# *D-Mash*: A Framework for Privacy-Preserving Data-as-a-Service Mashups

Mahtab Arafati, Gaby G. Dagher
*CIISE, Concordia University*
*Montréal, Quebéc, Canada*
*{ma_arafa, daghir}@ciise.concordia.ca*

Benjamin C. M. Fung
*SIS, McGill University*
*Montréal, Quebéc, Canada*
*ben.fung@mcgill.ca*

Patrick C. K. Hung
*University of Ontario Institute of Technology*
*Oshawa, Ontario, Canada*
*patrick.hung@uoit.ca*

***Abstract*—*Data-as-a-Service* (*DaaS*) mashup enables data providers to dynamically integrate their data on demand depending on consumers' requests. Utilizing DaaS mashup, however, involves some challenges. Mashing up data from multiple sources to answer a consumer's request might reveal sensitive information and thereby compromise the privacy of individuals. Moreover, data integration of arbitrary DaaS providers might not always be sufficient to answer incoming requests. In this paper, we provide a cloud-based framework for privacy-preserving DaaS mashup that enables secure collaboration between DaaS providers for the purpose of generating an anonymous dataset to support data mining. Experiments on real-life data demonstrate that our DaaS mashup framework is scalable and can efficiently and effectively satisfy the data privacy and data mining requirements specified by the DaaS providers and the data consumers.**

***Keywords*-data mashup; data privacy; anonymization; data mining; web services**

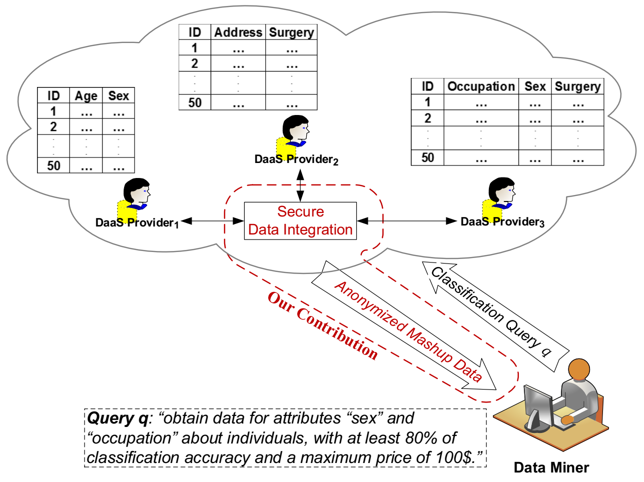


Figure 1. Mashup of patient records in DaaS environment

## I. Introduction

*Data-as-a-Service* (*DaaS*) is an emerging cloud computing service that provides data on demand to consumers across various cloud platforms via different protocols over the Internet. Utilizing DaaS not only supports data access from anywhere at anytime but also reduces the cost of data management. We foresee that a new class of integration technologies will emerge to serve data integration on demand using DaaS providers through web services, and we call it *DaaS Mashup*.

In this paper, we propose a privacy-preserving DaaS mashup framework that allows DaaS providers to securely integrate and trade their collected person-specific survey data to support analytical data mining tasks such as classification analysis. In the market, *DaaS providers* can register and advertise their available data, and *data consumers* can submit their classification analysis requests with a minimum accuracy requirement. Then a *mashup coordinator* in the framework dynamically determines the group of DaaS providers whose data can fulfill the request, with consideration of data availability, bid price, and data quality, such as classification accuracy. Figure 1 presents an example of a DaaS mashup market for secure integration of patient records.

The challenges of modeling a data-sharing market are summarized as follows. The first challenge is the privacy concern. DaaS providers are often reluctant to share the person-specific data of their survey respondents because of data privacy. Two types of privacy concerns have to be addressed in our proposed DaaS mashup framework. First, the final mashup data has to be anonymized in order to disable any potential linkage attacks. Second, during the mashup process, no DaaS provider should learn more information from the other DaaS providers other than what is revealed in the final mashup data [1]. The second challenge is the data quality of the anonymized data. It is important to ensure that the final mashup data contributed by multiple DaaS providers is useful for a given consumer's data request. The third challenge is how to satisfy a given data request. Data coming from a single DaaS provider may not be sufficient to fulfill a data request; subsequently, selecting the appropriate combination of DaaS providers is a non-trivial task. The selection process has to consider the consumer's data attribute requirement, data quality requirement, and bid price as well as the DaaS providers' privacy requirements.

The contributions of this paper can be summarized as follows:

**Contribution #1.** To the best of our knowledge, this is the first work that proposes a cloud-based DaaS framework to integrate private data from multiple DaaS providers with the goal of preserving both data privacy and the data mining quality of the underlying data. Section II provides a formal description of the objectives and behaviour of the participants in the proposed framework.

**Contribution #2.** Vaculin et al. [2] presented a web service framework to answer a request coming from a consumer with the assumption that a single provider can fulfill the request. In contrast, we remove such an assumption and dynamically identify the combination of DaaS providers whose data can best satisfy the data privacy, data quality, and price requirements. Section III presents the proposed framework and algorithms.

**Contribution #3.** We performed experimental evaluation on real-life data to measure the impact of the DaaS providers' revenue and the efficiency of our proposed market framework with respect to different privacy levels. Extensive experimental results suggest that our framework is efficient in terms of processing various sizes of queries with regard to data quality and bid price. Section IV shows the experimental results.

## II. The Participants

This paper introduces a privacy-preserving framework for trading person-specific survey data. The framework assumes three types of participants: *DaaS providers*, *data consumers*, and *mashup coordinator*. We assume that the data being shared is in the form of a relational table that is vertically partitioned into sub-tables, each of which is hosted by one DaaS provider. We also assume that the data mining task the data consumer is interested in performing is classification analysis. The data consumer submits a sequence of data queries to a mashup coordinator in the platform, where each query consists of the requested attributes, the required data quality (classification accuracy), and the maximum bid price. Since a single DaaS provider might not be able to provide all requested attributes, the mashup coordinator is responsible for determining the group of DaaS providers that can cover all the attributes while meeting the requested data quality and price. Finally, the mashup coordinator has to return an anonymized data table that satisfies a given privacy requirement that is agreed on by all the contributing DaaS providers. The rest of this section describes the goals, requirements, and behaviour of these three types of participants in our proposed framework.

### A. DaaS Providers

Let $DP = \{P_1, \ldots, P_n\}$ be the group of registered DaaS providers in our framework. Each provider $P_i$ owns an *attribute table* in the form of $T_i^A = (UID, EID_i, QID_i, Sen_i,\ Class)$, where $UID$ is a system-generated unique identifier of a survey respondent, $EID_i$ is a set of explicit identifiers, $QID_i$ is a set of quasi-identifiers, $Sen_i$ is a set of sensitive attributes, and $Class$ is a target class attribute for classification analysis. Explicit identifiers contain information, such as name and SSN, that can explicitly identify an individual. They should be removed before the data publishing. QID is a set of attributes, such as job, sex, and age, that *may* identify a respondent if some combinations of QID values are specific enough. They cannot be removed because they are useful for the data mining task. Any attribute in QID can be shared by two or more DaaS providers. The sensitive attribute $Sen_i$ contains some sensitive information about the survey respondents, such as diseases they might have. The target class attribute will be explained later in this section.

The DaaS providers want to sell the survey data in their attribute table for profit, but releasing the raw data may compromise the privacy of their survey respondents. In a common privacy attack called *record linkage* an adversary attempts to utilize his background knowledge, represented by a combination of QID values denoted by $qid$, of a target victim $V$, with the goal of identifying $V$'s record in the released data table $T$. Many privacy models [3][4] have been proposed in the last decade to thwart these linkage attacks. In our proposed framework, we choose to impose $LKC$*-privacy* [5] on the final mashup data for two reasons. First, $LKC$-privacy was specifically designed for preventing linkage attacks on *high-dimensional data*. This is important because a classification analysis request might require many attributes from different DaaS providers, often resulting in a high-dimensional mashup table. Second, $LKC$-privacy is a generalized privacy model that covers $K$-anonymity [3] and $\ell$-diversity [6]. Therefore, the DaaS providers, if necessary, have the flexibility to employ these traditional privacy models. Due to limited space, we do not present a full definition of $LKC$-Privacy here. Please refer to our technical report [1] for more details.

$LKC$-privacy guarantees the probability of a successful record linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$. $L$, $K$, and $C$ are DaaS provider-specified privacy thresholds. Increasing $K$, increasing $L$, or decreasing $C$ imposes a higher level of privacy protection, and vice versa. In general, imposing a higher level of privacy would result in lower data quality, and, therefore, it would lower the data mining value of the anonymized data. Thus, the DaaS providers would anonymize their attribute table $T_i^A$ with different combinations of $L$, $K$, and $C$ and advertise their prices in a *price table* $T_i^P = (L, K, C, Quality,\ Price)$ containing different combinations of privacy levels in terms of $L$, $K$, and $C$, with the corresponding data quality and price. The data quality is an objective measure depending on the supported data mining task. For example, the quality measure can be classification accuracy for classification analysis, and the quality measure can be F-measure for cluster analysis. Our proposed platform is applicable to any data mining task, provided there is a quality measure. In the implementation illustrated in the rest of this paper, we assume that the DaaS providers support classification analysis, and the quality measure is classification accuracy on the target attribute $Class$. Without loss of generality, we assume that there is only one $Class$ attribute shared among $\{T_1^A, \ldots, T_n^A\}$. Though $LKC$-privacy is chosen to be the privacy model in our implementation, our platform can adapt any privacy model provided there is a privacy parameter(s) to adjust the privacy level.

To construct the price table we introduce procedure $buildPT$. This procedure takes as input a set of $LKC$-privacy requirements. For each $LKC$-privacy requirement, procedure $buildPT$ utilizes an algorithm called *Privacy Aware Information Sharing* (*PAIS*) [5] to anonymize the attribute table $T_i^A$. *PAIS* is a top-down specialization method for achieving $LKC$-privacy with the goal of maximizing the classification accuracy on the $Class$ attribute. The resulting anonymized table is denoted by $T_i^{A'}$. Then, the procedure employs the *C4.5 decision tree classifier* [7] to determine the classification accuracy $Acc$ of $T_i^{A'}$. The advertised price is determined by the price per attribute of provider $P_i$, discounted by the accuracy. A new record with values $L$, $K$, $C$, $Acc$, and $Price$ is then inserted into the price table $T_i^P$. We assume that DaaS providers follow the *non-colluding semi-honest model* [8].

### B. Data Consumers

Data consumers are participants who want to perform some specific data analysis and would like to purchase some survey data from the market by submitting a data request. This can be as simple as a count query or as complex as a data mining operation, such as a classification analysis or a cluster analysis. In our proposed framework, a data request is represented in the form of $req = \{A_{req}, Acc_{req}, BPrice_{req}\}$, where $A_{req}$ is the set of requested attributes such that $A_{req} \subseteq (\bigcup_{i=1}^{n} QID_i) \cup (\bigcup_{i=1}^{n} Sen_i) \cup Class$, $Acc_{req}$ is the required minimum classification accuracy, and $BPrice_{req}$ is the bid price for the requested data. Our model assumes that any data consumer can be an adversary whose goal is to launch record and attribute linkage attacks on the received data. Therefore, the final mashup data must satisfy a given $LKC$-privacy requirement that is agreed upon by all contributing DaaS providers.

### C. Mashup Coordinator

A mashup coordinator is a mediator between data consumers and DaaS providers. Given a data request $req = \{A_{req}, Acc_{req}, BPrice_{req}\}$, the objective of a mashup coordinator is to coordinate one or multiple DaaS providers to generate a mashup table $T^M$ such that $T^M$ contains all the requested attributes $A_{req}$, the total price of the mashup table $TPrice(T^M) \leq BPrice_{req}$, and the classification accuracy on the final mashup table $Acc(T^M) \geq Acc_{req}$. Finally, the mashup coordinator is responsible for sending the final mashup table $T^M$ to data consumers and distributing the revenue to the contributing DaaS providers.

In case a mashup table $T^M$ satisfies $A_{req}$ and $Acc_{req}$ but fails to satisfy $TPrice(T^M) \leq BPrice_{req}$, a mashup coordinator should have the capability to make alternative recommendations to the data consumers, such as increasing the bid price $BPrice_{req}$ or decreasing the minimum accuracy $Acc_{req}$.

## III. D-Mash Framework Solution

### A. Solution Overview

The objective of our solution is to provide a market mashup framework with a *Service-oriented architecture (SOA)* that enables DaaS providers to securely integrate their survey data and generate an anonymized mashup table $T^M$ such that the privacy of the data is preserved, while the request coming from the data consumer is satisfied.

The framework for answering a data consumer's request consists of four steps:

**Step 1 - Identify Contributing DaaS Providers**. We introduce a greedy algorithm *DaaS Providers Selector (selectDaaSPs)* that determines the group of DaaS providers whose data satisfy all requested attributes such that the total cost is minimal.

**Step 2 - Compute Total Price**. The mashup coordinator executes a procedure called *Total Price Computation (compTPrice)* to compute the total price of the mashup table $T^M$.

**Step 3 - Construct Mashup Table**. To construct the final mashup table $T^M$ and determine its final accuracy, the mashup coordinator executes a procedure called *Mashup Table Construction (buildTM)*. The latter uses the privacy-preserving *PHDMashup* algorithm [9] to securely integrate and anonymize the attribute tables of contributing DaaS providers. It also utilizes classifier $C4.5$ to compute the final classification accuracy of $T^M$.

### B. The Architecture

*Service-oriented architecture (SOA)* is a pattern for business processes maintenance that contains large distributed systems. SOA has several properties including *services*, *interoperability*, and *loose coupling*. A service is a discrete software module utilized for different simple or complex functionalities. An *enterprise service bus* (ESB) enables the interoperability for services among distributed systems and eases the distribution of processes over multiple systems. Loose coupling minimizes the dependencies of system components and improves scalability and fault tolerance of the system [10]. The implemented architecture of our framework is illustrated in Figure 2.

The proxy component contains a proxy manager that generates a proxy class based on the *WSDL* description

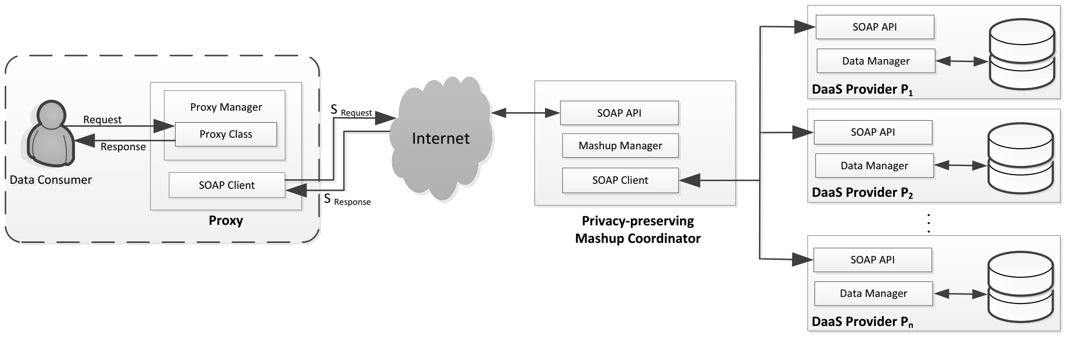


Figure 2. *D-Mash* framework for privacy-preserving Data-as-a-Service mashups: architecture

and exposes a programmatic interface based on the methods published by the web service of the mashup coordinator. When the data consumer sends a request, the coordinator invokes a method from the interface, where the method call is automatically converted (serialized) to a SOAP request $S_{Request}$ by the proxy using *XmlSerializer class*. The $S_{Request}$ is then *XML*-formatted and transferred through the network. Since SOAP web services utilize *simple object access protocol* to transmit data between SOAP clients and SOAP APIs, our proxy manager uses the SOAP client to send $S_{Request}$ to the SOAP API of the mashup coordinator.

The mashup coordinator component contains three entities: SOAP API, mashup manager, and SOAP client. The serialized request is automatically *deserialized* by *XmlSerializer class* in order to extract the data when it reaches the SOAP API. The mashup manager uses the extracted data to compute the contributing DaaS providers, calculate the total price, construct the anonymized mashup table $T^M$, and compute the final accuracy of $T^M$. The mashup manager is also responsible for ensuring that the consumer's request is fulfilled. In case the request cannot be fulfilled, it recommends alternative solutions. The SOAP client entity of the mashup coordinator component is used to communicate with the DaaS provider components.

Each DaaS provider component consists of two entities: data manager and SOAP API. The data manager receives requests from a mashup coordinator through the SOAP API, and then deserializes the request and queries the data accordingly. Once the final anonymized mashup table $T^M$ has been constructed, the mashup manager serializes the $T^M$ data, along with its accuracy and price values, and sends that as a SOAP response back to the proxy via its SOAP API. The proxy component receives the SOAP response $S_{Response}$ through its SOAP client, then the proxy manager deserializes the data and sends it back to the data consumer.

## C. Identify Contributing DaaS Providers

When the mashup coordinator receives a consumer's data request $req$, the first task is to identify one or more registered DaaS providers that can collectively fulfill all requested attributes $A_{req}$ such that the price of each attribute is the lowest possible price. We call such a group *contributing DaaS providers*. The following is the formal definition:

*Definition 3.1 (Contributing DaaS Providers.): Given a set of registered DaaS providers $DP$ and a set of requested attributes $A_{req}$, the* contributing DaaS providers *are the set of providers $\mathbb{D} \subseteq DP$ such that:*

1) *$\forall A \in A_{req}$, $\exists P_i \in \mathbb{D}$, where $T_i^A$ contains $A$, and*
2) *$\nexists P_j \in DP$ such that $T_j^A$ contains $A$ and the price per attribute $PA_j < PA_i$, where $PA_j$ and $PA_i$ are the price per attribute for providers $P_j$ and $P_i$, respectively.* ■

**Algorithm 1** *selectDaaSPs*: **DaaS Providers Selector**

```
Input: requested attributes $A_{req}$
Input: registered DaaS providers $DP$
Output: contributing DaaS providers $\mathbb{D}$
 1: initially $R = A_{req}$ and $\mathbb{D} = \emptyset$ and $\hat{D} = DP$
 2: while $R \neq \emptyset$ do
 3:     select $P_i \in \hat{D}$ with the least price per attribute $PA_i$
 4:     $M_i \leftarrow \{T_i^A \cap R\}$
 5:     if $M_i \neq \emptyset$ then
 6:         $\mathbb{D} \leftarrow (P_i, M_i)$
 7:         $R \leftarrow R \setminus M_i$
 8:     end if
 9:     $\hat{D} \leftarrow \hat{D} \setminus P_i$
10: end while
11: return $\mathbb{D}$;
```

In Algorithm 1, we introduce a greedy procedure *DaaS Providers Selector (selectDaaSPs)* that enables the mashup coordinator to compute the contributing DaaS providers for request $req$. This algorithm examines the set of attributes $A_{req}$ and the price per attribute $PA_i$ provided by each DaaS provider, and then identifies for each requested attribute the DaaS provider with the lowest price. The resulting $\mathbb{D}$ denotes a set of contributing DaaS providers. Because there might be more than one set of contributing DaaS providers that can satisfy $req$, *selectDaaSPs* is designed to find only one set of contributing DaaS providers and terminates once the set has been identified. *SelectDaaSPs* is a variation of the weighted set cover problem [11]. Initially, $R$ is equal to the requested attributes $A_{req}$, and $\hat{D}$ is the set of all registered DaaS providers (Line 1). In each iteration, the algorithm selects a provider $P_i \in \hat{D}$ whose price per attribute $PA_i$ is the least among all providers in $\hat{D}$ (Line 3). If $T_i^A$, the attribute table of $P_i$, contains some requested attributes $M_i$ (Line 4), then the pair of DaaS provider $P_i$ along with $M_i$ is added to $\mathbb{D}$ (Line 6), and $M_i$ is then removed from $R$ (Line 7). $P_i$ is also removed from $\hat{D}$ (Line 9), and a new iteration commences until $R$ is empty.

*Proposition 3.1: The cost of satisfying all requested attributes $A_{req}$ in procedure* selectDaaSPs *is $\sum PA_i \times CA_i$, where $PA_i$ is the price per attribute of provider $P_i$, and $CA_i$ is the number of covered attributes by provider $P_i$.* ∎

*Proposition 3.2: The runtime complexity of the greedy procedure* selectDaaSPs *is $O(n\ log\, m)$, where $n$ is the number of requested attributes $|A_{req}|$ and $m$ is the number of DaaS providers $|DP|$.* ∎

Due to limited space, we omitted the complexity proof. Please refer to our technical report [1] for the details.

## *D. Compute Total Price*

The total price of the mashup table $TPrice(T^M)$ is computed using procedure $compTPrice$. Given a minimum requested accuracy $Acc_{req}$ and the set of contributing DaaS providers $\mathbb{D}$ determined in Section III-C, procedure $compTPrice$ presented in Algorithm 2 randomly selects a provider $P_i$ from the set of contributing DaaS providers and removes it from $\mathbb{D}$ (Lines 1-2). Algorithm *findAcc* is utilized to examine the price table $T_i^P$ and find the smallest accuracy $Acc$ that is greater or equal to $Acc_{req}$ (Line 3). If such accuracy cannot be found, then *findAcc* selects the highest accuracy available in $T_i^P$. Next, algorithm *selectLKCP* selects from $T_i^P$ (Line 4) the values $L, K, C$, and $Price_i$ corresponding to $Acc$. $Price_i$ is the price of one attribute from DaaS provider $P_i$ with regard to $L, K, C$ values, whereas $CA_i = |M_i|$ is the number of covered attributes by provider $P_i$ (Line 5), where $M_i$ is the set of intersecting attributes between attribute table $T_i^A$ and requested attributes $A_{req}$.

Because the $LKC$-privacy model requires one set of $L, K, C$ values for anonymization, for each remaining contributing DaaS provider $P_j$ procedure $compTPrice$ checks price table $T_j^P$ to find the $L, K, C$ values. If a $T_j^P$ does not contain the specified $L, K, C$ values, then procedure $buildPT$ is invoked to generate a new row in the $T_j^P$ table by utilizing given specified $L, K, C$ values (Lines 7-9). Then for each $T_j^P$, *selectPrice* identifies the corresponding price value, multiplies it by $CA_j$, and then adds it to the total price (Line 10). The resulting $TPrice(T^M)$ is the total price of mashup table $T^M$. This procedure outputs the total price $TPrice(T^M)$ and the set of $L, K, C$ values (Line 12).

**Algorithm 2** *compTPrice*: **Total Price Computation**

```
Input: requested min. classification accuracy $Acc_{req}$
Input: contributing DaaS providers $\mathbb{D}$
Output: total price $TPrice(T^M)$
Output: privacy requirements $L, K, C$
 1: $P_i \leftarrow$ select a provider from $\mathbb{D}$
 2: $\mathbb{D} \leftarrow \mathbb{D} \setminus P_i$
 3: $Acc \leftarrow findAcc(T_i^P, Acc_{req})$
 4: $(L, K, C, Price_i) \leftarrow selectLKCP(T_i^P, Acc)$
 5: $TPrice(T^M) \leftarrow Price_i \times CA_i$
 6: for each $P_j \in \mathbb{D} : 1 \leq j \leq |\mathbb{D}|$ do
 7:     if $(L, K, C) \nexists T_j^P$ then
 8:         $T_j^P \leftarrow T_j^P \cup buildPT(T_j^A, \{L, K, C\}, PA_j)$
 9:     end if
10:     $TPrice(T^M) \leftarrow TPrice(T^M) + selectPrice(L, K, C, T_j^P) \times CA_j$
11: end for
12: return $TPrice(T^M), L, K, C$;
```

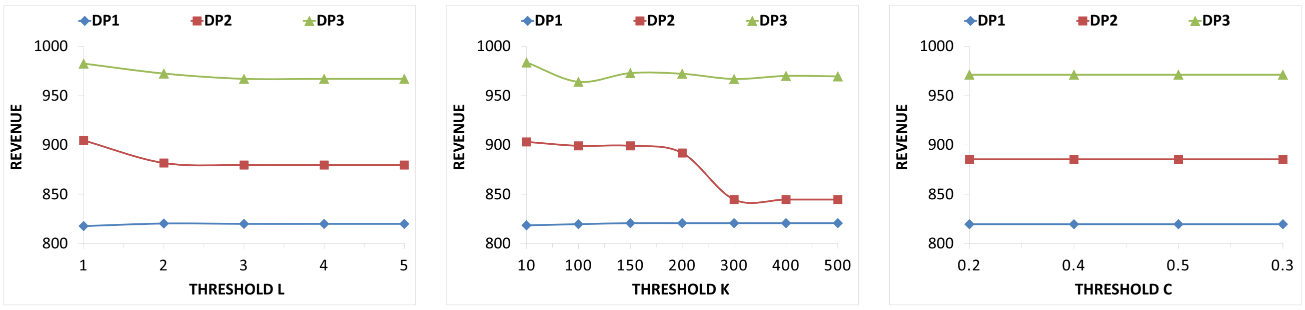
## *E. Construct Mashup table $T^M$*

To compute the mashup table $T^M$, we introduce procedure *buildTM* presented in Algorithm 3 that utilizes a secure algorithm called *Privacy-Preserving High-Dimensional Data Mashup* (*PHDMashup*) [9]. We would like to emphasize that Fung et al. [9] did not present a DaaS framework on how to identify the appropriate combination of DaaS providers with consideration of price and data quality requirements, which is a main contribution of this paper.

**Algorithm 3** *buildTM*: **Mashup Table Construction**

```
Input: contributing DaaS providers $\mathbb{D}$
Input: privacy requirements $L, K, C$
Output: mashup table $T^M$
Output: accuracy of mashup table $Acc(T^M)$
 1: $T^M \leftarrow PHDMashup(\mathbb{D}, L, K, C)$
 2: $Acc(T^M) \leftarrow 100 - C4.5(\mathbb{D}, L, K, C)$
 3: return $T^M, Acc(T^M)$;
```

Procedure *buildTM* is executed by the mashup coordinator for the purpose of computing mashup table $T^M$ and determining its accuracy $Acc(T^M)$. Given a set of contributing DaaS providers $\mathbb{D}$ and privacy requirements $L, K, C$, the mashup coordinator runs the *PHDMashup* algorithm (Line 1) in order to integrate and anonymize the raw data of contributing DaaS providers $\mathbb{D}$ and generates a mashup table $T^M$ that satisfies the given privacy requirements $L, K, C$. The *PHDMashup* algorithm preserves the privacy of every data provider by guaranteeing the mashup coordinator does not gain more information than the final mashup $T^M$ gives. The classifier $C4.5$ computes the classification error for the anonymized mashup table $T^M$ and privacy requirements $L, K, C$ (Line 2), where the resulting value $Acc(T^M)$ is the classification accuracy of the mashup table $T^M$. Procedure



(a) (b) (c)

Figure 3. Impacts of $LKC$-privacy requirements on DaaS provider's revenue.

*buildTM* returns both the mashup table $T^M$ and its accuracy $Acc(T^M)$ (Line 3).

## IV. Experimental Evaluation

We implemented our proposed architecture in *Microsoft Windows Azure*[1], a cloud-based computing platform. DaaS providers are distributed in a cloud environment, each of which is implemented on a Windows Server 2008 R2 running on AMD Opteron$^{TM}$ Processor 4171 HE@2.09 GHz with 1.75 GB RAM, and each hosts an *SQL Azure* database. The mashup coordinator is implemented as a web service, whereas the data consumer is implemented as a web client that interacts with the mashup coordinator via *http* protocol.

We utilize a real-life *adult* data set [12] in our experiments to illustrate the performance of our proposed framework. The adult data set contains 45,222 census records consisting of eight categorical attributes, six numerical attributes, and a class attribute *revenue* with two levels, $\leq 50K$ or $> 50K$.

The objective of our experiments is to evaluate the performance of the proposed market framework for privacy-preserving DaaS mashup. We first study the impact on the revenue of each data provider that results from enforcing various $LKC$-privacy requirements by varying the thresholds of maximum adversary's knowledge $L$, minimum anonymity $K$, and maximum confidence $C$. Next, we evaluate the efficiency of our solution and show that it is efficient with regard to the number of requested attributes $|A_{req}|$, classification analysis $Acc_{req}$, and bid price $BPrice_{req}$.

### A. Impact of Privacy Requirements on Revenue

Assuming that the attributes are randomly distributed over three DaaS providers, we evaluate the impact of $LKC$-privacy requirements on the revenue of each DaaS provider. After anonymizing the data set, we run $C4.5$ classifier on 2/3 of the anonymized records as the training set, measure the classification error on 1/3 of the anonymized records as the testing set, determine the final classification accuracy $FAcc$, and then compute the revenue of each DaaS provider $P_i$ with respect to its price per attribute $PA_i$.

Figure 3 illustrates the impact of $L, K, C$ thresholds on the revenue of each DaaS provider. Figure 3.a depicts the effect of threshold $L$. We observe that the revenue of each DaaS provider is insensitive to threshold $L$ when $L >= 2$. Figure 3.b depicts the effect of threshold $K$. The revenue of $P_1$ and $P_3$ is mainly unaffected by the change of value of $K$. However, an increase of the value of $K$ might negatively impact the revenue, as is the case with DaaS provider $P_2$, whose revenue dropped by 5% (from \$892 to \$844) when $K$ increased from 200 to 300. The reason for this drop is that when the specialization level $K$ is increased to 300, the number of "good" attributes that can lead to useful discrimination between the classes is reduced. Figure 3.c depicts that revenue is insensitive to the increase in the value of confidence threshold $C$. Consequently, we conclude that the primary privacy parameter that has a major impact on the revenue of a DaaS provider in our framework is the specialization parameter $K$.

### B. Efficiency

One major contribution of our work is the development of an efficient market framework for privacy-preserving DaaS mashup. The runtime complexity of our approach is dominated by the number of requested attributes $|A_{req}|$ in the consumer's data request $req$, the classification accuracy $Acc_{req}$, and the bid price $BPrice_{req}$. Therefore, we study the runtime under different numbers of requested attributes $A_{req}$ and different values of the pair $(Acc_{req}, BPrice_{req})$. We split the total runtime of our approach into three major phases: *Data Pre-Processing*, corresponding to procedure *buildPT*; *Contributing DaaS Providers*, corresponding to procedure *selectDaaSPs*; and *Final Mashup* $T^M$, corresponding to procedures *compTPrice* and *buildTM*. Figure 4 depicts the runtime of each phase when the number of requested attributes $A_{req}$ ranges between 4 and 13 attributes, with two different values of the pair $(Acc_{req}, BPrice_{req})$.

[1]http://www.microsoft.com/azure/

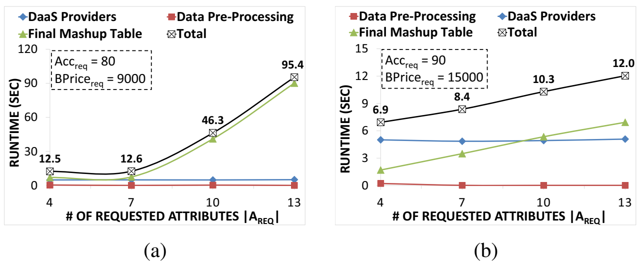


Figure 4. Efficiency w.r.t. the number of requested attributes $|A_{req}|$.

Figures 4.a and 4.b depict the runtime of each phase when the classification accuracy and bid price pair ($Acc_{req}$, $BPrice_{req}$) is equal to (80%, \$9,000) and (90%, \$15,000), respectively. We observe that the runtime of the *Data Pre-Processing* phase and the *Contributing DaaS Providers* phase is almost constant with regard to $|A_{req}|$, $Acc_{req}$, and $BPrice_{req}$. On the other hand, when $|A_{req}| \geq 7$, the runtime of the *Final Mashup* $T^M$ phase grows linearly as the number of requested attributes $|A_{req}|$ increases. We also observe that the runtime of the *Final Mashup* $T^M$ phase dominates the total runtime of our approach. Note that in Figure 4.b, the total runtime when $|A_{req}| = 13$ is 12 sec, in contrast to 95 sec in Figure 4.a. This is because both $Acc_{req} = 90\%$ and $P_{req} = \$15,000$ are beyond the threshold of accuracy and price in the DaaS providers' price tables. In this case, the highest accuracy from the data providers' price tables is selected, and the corresponding total cost is computed while avoiding the need to find higher or lower accuracies.

## C. Scalability

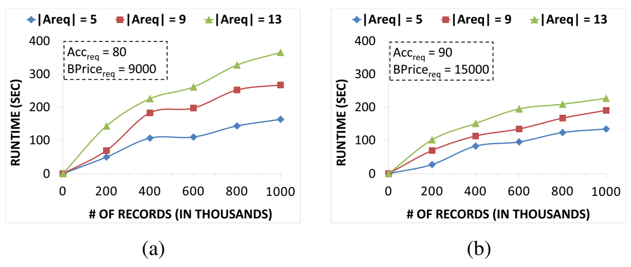


Figure 5. Scalability w.r.t. the number of records in the data set.

We evaluate the scalability of our algorithm with respect to the number of records in the data set when the number of requested attributes $|A_{req}|$ is equal to 5, 9, and 13.

Figure 5.a and Figure 5.b depict the total runtime of our algorithm when the number of records in the data set increases from 200,000 to 1 million. Figure 5.a depicts the total runtime when $Acc_{req} = 80$ and $BPrice_{req} = 9000$. The total runtime of answering a consumer's request with consideration to the consumer's data attribute requirement for 1 million records is 163s when the number of requested attributes $|A_{req}| = 5$, 267s when the number of requested attributes $|A_{req}| = 9$, and 365s when the number of requested attributes $|A_{req}| = 13$. Figure 5.b depicts the total runtime when $Acc_{req} = 90$ and $BPrice_{req} = 15000$. The total runtime of answering a consumer's request with consideration to the consumer's data attribute requirement for 1 million records is 134s when the number of requested attributes $|A_{req}| = 5$, 190s when the number of requested attributes $|A_{req}| = 9$, and 227s when the number of requested attributes $|A_{req}| = 13$. Note that the total runtime in Figure 5.b is less than the total runtime in Figure 5.a because $Acc_{req}$ and $BPrice_{req}$ in Figure 5.b go beyond the accuracy threshold and prices specified in the providers' price tables. We also observe that regardless of the value of $Acc_{req}$ and $BPrice_{req}$, our algorithm scales linearly with regard to the linear increase in the number of records in the data set.

## V. Related Work

In this section, we review the literature examining several areas related to our work.

First, we start with the area of *web services discovery for data integration*. Klusch et al. [13] and Vaculin et al. [2] respectively propose an OWL-S hybrid approach and a RDF-based framework for approximate matchmaking of requests and web services and discovering data providing services. Unlike their model, our proposed framework assumes that a consumer's data request could be best satisfied by multiple DaaS providers, and, therefore, our framework enables interactions between DaaS providers for securely integrating their data in order to answer the data request.

Another related area is *information integration*. Agrawal et al. [14] introduce the concept of minimal information sharing for only sharing metadata between data owners. On the other hand, secure multiparty computation (SMC) [15] allows the sharing of the computed result (e.g., a classifier) while prohibiting private data from being shared.

*Privacy-preserving data publishing* is another area related to our work. The privacy protection model, $K$-anonymity, was proposed in [16]. Sweeney [3] uses generalization and suppression to achieve $K$-anonymity for a datafly system. Preserving classification information in $K$-anonymous data is studied in [17]. Mohammed et al. [18] propose a top-down specialization algorithm to securely integrate two vertically partitioned distributed data tables into a $K$-anonymous table. Friedman and Schuster [19] propose an interactive algorithm for building a decision tree that satisfies $\varepsilon$-differential privacy. Trojer et al. [20] present a service-oriented architecture for achieving $K$-anonymity in the privacy-preserving data mashup scenario. Our work has a combination of single data source and integrated data source privacy levels. To preserve the privacy of the data of each DaaS provider, we utilize [5], which proposes an $LKC$-privacy model with an anonymization algorithm to address the problem of high-dimensional anonymization. Barhamgi et al. [21] propose a privacy-preserving approach for mashing-up DaaS web services. They arrange services in the mashup by defining a dependency graph, and then insert privacy filters to generate the mashup data. In contrast, we use PHDMashup as a secure protocol in order to integrate the data tables of DaaS providers while preserving privacy of mashup data using the $LKC$-privacy model.

## VI. Conclusions and Further Work

In this paper, we address the problem of secure collaboration between most suitable DaaS providers to answer data mining queries, while achieving $LKC$-privacy on the mashup data without revealing more detailed information in the process. Our proposed solution is different from the classical secure multiparty computation due to the fact that we allow *data sharing* instead of *data mining result sharing*. Data sharing provides the data recipient greater flexibility to perform different data analysis tasks. For future work, we plan to address the privacy-preserving DaaS mashup problem in a publicly verifiable malicious adversarial model.

## Acknowledgment

The research is supported in part by NSERC Discovery Grants (356065-2013).

## References

[1] M. Arafati, G. G. Dagher, B. C. M. Fung, and P. C. K. Hung, "A framework for privacy-preserving data-as-a-service mashups," Concordia University, Technical report, 2014. [Online]. Available: http://users.encs.concordia.ca/~daghir/TechReports/PPDaaSMashup.pdf

[2] R. Vaculin, C. Huajun, R. Neruda, and K. Sycara, "Modeling and discovery of data providing services," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2008, pp. 54–61.

[3] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 571–588, 2002.

[4] P. Samarati, "Protecting respondents identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 13, no. 6, pp. 1010–1027, 2001.

[5] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C.-k. Lee, "Anonymizing healthcare data: a case study on the blood transfusion service," in *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, New York, NY, USA, 2009, pp. 1285–1294.

[6] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, Mar. 2007.

[7] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[8] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," *IACR Cryptology ePrint Archive*, vol. 2011, p. 272.

[9] B. C. M. Fung, T. Trojer, P. C. K. Hung, L. X., K. Al-Hussaeni, and R. Dssouli, "Service-oriented architecture for high-dimensional private data mashup," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 373–386, 2012.

[10] N. Josuttis, *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., 2007.

[11] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[12] K. Bache and M. Lichman, *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[13] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with owls-mx," in *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, USA, 2006, pp. 915–922.

[14] R. Agrawal, A. Evfimievski, and R. Srikant, "Information sharing across private databases," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD)*, New York, NY, USA, 2003, pp. 86–97.

[15] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *Journal of Privacy and Confidentialityl.*, vol. 4, no. 1, pp. 59–98, 2009.

[16] P. Samarati and L. Sweeney, "Generalizing data to provide anonymity when disclosing information (abstract)," in *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, New York, NY, USA, 1998.

[17] B. C. M. Fung, K. Wang, and P. S. Yu, "Anonymizing classification data for privacy preservation," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 5, pp. 711–725, 2007.

[18] N. Mohammed, B. C. M. Fung, K. Wang, and P. C. K. Hung, "Privacy-preserving data mashup," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT)*, New York, NY, USA, 2009, pp. 228–239.

[19] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 493–502.

[20] T. Trojer, B. C. M. Fung, and P. C. K. Hung, "Service-oriented architecture for privacy-preserving data mashup," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2009, pp. 767–774.

[21] M. Barhamgi, D. Benslimane, C. Ghedira, S.-E. Tbahriti, and M. Mrissa, "A framework for building privacy-conscious daas service mashups," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2011, pp. 323–330.