PRIVACY-PRESERVING DATA-AS-A-SERVICE MASHUPS

MAHTAB ARAFATI

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2013

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mahtab Arafati**

Entitled: **Privacy-Preserving Data-as-a-Service Mashups**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Information Systems Security

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Abdessamad Ben Hamza	Chair
Dr. Jeremy Clark	Examiner
Dr. Anjali Agarwal	Examiner
Dr. Benjamin Fung	Supervisor

Approved by _____

Chair of Department or Graduate Program Director

_____ September 12, 2013 _____ _____

Dr. Christopher Trueman, Dean

Faculty of Engineering and Computer Science

Abstract

Privacy-Preserving Data-as-a-Service Mashups

Mahtab Arafati

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 thesis, 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 is scalable to large set of databases and it can efficiently and

effectively satisfy the data privacy and data mining requirements specified by the DaaS providers and the data consumers.

Acknowledgments

I wish to express my gratitude and indebtedness to Dr. Benjamin C. M. Fung, for his support, generous consultation and giving me his precious time throughout this study. In this clear sighted way, he has always been inherently sympathetic and social in character, a perspective which I am now beginning to understand after two years of study at this university.

Furthermore, I am endlessly grateful to my family for their unwavering supports and motivation throughout this entire process.

To my husband *Mahdi* with love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mashup is a web technology that integrates information from multiple web applications into a new web application. For instance, *Trendsmap.com*[1] 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 [But06] and addressing enterprises' business needs [Jhi06].

Data-as-a-Service (*DaaS*) is an emerging cloud computing service that provides data on demand to consumers across various cloud platforms via different network 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 thesis, we propose a privacy-preserving DaaS mashup framework that allows DaaS providers to securely integrate and trade their collected person-specific survey data

[1] http://www.trendsmap.com/

to support data mining. In the market, *DaaS providers* can register and advertise their available data, and *data consumers* can submit their data mining requests, such as classification analysis with a minimum accuracy requirement. Then a *mashup coordinator* in the framework dynamically determines the group of DaaS providers whose data 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 [Swe02a] [Sam01] 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 thesis 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 3 provides a formal description of the objectives and behaviour of the participants in the proposed framework.

Contribution #2. Vaculin et al. [VHNS08] 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 4 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, the efficiency, and scalability 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 5 shows the experimental results.

The rest of the thesis is organized as follows. Chapter 2 reviews the preliminaries together with related works. Chapter 3 describes the participants of our framework and provides the formal definition for our DaaS mashup framework. Chapter 4 illustrates our

proposed solution. Comprehensive experimental results are presented in Chapter 5. Finally, we conclude the thesis and identify some future works in Chapter 6.

Chapter 2

Background

In this chapter, we review the preliminary constructs that are required to understand the research problem and solution discussed in the subsequent chapters. Next, we review the literature in the related research areas.

2.1 Preliminaries

In privacy-preserving data publishing, choosing the appropriate privacy model and anonymization techniques are important issues. When a data provider publishes nothing, data privacy is maximized, whereas privacy protection can not be guaranteed when a data provider releases raw data without anonymization. Consequently, it is essential for a data provider to employ a proper data privacy model and an anonymization mechanism. In this section, we discuss different privacy models and anonymization techniques.

The scenario presented in this thesis involves multiple DaaS providers in a cloud environment. Thus, we will study a specific data mashup algorithm that is designed for performing privacy-preserving data mashup on high-dimensional data [FTH+12]. Furthermore, we will discuss the concept of Platform-as-a-Service in cloud computing [MG11].

2.1.1 Privacy Models

In 1977 Dalenius [Dal77] provided a very stringent definition for privacy protection: "Access to the published data should not enable the attacker to learn anything extra about any target victim compared to no access to the database, even with the presence of any attacker's background knowledge obtained from other sources." Dwork [Dwo06], in 2006, showed it is impossible to claim such an absolute privacy protection due to the presence of an attacker's background knowledge. Suppose the job of an individual is sensitive information. Assume an attacker knows that Bob's job is a professional job. If an attacker has access to a statistical database that discloses the professional jobs, then according to Dalenius [Dal77], Bob's privacy is compromised, regardless of whether or not his record is in the database [Dwo06]. Consequently, an attacker having background knowledge can perform various kinds of privacy attacks. Accordingly, different kinds of privacy models have been proposed to address this issue.

In the most basic form of privacy-preserving data publishing (PPDP), the data publisher's table has a form of:

$$D(Explicit\_Identifier, Quasi\_Identifier, Sensitive\_Attributes, Non-sensitive\_Attributes),$$

where $Explicit\_Identifier$ is a set of attributes that explicitly identify an individual, such as SSN and name. These attributes must be removed before publishing the data. $Quasi\_Identifier$ (QID) is a set of attributes whose combined value may potentially identify an individual. For example, the combination of gender, native-country, and job. The values of these attributes may be available publicly from other sources. $Sensitive\_Attributes$ contain sensitive person-specific information, and an adversary is not permitted to link their values with an identifier. Examples are disease, salary, etc. $Non-sensitive\_Attributes$ consist of all attributes that do not fall into the previous three categories [BBSPA03]. Based

on quasi-identifiers, an attacker may still be able to perform different types of privacy attacks. We explain the most common linkage attacks below.

Record linkage attack. In this attack, a small number of records in the released data table T, named a *group*, can be identified by some value qid on QID. If the target victim's QID matches the value qid, then the target victim is vulnerable to being linked to the group. In this case, an adversary needs to use background knowledge to be able to identify the victim's record from the group.

Job	Sex	Age	Salary
Writer	Male	25	50K
Writer	Male	21	50K
Dancer	Male	27	35K
Dancer	Male	25	30K
Engineer	Female	38	50K
Doctor	Female	30	45K
Doctor	Female	30	45K

Table 1: Raw Employee Data

Example 1. Suppose an organization wants to publish an employee's record in Table 1 to a research center. If an adversary knows there is a record in the table that belongs to Bob, a male writer who is 21 years old, he can deduce that Bob has a salary of 50K dollars because there is only one record with $qid =< Writer, Male, 21 >$ in the table. ■

Attribute linkage attack. This attack is not similar to record linkage attack as an adversary may not need to exactly identify the record of a target victim V. In an attribute linkage attack, an adversary could infer some sensitive information about V based on the set of sensitive attributes associated with the group to which V belongs. In other words, he wants to utilize background knowledge of the victim's qid to infer sensitive values with a certain degree of confidence. This attack is effective if the confidence, calculated by $P(s|qid) =$

$\frac{|T[qid \wedge s]|}{|T[qid]|}$, is high, i.e., $|T[qid]|$, the number of records in T containing qid, is small.

Name	Job	Sex	Age
Bob	Writer	Male	21
Cathy	Doctor	Female	30
Peter	Writer	Male	25
Alice	Doctor	Female	30
John	Writer	Male	29
Henry	Dancer	Male	27
Renee	Engineer	Female	38
Bob	Dancer	Male	25
Linda	Doctor	Female	31

Table 2: External Data Table

Example 2. Following Table 1, the adversary can infer that all female doctors at age 30 have the sensitive attribute salary 45K. Applying this information to Table 2, an adversary can infer that Alice has an income of 45K with 100% confidence, provided she comes from the same population in Table 1. This example shows an effective attack because the number of records in Table 2 containing $qid =< Doctor, Female, 30 >$ is small. ■

In general, different privacy models have been proposed to prevent an adversary linking to an individual with sensitive information, given the knowledge of the quasi-identifier attributes. In this section, we discuss four different privacy models: K-anonymity [Sam01] [Swe02b], ℓ-Diversity [MKGV07], Confidence Bounding [WFY07], and LKC-Privacy [MFHL09].

K-Anonymity

The notion of K-anonymity, first proposed by Samarati and Sweeney [Sam01] [Swe02b], is to prevent a record linkage attack through QID. K-anonymity declares the minimum group size on QID in a table is K, and a table satisfying this requirement is called *K-anonymous*. In a K-anonymous table, the probability of linking a victim to a certain record through QID is at most $1/K$ because it is indistinguishable for an adversary to identify a record from at least $k - 1$ other records with respect to QID.

Job	Sex	Age	Salary
Artist	Male	[20-30)	>40K
Artist	Male	[20-30)	>40K
Artist	Male	[20-30)	<40K
Artist	Male	[20-30)	<40K
Professional	Female	[30-40)	>40K
Professional	Female	[30-40)	>40K
Professional	Female	[30-40)	>40K

Table 3: 3-Anonymous Employee Data by Generalization

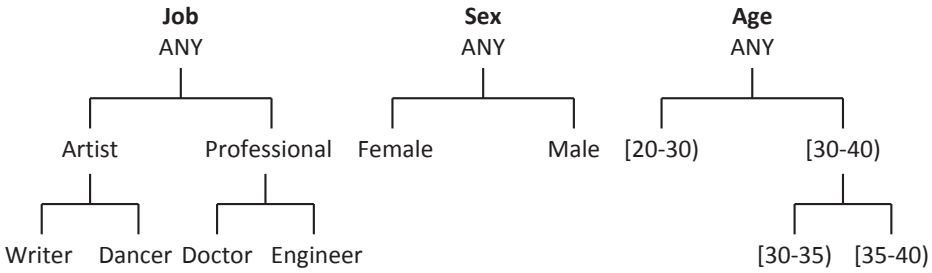

Figure 1: Taxonomy Trees for *Job, Age, Sex*

Example 3. Table 3 shows a 3-anonymous table; each group of distinct QID groups, $< Artist, Male, [20, 30) >$ and $< Professional, Female, [30, 40) >$, contains at least 3 records. This table generalizes $QID = \{Job, Sex, Age\}$ from Table 1 using taxonomy trees in Figure 1. ■

ℓ-Diversity

Machanavajjhala et al. [MKGV07] proposed a ℓ-diversity privacy model to prevent attribute linkage attacks and pointed out that the K-anonymity model cannot prevent such attacks. The ℓ-diversity model needed every qid group to contain at least ℓ "well-represented" sensitive values. The notion of "well-represented" has different interpretations, which some instantiations represented in [MKGV07]. The simplest one is to ensure that the sensitive attribute has at least ℓ distinct values in each qid group.

Example 4. Table 4 shows a 2-diverse table, in which each qid group contains at least two distinct values. Thus, even if an adversary figures out the qid group containing the record

Job	Sex	Age	Salary
Artist	Male	[20-30)	>40K
Artist	Male	[20-30)	>40K
Artist	Male	[20-30)	<40K
Artist	Male	[20-30)	<40K
Professional	Female	[30-40)	>40K
Professional	Female	[30-40)	>40K
Professional	Female	[30-40)	>40K

Table 3: 3-Anonymous Employee Data by Generalization

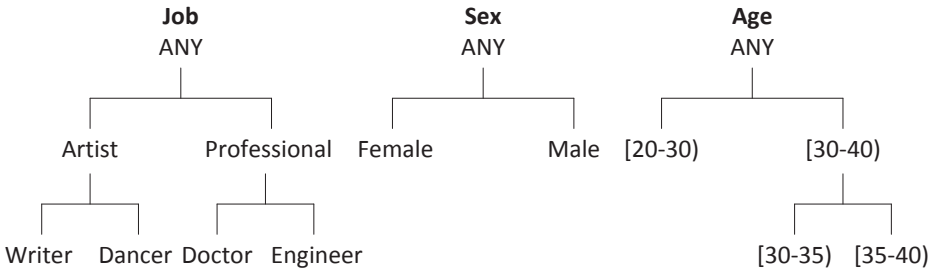

Figure 1: Taxonomy Trees for *Job, Age, Sex*

Example 3. Table 3 shows a 3-anonymous table; each group of distinct QID groups, $< Artist, Male, [20, 30) >$ and $< Professional, Female, [30, 40) >$, contains at least 3 records. This table generalizes $QID = \{Job, Sex, Age\}$ from Table 1 using taxonomy trees in Figure 1. ■

ℓ-Diversity

Machanavajjhala et al. [MKGV07] proposed a ℓ-diversity privacy model to prevent attribute linkage attacks and pointed out that the K-anonymity model cannot prevent such attacks. The ℓ-diversity model needed every qid group to contain at least ℓ "well-represented" sensitive values. The notion of "well-represented" has different interpretations, which some instantiations represented in [MKGV07]. The simplest one is to ensure that the sensitive attribute has at least ℓ distinct values in each qid group.

Example 4. Table 4 shows a 2-diverse table, in which each qid group contains at least two distinct values. Thus, even if an adversary figures out the qid group containing the record

Job	Sex	Age	Salary
Writer	Male	*	50K
Writer	Male	*	50K
Writer	Male	*	35K
*	Female	25	30K
*	Female	25	30k
*	Female	25	45K
Doctor	*	38	70k
Doctor	*	38	70k
Doctor	*	38	40k

Table 4: 2-Diverse Data table

of an individual, he can determine the real sensitive value of the individual with no more than 50% confidence.

Confidence Bounding

Wang et al. [WFY07] presented an alternative privacy model, Confidence Bounding, to prevent attribute linkage attacks. They considered bounding the confidence of inferring a sensitive value from a qid group by determining one or more *privacy templates* of the form $< QID \rightarrow s, h >$, where QID is a quasi-identifier, s is a sensitive value, and h is a threshold.

Example 5. Consider $QID = Job, Sex, Age$, $< QID \rightarrow> 40K, 20\% >$, which states the confidence of inferring someone's salary >40K from any group on qid is no more than 20%. Table 3 shows this privacy template is violated because the confidence of inferring a salary >40K is 50% in the group $< Artist, Male, [20-30) >$. ■

LKC-Privacy

Many privacy models such as K-anonymity [Swe02a] [Sam01] [Swe02b] and its extensions [LDR06a] [MKGV07] [XT06] have been proposed in the last decade to thwart record and attribute linkage attacks in the context of relational databases. LKC-privacy [MFHL09] 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 data providers, often resulting in a high-dimensional integrated table. Extensive experimental results [MFHL09] have shown that enforcing other traditional privacy models would result in poor data mining quality in the anonymized data.

The general intuition of LKC-privacy is to ensure that every qid with a maximum length L in the data table T is to be shared by at least a certain number of records K. The confidence of inferring any sensitive values in S is at most C, where L, K, C are data provider-specified privacy thresholds, and S is a set of sensitive values determined by the data providers. The *LKC*-privacy model bounds the probability of a successful record linkage attack to be $\leq 1/K$ and the probability of a successful attribute linkage attack to be $\leq C$, provided an adversary's prior knowledge does not exceed L.

LKC-privacy generalizes several traditional privacy models. K-anonymity [Swe02a] [Sam01] [Swe02b] is a special case of LKC-privacy if $L = |QID|$ and $C = 100\%$, where $|QID|$ is the number of QID attributes in the data table T. Confidence bounding [WFY07] is also a special case of LKC-privacy if $L = |QID|$ and $K = 1$. Consequently, traditional models are available for data providers, if needed.

2.1.2 Anonymization Techniques

In order to achieve the privacy models, anonymization techniques must be applied to the raw data to make them less precise. There may be more than one anonymization technique usable to achieve a privacy model, and choosing the right technique leads to a better trade-off between data privacy and utility. In the following we present some widely used techniques often used for anonymization.

Suppression

The simplest anonymization technique is suppression. It is achieved by replacing (suppressing) an attribute value of a cell with a special symbol e.g., "*", or "Any." For example,

Job	Sex	Age	Salary
Artist	Male	[20-30)	>40K
Artist	Male	[20-30)	>40K
Artist	Male	[20-30)	<40K
Artist	Male	[20-30)	<40K
Professional	Female	*	>40K
Professional	Female	*	>40K
Professional	Female	*	>40K

Table 5: 3-Anonymous Employee Data by Suppression

in Table 5 certain values of the Age attribute are suppressed to ensure 3-anonymity.

There are different schemes for suppression: *Record suppression* [BA05] [Iye02] [LDR05] [Sam01], *Value suppression* [WFY05] [WFY07], and *Cell suppression* (or local suppression) [Cox80] [MW04], which respectively refer to suppressing an entire record, suppressing every instance of a given value in a table, and suppressing some instances of a given value in a table.

Generalization

Generalization provides better data utility than suppression because it replaces the attribute values by more general values using taxonomy trees. Generalization uses some intermediate states according to the given taxonomy tree to anonymize a table. For a categorical attribute, a specific value can be replaced with a general one and for a numerical attribute, an interval covering the exact values can be used. Table 3 shows 3-anonymous data by generalization using Figure 1 as a taxonomy tree. For instance, it shows the values *Writer* and *Dancer*, for the categorical attribute *Job*, replaced by a more general value, *Artist*; the value 25, for numerical attribute *Age*, is replaced by an interval [20,30), according to the taxonomy tree.

Generalization techniques have two main categories: global and local generalization [LDR05]. Global generalization refers to mapping all instances of a value to the same general value; in local generalization different instances can be mapped to different general values.

Bucketization

Bucketization [XT06] [MKM+07] is the notion of dividing all the records into several buckets in such a way that each bucket identifies by an ID. Bucket IDs are stored, along with encrypted data, on the server. There are two general bucketization methods to select the bucket ranges: equi-width and equi-depth. Equi-width bucketization works well with uniformly distributed data because each bucket is the same size. However, equi-depth bucketization may be more suitable for non-uniformly distributed data because each bucket has the same number of items.

Other anonymization techniques include a randomization-based approach and an output perturbation-based approach. A randomization-based approach adds noise to the underlying data if the attributes are numerical attributes and replaces the values with other values from the domain for the categorical attributes [AS00] [EGS03]. Randomization-based approach is useful if the applications need to preserve data truthfulness at the aggregated level, but not at the record level. On the other hand, the output perturbation-based approach first computes the correct result and then adds noise in order to output the perturbed result.

2.1.3 Privacy-Preserving High-Dimensional Data Mashup (PHDMashup)

Data Mashup is a technology used to integrate data from multiple providers based on a user's request. Integrating data from multiple sources always brings some challenges. First, integrating multiple private data sets from different data providers without using any privacy models and anonymization techniques would reveal sensitive information to the other data providers. Moreover, the mashup data might reveal some sensitive information that was not available before the mashup. Utilizing a traditional privacy model such as K-anonymity on the raw data would result in the curse of high dimensionality problem [Agg05]; the high-dimensional data could result in useless data for various data analyses.

Consider $\mathbb{D} = \{P_1, \ldots, P_m\}$ be a set of data providers, where each provider $P_i \in \mathbb{D} : 1 \leq i \leq m$ 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*) [FTH+12] is a secure protocol that addresses all the aforementioned challenges and generates a mashup table that fulfills a specified LKC-privacy requirement while containing as much information as possible for simple or complex data analysis. During the integration process each data provider learns nothing about the other provider's data more than the data in the final mashup table. In this thesis, *PHDMashup* serves as the core data mashup protocol of our framework.

2.1.4 Platform-as-a-Service (PaaS)

PaaS is a class of cloud computing services that provide a computing platform-as-a-service in the cloud. In this model, cloud providers offer a computing platform that includes operating systems, databases, and web servers. By subscribing to this service, consumers can easily deploy applications in a cloud-based infrastructure without paying high costs that include purchasing, maintaining, and configuring hardware and the software required to deploy applications. We require the *PaaS* model to have large-scale storage capabilities and availability, reliability, and easy maintenance.

In this thesis we choose *Microsoft Windows Azure* because it supports all the aforementioned properties in addition to our familiarity with Windows (ease of use), as well as different pricing models. The *Windows Azure Cloud Services* are utilized to deploy the web services of our framework in the cloud, and *Microsoft SQL Azure* is used to store the DaaS providers' databases in a distributed environment.

In a traditional on-premise application, both the database and the application code are

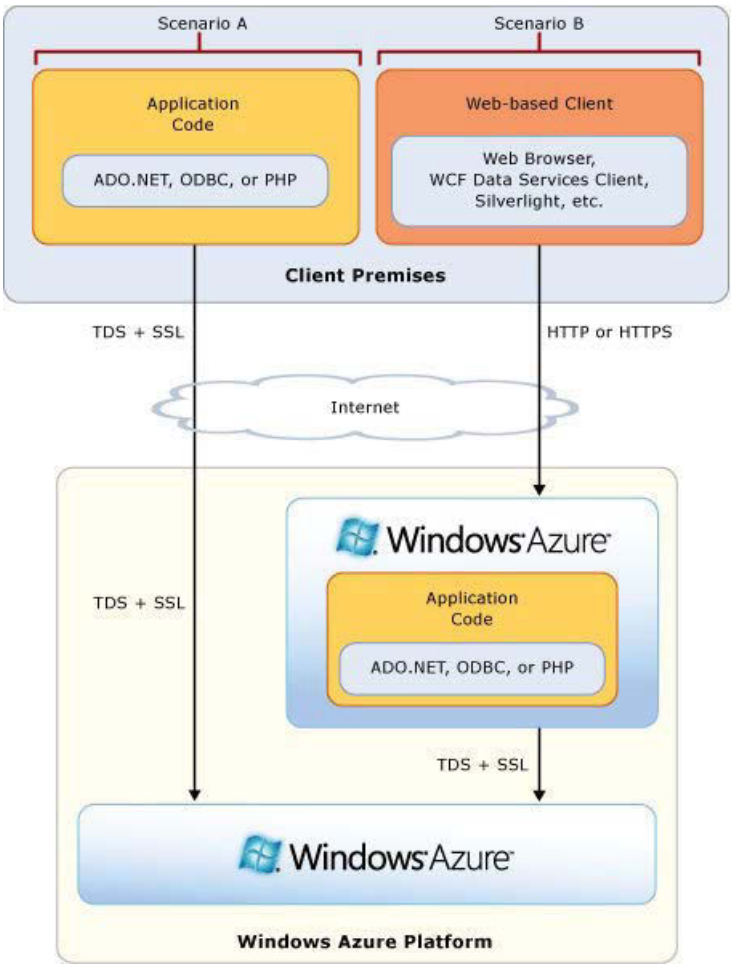

Figure 2: Windows Azure SQL Database Data Access

located in the same physical data center. The Windows Azure platform offers new alternatives to that architecture. Figure 2[1] illustrates two scenarios of how the application service can access data in *Windows Azure SQL Database*.

Scenario A, shown on the left, depicts a model for the time the application code locates on the premises of a corporate data center, but the database resides in a Windows Azure SQL database. An application code needs to use client libraries to access database(s) in a Windows Azure SQL Database. Regardless of the client library chosen, data is transferred using a tabular data stream (TDS) over a secure sockets layer (SSL).

Scenario B, shown on the right, depicts a model for the time the application code is

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

hosted in Windows Azure and the database locates in a Windows Azure SQL Database. The application code in this scenario also needs to use client libraries to access the database(s) in a Windows Azure SQL Database. There are many different types of applications that can be hosted by Windows Azure platform (e.g., web applications or mobile applications). The client premises in this scenario may represent an end user's web browser, someone who wants to access a web application, or a desktop or Silverlight application that uses the benefits of the WCF Data Services client to access data hosted in a Windows Azure SQL Database via HTTP or HTTPS protocol.

2.2 Related Work

In this section, we review the literature examining several areas related to our work. We discuss privacy-preserving data publishing and privacy-preserving distributed data mining researches. We also discuss solutions that enable integration of data along with preserving data privacy. We then discuss the studied techniques of discovering web services for data integration.

2.2.1 Privacy-Preserving Data Publishing (PPDP)

Privacy-preserving data publishing (PPDP) provides methods and tools to publish data in such a way that the published data remain useful while individual privacy is preserved. The data publisher first collects data from data owners and then publishes the collected data to a *data recipient*, who might be a data miner who wants to conduct data mining on the published data. The data mining task could be a simple or complex task, e.g., classification or clustering analysis. For example, an organization collects information from its employees and wants to publish an employee's record to an external research center. In this case, the organization is the data publisher, the employees are data owners, and the

research center is the data recipient that wants to do the data mining task on the collected data.

Data publishers can be classified into two models: untrusted vs. trusted [Geh06]. In the untrusted model, the data publisher may attempt to obtain some sensitive information from the data owner anonymously. Examples are various cryptographic solutions [YZW05], anonymous communications [Cha81] [JJR02], and statistical methods [War65]. In the trusted model, the data publisher is trusted and the data owners are willing to share their sensitive information with the data publisher. In this thesis, we follow the trusted model for the data publisher, and we call the data publishers DaaS providers.

In a practical data publishing situation, a data recipient could be an attacker. For example, there might be untrustworthy people in the research center where a data publisher wants to publish the employee's information. The solution to this problem is very different from cryptographic approaches, where authorized parties can only access published data. The solution is to preserve both privacy and information utility in the anonymous data. In our work, we take advantage of the LKC-privacy model to protect privacy and information usefulness in our result.

For privacy-preserving relational data publishing, there is a large body of work on anonymizing relational data, based on partitioned-based privacy models. As we discussed earlier, K-anonymity [Sam01] [Swe02b], ℓ-Diversity [MKGV07], and Confidence Bounding [WFY07] [WFY05] are based on single QID-based approaches that suffer from the curse of high dimensionality [Agg05], which would result in useless data for data mining. In Chapter 3 we address this problem by utilizing a LKC-privacy model, which assumes an adversary knows at most L values of QID attributes of any target victim V.

There are a few algorithms that have proposed solutions for classification analysis [FWCY10]. The examples are [FWY05] [FWY07] [LDR06b] [Iye02] [WYC04]. In these works, researchers have built a classifier based on anonymized data, and then evaluated performance

on the testing sets.

Other examples for privacy-preserving data publishing models are as follows: Sweeney [Swe02a] uses generalization and suppression to achieve K-anonymity for datafly systems. Preserving classification information in K-anonymous data is studied in [FWY07] [LDR06b]. Mohammed et al. [MFWH09] propose a top-down specialization algorithm to securely integrate two vertically partitioned distributed data tables into a K-anonymous table. Trojer et al. [TFH09] present a service-oriented architecture for achieving K-anonymity in the privacy preserving data mashup scenario. Our work has a combination of a single data source and integrated data source privacy levels. To preserve the privacy of data of each DaaS provider, we utilize [MFHL09], which proposes a 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 [FTH$^+$12], which provides a service-oriented architecture for achieving LKC-anonymity in the privacy preserving data mashup. We choose these two models for two reasons: a 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.

2.2.2 Privacy-Preserving Distributed Data Mining (PPDDM)

Privacy-Preserving Distributed Data Mining (PPDDM) presents a scenario for multiple data providers who need to do data mining tasks on the integrated data in collaboration with each other while preserving the privacy of individuals. For example, multiple organizations want to build a classifier on the salary of employees and publish that classifier to the public, but not the data itself.

Information integration has been an active area of database research [Jhi06] [Wie93] [ERS99]. Two different methods are commonly used for releasing the result of data integration: data sharing and result sharing:

PPDDM usually utilizes an assumption about data providers and cryptographic techniques to achieve its goals. The assumption is that the data providers follow a non-colluding semi-honest model [GMW87], where they do follow the protocol but may try to obtain some additional information about the other providers from the collected data. The secure protocols for constructing different data mining models based on cryptographic techniques are known as *Secure Multiparty Computation (SMC)* [GMW87] [Gol04] [LP09] [Yao82]. It allows the sharing of the computed result (e.g., a classifier) while it prohibits private data from being shared. An example is the secure multiparty computation of classifiers [CKV+02] [DZ02] [ZSR05].

On the other hand, privacy-preserving data publishing (PPDP) allows the publishing of data records about individuals and not the data mining result sharing [FWCY10]. Agrawal et al. [AES03] introduce the concept of minimal information sharing that allows only for the metadata to be shared between data owners for the purpose of answering queries that span multiple private databases. In the thesis, we utilize a privacy-preserving algorithm that enables DaaS providers to share data, not only the data mining results. In many applications data sharing gives greater flexibility than does result sharing because the data recipients can perform their required analysis [FWY07].

2.2.3 Privacy-Preserving Data Integration

Privacy-Preserving Data Integration [BGIC06] refers to solutions that enable integration of data along with preserving the privacy of the data effectively, while data integration [DD99] [Hul97] does not necessarily pay attention to the data privacy.

Bhowmick et al. [BGIC06] analyze the process of designing a privacy-preserving data integration model and highlight the privacy and security challenges and concerns. They preserve privacy by designing a query-rewriting module that receives and rewrites the *XML* query and filters out the result instances that violate the access rules and data privacy.

In [BGIC06], unlike in the data integration researches [DD99] [Hul97], privacy plays the main role and avoids freely releasing data and schema of the sources.

Barhamgi et al. [BBG+11] propose a privacy preserving approach for mashing-up DaaS web services. For protecting privacy, they proposed a model to rewrite queries based on privacy policies defined by data providers. Their privacy model contains a set of rules to specify the recipients to whom data may be disclosed and the purposes that may be used for data. This approach contains two rewriting phases, rewriting the query to satisfy the privacy constraints and rewriting the modified query in terms of available 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 based on the request coming from a data consumer, while preserving privacy of mashup data using LKC-privacy model.

2.2.4 Web Service Discovery for Data Integration

Web services discovery for data integration is an area related to our work. Benatallah et al. [BHRT03] propose a solution to discover web services based on their capabilities. This approach enables a combination of web services to fulfill a consumer's request by making a comparison between the request and available web services in the context of *DAML-S* ontologies of services. In [PKPS02] Paolucci et al. show how to discover a web service and make a semantic match between data providers' advertisements and requests. Their solution is based on *DAML-S* language for service description. Klusch et al. [KFS06] and Vaculin et al. [VHNS08] extend the work of Paolucci et al. [PKPS02].

In [KFS06] Klusch et al. propose a *OWL-S* hybrid approach for approximate match-making of requests and web services using approaches from information retrieval. In

[VHNS08], Vaculin et al. propose an *RDF*-based framework for modeling and discovering data providing services (*DPS*). They use a matchmaker component for service discovery, and then require service requesters to 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 enables interactions between DaaS providers for securely integrating their data in order to answer the data request.

Chapter 3

The Participants

This thesis 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 3. 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.

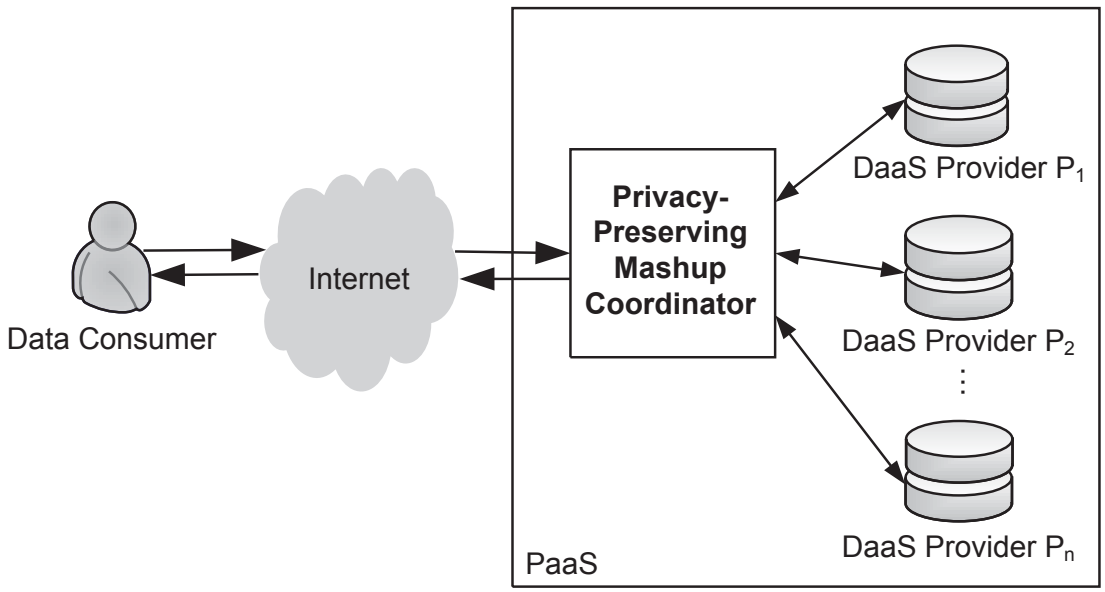

Figure 3: The Framework

3.1 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. 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 $P(s|qid) = \frac{|T[qid \wedge s]|}{|T[qid]|}$, is high.

Many privacy models [Swe02a] [Sam01] have been proposed in the last decade to thwart these linkage attacks. In our proposed framework, we choose to impose LKC-*privacy* [MFHL09] 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 [MFHL09] 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 [Swe02a], confidence bounding [WFY05], and ℓ-diversity [MKGV07]. Therefore, the DaaS providers, if necessary, have the flexibility to employ these traditional privacy models.

Definition 1 (LKC-Privacy [MFHL09])**.** *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.* ■

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 thesis, 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.

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*) [MFHL09] 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* [Qui93] to determine the

Algorithm 1 *buildPT*: **Price Table Construction**

Input: attribute table T_i^A
Input: a set of LKC-privacy requirements PR_i
Input: price per attribute PA_i
Output: price table T_i^P

1: **for** each combination $\{L, K, C\} \in PR_i$ **do**
2: $T_i^{A'} \leftarrow PAIS(T_i^A, L, K, C)$;
3: $Acc \leftarrow 100 - C4.5(T_i^{A'})$
4: $Price = Acc \times PA_i$
5: $T_i^P \leftarrow insert(L, K, C, Acc, Price)$
6: **end for**
7: **return** T_i^P;

classification accuracy Acc of $T_i^{A'}$. The output of $C4.5$ classifier is a classification error (CE) which is a positive number. Thus, Acc has range [0-100]. 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, K, C, Acc, and $Price$ is then inserted into the price table T_i^P (Line 5).

We assume that DaaS providers follow the *non-colluding semi-honest model* [KMR], 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.

3.2 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.

3.3 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}.

3.4 Problem Statement

The problem is defined as follows. Given a person-specific relational database that is vertically partitioned into n sub-databases, each of which is hosted by one DaaS provider $P_i : 1 \leq i \leq n$, the objective is to provide a framework for privacy-preserving DaaS mashup, where the answer to a data consumer's request $req = \{A_{req}, Acc_{req}, BPrice_{req}\}$

is a mashup table T^M such that (1) T^M satisfies all requested attributes A_{req}, and the total price $TPrice(T^M)$ and the classification accuracy $Acc(T^M)$ of T^M satisfy the bid price BP_{req} and desired accuracy Acc_{req}, respectively, (2) T^M satisfies a given $\{L, K, C\}$ privacy requirement agreed by the contributing DaaS providers, and (3) the integration process between DaaS providers is secure.

Chapter 4

The DaaS Mashup Framework

4.1 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 [FTH+12] to securely integrate and anonymize the attribute tables of contributing DaaS providers. It also utilizes classifier $C4.5$ to compute the final classification accuracy

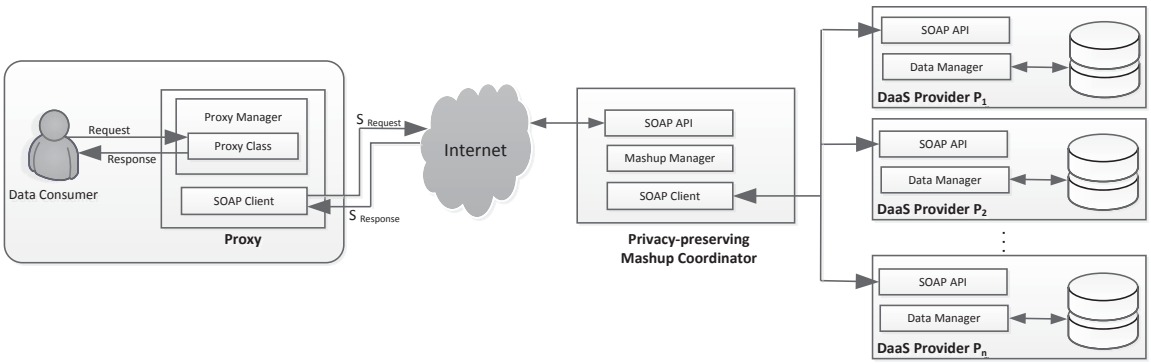

Figure 4: Framework for Privacy-Preserving Data-as-a-Service Mashups: Implementation Architecture

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.

4.2 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 [Jos07]. The implemented architecture of our framework is illustrated in Figure 4.

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 $S_{Request}$ 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.

4.3 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

Algorithm 2 *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 𝔻 = ∅ and D̂ = DP
2: while R ≠ ∅ do
3:    select P_i ∈ D̂ with the least price per attribute PA_i
4:    M_i ← {T_i^A ∩ R}
5:    if M_i ≠ ∅ then
6:       𝔻 ← (P_i, M_i)
7:       R ← R \ M_i
8:    end if
9:    D̂ ← D̂ \ P_i
10: end while
11: return 𝔻;
```

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 2 (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.* ■

In Algorithm 2, 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 [Chv79].

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 4.3.1. *The cost of satisfying all requested attributes A_{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 Algorithm 2 is $O(nlog\ m)$, where n is the number of requested attributes $|A_{req}|$ and m is the number of DaaS providers $|P_i|$. The main loop has complexity $O(n)$ because $|A_{req}| = n$. For *selectDaaSPs*, the major computational cost comes from the selection of DaaS providers with the least price per attribute PA_i. The complexity of selecting DaaS providers with the least PA_i is $O(log\ m)$ using a priority heap.

Example 6. 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 *selectDaaSPs* picks T_4^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 $D = \{(T_4^A, \{a_5, a_6, a_7, a_8\}),$

Algorithm 3 *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** $TPrice(T^M), L, K, C$;

$(T_3^A, \{a_3, a_4\}), (T_5^A, \{a_1\}), (T_6^A, \{a_2\})\}$. The cost of satisfying A_{req} is $(1 \times 4) + (2 \times 2) + (3 \times 1) + (3 \times 1) = 14$. ■

4.4 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 $\mathbb{D}$ determined in Section 4.3, the $compTPrice$ algorithm 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, 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).

4.5 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*) [FTH+12]. 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 $\mathbb{D} = \{P_1, \ldots, P_m\}$ be a set of contributing DaaS providers, where each provider $P_i \in \mathbb{D} : 1 \leq i \leq m$ 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 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. [FTH+12] 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

thesis.

Algorithm 4 *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* 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 $\mathbb{D}$ 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 *buildTM* returns both the mashup table T^M and its accuracy $Acc(T^M)$ (Line 3).

4.6 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 $TPrice(T^M) \leq BPrice_{req}$ and $Acc(T^M) \geq Acc_{req}$. 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 5 illustrates the activity diagram for request satisfaction.

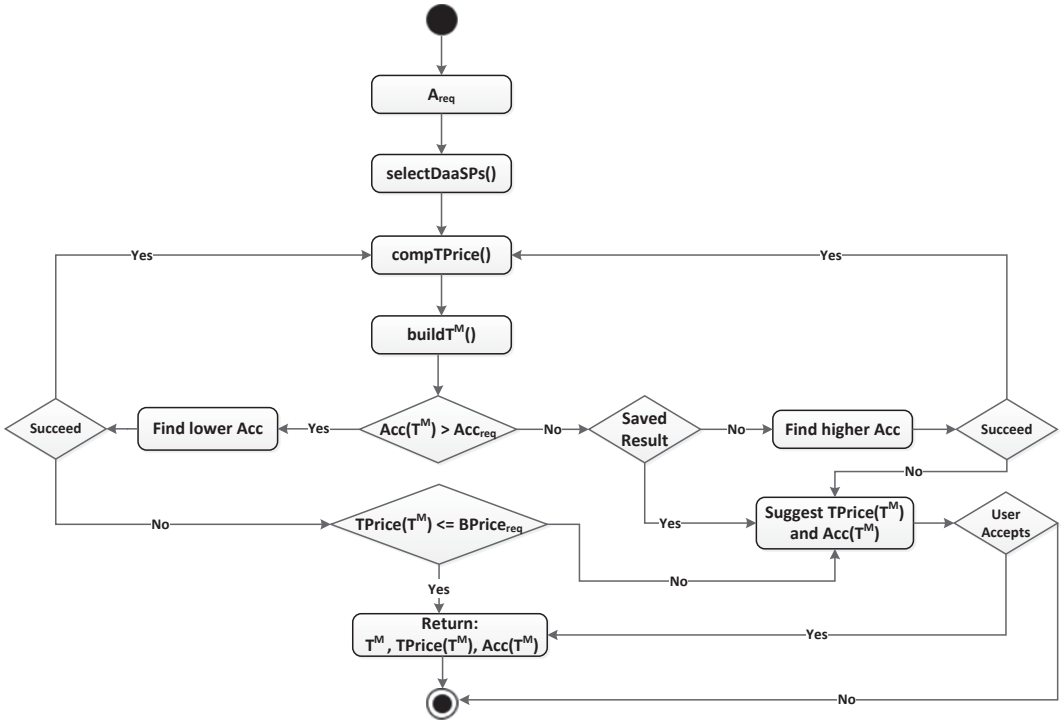

Figure 5: Data Request Satisfaction

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 A_{req}. As illustrated in Section 3.1, $Price = Acc \times PA_i$ 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 $buildT^M$ 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^P 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 A_{req}, if $Acc(T^M)$ is always less than Acc_{req}, then the mashup coordinator suggests to 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 $BPrice(T^M)$.

Chapter 5

Experimental Evaluation

5.1 Implementation

We implemented our proposed architecture in *Microsoft Windows Azure*[1], a cloud-based computing platform. *Platform-as-a-service (PaaS)* [MG11] 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 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^{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 [BL13] in our experiments to illustrate the performance of our proposed framework. The adult data set contains 45,222 census records

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

consisting of eight categorical attributes, six numerical attributes, and a class attribute *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 6 shows the attributes of each data provider. Each table contains 6 attributes. The common attributes are coloured in gray. The tables share a common UID for joining. The sensitive attribute in each table is *Marital-Status*, with two values: *Divorced* and *Separated*. The remaining 6 attributes in each table are the QID attributes. The taxonomy trees of all categorical attributes can be found in [FWY07].

Attribute	Type	numerical range	
		# Leaves	# Leaves
Age	numerical	17-90	
Final-weight	numerical	13492 - 1490400	
Education-Num	numerical	1-16	
WorkClass	categorical	8	5
Education	categorical	16	5
Marital-Status	categorical	7	4

DaaS Provider P1

Attribute	Type	numerical range	
		# Leaves	# Leaves
Education-Num	numerical	1-16	
Marital-Status	categorical	7	4
Occupation	categorical	14	3
Relationship	categorical	6	3
Race	categorical	5	3
Sex	categorical	2	2

DaaS Provider P2

Attribute	Type	numerical range	
		# Leaves	# Leaves
Capital-Gain	numerical	0-99999	
Capital-Loss	numerical	0-4356	
Hours-per-week	numerical	1-99	
Marital-Status	categorical	7	4
Sex	categorical	2	2
Native-Country	categorical	40	5

DaaS Provider P3

Table 6: Attributes of Three DaaS Providers

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 $BPrice_{req}$.

5.2 Impact of Privacy Requirements on Revenue

To evaluate the impact of LKC-privacy requirements on the revenue of each DaaS provider, we use all 45,222 records of each data for anonymization, build classifier $C4.5$ 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 $FAcc$, and then compute the revenue of each DaaS provider P_i with respect to its price per attribute PA_i.

Figure 6, Figure 7 and Figure 8 illustrates the impact of L, K, C thresholds on the revenue of each DaaS provider. The data providers can use these results as a guideline to estimate their revenue with respect to the enforced LKC-privacy requirements.

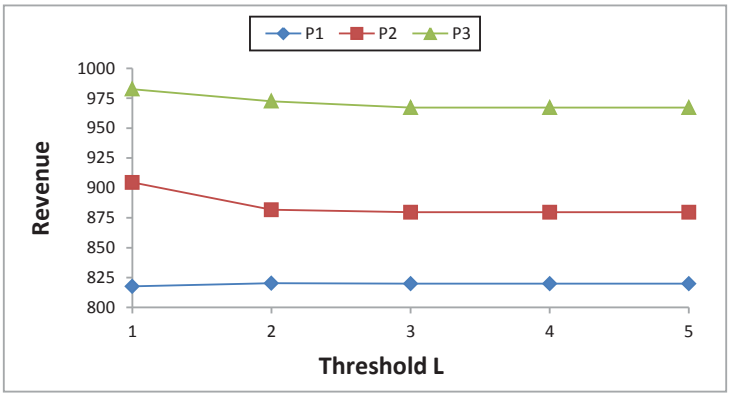

Figure 6: Impacts of Threshold L on DaaS Provider's Revenue

Figure 6 depicts the effect of threshold L. We observe that the revenue of each DaaS provider is insensitive to threshold L when $L >= 2$.

Figure 7 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, the increase of the value of K might negatively impact the revenue, as is the case with DaaS provider P_2, whose revenue dropped

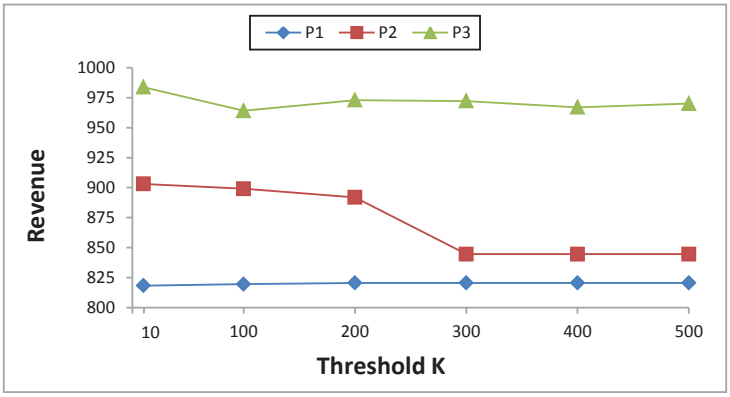

Figure 7: Impacts of Threshold K on DaaS Provider's Revenue

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.

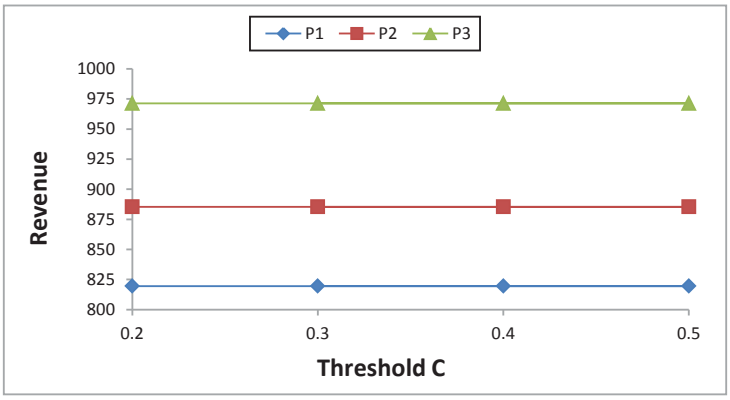

Figure 8: Impacts of Threshold C on DaaS Provider's Revenue

Figure 8 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.

5.3 Efficiency and Scalability

One major contribution of our work is the development of an efficient and scalable 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 $(Acc_{req}, BPrice_{req})$.

Efficiency. 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 9, Figure 10, and Figure 11 depict 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 $(Acc_{req}, BPrice_{req})$.

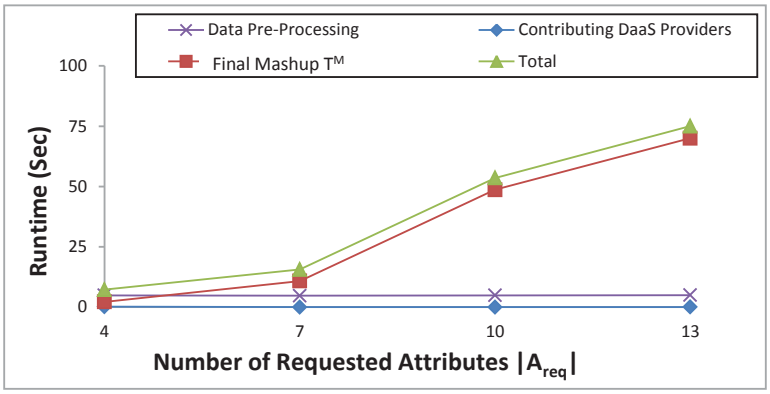

Figure 9: Efficiency $(Acc_{req} = 70, BPrice_{req} = 3000)$

Figure 9, Figure 10, and Figure 11 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 $|A_{req}|$,

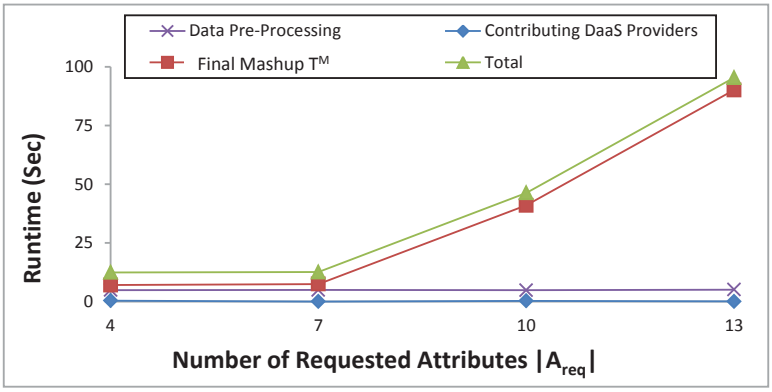

Figure 10: Efficiency ($Acc_{req} = 80, BPrice_{req} = 9000$)

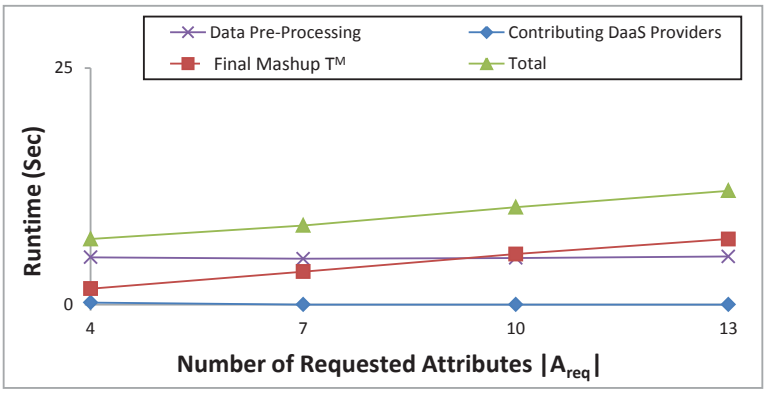

Figure 11: Efficiency ($Acc_{req} = 90, BPrice_{req} = 15000$)

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 11, the total runtime when $|A_{req}| = 13$ is 12 sec, in contrast to 75 sec in Figure 9 and 95 sec in Figure 10. This is because $Acc_{req} = 90\%$ and $BPrice_{req} = \$15,000$ 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 $BPrice_{req}$.

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 12, Figure 13, and Figure 14 illustrates the total runtime of our solution for different number of requested attributes $|A_{req}|$ by consumer.

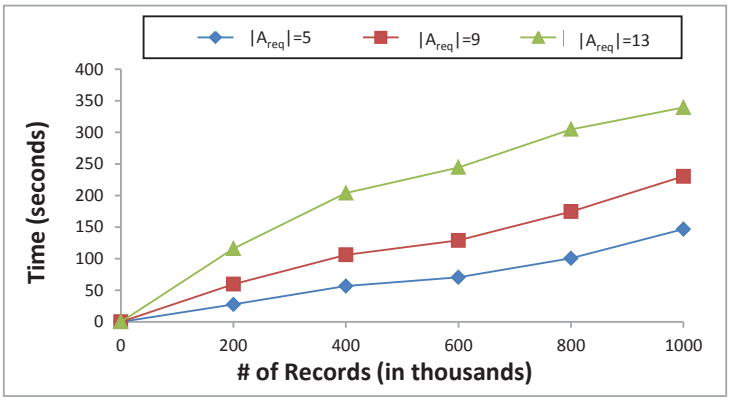

Figure 12: Scalability ($Acc_{req} = 70, BPrice_{req} = 3000$)

Figure 12 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.

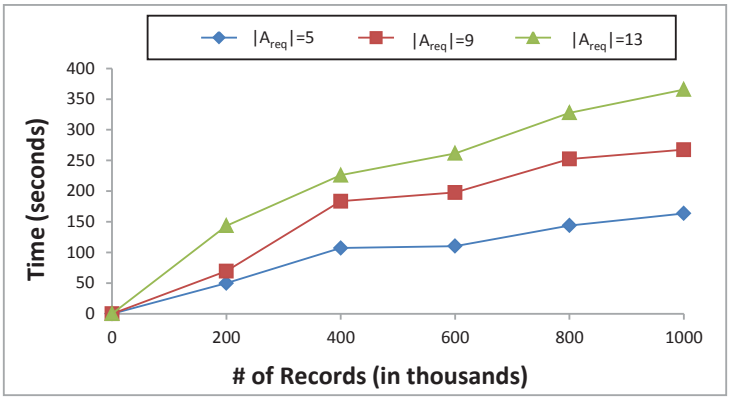

Figure 13: Scalability ($Acc_{req} = 80, BPrice_{req} = 9000$)

Figure 13 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 $|A_{req}| = 13$.

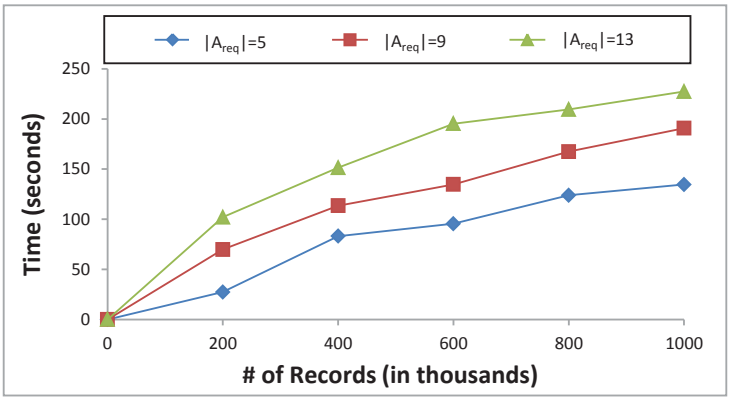

Figure 14: Scalability ($Acc_{req} = 90, BPrice_{req} = 15000$)

Figure 14 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 14 is less than the total runtime in Figure 12 and the total runtime in Figure 13 because Acc_{req} and $BPrice_{req}$ in Figure 14 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.

Chapter 6

Conclusion

In this thesis we implemented a DaaS mashup cloud-based framework for the online market and generalized the privacy and information requirements to the problem of a 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 differs from the classic 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.

In this chapter, we summarize the contributions, followed by a description of future research directions.

6.1 Summary of Contributions

First, we present a greedy algorithm 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. The proposed solution identifies the combination

of contributing DaaS providers whose data can fulfill the data privacy, data quality, and bid price requirements.

Second, we propose a solution and implement a DaaS mashup cloud-based framework to mash-up private data from distributed DaaS providers to satisfy a consumer's request, while preserving both data privacy and data mining quality of the underlying data. We also consider alternative solutions for cases where no providers can satisfy a consumer's request. In these cases, the nearest suggestions to the consumer's needs will be offered.

Finally, we conduct extensive experimental study on a real-life data set and examine the impact of employing different privacy thresholds on the revenue of each DaaS provider. We then demonstrate that our approach is efficient and scalable in terms of processing various sizes of queries with regard to data quality and bid price. Next, we show that our solution is highly scalable for large scaled data sets.

6.2 Future Work

For our future work, we identify two potential research directions:

First, in the greedy algorithm proposed in this thesis we presented a solution to output one set of contributing DaaS providers in terms of price. It implies that the identified set of DaaS providers is the cheapest set of providers whose data can satisfy data privacy and data quality as well as bid price requirements. However, this solution provides an answer which is be on behalf of the user in terms of price, but it does not identify the minimum number of DaaS providers whose data can fulfill a consumer's request. One possible direction for future work is to modify the algorithm in a way that the output is the smallest number of DaaS providers that can satisfy the query such that the total cost is minimal.

Second, in this thesis, we utilize the price per attribute values of each DaaS provider in the greedy algorithm to find a set of contributing DaaS providers whose data can satisfy the request with minimum cost. Our solution selects a DaaS provider from that set and

tries to satisfy the requested accuracy and bid price. However, our solution determines the lowest price needed to answer a request. It would be interesting to study how price and accuracy can collaborate in one greedy algorithm to propose a privacy-preserving DaaS mashup framework that can fulfill a consumer's request. This feature allows the consumer to select a better total price for the request.

Bibliography

[AES03] 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)*, pages 86–97, 2003.

[Agg05] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 901–909, 2005.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 439–450, 2000.

[BA05] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 217–228, 2005.

[BBG+11] 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)*, pages 323–330, 2011.

[BBSPA03] L. Burnett, K. Barlow-Stewart, A. Pros, and H. Aizenberg. The gene trustee: A universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine*, 10, 2003.

[BGIC06] S. S. Bhowmick, L. Gruenwald, M. Iwaihara, and S. Chatvichienchai. PRIVATE-IYE: A framework for privacy preserving data integration. In *Proceedings of the 22nd International Conference on Data Engineering Workshops*, pages 91–91, 2006.

[BHRT03] B. Benatallah, S. Hacid, C. Rey, and F. Toumani. Request rewriting-based web service discovery. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 242–257, 2003.

[BL13] K. Bache and M. Lichman. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[But06] D. Butler. Mashups mix data into global service. *Nature*, 439:6–7, 2006.

[Cha81] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[CKV+02] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4:28–34, 2002.

[Cox80] L. X. Cox. suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75:377–385, 1980.

[Dal77] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977.

[DD99] R. Domenig and K. R. Dittrich. An overview and classification of mediated query systems. *SIGMOD Record*, 28:63–72, 1999.

[Dwo06] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1–12, 2006.

[DZ02] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE International Conference on Privacy, Security and Data mining*, pages 1–8, 2002.

[EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGMOD-SIGACTSIGART Symposium on Principles of Database Systems (PODS)*, pages 211–222, 2003.

[ERS99] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, editors. *Management of heterogeneous and autonomous database systems*. Morgan Kaufmann Publishers Inc., 1999.

[FTH+12] B. C. M. Fung, T. Trojer, P. C. K. Hung, X. Li, K. Al-Hussaeni, and R. Dssouli. Service-oriented architecture for high-dimensional private data mashup. *IEEE Transactions on Services Computing*, 5:373–386, 2012.

[FWCY10] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42:1–53, 2010.

[FWY05] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 205–216, 2005.

[FWY07] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19:711–725, 2007.

[Geh06] J. Gehrke. Models and methods for privacy-preserving data analysis and publishing. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 105–105, 2006.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th annual ACM symposium on Theory of computing (STOC)*, pages 218–229, 1987.

[Gol04] O. Goldreich. Foundations of cryptography: Volume 2, basic applications, 2004.

[Hul97] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 51–61, 1997.

[Iye02] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 279–288, 2002.

[Jhi06] A. Jhingran. Enterprise information mashups: integrating information, simply. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 3–4, 2006.

[JJR02] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, 2002.

[Jos07] N. Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., 2007.

[KFS06] 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*, pages 915–922. ACM, 2006.

[KMR] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011:272.

[LDR05] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proceedings of the 31st ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 49–60, 2005.

[LDR06a] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 25–25, 2006.

[LDR06b] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 277–286, 2006.

[LP09] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 4:59–98, 2009.

[MFHL09] 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), pages 1285–1294, 2009.

[MFWH09] 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 (EDBT)*, pages 228–239, 2009.

[MG11] P. Mell and T. Grance. The nist definition of cloud computing. pages 800–145. National Institute of Standards and Technology (NIST), 2011.

[MKGV07] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1, 2007.

[MKM+07] D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. Worstcase background knowledge in privacy-preserving data publishing. In *Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE)*, pages 126–135, 2007.

[MW04] D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS)*, pages 210–219, 2004.

[PKPS02] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347, 2002.

[Qui93] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[Sam01] P. Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13:1010–1027, 2001.

[Swe02a] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10:571–588, 2002.

[Swe02b] L. Sweeney. k-anonymity: a model for protecting privacy. *Intertation Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10:557–570, 2002.

[TFH09] 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)*, pages 767–774, 2009.

[VHNS08] 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)*, pages 54–61, 2008.

[War65] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60:63–69, 1965.

[WFY05] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 466–473, 2005.

[WFY07] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *Knowledge and Information Systems (KAIS)*, 11:345–368, 2007.

[Wie93] G. Wiederhold. Intelligent integration of information. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 434–437, 1993.

[WYC04] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM)*, pages 249–256, 2004.

[XT06] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 229–240, 2006.

[Yao82] A. C. Yao. Protocols of secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164. IEEE Computer Society, 1982.

[YZW05] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *Proceedings of the 11th ACM SIGKDD Conference (SIGKDD)*, pages 334–343, 2005.

[ZSR05] Y. Zhiqiang, Z. Sheng, and N. W. Rebecca. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 5th SIAM International Conference on Data Mining*, 2005.

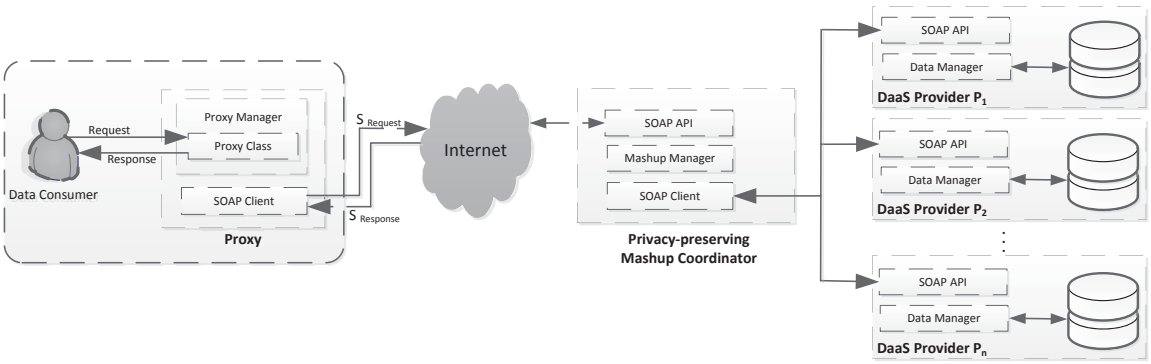

Figure 4: Framework for Privacy-Preserving Data-as-a-Service Mashups: Implementation Architecture

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.

4.2 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 [Jos07]. The implemented architecture of our framework is illustrated in Figure 4.

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

with a higher price or lower accuracy. Figure 5 illustrates the activity diagram for request satisfaction.

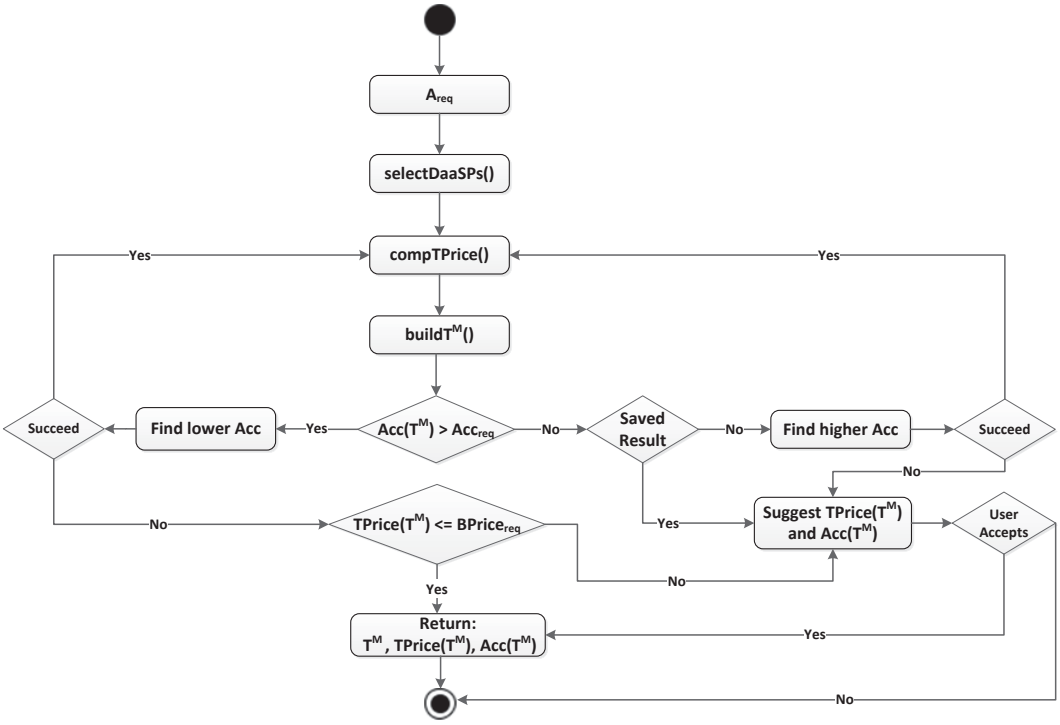

Figure 5: Data Request Satisfaction

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 A_{req}. As illustrated in Section 3.1, $Price = Acc \times PA_i$ 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 $buildT^M$ 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^P 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