PAIRSE: A Privacy-Preserving Service-Oriented Data Integration System

Djamal Benslimane
Claude Bernard University
69622 Villeurbanne, France
djamal.benslimane@univ-lyon1.fr

Mahmoud Barhamgi
Claude Bernard University
69622 Villeurbanne, France
mahmoud.barhamgi@univ-lyon1.fr

Frederic Cuppens
Telecom-Bretagne
2 Rue de la Chataigneraie
35576 Cesson Sevigne, France
frederic.cuppens@telecom-bretagne.eu

Franck Morvan
IRIT, Paul Sabatier University
118 Route de Narbonne, 31062
Toulouse, France
franck.morvan@irit.fr

Bruno Defude
Institut TELECOM
CNRS UMR Samovar
Evry, France
bruno.defude@it-sudparis.eu

Ebrahim Nageba
Claude Bernard University
69622 Villeurbanne, France
ebrahim.nageba@univ-lyon1.fr

ABSTRACT

Preserving data privacy is among the key challenges that still hamper answering business data integration needs in many sectors, including healthcare, e-commerce, and e-government. The PAIRSE project aims at providing a flexible, loosely-coupled and privacy-preserving data integration mechanism in P2P data integration environments. The project exploits recent Web standards and technologies such as Web services and ontologies to export data from autonomous data providers as reusable services, and proposes the use of service composition as a viable solution to answer data integration needs on the fly. The project proposed new composition algorithms and service/composition execution models that preserve privacy of the manipulated data and prevent the different actors (e.g., data consumers, service providers, etc.) from learning/inferring any extra information, beyond what is permitted. The proposed integration system was demonstrated at EDBT 2013 and VLDB 2011.

1. INTRODUCTION

Data integration has been a long-standing challenge for the database community. This is motivated by the number of contexts in which the need for a flexible data integration mechanism has become critical, including Web and enterprise data integration, data sharing for scientific research, data exchange in government agencies, building eCommerce marketplaces, etc.

Much of the literature on data integration across autonomous data sources has tacitly assumed that data on the side of each data source can be revealed and shared with other sources. In practice, however, data integration scenarios in many contexts and application domains are often hampered by legitimate and widespread data privacy concerns. In the healthcare application domain for example, medical data are subject to many legislations and privacy protection laws (e.g., [2, 1]) around the world that restrict collection, processing, and disclosure of personal data, and hold data holders accountable for any unintended data disclosure or misuse.

The PAIRSE project addresses the challenge of flexible and privacy-preserving data integration in peer-to-peer data sharing environments. Driven by the recent trends of using SOA-oriented architectures for data integration in modern enterprises, PAIRSE assumes that data sources are exposed to the data sharing environment as Web services. This type of services is commonly known as data services [13] (a.k.a. Data as a Service [16]), where data services provide a well-documented, platform (and source) independent, interoperable and uniform method of interacting with data. PAIRSE proposes a service composition-based approach for on-demand data integration; i.e., heterogeneous data services from autonomous service providers are selected and composed on the fly to answer users' queries. Data privacy preservation is a key objective of PAIRSE. Users in PAIRSE are allowed only to access the information they are entitled to for a given purpose. PAIRSE focuses on modeling, discovering, selecting and composing data services to efficiently answer

users' queries. Specifically, the project addresses the following key challenges:

- ***Query resolution by automatic service composition***. In PAIRSE queries are resolved by *automatically* selecting and composing relevant data services from the P2P data integration environment. This involves handling different challenging issues. First, the semantics of data services should be explicitly represented to automate their discovery, selection and composition. Second, a P2P service discovery mechanism is needed to efficiently locate relevant data services on the P2P network based on their semantics. Third, relevant services that can be combined to answer the query need to be selected and orchestrated.

- ***Privacy preservation***. Privacy of data subjects whose data are manipulated by services is among the key barriers to data service composition. Two particular issues need to be addressed. First, as compositions may generally involve autonomous data services (i.e., services provided by autonomous service providers), services involved in a composition should not be able to learn or infer any information about the data that the other services provide, beyond what is permitted. Second, service providers should be allowed to enforce locally their privacy and security policies without changing the implementation of their services.

The contributions of our PAIRSE data integration system, which was demonstrated at EDBT 2013 [7] and VLDB 2011 [11], are summarized as follows:

- ***Semantic description model for data services***. We propose an ontology-based description for data services [10]. Services are modeled as "*RDF Views*" over domain ontologies. An RDF view captures the semantics of a data service by defining the semantic relationship between the service's input and output parameters in a "declarative" way using "concepts" and "relations" whose semantics are formally defined in domain ontologies. The service description files (e.g., WSDLs for SOAP services, HTML for RESTful services, etc.) are annotated with the defined views.

- ***Query resolution by service composition***. We proposed a novel service composition algorithm [10]. The algorithm exploits the mature query rewriting techniques to relieve users from having to manually select and compose services, a task that would generally require important programming skills. Users need only to formulate their composition queries over domain ontologies using the de facto ontology query language SPARQL[1]. The algorithm will then select and compose the services based on the proposed semantic modeling (i.e., RDF views). We proposed also an efficient algorithm to locate the services that are relevant to a given query in a P2P environment [19].

- ***Privacy preservation***. We proposed a privacy preserving composition execution model [7, 21, 8]. Our model is twofold. First, it allows services providers (whose services are involved in a composition) to enforce locally the privacy and security policies that may be associated with their provided data without changing the implementation of their services. Second, our model provides means to execute the composition without revealing extra information to any of the involved services; i.e., none of involved services (and their providers) is able to learn/infer any information about the data the other services provide beyond what is permitted.

The rest of the paper is organized as follows. Section 2 gives an overview of our service-based integration system. Section 3 describes our semantic modeling of data services. Section 4 presents our composition approach and describes its originalities. Section 5 presents our techniques to ensure privacy protection. Section 6 applies our work in two application domains, and summarizes our conducted evaluations and obtained results. Section 7 concludes the paper.

2. PAIRSE'S ARCHITECTURE

The PAIRSE data integration system has a hybrid peer-to-peer infrastructure (Figure 1)[19], where peers form communities of interest, called *Virtual Organizations VOs*. Each VO has a common domain ontology modeling its expertise, and peer members that may have relations with members from other VOs. Relations between peers exist only if there is a mapping between the ontologies modeling the expertise of their respective VOs. PAIRSE does not impose any constraint on the topology graph formed by the ontologies and the different mappings. Peers export their (sharable) data sources as data services. The adoption of a service-oriented view of data has many advantages, including providing an abstraction layer between data consumers

[1]http://www.w3.org/TR/rdf-sparql-query/

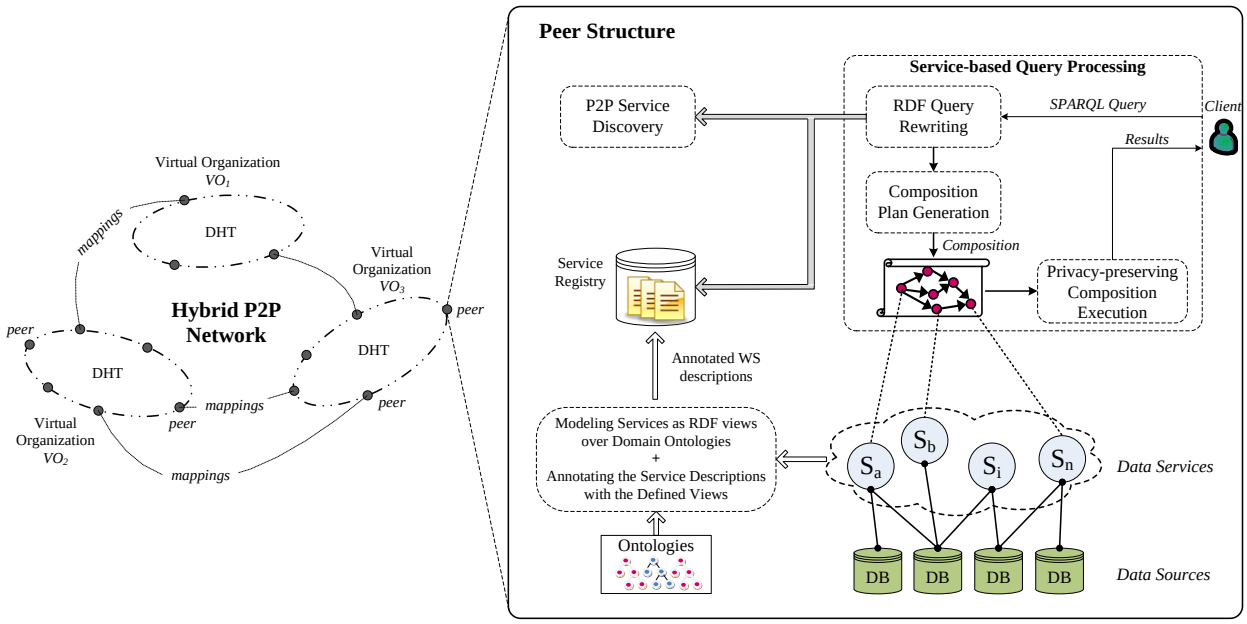

Figure 1: An overview of the proposed data integration system

and the peer's heterogeneous data sources, and preparing the ground for a *flexible* and *loosely-coupled* data integration by service composition [13]; i.e., data services from different peers can be quickly composed to answer evolving data business needs.

To automate the discovery and the composition of data services, we model them as *RDF Views* over domain ontologies. RDF views capture in a faithful and declarative way the semantic relationships between input and output parameters using ontological concepts and relations whose semantics are well defined in domain ontologies. RDF views are incorporated into service description files as annotations.

The PAIRSE system follows a declarative approach to compose data services. Specifically, users pose their queries on domain ontologies using SPARQL query language. Then, our system exploits the defined RDF views in service description files to select the services that can be combined to answer the query using an RDF query rewriting algorithm that we have devised for that purpose. Then, it generates an execution plan for the composition and executes it to provide the user with the requested data. As data services may manipulate privacy-sensitive information, PAIRSE proposed new service and composition execution models to preserve privacy (see section 5).

Queries may necessitate the use of remote data services, in which case an efficient P2P service discovery algorithm [19] is used to locate and retrieve the descriptions of relevant services from remote peers.

3. SEMANTIC MODELING FOR DATA SERVICES

Modeling and explicitly specifying the semantics of data services are the first step towards the automation of service selection and composition. Different semantic Web service description languages and models have been proposed in the literature, including OWL-S[2], WSMO[3], SAWSDL[4]. However, these languages were designed with the conventional services[5] in mind, where services are seen as "*actions*" that can be characterized by their *inputs*, *outputs*, *preconditions* and *effects* (a.k.a. IOPEs). Their focus was, therefore, on semantically modeling IOPEs. Unfortunately, this action-oriented modeling is insufficient for data services, where the focus should be on representing the semantic relationship between inputs and outputs. This is crucial, as different data services may have the same input and output sets, but completely different semantics.

In PAIRSE, we proposed in [10] to model data services as *RDF Parameterized Views* (RPVs) over domain ontologies. A parameterized RDF view uses

[2]http://www.w3.org/Submission/OWL-S/
[3]http://www.w3.org/Submission/WSMO/
[4]http://www.w3.org/2002/ws/sawsdl/
[5]Services encapsulating software artifacts (i.e. software-as-a-service services)

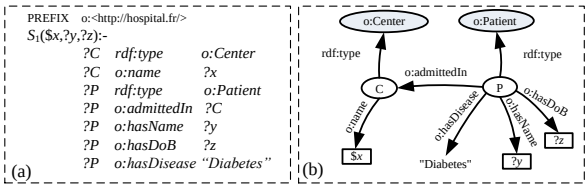

Figure 2: (a) a parameterized RDF view; (b) its graphical representation

concepts and *relations* whose meanings are formally defined in domain ontologies to define the semantic relationships between input and output parameters sets of a data service. A parameterized view is a technique that has been used to describe content and access methods in Global-as-View (GaV) integration architectures [18]. It has been also used to model privacy constraints in [23]. A parameterized view requires a particular set of inputs (the parameter values) in order to retrieve a particular set of outputs; i.e., outputs cannot be retrieved unless inputs are bound. Figure 2 shows an RPV of a service returning the personal information (i.e., name and dates of birth) of patients admitted in a given medical center. Note that input parameters are prefixed with the symbol "$" and output parameters are prefixed with the symbol "?".

RDF views may also specify constraints to characterize the data manipulated by their corresponding services. These constraints may have different forms, including *simple interval constraints* (e.g., $X \in [a, b]$, where X is a variable used in an RDF view), and *fuzzy constraints* interpreted according to a fuzzy membership function (e.g., the medications returned by a service have "$High$" concentration of hydroxypropy1-β-cyclodextrin; i.e., X *is* $High$, where the fuzzy term "$High$" is interpreted by a membership function specifying for each value of the concentration parameter the degree to which it is high).

We adopted an approach similar to SAWSDL to associate data services with their RPVs. We exploited the extensibility feature of the WSDL standard to annotate the WSDL files with RPVs.

4. QUERY RESOLUTION IN PAIRSE

In PAIRSE, users' queries are resolved by composing relevant data services on the fly. Each virtual organization in PAIRSE's hybrid P2P architecture has a DHT (Distributed Hash Table) to index its published services [19]. Services are indexed according to the ontological concepts used in their RPVs. When a query is issued at a given peer, the peer extracts the different ontological concepts used in the query and launches service discovery requests for services annotated (via their RPVs) with these concepts. Services are first sought in the same VO where the query is posed, then the service discovery request is propagated to connected VOs. The descriptions of discovered services are then sent back to the initial peer, where the relevant services will be selected and composed. Furthermore, for each discovered service we return the mapping path between the ontologies associated with the expertise domains (i.e., VOs) of the discovered service and the initial peer. This mapping path allows the translation of RPV views.

We proposed a *Query Rewriting* based service composition algorithm to select and compose data services on the fly [10, 11, 9]. The algorithm, given a SPARQL query formulated over a domain ontology, and a set of data services represented by their RPVs, rewrites the query in terms of calls to relevant services.

The problem of query rewriting using views was well studied and formalized in the last decade ([18] is a good survey on the topic). Different rewriting algorithms (e.g., *Bucket*, *Inverse Rules*, *Minicon*, etc.) were proposed and used in different integration systems (e.g., *InfoMaster*, *Manifold*, *Ariadne*, etc.). Our algorithm extends the earlier works on query rewriting and data integration in the following aspects:

Compliance with the RDF/S data models: while most of previous work has focused on relational [22, 18] and XML data integration [26], we considered the case of RDF/RDFS-based data integration. Specifically, we designed and implemented an RDF/RDFS query rewriting algorithm that takes into account the RDF schema constraints such as *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain*, and *rdfs:range* when comparing RPVs to a user's SPARQL query. It rewrites the SPARQL query into a set of service calls. Evaluating the union of these calls has essentially the same effect as running the query on a target RDF instance. The consideration of RDFS constraints is of great importance in Web data integration. For example, suppose there is a statement in an RDFS domain ontology specifying that *:Medication rdfs:subClassOf :Drug*. Given a data service S returning the medications administered to a given patient, and a query Q for the drugs administered to a given patient, our rewriting algorithm automatically infers that S can be used to generate rewritings for Q. In other words, the consideration of the RDFS constraints allows our system to infer more results than the previous rewriting techniques.

Answering parameterized queries: the key focus of previous data integration systems has been on answering specific user queries. In contrast, PAIRSE focused on answering *parameterized queries*; i.e. the focus is on constructing compositions of services (i.e., parameterized integration plans) that are independent of a particular input value (i.e., they respond to a range of input values). One implication of considering parameterized queries is that component services cannot be chosen at the composition time. The selection of those services depends on the actual values of input parameters, which are only known at the execution time.

Example: assume a parameterized query $Q(\$x, ?y)$ for the medications y that may interact with a given medication x. Assume also two data services:
S_1(\$x,?y), where $x \in$ [1, 5] and $y \in$ [100,150],
S_2(\$x,?y), where $x \in$ [6,10] and $y \in$ [150,200]
If Q was a specific query ($Q_{x=2}$), then S_2 would not be considered in the rewriting (i.e., composition) as $x = 2$ is not covered by S_2. In contrast, both of S_1 and S_2 are usable for Q, to cover as much as possible of the potential values of x. Furthermore, the user may customize the parameterized query by specifying ranges on its different parameters and variables. For example, by specifying that the final composition corresponding to Q will be invoked with values $x \in$ [1, 10], and by being only interested in values of $y \in$ [100, 200].

In this context, an important issue is to determine the minimum number of similar data services that satisfy together the range constraints of a subgraph of the query. This assumes that, in the general case, a query may contain different sub-graphs, where each one could be matched to different similar services with different ranges of input and output parameters. Our rewriting algorithm extends the previous work on data integration with a probabilistic subsumption test allowing to compute for each sub-graph of the query, the minimum number of services in polynomial time [9]. Note that this problem is known to be NP-Complete [24], as services may have in the general case N constrained input and output parameters (not only 2 in the example.). The optimization of the generated composition plan is another important issue involved in handling parameterized queries. Our composition algorithm analyzes the service descriptions in order to automatically insert filtering conditions in the composition plans that result in fewer requests to component services.

Inclusion of user's preferences: often the number of candidate compositions that may be used to answer the same query is very large. We proposed in PAIRSE a novel approach [11] to compute the top-k data service compositions based on user preferences. In our approach, we modeled user's preferences using fuzzy sets and incorporated them into the composition query. We match the (fuzzy) constraints of the relevant services (selected by our RDF rewriting algorithm) to those of the query and determine their matching degrees using a set of matching methods from the fuzzy set theory. We then rank-order the candidate services based on a fuzzification of *Pareto dominance* and compute the top-k Data service compositions. In addition, we introduce a new method to increase the diversity of returned top-k compositions while maintaining as much as possible the compositions with the highest scores.

5. PRIVACY PRESERVATION IN PAIRSE

In this section, we present briefly our models to (i) allow service providers to ensure the security and privacy of data returned by their services, and (ii) to execute a composition without revealing extra information to any of the involved services about the data that each other holds.

5.1 Privacy-preserving Service Execution Model

Data returned by a data service may be subject to different security and privacy concerns. For example, different people may have different access rights over the same data item; data subjects[6] may have different preferences about the disclosure of their data, etc. A common approach in the database field to handle such concerns is to push them to the underlying DBMS by rewriting the query to include these constraints [20]. However, this may not be applicable to data services as the same service may access a multitude of heterogeneous data sources that may not be necessarily managed by a DBMS (e.g., XML files, flat files, silos of legacy applications, etc.). An alternative approach is to enforce privacy and security policies at the application level [6], by modifying, in our case, the source code of data services. However, this may not always be applicable nor advisable as most of current data service creation platforms (e.g., *AquaLogic* [14]) provide data services as *black boxes* that cannot be modified; i.e., their internal data integration procedures and logics are not accessible. Even if the code was modifiable, this solution often leads

[6]We use the term *data subject* to mean the individual whose private information is manipulated by data services.

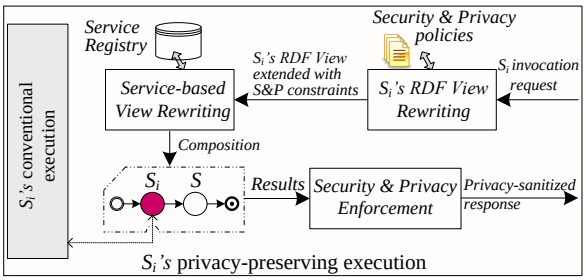

Figure 3: A Privacy-preserving Service Execution Process

to privacy leaks [20], as the dropped programming code may contain flaws; i.e., its correctness is hard to be proven (especially for complex queries), compared to declarative rewritten queries in the first approach.

We proposed a secure, privacy-preserving execution model for data services allowing service providers to enforce their privacy and security policies without changing the implementation of their data services (i.e., data services are considered as black boxes). Our model takes its inspiration from the database approach to "*declaratively*" handle the security and privacy concerns. It involves the following steps (refer to Figure 3):

Step 1: *View rewriting to integrate the security and privacy constraints.* When a data service is invoked, our model rewrites its corresponding RDF view to take into account applicable security and privacy rules from the service's associated policies, which are expressed using the OrBAC and PrivOrBAC models [3, 5] over domain ontologies and take into account the data *recipient* (i.e., the service consumer), his *purpose* for requesting the data, and the *consents* of data subjects. Security and privacy rules may contain conditions to specify fine-grained access control constraints (e.g., constraints on working hours, time, locations, etc.), or to query the data subjects' consents. These conditions have the form of SPARQL expressions over domain ontologies, which eases their integration to the view. The soundness and the correctness features of our rewriting algorithm are demonstrated in our work [21, 8].

Step 2: *Rewriting the extended view in terms of data services.* The extended RDF view $v_{extended}$ may include additional data items (denoted by $\Delta v = v_{extended} - v_{original}$) required to enforce security and privacy constraints. These data items may not be necessary covered by the initial service. In this step, we find the data services covering Δv to prepare for the enforcement of privacy and security conditions (in a later step), and rewrites $v_{extended}$ in terms of these services along with the initial service. Our rewriting algorithm is detailed and evaluated in [10].

Step 3: *Enforcing security and privacy constraints.* The services selected in the previous step are orchestrated into a composition plan and executed using the conventional service execution process. The composition returns (i) the data items returned by the invoked service along with (ii) the data items necessary to evaluate the security and privacy constraints. We defined a privacy filter that evaluates the privacy constraints of the different items that are subject to privacy constraints in the view. *Null* values will be returned for items whose privacy constraints evaluate to *False*.

We demonstrated the validity of our model by extending the architecture of the famous service container AXIS[7] 2.0 with a new module implementing our privacy-preserving service execution model.

5.2 Privacy-preserving Composition Execution Model

Executing compositions may disclose confidential information to component services. Assume, for example, a composition of two services: S_1 returns HIV patients in a given city, and S_2 checks whether a given patient has been treated for psychiatric disorders (S_1 and S_2 are provided by different healthcare providers). Such composition could be needed (by a pharmaceutical researcher) to investigate the connection between a chemical component present in HIV medicines and the development of severe psychiatric disorders at HIV patients. Assume also *Bob* is a common patient for both S_1 and S_2. If S_2 is invoked with Bob's identifier, and the provider of S_2 has an access to the composition plan (i.e., he knows that Bob was outputted by S_1), then he will infer that Bob is an HIV patient. On the other hand, if the data returned by S_1 were completely privacy-sanitized (e.g., by removing identifiers and sensitive information), then the composition could not be executed.

We proposed a privacy-preserving composition execution model [7] implementing the following interesting property (which we called the *k-protection property*): *the knowledge leaked to any data service S_j about the data held by another service S_i in the composition (denoted by $\Re_{S_j}(S_i)$) must be lower than or equal to $1/k_i$, where k_i is defined by S_i.*

Our model assumes a honest-but-curious environment [17] and distinguishes between the following entities: (i) the services involved in the composition, (ii) the execution engine, and (iii) the recipi-

[7]http://axis.apache.org/axis2/java/core/

ent of the final results. It relies on two key ideas:
First, data services use the same order-preserving encryption scheme OPES [4] to encrypt the identifier attributes that are needed to connect the information of the same data subject across the different services. They are still free to protect non-identifier attributes with their own techniques (e.g., anonymization, suppression, etc.). This way the execution engine has only access to protected data and can still link data subjects across the different services using the encrypted identifier attributes (note that 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, which is held only by services. By the end of the composition execution, it removes from the final results the encrypted identifier attributes before returning them to the recipient, who will thus get only privacy-sanitized data.
Second, we proposed a value *k-generalization algorithm* implementing the *K-protection property*. Our algorithm allows the execution engine to generalize the encrypted value v_e received from a service S_i before proceeding with the invocation of the subsequent service S_j in the composition, such that the generalized value $\text{Gen}(v_e)$ corresponds to k input encrypted values for which S_j has outputs; e.g., the identifier of Bob is generalized to cover k-1 other patients for which S_2 has an output (i.e., S_2 will not be able to distinguish between Bob and k-1 other patients).

6. IMPLEMENTATION AND EVALUATION

We evaluated our different techniques and algorithms in the healthcare and bioinformatics application domains. Both of these domains have widely embraced Web standards, such as XML and Web services [15, 12], and both are characterized by the need for a flexible and privacy-preserving data integration approach. In the healthcare domain, health actors (e.g., physicians, healthcare planers, medical researchers, etc.) need daily to combine data from autonomous health facilities and organizations (e.g., hospitals, labs, health authorities, etc.) for different purposes, such as providing better treatments for patients, preventing disease outbreaks, etc. In Bioinformatics, scientists conduct the so-called *in silico* experiments by combining biomedical data Web services [12].

To evaluate our techniques in the healthcare domain, the cardiology hospital of Lyon provided us with access to two medical databases. The identities of patients in these databases were changed. We also generated synthetic medical data about the same patients. We implemented about /400/ data Web services on top of our real and synthetic data. Services were all deployed on an extended version of AXIS 2.0 implementing our service execution model presented in Section 5.1. We built a medical ontology based on the building blocks and the data types defined in the HL7 standard, and used it for the annotation of service description files. To evaluate our techniques in the bioinformatics domain, we used a set of /300/ services from the *BioCatalogue registry*[8].

Figure 4 shows the query interface to PAIRSE. Users are assisted in formulating their SPARQL queries over domain ontologies. The figure shows also the composition plan of a selected composition, along with the obtained results.

We conducted exhaustive experiments to evaluate the performance of our integration system in the different settings. We summarize below the obtained results[9]:

Composition construction: Our experiments in [10] showed that our query rewriting based composition algorithm can handle hundreds of data Web services in a reasonable time. For example, for chain queries [25] and RPVs with a length of 3 or 4 object properties the algorithm was able to handle up to 400 services in less than 4 seconds; and for star queries [25] the algorithm was able to handle up to 400 services in less than one second. In the context of parameterized queries, our experiments in [9] showed that our algorithm to find the minimum set of services introduced only a small cost at the composition construction time (i.e., in all experiments the algorithm required less than 10% of the time needed to rewrite the query), and improved substantially the composition execution time (i.e., in all experiments the composition execution time was reduced to less than 0.75% of the time needed without optimization), as it removes redundant services. In the context of preferences queries, our experiments in [11] considered that services can be grouped in service classes. The experiments showed that the top-k compositions can be computed efficiently. For instance, for services classes containing about 400 services, the top-k compositions are computed in less than 4 seconds.

[8]http://www.biocatalogue.org/
[9]As our experiments were conducted on different machines, please refer to the corresponding paper of each cited set of experiments for detailed information about the considered hardware and software settings.

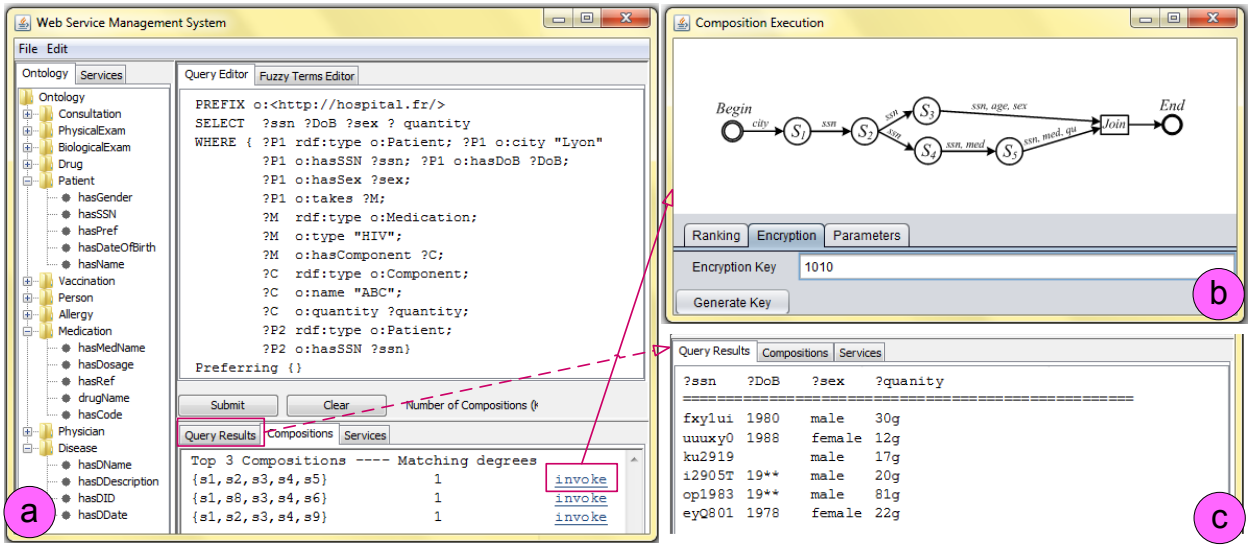

Figure 4: (a) Query interface; (b) The selected composition; (c) Obtained results

Privacy preserving service execution: The conducted experiments in [8] showed that security and privacy handling adds only a small increase (of an order of magnitude less than 10%) in the service invocation time. This could be due to the following reasons: (*i*) the time needed to inject the security and privacy constraints in the service's RDF view is almost negligible, (*ii*) rewriting the $v_{extended}$ in terms of services is not expensive, as most of $v_{extended}$'s graph is already covered by $v_{original}$ and the size of (Δv) does not exceed generally 20% of the size of $v_{original}$, and finally (*iii*) there is no network overhead incurred in invoking the additional services as they are already deployed on the server. They showed also that the cost incurred in security and privacy constraints enforcement is still relatively low if compared to the time required for executing the services without addressing the security and privacy concerns.

Privacy preserving composition execution: The conducted experiments for the evaluation of our composition execution model [7] showed that the time required to execute the composition with privacy preservation is at most three orders of magnitude of the time required without privacy preservation (K_i was set to 4 in all tests). We cut down further that cost to two orders of magnitude by *reusing* the selectivities and ranges computed in past invocations of the same services (and during the same composition execution).

7. CONCLUSION

The main objective of the PAIRSE project was to develop new methods and techniques for flexible and privacy-preserving data integration. This project has led to significant advances in the field. The results were published in international journals and conferences. Original and efficient tools have been developed and experimented in the healthcare application domain. Finally, the project has enabled the development of collaborations between partners.

8. ACKNOWLEDGMENTS

This research work is funded by the French National Research Agency under the grant number ANR-09-SEGI-008

9. ADDITIONAL AUTHORS

Francois Paulus (Semsoft Company, `francois.paulus@semsoft-corp.com`),
Stephane Morucci (Swid Company, `stephane.morucci@swid.fr`),
Michael Mrissa (Lyon 1 University, `michael.mrissa@univ-lyon1.fr`),
Nora Cuppens (Telecom-Bretagne, `nora.cuppens@telecombretagne.eu`),
Chirine Ghedira (IAE Lyon - Ecole universitaire de management, `chirine.ghedira-guegan@univ-lyon3.fr`),
Riad Mokadem (IRIT, Paul Sabatier University, `Riad.Mokadem@irit.fr`),
Said Oulmakhzoune (Telecom-Bretagne, `said.oulmakhzoune@swid.fr`) and Jocelyne FAYN (Universit Lyon 1, INSA-Lyon,

Jocelyne.Fayn@insa-lyon.fr).

10. REFERENCES

[1] E.u. directive on data protection, official journal of the european communities, 23 november 1995.

[2] Health insurance portability and accountability act of 1996, united states public law 104-191.

[3] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization based access control. In *4th IEEE POLICY*, Italy, June 2003.

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

[5] N. Ajam, N. Cuppens-Boulahia, and F. Cuppens. Contextual privacy management in extended role based access control model. In *DPM/SETOP*, pages 21–35, 2009.

[6] P. Ashley and D. Moore. Enforcing privacy within an enterprise using ibm tivoli privacy manager for e-business. In *VLDB*, pages 108–119, 2003.

[7] M. Barhamgi, D. Benslimane, Y. Amghar, N. Cuppens-Boulahia, and F. Cuppens. Privcomp: A privacy-aware data service composition system. In *EDBT*, 2013 (To appear - Demo paper).

[8] M. Barhamgi, D. Benslimane, C. Ghedira, S.-E. Tbahriti, and M. Mrissa. A framework for building privacy-conscious daas service mashups. In *IEEE ICWS*, pages 23–30, 2011.

[9] M. Barhamgi, D. Benslimane, C. Ghedira, S.-E. Tbahriti, and M. Mrissa. Optimizing daas web service based data mashups. In *IEEE SCC*, pages 464–471, 2011.

[10] M. Barhamgi, D. Benslimane, and B. Medjahed. A query rewriting approach for web service composition. *IEEE Transactions on Services Computing*, 3(3):206–222, 2010.

[11] K. Benouaret, D. Benslimane, A. HadjAli, and M. Barhamgi. Fudocs: A web service composition system based on fuzzy dominance for preference query answering. *PVLDB*, 4(12):1430–1433, 2011.

[12] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, and M. Roos. Biocatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research*, 38(5):689–694, 2010.

[13] M. J. Carey, N. Onose, and M. Petropoulos. Data services. *Commun. ACM*, 55(6):86–97, 2012.

[14] M. J. Carey, P. Reveliotis, and S. Thatte. Data service modeling in the aqualogic data services platform. In *SERVICES I*, pages 78–80, 2008.

[15] A. Dogac. Interoperability in ehealth systems (invited tutorial). *PVLDB*, 5(12):2026–2027, 2012.

[16] S. Dustdar, R. Pichler, V. Savenkov, and H. L. Truong. Quality-aware service-oriented data integration: requirements, state of the art and open challenges. *SIGMOD Record*, 41(1):11–19, 2012.

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

[18] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[19] I. Ketata, R. Mokadem, and F. Morvan. Resource discovery considering semantic properties in data grid environments. In *Globe*, pages 61–72, 2011.

[20] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, and Y. Xu. Limiting disclosure in hippocratic databases. In *VLDB*, pages 08–19, 2004.

[21] S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, and S. Morucci. fQuery: SPARQL Query Rewriting to Enforce Data Confidentiality. *Proc. of the 24th IFIP WG11.3 Working Conference on Data and Applications Security and Privacy. Rome, Italy*, 21-23 June 2010.

[22] R. Pottinger and A. Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001.

[23] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD Conference*, pages 551–562, 2004.

[24] D. Srivastava. Subsumption and indexing in constraint query languages with linear arithmetic constraints. *Ann. Math. Artif. Intell.*, 8(3-4):315–343, 1993.

[25] M. Steinbrunn, G. Moerkotte, and A. Kemper. Heuristic and randomized optimization for the join ordering problem. *VLDB J.*, 6(3):191–208, 1997.

[26] C. Yu and L. Popa. Constraint-based xml query rewriting for data integration. In *SIGMOD Conference*, pages 371–382, 2004.