Privacy Preserving Data Integration Across Autonomous Cloud Services

Samer Abdul Ghafour
Claude Bernard University (Lyon1)
69622 Villeurbanne, France
samer.abdul-ghafour@univ-lyon1.fr

Parisa Ghodous
Claude Bernard University (Lyon1)
69622 Villeurbanne, France
parisa.ghodous@univ-lyon1.fr

Christine Bonnet
Claude Bernard University (Lyon1)
69622 Villeurbanne, France
christine.bonnet@univ-lyon1.fr

***Abstract*—In this paper, we tackle privacy issues of data sharing services in a cloud environment. We propose a service-oriented privacy-preserving model for data integration across autonomous clouds. Our model allows to execute aggregations of cloud data sharing services without revealing any extra information to any of the involved services. Thus, involved services enforce locally their privacy policies by applying their own access control models and data anonymization algorithms. We illustrate lack of data protection in current data sources through examples of querying data in a healthcare system.**

***Keywords*-Privacy; Data Integration; Services; Cloud;**

I. Introduction

Data integration across autonomous data sources has been a long-standing challenge for the database community [6], [5], [4]. This is motivated by the number of contexts in which the need for a flexible data integration mechanism is critical, including Web and enterprise data integration, integrating data pieces from smart objects in IoT-based smart environments, data sharing for scientific research, data exchange in government agencies, etc.

In practice, however, the development of data integration systems is often hindered by the lack of robust and flexible techniques to protect the privacy of the shared data. Existing data integration solutions are usually implemented as centralized data warehouses collecting and storing data from various data sources. Typically, data sources and data warehouses expect to sign business agreements in which the scope of the shared data and corresponding privacy policies are specified. For example, all shared data will be kept confidential and will not be disclosed to other unrelated third parties or be used for other purposes. While this solution works well for a single organization or a federation of organizations, where trust relations have been well established, serious problems will arise when some data warehouses cannot be trusted by data sources. In such cases, data sources will refuse to share their data because they have no control of its usages and disclosures once the data is shared. In fact, data warehouses indeed can reveal or abuse the shared data. Furthermore, even if data warehouses adhere to the agreement, there is no guarantee that they have sufficient capability to protect the data.

In this paper, we propose a service-oriented privacy-preserving model for data integration across autonomous clouds. Our model allows to execute aggregations (*i.e.*, compositions) of data sharing services held by autonomous clouds without revealing any extra information to any of the involved services (*i.e.*, none of involved services (and their providers) should be able to learn/infer any information about the data the other services provide beyond what these services already know).

Table I
Available Data Services

Service	**Semantics**
$S_1(\$city, ?ssn)$	*Returns the SSN of HIV patients in a given city.*
$S_2(\$ssn, ?description)$	*Returns a description of the psychiatric disorder of a given patient if she/he has any.*
$S_3(\$ssn, ?age, ?sex)$	*Returns the age and sex of a given patient.*
$S_4(\$ssn, \$type, ?medication)$	*Returns the medications of a given type taken by a given patient*
$S_5(\$medication, \$ingredient, ?quantity)$	*Returns the quantity of a given ingredient in a given medication.*

A. Running Example

Let us consider a healthcare information system which collaborates with multiple healthcare organizations including medical research institutes, hospitals, pharmacies, pharmaceutical labs, that may manage their data through autonomous clouds. For the sake of simplicity, we assume that the system has access to the data sharing services in Table-1. Assume also that a pharmaceutical researcher, *Alice*, wants to investigate the correlation between a chemical component ABC present in HIV medicines and the development of severe psychiatric disorders at HIV female patients.

Obviously, *Alice* can answer her research questions by composing these services as follows (*cf.* Fig. 1). She invokes S_1 with the desired city to get the identifiers of HIV patients. Then for each obtained *ssn*, she verifies whether the patient has psychiatric disorders by invoking S_2; then for each of these patients she retrieves the age and sex by invoking S_3 and the HIV medications by invoking S_4. For each of the obtained HIV medications, she retrieves its ABC content by invoking S_5. Then she joins the outputs of S_3 and S_5 to link the medical and the personal information to the same patient.

This example showcases an interesting challenge. If the data returned by individual services were completely privacy-sanitized (by removing identifiers, *e.g.*, ssn, and

2159-6190/15 $31.00 © 2015 IEEE
DOI 10.1109/CLOUD.2015.160

The data accessed by S_1

city	ssn
villeurbanne	P_{15}
villeurbanne	P_{201}
villeurbanne	P_{512}

The data accessed by S_2

ssn	description
P_3	...
P_{11}	...
P_{15}	...
P_{16}	...
P_{21}	...
P_{32}	...
P_{201}	...
P_{199}	...
P_{300}	...
P_{510}	...
P_{512}	...
P_{571}	...
P_{575}	...

The data accessed by S_3

	ssn	age	sex
	P_0	[0-10]	m
	P_8	[0-10]	m
t_1	P_{15}	[0-20]	null
	P_{20}	[0-20]	f
	P_{23}	[0-20]	f
	P_{188}	[0-20]	m
t_2	P_{201}	[10-15]	null
	P_{204}	[10-15]	m
	P_{209}	[10-15]	m
	P_{411}	[0-10]	m
t_3	P_{512}	[15-20]	f
	P_{513}	[0-30]	f
	P_{514}	[10-15]	f

The data accessed by S_4

ssn	medicine	type
P_5	dox	cardiac
P_{15}	alpha1	HIV
P_{17}	dox	cardiac
P_{227}	drab	cardiac
P_{201}	alpha 2	HIV
P_{242}	drab	cardiac
P_{411}	dox	cardiac
P_{512}	alpha 3	HIV
P_{711}	dox	cardiac

The data accessed by S_5

	medicine	quantity
l_1	alpha1	[0-30] mg
l_2	alpha2	5 mg
l_3	alpha3	[10-100]mg

Figure 2. Sample of the data accessed by the data services

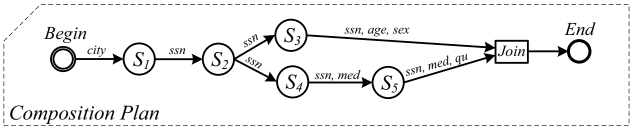

Figure 1. The Composition Plan

anonymizing sensible information) then the query plan could not be executed, and the research query of Alice could not be answered. On the other hand, if returned data were not protected, then participant services, the entity executing the query and the query issuer (*i.e. Alice*) will learn sensitive information that they normally must not know. For instance, if the provider of S_2 has an access to the query plan, and knows that its input tuples are coming form S_1, and assuming that the data accessed by our services are given in Fig. 2, then he will learn who of his patients have been tested positive for HIV (i.e, P_{15}, P_{201} and P_{512}). Similarly, the providers of S_3 and S_4 will learn who of their patients are receiving treatments for psychiatric disorders and are HIV patients. *Alice* and the entity responsible of executing the query will learn sensitive information about patients including their ssn, ages, medications, etc. Based on that observation, the goal of this paper is to enable the services involved in answering a query (such as the one formulated by *Alice*) to enforce locally their privacy policies (by applying their own access control models and data anonymization algorithms) while at the same time keeping it possible to link data subjects[1] (*e.g.*, patients) across the different services. In this paper, we propose a privacy-preserving query execution model to execute multi-source queries over autonomous data sharing services (managed by different clouds). Our model ensures that (i) no extra information is revealed to any of the involved services, *i.e.*, none of involved services is able to learn/infer any information about the data the other services provide beyond what these services already know, and (ii) the entity that executes the query and aggregates intermediate results has only the necessary information to interconnect data subjects.

II. A Privacy-Preserving Composition Execution Model for Honest-but-curious Data Sharing Services

A. Context and Assumptions

We consider a honest-but-curious environment. An honest-but-curious environment (a.k.a. semi-honest environment [2]) is one where the parties involved in the query processing (*i.e.*, composed data services and the composition execution engine) follow correctly the given protocol, but may keep any result or information they obtain during the course of the protocol. We assume that the services, the composition execution engine and the recipient of the final results are three independent entities.

We consider that the attributes of a dataset can be divided into: *identifier attributes* and *non-identifier attributes*. The integration of the data subjects across the different data services is carried out using the identifier attributes. We assume the existence of universal identifiers in each application domain (*e.g.*, the social security number in the healthcare domain).

B. Preliminaries

A Composition of Data Services $\mathcal{H}$: A composition of *n* data services is represented as a directed acyclic graph (DAG) $\mathcal{H}$ in which there is a node corresponding to each data service, and there is a directed edge e_{ij} from S_i to S_j if there is a precedence constraint $S_i \prec S_j$ (*i.e.*, S_j is preceded by S_i when one of its inputs is an output of S_i), and where each service S_i 1≤ i≤ n has a set of inputs and outputs that could be privacy-sensitive or identifier attributes. Edges e_{ij} may be associated with constraints to filter relayed tuples.

Service Selectivity $Se(S_i, R_j)$: Given a data service S_i, and a range of input values R_j, the selectivity of S_i relative to R_j

[1] We use the term data subject to mean the individual whose private information is stored and managed by data services

is the number of outputted tuples when S_i is invoked with R_j. For example, assuming that the ssn values in Fig. 2 are ordered, then $Se(S_3, [P_0, P_{10}]) = 2$, $Se(S_3, [P_5, P_{20}]) = 3$, and $Se(S_3, [P_0, P_{1000}]) = 13$ are the selectivities of S_3 relative to the ranges $[P_0, P_{10}]$, $[P_5, P_{20}]$ and $[P_0, P_{1000}]$. We assume that data services can provide operations (*i.e.*, functions) to provide statistical information about their managed data (including the selectivity of a service.).

Order Preserving Encryption Scheme OPES. An OPES [1] allows to encrypt numeric data values while preserving the order relation between them. This allows to apply equality and range queries as well as the MAX, MIN and COUNT queries on encrypted data, without decrypting the operands.

C. Privacy-preserving Composition - Execution Model

Our model relies on two key ideas. First, we use a combination of OPES for identifier attributes and anonymization techniques for non-identifier attributes. Composed services could apply the desired anonymization algorithms on non-identifier attributes, but they must all use the same OPES for identifier attributes. This way the composition execution engine has only access to anonymized data and can link the anonymized information of the same data subject across the different services using the encrypted identifier attributes (recall that the OPES allows for applying equality queries on encrypted data). It cannot decrypt the encrypted identifier attribute values, as it does not have the encryption key. By the end of the composition's execution, it removes from the final results the encrypted identifier attributes before returning them to the recipient, who will thus get only the anonymized data.

Second, our model implements the *K-protection* notion that we introduce below, and which limits the knowledge leaked to participant services during the execution of $\mathcal{H}$.

***K*-protection**: *Given a vector $K = (k_1, k_2, ..., k_n)$, where k_i is an integer representing the protection degree the service S_i must provide for its outputted tuples. For each edge e_{ij} in $\mathcal{H}$, the knowledge leaked to S_j during the execution of $\mathcal{H}$ (denoted by $\Re(S_j)$) must be $\leq min(1/k_l)$, where k_l is associated with S_l, which denotes the (direct or indirect) parents of S_j in $\mathcal{H}$. Note that S_j has at least one parent in $\mathcal{H}$, which is S_i.*

The above definition can be interpreted as follows: when a service S_j is invoked, it must not be able to determine precisely its input value between k input values for which it has outputs; *i.e.*, it must not be able to determine precisely the tuple t in which the invoker is interested between k tuples of its own data. This can be realized by invoking S_j by a range of values R instead of a precise value v, where $Se(S_j, R) = K$.

Example: Examples of privacy breaches that could happen if the composition in Fig. 1 was executed without ensuring the k-protection requirement include: S_2 will know that its patients P_{15}, P_{201} and P_{512} have HIV virus; S_3 will know that these same patients have HIV and suffer from severe psychiatric disorders, etc. Now, assume that $K_1 = 3$, the k-protection requirement implies that S_2 must not be able to distinguish each of its input values (*e.g.*, P_{15}) from at least 3 values for which it has answers. Fig. 3 shows how the k-protection is enforced on the edge e_{12}. The value P_{15} is *k-generalized* into a range of values V which has at least three values (*e.g.*, P_{11}, P_{15} and P_{16}) for which S_2 has corresponding tuples. After the invocation of S_2, the extraneous tuples are filtered out.

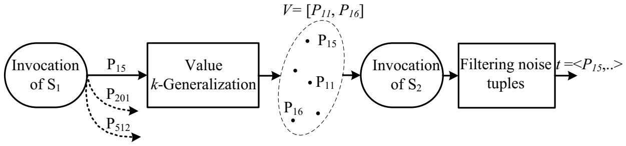

Figure 3. Ensuring the k-protection on the edge e_{12} (k=3)

The Model Description: As illustrated in Fig. 4, the system is composed of two main modules: the *Service Composition Module* which generates the composition execution plan and the *Composition Execution Module* which executes the composition in a privacy-preserving manner. The recipient specifies an encryption key, submits it directly to participant services in $\mathcal{H}$, and launches the execution of $\mathcal{H}$. When participant services are invoked, they anonymize their sensitive data and encrypt the identifiers with the supplied key. The composition execution engine implements (in the *Value K-Generalization module*) an algorithm to ensure the k-protection requirement when it invokes participant services. Specifically, for each invoked service S_i, it determines the protection factor k that must be ensured: k = MAX($S_j.k_j$), where S_j denotes the parents of S_i in $\mathcal{H}$. Then, for each input tuple t, the algorithm determines the minimum range of value $R[a, b]$ that should be used to invoke S_i instead of t. For this purpose the execution engine requests the selectivity of S_i with respect to a wide range of identifier values R (we use the range]-∞, +∞[to denote the range covering the whole tuples set managed by S_i) along with a value v occurring in the middle of the ordered value sets held by S_i. Then if the returned selectivity is greater than k, the execution engine compares the identifier attribute (denoted by x) of t to v to determine the half of R covering t, which becomes the new range R. This step is repeated with the new R until there is no R with a selectivity greater than k. Then, S_i is invoked[2] with the obtained range, and the execution engine retains only the output related to t.

Example: Fig. 5 shows how the k-protection requirement is enforced on the edge e_{23}. Assume that S_1 and S_2 require a protection factor k = 3. The invocation of S_2 returns the tuples corresponding to c_{15}, c_{201} and c_{512} (*i.e.*, the

[2] We assume that data services provide different operations to query the underlying data sets by precise values or by ranges of values.

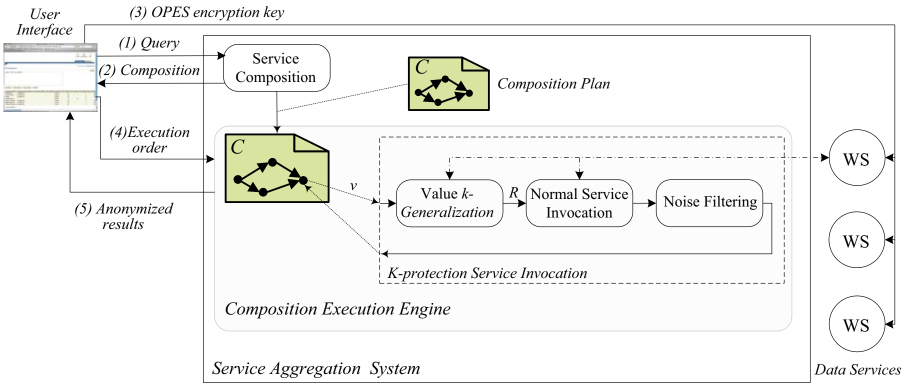

Figure 4. Implementation

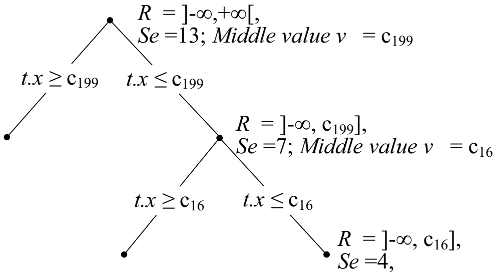

Figure 5. Finding the minimum range for invoking S_3

encrypted values of P_{15}, P_{201} and P_{512}). Instead of invoking S_3 directly with the tuple c_{15}, the execution engine *k-generalizes* c_{15} as follows. It requests the selectivity of S_3 with respect to R =]-∞, +∞[; S_3 acknowledges it has 13 distinct values and that the value (v = c_{199}) occurs in the middle of these ordered values set. The execution engine compares c_{15} to c_{199}, and determines the new range R=]-∞, c_{199}]. It then requests the selectivity of the new R along with the new v; the new values of Se and v are 7 and c_{16}. It determines again the new range by comparing c_{15} to c_{16}. The new range is R=]-∞, c_{16}] and its selectivity is 4. The algorithm stops here as if the new range was divided then *Se* will be less than *k*.

III. Implementation and Preliminary Evaluation

We implemented the system in Java and evaluated its performance based on a set of 40 medical data sharing services that we built in our previous work [3]. These services access synthetic medical information of more than 30.000 patients.

We conducted a preliminary evaluation of our techniques using the query in the running example. The query was executed 1000 times with and without privacy preservation. Evaluation showed that the execution average time with privacy is at most three orders of magnitude of time without privacy (with K_i set to 4). Furthermore,we were able to reduce time to two orders of magnitude by *reusing* the selectivities and the ranges computed in past invocations of the same services.

IV. Conclusion

This paper addresses privacy issues with aggregated services in a cloud environment. We have considered an example in the healthcare system to illustrate privacy issues when participants may access to sensitive information that they normally must not know. Our proposed solution enforces privacy policies for services involved in answering a query by applying their own access control models and data anonymization algorithms. We are currently working on extending our model to make it multi-dimensional, *i.e.*, when the identifier of a data subject includes the combination of multiple attributes.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD Conf.*, pages 563–574, 2004.

[2] F. Emekçi, D. Agrawal, A. E. Abbadi, and A. Gulbeden. Privacy preserving query processing using third parties. In *ICDE*, page 27, 2006.

[3] S. A. Ghafour and P. Ghodous. On-demand data integration on the cloud. In *2014 IEEE 7th International Conference on Cloud Computing, Anchorage, AK, USA, June 27 - July 2, 2014*, pages 924–927, 2014.

[4] M. Kuzu, M. Kantarcioglu, A. Inan, E. Bertino, E. Durham, and B. Malin. Efficient privacy-aware record integration. In *Joint 2013 EDBT/ICDT Conf., EDBT '13 Proceedings, Genoa, Italy*, pages 167–178, 2013.

[5] G. Navarro-Arribas and V. Torra, editors. *Advanced Research in Data Privacy*, volume 567 of *Studies in Computational Intelligence*. Springer, 2015.

[6] J. Weis and J. Alves-Foss. Securing database as a service: Issues and compromises. *IEEE Security & Privacy*, 9(6):49–55, 2011.