Towards Privacy-preserving Data-as-a-Service Mashups

Mahtab Arafati, Gaby G. Daghir
CIISE, Concordia University
Montreal, QC, Canada
{ma_arafati, daghir}@ciise.concordia.ca

Benjamin C. M. Fung
SIS, McGill University
Montreal, QC, 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) is a paradigm that provides data on demand to consumers across different cloud platforms over the Internet. Yet, a single DaaS provider may not be able to fulfill a data request. Consequently, the concept of DaaS mashup was introduced to enable DaaS 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. We propose a greedy algorithm to determine a suitable group of DaaS providers whose data can satisfy a given request. Furthermore, our framework securely integrates the data from multiple DaaS providers while preserving the privacy of the resulting mashup data. Experiments on real-life data demonstrate that our DaaS mashup framework 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;

I. INTRODUCTION

Mashup is a web technology that integrates information from multiple web applications into a new web application. For instance, Trendsmap.com1 is a website that integrates the data from Twitter and Google Maps. It displays the map of cities all over the world on Google Maps with the most tweeted subjects from Twitter. Data mashup is a special kind of mashup application for integrating information of multiple data providers based on consumers’ requests. Data mashup is applicable for different purposes, such as managing scientific research [1] and addressing enterprises’ business needs [2].

1http://www.trendsmap.com/
can fulfill the data mining request, with the consideration of data availability, bid price, and data quality, such as classification accuracy. The challenges of constructing a market for sharing survey data are summarized as follows:

**Challenge #1: Privacy concerns.** DaaS providers are often reluctant to share the person-specific data of their survey respondents because of data privacy. Many organizations and companies believe that removing explicit identifying information, such as a respondents’ name and SSN, from the released data is sufficient for privacy protection. Yet, substantial research works [3][4] demonstrate that this naive approach is insufficient because a respondent can be re-identified by simple linkage attacks on other attributes called quasi-identifiers (QID). 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 that is revealed in the final mashup data.

**Challenge #2: Data quality concerns.** Protecting privacy is important. Yet, it is also equally important to ensure that the final mashup data contributed by multiple DaaS providers is useful for a given consumer’s data request. A data request can range from a simple data query to a complex data mining request. The challenge is how to ensure that the data quality of the final anonymized mashup data meets the data request.

**Challenge #3: Matching data requests.** Every registered DaaS provider owns different data attributes, imposes different levels of privacy protection, and advertises their data with different prices. 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. [5] 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. If no providers can fulfill the request with the offered price, alternative solutions with a higher price or lower data quality requirements will be recommended. 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, namely DaaS providers, data consumer, and mashup coordinator, as depicted in Figure 2. 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. A data consumer submits a sequence of data queries to a mashup coordinator in the platform, where each query consists of a data mining task, the requested attributes, the required data quality, 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 $T^A_i = (UID, EID_i, QID_i, Sen_i, Class)$, 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. 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. Even if attributes $EID_i$ are removed, an adversary may still be able to launch effective privacy attacks on some target victims. 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 $T$. This attack is effective if the number of records in $T$ containing $qid$, denoted by $|T[qid]|$, is small. Another common privacy attack is attribute linkage, in which an adversary attempts to utilize the background knowledge of $V$'s $qid$ to infer $V$'s sensitive value $s$ with a certain degree of confidence. This attack is effective if the confidence, which is calculated by $P(s|qid) = \frac{|T[qid \land s]|}{|T[qid]|}$, is high.

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 [6] on the final mashup data for two reasons. First, LKC-privacy was specifically designed for preventing linkage attacks on high-dimensional data, i.e., data with a large number of attributes. A data mining request can be complicated and requires many attributes from different DaaS providers, often resulting in a high-dimensional mashup table. Previous experimental results [6] have shown that enforcing other traditional privacy models would result in poor data mining quality in the anonymized data. Second, LKC-privacy is a generalized privacy model that covers $K$-anonymity [3], confidence bounding [7], and $\ell$-diversity [8]. Therefore, the DaaS providers, if necessary, have the flexibility to employ these traditional privacy models.

**Definition 2.1 (LKC-Privacy [6]):** Let $L$ be the maximum number of QID values of the adversary’s background knowledge on any participant in a data table $T$. Let $S$ 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 a minimum anonymity threshold, and

2) $\forall s \subseteq S$, the probability $P(s|qid) \leq C$, where $0 < C \leq 1$ is a maximum confidence threshold.

Algorithm 1 presents a procedure called $buildPT$ for constructing the price table. The procedure takes in a set of LKC-privacy requirements. For each LKC-privacy requirement, the procedure (Line 2) utilizes an algorithm called Privacy Aware Information Sharing (PAIS) [6] 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 (Line 3) employs the C4.5 decision tree classifier [9] to determine the classification accuracy $Acc$ of

<table>
<thead>
<tr>
<th>Algorithm 1 $buildPT$: Price Table Construction</th>
</tr>
</thead>
</table>
|**Input:** attribute table $T_A^i$  
|**Input:** a set of LKC-privacy requirements $PR_i$  
|**Input:** price per attribute $PA_i$  
|**Output:** price table $T^P_i$  |
|1: for each combination $\{L,K,C\} \in PR_i$ do  
|2: $T^i_A \leftarrow PAIS(T^i_A, L, K, C)$;  
|3: $Acc \leftarrow C4.5(T^i_A)$  
|4: $Price = Acc \times PA_i$  
|5: $T^P_i \leftarrow insert(L, K, C, Acc, Price)$  
|6: end for  
|7: return $T^P_i$;  

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 $T_A^i$ with different combinations of $L$, $K$, and $C$, and advertise their prices in a price table $T^P_i = (L, K, C, Quality, Price)$ 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.
The advertised price in Line 4 is determined by the price per attribute of provider $P_i$, discounted by the accuracy. A new record with values $L_i$, $K_i$, $C_i$, $Acc_i$, and $Price_i$ is then inserted into the price table $T_i^P$ (Line 5).

We assume that DaaS providers follow the non-colluding semi-honest model [10], meaning the providers follow the algorithm but are curious to derive sensitive information from the results obtained from other providers, without colluding with other parties in the platform. During the mashup process, the DaaS providers should not learn more information from other providers other than what is in the final mashup data.

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, which 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 [11] 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 $T^M$.

Step 4 - Satisfy the Data Request. The mashup coordinator ensures that the requested accuracy $Acc_{req}$ and the bid price $BPrice_{req}$ are fulfilled. Otherwise, the mashup coordinator recommends alternative solutions with a higher price or lower accuracy.

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 [12]. The implemented architecture of our framework is illustrated in Figure 3.

The proxy component contains a proxy manager that generates a proxy class based on the WSDL description 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
The mashup coordinator component contains three entities: SOAP API, mashup manager, and SOAP client. The serialized request is automatically deserialized by `XmmlSerializer` 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 \( S_{\text{Response}} \) 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 \( r_{\text{req}} \), the first task is to identify one or more registered DaaS providers that can collectively fulfill all requested attributes \( A_{\text{req}} \) 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 \( A_{\text{req}} \), the contributing DaaS providers are the set of providers \( D \subseteq DP \) such that:

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

In Algorithm 2, we introduce a greedy procedure `DaaS Providers Selector (selectDaaS)` that enables the mashup coordinator to compute the contributing DaaS providers for request \( r_{\text{req}} \). This algorithm examines the set of attributes \( A_{\text{req}} \) 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 \( D \) denotes a set of contributing DaaS providers. Because there might be more than one set of contributing DaaS providers that can satisfy \( r_{\text{req}} \), `selectDaaS` is designed to find only one set of contributing DaaS providers, and terminates once the set has been identified. `SelectDaaS` is a variation of the weighted set cover problem [13].

Initially, \( R \) is equal to the requested attributes \( A_{\text{req}} \), 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 \( D \) (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 \( A_{\text{req}} \) 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 \( P_i \). ■

The runtime complexity of the greedy procedure `selectDaaS` is \( O(n \log m) \), where \( n \) is the number of requested attributes.
attributes $A_{req}$ and $m$ is the number of DaaS providers $|DP|$. The main loop (Line 2) has complexity $O(n)$ because $|R|=|A_{req}|=n$. The main computational cost within the loop comes from the selection of DaaS providers with the least price per attribute $PA_i$ (Line 3). The complexity of selecting DaaS providers with the least $PA_i$ can be achieved in $O(\log m)$ using a priority heap.

Example 3.1: In a data request $req$, let the set of requested attributes be $A_{req} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$. Let the attributes included in each attribute table of a DaaS provider be as follows: $T_1^A = \{a_1\}$, $T_2^A = \{a_2\}$, $T_3^A = \{a_3, a_4\}$, $T_4^A = \{a_5, a_6, a_7, a_8\}$, $T_5^A = \{a_1, a_3, a_5, a_7\}$, $T_6^A = \{a_2, a_4, a_6, a_8\}$. Let the price per attribute of each provider be: $PA_1 = PA_2 = 4$, $PA_3 = 2$, $PA_4 = 1$, $PA_5 = PA_6 = 3$. Procedure selectDaaSProviders picks $T_1^A$, $T_3^A$, $T_5^A$, and $T_6^A$ in the first, second, third, and forth iteration, respectively, and returns the set of contributing DaaS providers $D = \{\{T_1^A, \{a_5, a_6, a_7, a_8\}\}, \{T_3^A, \{a_1\}\}, \{T_5^A, \{a_2\}\}\}$. The cost of satisfying $A_{req}$ is $(1 \times 4) + (2 \times 2) + (3 \times 1) + (3 \times 1) = 14$.

D. Compute Total Price

Once the set of contributing DaaS providers has been determined, the next step for the mashup coordinator is to compute the total price of the mashup table $TPrice(T^M)$. Given a minimum requested accuracy $Acc_{req}$ and the set of contributing DaaS providers $D$ determined in Section III-C, the $compTPrice$ algorithm randomly selects a provider $P_i$ from the set of contributing DaaS providers and removes it from $D$ (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 $Price$, 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 $P_j$, algorithm $compTPrice$ checks the price table $T_j^P$ to find the $L, K, C$ values selected in Line 4. If a $T_j^P$ does not contain the specified $L, K, C$ values (Line 7), then algorithm buildPT (Line 8) is invoked to generate a new row in the $T_j^P$ table by utilizing given specified $L, K, C$ values. 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 algorithm outputs the total price $TPrice(T^M)$ and the set of $L, K, C$ values (Line 12).

E. Construct mashup table $T^M$

To construct the mashup table $T^M$, the mashup coordinator utilizes a secure algorithm called Privacy-Preserving High-Dimensional Data Mashup (PHDMashup) [11]. In this section, we first introduce the PHDMashup algorithm. We then show how to construct the anonymized mashup table $T^M$ that satisfies the requested attributes of the data consumer, and we compute its final accuracy.

Let $D = \{P_1, \ldots, P_m\}$ be a set of contributing DaaS providers, where each provider $P_i \in D : 1 \leq i \leq m$ owns a person-specific data table $T_i$. The target attribute $Class$ for classification analysis is shared among all tables. Privacy-Preserving High-Dimensional Data Mashup (PHDMashup) is a data integration protocol that securely integrates the data tables of any set of DaaS providers with this setting and ensures that the final mashup table satisfies a specified LKC-privacy requirement with the goal of maximizing the

Algorithm 2 selectDaaSProviders: DaaS Providers Selector
Input: requested attributes $A_{req}$
Output: contributing DaaS providers $D$:

1. Initially $R = A_{req}$ and $\emptyset = \emptyset$ and $\hat{D} = DP$
2. While $R \neq \emptyset$
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 $M_i \neq \emptyset$
6. $\emptyset \leftarrow (P_i, M_i)$
7. $R \leftarrow R \setminus M_i$
8. End if
9. $\hat{D} \leftarrow \hat{D} \setminus P_i$
10. End while
11. Return $\emptyset$;

Algorithm 3 compTPrice: Total Price Computation
Input: requested min. classification accuracy $Acc_{req}$
Input: contributing DaaS providers $D$
Output: total price $TPrice(T^M)$:

1. $P_i \leftarrow$ select a provider from $D$
2. $\emptyset \leftarrow \emptyset \setminus P_i$
3. $Acc \leftarrow findAcc(T_i^P, Acc_{req})$
4. $(L, \hat{K}, \hat{C}) \leftarrow$ selectLKCP($T_i^P$, $Acc$)
5. $TPrice(T_i^P) \leftarrow Price_i \times CA_i$
6. For each $P_j \in \emptyset : 1 \leq j \leq |\emptyset|$
7. If $(L, \hat{K}, \hat{C}) \neq T_j^P$
8. $T_j^P \leftarrow T_j^P \cup buildPT(T_j^A, \{L, \hat{K}, \hat{C}\}, PAP_i)$
9. End if
10. $TPrice(T^M) \leftarrow TPrice(T^M) \times Price_i$
11. End for
12. Return $TPrice(T^M), L, K, C;
data quality for classification analysis. PHDMashup serves as the core data mashup protocol in our framework. Yet, we would like to emphasize that Fung et al. [11] 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 4 buildTM: Mashup Table Construction

Input: contributing DaaS providers D
Input: privacy requirements L, K, C
Output: mashup table T^M

Procedure buildTM presented in Algorithm 4 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 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 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 buildTM returns both the mashup table T^M and its accuracy Acc(T^M) (Line 3).

F. Data Request Satisfaction

Having constructed the mashup table T^M and determined its accuracy Acc(T^M) and price TPrice(T^M), the mashup coordinator must ensure the requested accuracy Acc_req and the bid price BPrice_req are fulfilled such that TPrice(T^M) ≤ BPrice_req and Acc(T^M) ≥ Acc_req. If Acc_req and BPrice_req are not fulfilled, the mashup coordinator constructs another mashup table T^M and verifies the fulfillment again. If no T^M table can fulfill Acc_req and BPrice_req simultaneously, the mashup coordinator recommends alternative solutions with a higher price or lower accuracy. Figure 4 illustrates the activity diagram for request satisfaction.

The goal of the mashup coordinator is to find the mashup table T^M whose price TPrice(T^M) is the lowest possible among all mashup tables satisfying requested attributes Acc_req. As illustrated in Section II-A, Price = Acc × PA_i for any privacy requirements L, K, C, where PA_i is the price per attribute of provider P_i. Therefore, in order for TPrice(T^M) to be the lowest possible, Acc(T^M) must be as close as possible to Acc_req. The mashup coordinator iteratively executes compTPrice and buildTM procedures to identify a mashup table T^M such that its accuracy Acc(T^M) is closest to Acc_req and greater than or equal to Acc_req.

If no lower accuracy can be found in the price table T_i of the first selected contributing DaaS provider P_i, but both Acc_req and BPrice_req are satisfied, then the mashup coordinator returns the anonymized mashup table T^M with its total price TPrice(T^M) and final accuracy Acc(T^M) to the data consumer.

The mashup coordinator might recommend alternative solutions if Acc_req and BPrice_req could not be mutually fulfilled. For instance, for any mashup table T^M that satisfies all requested attributes Acc_req, if Acc(T^M) is always less than Acc_req, then the mashup coordinator suggests the data consumer the mashup table T^M whose accuracy Acc(T^M) is the highest achievable accuracy, given that TPrice(T^M) might be higher than the bid price BPrice(T^M).

IV. EXPERIMENTAL EVALUATION

We implemented our proposed architecture in Microsoft Windows Azure2, a cloud-based computing platform. Platform as a service (PaaS) [14] is a class of cloud computing services that provides a computing platform, including operating systems, databases, and web servers, as a service to the users. PaaS offers large storage, high reliability, and easy maintenance. Our developed web services are deployed

2http://www.microsoft.com/azure/
in Microsoft Windows Azure Cloud Services together with Microsoft SQL Azure as the storage for DaaS providers in a distributed environment. Our works are applicable to other PaaS providers who support similar services. DaaS providers are distributed in a cloud environment, each of which is implemented on a Windows Server 2008 R2 running on AMD Opteron\textsuperscript{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 [15] 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 \textit{revenue} with two levels, \( \leq 50K \) or \( > 50K \). We perform our experiments with the assumption of having three DaaS providers in the system. Thus, the adult data is vertically partitioned into three overlapping partitions, each of which contains 6 attributes. The partitions are used to construct the attribute tables \( T_1^A, T_2^A, \) and \( T_3^A \) corresponding to providers \( P_1, P_2, \) and \( P_3 \), respectively.

Table I shows the attributes of each data provider. Each table contains 6 attributes. The common attributes are coloured in gray. The tables share a common \textit{UID} for joining. The sensitive attribute in each table is \textit{Marital-Status}, with two values: \textit{Divorced} and \textit{Separated}. The remaining 6 attributes in each table are the \textit{QID} attributes. The taxonomy trees of all categorical attributes can be found in [16].

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 \textit{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 \( |A_{\text{req}}| \), classification analysis \( A_{\text{req}} \), and bid price \( B_{\text{Price}_{\text{req}}} \).

A. Impact of Privacy Requirements on Revenue

To evaluate the impact of \textit{LKC}-privacy requirements on the revenue of each DaaS provider, we use all 45,222 records of each data for anonymization, build classifier \textit{C4.5} 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 \( F_{\text{Acc}} \), and then compute the revenue of each DaaS provider \( P_i \) with
respect to its price per attribute $P_{A_i}$.

Figure 5 illustrates the impact of $L, K, C$ thresholds on the revenue of each DaaS provider. Figure 5.a depicts the effect of threshold $L$. We observe that the revenue of each DaaS provider is insensitive to threshold $L$ when $L \geq 2$. Figure 5.b depicts the effect of threshold $K$. The revenue of $P_1$ and $P_2$ is mainly unaffected by the change of value of $K$. However, the increase of the value of $K$ might negatively impact the revenue, as is the case with DaaS provider $P_3$, 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 5.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 Algorithm 1; Contributing DaaS Providers, corresponding to Algorithm 2; and Final Mashup $T^M$, corresponding to Algorithm 3 and Algorithm 4. Figure 6 depicts the runtime of each phase when the number of requested attributes $A_{req}$ ranges between 4 and 13 attributes, with three different values of the pair $(Acc_{req}, BPrice_{req})$.

Figures 6.a, 6.b, and 6.c depict the runtime of each phase when the classification accuracy and bid price pair $(Acc_{req}, BPrice_{req})$ is equal to (70%, $3,000$), (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. This is due to the fact that sometimes the integration procedure $buildTM$ in Algorithm 4 might be executed more than once to satisfy the consumer’s request with regard to the bid price and data utility level. Note that in Figure 6.c, the total runtime when $|A_{req}| = 13$ is 12 sec, in contrast to 75 sec in Figure 6.a.
and 95 sec in Figure 6.b. This is because $Acc_{req} = 90\%$ and $P_{req} = \$15,000$ are both beyond the threshold of accuracy and price in the DaaS providers’ price tables. In this case algorithm 3 selects the highest accuracy from the data providers’ price tables and computes the corresponding total cost while avoiding the need to find higher or lower accuracies, which reduces the number of times Algorithm 4 needs to run. Consequently, we conclude that our proposed solution is efficient with regard to the number of requested attributes $|A_{req}|$, classification analysis $Acc_{req}$, and bid price $BPrice_{req}$.

C. Scalability

We evaluate the scalability of our algorithm with respect to data volume by blowing up the size of the Adult data set. First, we combined the training and testing sets, giving 45,222 records. For each original record $r$ in the integrated set, we created $\alpha - 1$ “variations” of $r$, where $\alpha > 1$ is the blowup scale.

For scalability evaluation in order to show the sensitivity to the change of requested accuracy and bid price values, we consider three different combinations for requested accuracy $Acc_{req}$ and requested price $BPrice_{req}$ according to the price table of data providers. Each line in Figure 7.a, Figure 7.b, and Figure 7.c illustrates the total runtime of our solution for different number of requested attributes $|A_{req}|$ by consumer.

Figure 7.a depicts the total runtime of our algorithm from 200,000 to 1 million records for $Acc_{req} = 70$ and $BPrice_{req} = 3000$. The total runtime of answering consumer’s request with consideration to the consumer’s data attribute requirement for 1 million records is 146s when the number of requested attributes $|A_{req}| = 5$, 230s when the number of requested attributes $|A_{req}| = 9$, and 339s when the number of requested attributes $|A_{req}| = 13$.

Figure 7.b depicts the total runtime of our algorithm from 200,000 to 1 million records for $Acc_{req} = 80$ and $BPrice_{req} = 9000$. The total runtime of answering 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 7.c depicts the total runtime of our algorithm from 200,000 to 1 million records for $Acc_{req} = 90$ and $BPrice_{req} = 15000$. The total runtime of answering 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 $|A_{req}| = 13$.

The total runtime in Figure 7.c is less than the total runtime in Figure 7.a and the total runtime in Figure 7.b because $Acc_{req}$ and $BPrice_{req}$ in Figure 7.c go beyond the accuracy threshold and prices specified in the providers’ price tables.

The runtime of all three figures scale linearly with respect to the data set’s size. The experimental results on real-life data sets suggest that our algorithm is scalable with respect to the number of requested attributes and the number of records with different accuracy and bid price requirements.

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. [17] propose a OWL-S hybrid approach for approximate matchmaking of requests and web services using approaches from information retrieval. In [5], Vaculin et al. propose an RDF-based framework for modeling and discovering data providing services (DPS). They use a matchmaker component for service discovering, and then make service requesters interact with DPSs directly while assuming that there is no direct communication between different DPSs. 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. [18] introduce the concept of minimal information sharing that allows for only the metadata to be shared between data owners for the purpose of answering queries that span over multiple private databases. On the other hand, secure multiparty computation (SMC) [19] [20] 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 [21]. Sweeney [3] uses generalization and suppression to achieve K-anonymity for a datafly system. Preserving classification information in K-anonymous data is studied in [16] [22]. Mohammed et al. [23] propose a top-down specialization algorithm to securely integrate two vertically partitioned distributed data tables into a K-anonymous table. Trojer et al. [24] 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 [6], which proposes an LKC-privacy model with an anonymization algorithm to address the problem of high-dimensional anonymization. To achieve LKC-anonymity for the integrated data we utilize [11], which provides a service-oriented architecture for achieving LKC-anonymity in the privacy-preserving data mashup. We choose these two models because an LKC-privacy model provides a stronger privacy guarantee than K-anonymity with regard to linkage attacks, and to avoid significant information loss when K-anonymity is applied on high-dimensional data.

Finally, Bhowmick et al. [25] analyze the process of designing a privacy-preserving data integration model and highlight the privacy and security challenges and concerns. Barhamgi et al. [26] 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

We have implemented a DaaS mashup cloud-based framework for the online market, and generalize the privacy and information requirements to the problem of privacy-preserving DaaS mashup with the objective of generating anonymous answers to a variety of data mining queries requested by consumers. We propose a solution for secure collaboration between the most suitable set of DaaS providers 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.

ACKNOWLEDGMENT

REFERENCES


