chen, and P.S. Yu, "Privacy-Preserving Data Publishing: A Survey of Recent Developments," *ACM Computing Surveys,* vol. 42, no. 4, pp. 14:1-14:53, June 2010.
[13] N. Mohammed, B.C.M. Fung, P.C.K. Hung, and C. Lee, "Anonymizing Healthcare Data: A Case Study on the Blood Transfusion Service," *Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 1285-1294, June 2009.
[14] N. Mohammed, B.C.M. Fung, P.C.K. Hung, and C.K. Lee, "Centralized and Distributed Anonymization for High-Dimensional Healthcare Data," *ACM Trans. Knowledge Discovery from Data,* vol. 4, no. 4, pp. 18:1-18:33, Oct. 2010.
[15] A. Jhingran, "Enterprise Information Mashups: Integrating Information, *Simply*," *Proc. 32nd Int'l Conf. Very Large Data Bases,* pp. 3-4, 2006.
[16] G. Wiederhold, "Intelligent Integration of Information," *Proc. ACM Int'l Conf. Management of Data (SIGMOD),* pp. 434-437, 1993.
[17] R. Agrawal, A. Evfimievski, and R. Srikant, "Information Sharing Across Private Databases," *Proc. ACM Int'l Conf. Management of Data (SIGMOD),* 2003.
[18] O. Goldreich, *Foundations of Cryptography: Vol. II Basic Applications.* Cambridge Univ. Press, 2004.
[19] Y. Lindell and B. Pinkas, "Secure Multiparty Computation for Privacy-Preserving Data Mining," *J. Privacy and Confidentiality,* vol. 1, no. 1, pp. 59-98, 2009.
[20] A.C. Yao, "Protocols for Secure Computations," *Proc. 23rd Ann. Symp. Foundations of CS,* pp. 160-164, 1982.
[21] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M.Y. Zhu, "Tools for Privacy Preserving Distributed Data Mining," *ACM SIGKDD Explorations Newsletter,* vol. 4, no. 2, pp. 28-34, Dec. 2002.
[22] W. Du and Z. Zhan, "Building Decision Tree Classifier on Private Data," *Proc. IEEE Int'l Conf. Privacy, Security and Data Mining (CRPIT '14),* Dec. 2002.
[23] Z. Yang, S. Zhong, and R.N. Wright, "Privacy-Preserving Classification of Customer Data without Loss of Accuracy," *Proc. Fifth SIAM Int'l Conf. Data Mining,* pp. 92-102, 2005.
[24] P. Samarati and L. Sweeney, "Generalizing Data to Provide Anonymity when Disclosing Information," *Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems,* p. 188, June 1998.
[25] A. Hundepool and L. Willenborg, "μ- and τ-Argus: Software for Statistical Disclosure Control," *Proc. Third Int'l Seminar Statistical Confidentiality,* 1996.
[26] W. Jiang and C. Clifton, "Privacy-Preserving Distributed k-Anonymity," *Proc. 19th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security,* pp. 166-177, Aug. 2005.
[27] W. Jiang and C. Clifton, "A Secure Distributed Framework for Achieving k-Anonymity," *J. Very Large Data Bases,* vol. 15, no. 4, pp. 316-333, Nov. 2006.
[28] N. Mohammed, B.C.M. Fung, K. Wang, and P.C.K. Hung, "Privacy-Preserving Data Mashup," *Proc. 12th Int'l Conf. Extending Database Technology (EDBT),* pp. 228-239, Mar. 2009.
[29] N. Mohammed, B.C.M. Fung, and M. Debbabi, "Anonymity Meets Game Theory: Secure Data Integration with Malicious Participants," *Int'l J. Very Large Data Bases,* vol. 20, pp. 567-588, 2011.
[30] T. Trojer, B.C.M. Fung, and P.C.K. Hung, "Service-Oriented Architecture for Privacy-Preserving Data Mashup," *Proc. IEEE Seventh Int'l Conf. Web Services,* pp. 767-774, July 2009.
[31] P. Jurczyk and L. Xiong, "Privacy-Preserving Data Publishing for Horizontally Partitioned Databases," *Proc. 17th ACM Conf. Information and Knowledge Management,* Oct. 2008.
[32] P. Jurczyk and L. Xiong, "Distributed Anonymization: Achieving Privacy for Both Data Subjects and Data Providers," *Proc. 23rd Ann. IFIP WG 11.3 Working Conf. Data and Applications Security (DBSec),* 2009.
[33] C. Jackson and H.J. Wang, "Subspace: Secure Cross-Domain Communication for Web Mashups," *Proc. 16th Int'l Conf. World Wide Web,* pp. 611-620, 2007.
[34] 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," *Proc. 12th ACM Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD),* pp. 754-759, 2006.
[35] C.E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical J.,* vol. 27, p. 379, p. 623, 1948.
[36] A. Skowron and C. Rauszer, "The Discernibility Matrices and Functions in Information Systems," *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory,* 1992.
[37] N. Josuttis, *SOA in Practice: The Art of Distributed System Design.* O'Reilly Media, Inc., 2007.
[38] H. Kargupta, K. Das, and K. Liu, "Multi-Party, Privacy-Preserving Distributed Data Mining Using a Game Theoretic Framework," *Proc. 11th European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD),* pp. 523-531, 2007.
[39] N. Nisan, "Algorithms for Selfish Agents," *Proc. 16th Symp. Theoretical Aspects of CS,* Mar. 1999.
[40] N. Zhang and W. Zhao, "Distributed Privacy Preserving Information Sharing," *Proc. 31st Int'l Conf. Very Large Databases (VLDB),* pp. 889-900, 2005.
[41] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, "UCI Repository of Machine Learning Databases," http://archive.ics.uci.edu/ml, 1998.
[42] P. Samarati and L. Sweeney, "Protecting Privacy When Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression," Technical Report, SRI Int'l, Mar. 1998.

Benjamin C.M. Fung received the PhD degree in computing science from Simon Fraser University in 2007. He is an associate professor in the Concordia Institute for Information Systems Engineering (CIISE) at Concordia University in Montreal, and a research scientist of the National Cyber-Forensics and Training Alliance Canada (NCFTA Canada). His current research interests include data mining, information security, and cyber forensics, as well as their interdisciplinary applications on emerging technologies. He has more than 50 publications that span across the prestigious research forums of data mining, privacy protection, cyber forensics, web services, and building engineering. His research has been supported in part by the NSERC, FQRNT, and DRDC. He is a member of the IEEE.

Thomas Trojer received the BSc and MSc degrees from the University of Innsbruck, Austria, in 2007 and 2010, respectively. He is currently associated with the research group Quality Engineering as part of the Institute of Computer Science of the University of Innsbruck and was a visiting researcher at the Faculty of Business and IT of the University of Ontario Institute of Technology, Canada. His research interests include security in services computing, data privacy and access control, security in electronic healthcare systems, model-driven security engineering, and recently, issues regarding usability and security. He gathered project experience during his work on several projects in the fields of, e.g., interorganizational workflow security, privacy-aware access control in healthcare systems, and mobile healthcare applications. He is a member of the IEEE.



Patrick C.K. Hung is an associate professor with the Faculty of Business and Information Technology, Institute of Technology, University of Ontario, and an adjunct faculty member in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is also a guest professor at the Institute of Computer Science at the University of Innsbruck, Austria. He is a founding member of the IEEE International Conference of Web Services (ICWS) and the IEEE International Conference on Services Computing (SCC). He is an associate editor of the *IEEE Transactions on Services Computing*, *International Journal of Web Services Research*, and *International Journal of Business Process and Integration Management.* He is a member of the IEEE.

Li Xiong received the BS degree from the University of Science and Technology, China, the MS degree from Johns Hopkins University, and the PhD degree from the Georgia Institute of Technology, all in computer science. She is an assistant professor in the Department of Mathematics and Computer Science, where she directs the Assured Information Management and Sharing (AIMS) research group. Her research interests include data privacy and security, distributed information management, and health informatics. She is a recipient of the Career Enhancement Fellowship by the Woodrow Wilson Foundation, a Cisco Research Award, and an IBM industry skills faculty innovation award. She is a member of the IEEE.

Khalil Al-Hussaeni received the bachelor's degree in computer engineering from the University of Sharjah, U.A.E. He received the master's degree in information systems security in 2009 from the Concordia Institute for Information Systems Engineering, Concordia University, Canada. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering at Concordia University, Montreal, Canada. He is a research assistant in the Computer Security Laboratory. His research interests include privacy preservation in services computing, data mining, privacy-preserving data publishing, and computer and network security. He is a member of the IEEE.

Rachida Dssouli received the Doctorat d'Université degree in computer science from the Université Paul-Sabatier of Toulouse, France, in 1981, and the PhD degree in computer science in 1987 from the University of Montréal, Canada. She is a professor and the founding director of the Concordia Institute for Information Systems Engineering (CIISE), Concordia University. Her research interests include communication software engineering, requirements engineering, and services computing. Ongoing projects include incremental specification and analysis of reactive systems based on scenario language, service computing, service composition, multimedia applications, conformance testing, design for testability, conformance testing based on FSM/EFSM, and timed automata. She is a member of the IEEE.