1-1-2016

Dynamic Privacy Management In Services Based Interactions

Nariman Tm Ammar
Wayne State University,

DYNAMIC PRIVACY MANAGEMENT IN SERVICES-BASED INTERACTIONS

by

NARIMAN AMMAR

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2016

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

To my family,

for all their loving care since my birth

and unfaltering support throughout my academic studies.

ACKNOWLEDGEMENTS

I am very grateful to my teacher, advisor, and mentor, Dr. Zaki Malik, who kindly and wisely guided me in the new field of Service Computing, with all his patience and support. During my doctoral studies, whenever I thought I was making jumps and leaps in my research, he challenged me with critical inquiries, and encouraged me during my slow progress whenever I felt low. I would like to express my gratitude to my dissertation committee members, Dr. Shiyoung Lu, Dr. Alexandar Kotov, and Dr. Abdelmounaam Rezgui for their patience and understanding and useful insights. I am also very grateful to Dr. Andrian Marcus for his advise and encouragement, especially during my most intense months of study. I would like to express my sincere thanks to Dr. Chandan Reddy, Dr. Marwan Abi-Antoun, and Dr. Shiyoung Lu for their advice throughout my Ph.D. candidacy years.

I am also extremely fortunate for meeting several wonderful graduate friends who supported me in many ways. I would like to thank my lab mates, Erfan, Khayyam, Fayez, Hussain, Benyamin, and Rojiar for wonderful years together and endless support. I am very grateful to my friends Talia, Xiaohui, Safraz, Vineeth, Diana, Cristina, Becky, Paul, Andre, Anthony, Amney, Alla, and many others.

I am thankful to the Computer Science department for supporting and funding me in several ways. Special thanks go to Dean Fotouhi, Jasmine Roberson, Dr. Schwiebert, LaNita Stewart, Amber Dawkins, and Nafis Ahmed.

Finally, I would like to thank my family for supporting me throughout my PhD journey.

TABLE OF CONTENTS

Dedication . . . . . **ii**

Acknowledgments . . . . . **ii**

List of Tables . . . . . **viii**

List of Figures . . . . . **x**

CHAPTER 1: INTRODUCTION **1**

1.1 Data Access Patterns 1

1.2 Dynamic Web Service Composition 3

1.3 Private Dynamic Web Service Composition 5

1.4 Problem Statement 6

1.5 Contributions To Research and Practice 6

1.6 Dissertation outline 7

CHAPTER 2: BACKGROUND **8**

2.1 Information Theory 8

2.1.1 Entropy 8

2.1.2 Joint Entropy and Conditional Entropy 8

2.1.3 Data Diversity 8

2.1.4 Information Gain 11

2.2 Data Mining 11

2.2.1 Naiive Bayes Calssification 11

2.2.2 Feature Selection 13

2.2.3 Hierarchical Clustering 14

2.3 Privacy Policy Languages 15

2.3.1 XACML Context Handling 15

2.3.2 XACML Static Privacy Policy Definitions 15

2.4 Private Data Publishing 17

2.4.1 K-Anonymity 17

CHAPTER 3: DYNAMIC PRIVACY AT THE MANUAL DATA QUERY LEVEL . 22

3.1 Privacy at The Data Level . . . 22

3.2 Motivating Scenario . . . 23

3.3 Why Context Matters? . . . 26

3.4 Architecture . . . 28

3.5 Formal Model Definition . . . 30

3.5.1 Dynamic Privacy Policy Rule Definition. . . . 30

3.5.2 Context Handling . . . 31

3.5.3 Semantic Handling . . . 36

3.5.4 Dynamic Rule Evaluation . . . 37

3.6 Prototype Implementation . . . 37

3.6.1 Extending XACML PEP . . . 38

3.6.2 MobiDyc: Private Mobile Based Cloud Based Framework . . . 40

3.7 Evaluation . . . 46

3.7.1 Case Study 1: Bio2RDF . . . 46

3.7.2 Case Study 2: E-HIP . . . 52

3.7.3 Case Study 3: Comparison to Dynamic Data Publishing . . . 55

CHAPTER 4: K-ANONYMITY BASED PRIVATE WEB SERVICE SELECTION 57

4.1 Motivating Scenario . . . 57

4.2 Web Service Composition Standards . . . 59

4.2.1 Web Service Conversation Language (WSCL) . . . 59

4.2.2 Business Process Execution Language (BPEL) . . . 61

4.3 Private Web Service Selection Model . . . 64

4.4 K-Anonymity Based Privacy Compatibility Matching (KPCM) . . . 69

4.4.1 K-Anonymity Based PCM (KPCM) . . . 73

4.5 Private Web Service Outsourcing Approach . . . 75

4.5.1 Running Example . . . 75

4.5.2 System Architecture . . . 77

4.5.3 K-Anonymity At the Operation Invocation Level (Revisited) . . . 78

4.5.4 K-Anonymity At the Business Process Level . . . 79

4.5.5 K-Anonymity Score calculation for Inhouse Operations . . . 79
4.5.6 K-Anonymity Score Calculation for Outsourced Operations . . . 80
4.5.7 K-Anonymity Types . . . 81
4.5.8 K-Anonymity as a Measure of Uncertainty . . . 83
4.5.9 Anonymous (Uncertain) Operation Invocation Precedence Graph . . . 85
4.5.10 K-Anonymity Check Algorithm (Revisited) . . . 89
4.5.11 Private Web Service Outsourcing Model . . . 89
4.6 Prototype Implementation . . . 90
4.6.1 Design-time Validation System . . . 90
4.6.2 Run-time Verification System . . . 94
4.7 Experiments . . . 95
4.7.1 Environment . . . 95
4.7.2 Jmeter Test Plan . . . 95
4.7.3 Data Set . . . 95
4.7.4 Variables and Metrics . . . 96
4.7.5 Results . . . 97
4.8 Conclusion . . . 99
CHAPTER 5: PRIVACY FLOW ANALYSIS . . . 101
5.1 Privacy Policies . . . 104
5.2 Composition Logic As Program Execution . . . 104
5.3 Contributions . . . 106
5.4 Business Process Execution Language . . . 106
5.5 Formalization of the Analysis . . . 108
5.5.1 Pri-calculus Abstract Syntax . . . 108
5.5.2 Privacy Levels Type System . . . 111
5.5.3 Private Data Flow Analysis . . . 112
5.5.4 Working Example . . . 117
5.6 Implementation . . . 118
5.6.1 The Privacy Flow Analysis . . . 120
5.6.2 The Privacy Analysis Type Checker . . . 121

5.6.3 Technology Implications . . . 122
CHAPTER 6: LITERATURE REVIEW . . . 123
6.1 Context Awareness . . . 123
6.2 Dynamic Rule Evaluation. . . . 123
6.3 Relative Sensitivity. . . . 124
6.4 Semantic Policy Definitions . . . 124
6.5 XACML Enhancements. . . . 125
6.6 Dynamic Private Data Publishing. . . . 125
6.7 XACML-based Privacy for Mobile Applications. . . . 126
6.8 Anonymization Techniques . . . 126
6.8.1 K-Anonymity . . . 126
6.8.2 L-Diversity and T-Closeness . . . 127
6.8.3 K-Anonymity Applied to Relational Data . . . 128
6.8.4 K-anonymity Applied to Graphs . . . 128
6.9 Private Service outsourcing . . . 129
6.10 Operation Invocation Correlation . . . 129
6.11 Model Checking of Services Based Interactions . . . 130
6.11.1 Process Calculi . . . 130
6.11.2 Type systems . . . 130
6.11.3 Dataflow Analyses an Lattice Models . . . 131
6.11.4 Privacy in Scientific Workflows . . . 131
CHAPTER 7: CONCLUSIONS AND FUTURE DIRECTIONS . . . 133
CHAPTER A: K-ANONYMITY BASED PRIVATE WEB SERVICE SELECTION 134
A.1 GENERATED BPEL AND WSCL FILES . . . 134
CHAPTER B: PRIVACY FLOW ANALYSIS SCENARIO EXAMPLES . . . 138
B.1 BPEL AND WSDL EXAMPLES . . . 138
REFERENCES . . . 141
ABSTRACT . . . 151
AUTOBIOGRAPHICAL STATEMENT . . . 152

LIST OF TABLES

Table 1.1: Ontology concepts from health data projects. 2

Table 1.2: Selected data type properties from the ontologies in Table 1.1. 3

Table 2.1: A set of queries QS collected over several submissions. 9

Table 2.2: Sample training data 13

Table 2.3: Conditional Probability Table 13

Table 2.4: k-Anonymity for k=2 and QI={Race, Birth, Gender, ZIP} 19

Table 3.1: A subset of the Web services in CP 25

Table 3.2: Sample queries from our adversarial scenario 26

Table 3.3: Queries from Table 3.2 as tuples. 31

Table 3.4: Results from different components for the iterations 3, 4, and 5. 44

Table 3.5: Web services implemented in our experiment 45

Table 3.6: Diversity handler results for first three iterations. 48

Table 3.7: Setup for Jmeter test plan 50

Table 3.8: Setup for Java metrics library test plan. 53

Table 4.1: Two business processes 65

Table 4.2: Domain subsumption relationships 68

Table 4.3: Transisitions (left) and all possible routes (right) 71

Table 4.4: KAnonymity counts 71

Table 4.5: WS Privacy Ruleset Items derived from Table 4.4 72

Table 4.6: KAnonymity counts 72

Table 4.7: WS Privacy Ruleset Items derived from Table 4.6 73

Table 4.8: KAnonymity counts-Only Endpoint Operations can Be called 73

Table 4.9: WS Privacy Ruleset Items derived from Table 4.8 73

Table 4.10: K-Anonymity values for the WSCL configuration in Table 4.3 75

Table 4.11: TR table (left) from the graph in Fig. 4.2 and $T_{EERouts}$ table derived from it. 79

Table 4.12: Transitions table 82

Table 4.13: K-Anonymity scores derived from the WSCL configuration in Table 4.11 . . 83
Table 4.14: K-Anonymity scores derived from the BPEL configuration in Table 4.12 . . 84
Table 4.15: Different Equivalent classes 86
Table 4.16: Routes in the $Op_i = 1, 1$ Equivalence class (derived from Table 4.14) 86
Table 4.17: Routes in the OS_1 Equivalence class (derived from Table 4.14). 86
Table 4.18: Uncertainty probabilities 87
Table 4.19: Example of K-Anonymity rules that can be applied to Web service definitions. 89
Table 4.20: Examples of Web Service and Client Assertions 94
Table 4.21: Jmeter test plan setup 96
Table 5.1: Web services involved in our scenario. 102
Table 5.2: Value flow between privacy levels 114
Table 5.3: Results of running the worklist algorithm 117
Table 6.1: An analytic evaluation of our approach compared to existing approaches. . . 132
Table A.1: Results of 10 test runs 134
Table A.2: Results of 10 test runs 135
Table A.3: K-Anonymity scores 136
Table A.4: WSCL file properties 137
Table A.5: Parameters of BPEL documents. 137

LIST OF FIGURES

Figure 1.1: Different data access scenarios in Services-based interactions. 3

Figure 1.2: A sample composition. 4

Figure 1.3: A sample composition plan 4

Figure 2.1: Diversity of combining queries from different phases 9

Figure 2.2: A bayesian network 12

Figure 2.3: Policy defined using XACML AttributeSelectors. 18

Figure 2.4: Request corresponding to the policy in Fig. 2.3 19

Figure 2.5: Policy defined using XACML AttributeDesignators. 20

Figure 2.6: Request corresponding to the policy in Fig. 2.5 21

Figure 3.1: Dynamic Semantic-based Privacy Management. 29

Figure 3.2: Approach. 29

Figure 3.3: WSO2 based implementation of our framework 39

Figure 3.4: Dynamic privacy policy management solution. 42

Figure 3.5: Dynamic privacy policy management solution. 43

Figure 3.6: ROC curve comparison 47

Figure 3.7: Cost of Context Inference over several test runs. 51

Figure 3.8: PEP Accuracy for PEP (left) vs. PEP+CH (right) 51

Figure 3.9: Scalability 52

Figure 3.10: PEP performance 54

Figure 3.11: Attribute fetch time 55

Figure 3.12: Accuracy of Decat et al. comparison 55

Figure 4.1: A choreography of a 5-operation (left) and 9-operation (right) invocation configurations. 59

Figure 4.2: A hybrid composition model using BPEL orchestration and WSCL choreography. 60

Figure 4.3: Web service privacy model. 66

Figure 4.4: A sample CP with dependencies. 68

Figure 4.5: KPCM Architecture. 70
Figure 4.6: A framework for private Web service outsourcing. 78
Figure 4.7: K-Anonymity as uncertainty from our scenario. 85
Figure 4.8: Operation precedence graphs for the three K-Anonymity types with uncertainties added to edge labels. 87
Figure 4.9: Operation precedence graph (PTOC) with uncertainties added to edge labels. 88
Figure 4.10: Operation precedence graph (PTONC) with uncertainties added to edge labels. 88
Figure 4.11: Operation precedence graph (OEOC) with uncertainties added to edge labels. 88
Figure 4.12: Prototype main view 91
Figure 4.13: K-Anonymity compatibility messages 92
Figure 4.14: WSCL admin view 93
Figure 4.15: Evaluation time of different WSCL compositions 97
Figure 4.16: Performance results of different BPEL compositions 97
Figure 4.17: Performance results of different CP sizes. 98
Figure 4.18: Performance results of different $TR_EERoutes$ table sizes for a fixed CP of size 5. 98
Figure 4.19: KPCM Performance 99
Figure 5.1: Composition as a program execution. 105
Figure 5.2: Composition as a program execution with privacy levels derived from policy files. 106
Figure 5.3: Abstract Syntax of the BPEL calculus. 108
Figure 5.4: Abstract Syntax of the Pi-calculus. 109
Figure 5.5: Dynamic semantics through reduction rules. 110
Figure 5.6: Abstract Syntax of Pri-Calculus (Pi-calculus with privacy types) 111
Figure 5.7: Typing rules 113
Figure 5.8: Simplified Annotation Syntax adapted from the formal system in Fig. 5.6. 114
Figure 5.9: Composition with Private data flow 119

CHAPTER 1 INTRODUCTION

Technology advancements facilitate the online collection and publication of data about individuals, which could potentially be distributed among several organizations (e.g., testing labs, research institutes, etc.). Each organization may manage it's data access and usage through a specialized Web service. In line with the different data sharing environments, health science data is a prime example, where the focus has been on transforming the data into ontology-based repositories using RDF (as a universal healthcare exchange language). Each repository defines an ontology (in OWL format) of all the concepts that can be searched for in a requester's query. OWL defines *classes* as a generic concept of individuals (e.g., Patient) and *data type properties* to link individuals of those classes to their data values (e.g., hasPatientStatus). For example, the Bio2RDF project incorporates data from the following ontologies (NCBIGene,PharmKGB, DrugBank, CDT, and GeneCDS). Each of the above mentioned repositories manages data access through a SPARQL endpoint. Table 1.2 shows examples of concepts from these ontologies including both data classes and data type properties. To query instances in a data repository, a requester can ask for any of the data type properties by which those instances can be identified. For example, an instance in a pharmacogenomics repository (e.g., PharmGKB, Table. 1.1) can be identified by the set of data type properties (drug, disease, gene, etc.) defined on the set of classes (Dosage, Drug, DrugGeneAssociations, etc.).

1.1 Data Access Patterns

In such services-based interactions, data can be accessed in several ways, including manual query submission (Fig. 1.1,I) through SPARQL endpoints (e.g., Bio2RDF), automated analysis pipelines and scientific workflows (e.g.,Taverna), and mashup service APIs (e.g., AIDSInfo) with minimal human interaction (Fig. 1.1,II). We briefly explain each access pattern:

SPARQL End points. SPARQL Endpoints are RESTful Web services that allow SPARQL queries to be executed against RDF datasets. They accept queries and return results via http. Some end points are generic (query any Web-accessible RDF data), while others are specific (hardwired to query against particular datasets). Results are often rendered as XML, JSON, RDF, HTML etc. Accessing Endpoints can be done via a command interface, an API, or on a graph. SPARQL end-

Table 1.1: Ontology concepts from health data projects.

Project (Ontology)	
Bio2RDF (NCBI Gene)	
Classes	**DataType properties**
Gene, ProtienCodingGene, GenefunctionAssociation	Symbol, function, startPosition, endPosition, Chromosome
Bio2RDF (DrugBank)	
Classes	**DataType properties**
Drug, DrugDosage, DrugTarget	Label, Function, Dosage, Enzyme, Gene, Protien, Chromosome
Bio2RDF (PharmGKB)	
Classes	**DataType properties**
Variant, Gene, Drug, Association, Disease, DiseaseVariantLocAssociation, DiseaseGeneAssociation, GeneGeneAssociation, DrugGeneAssociation, DrugVariantLocAssociation	gene, symbol, protien, drug, variant location, disease, isgenotyped
Bio2RDF (GeneCDS)	
Classes	**DataType properties**
Gene, Allele, Phenotype, Drug, Polymorphism, HumanWGeneticPolymorphism	Gene Symbol, rs, SPLTarget.
Bio2RDF (CDT)	
Classes	**DataType properties**
Chemical Disease Association, Chemical Gene Association, Gene, Chemical, Disease	gene symbol, disease, chemical, function, location, participant,
E-Hip (HPMS)	
Classes	**DataType properties**
Cardiologist, Date Helper, ElderCareSpecialist, GP, MedicalPersonnel, Nurse, Oncologist, Patient, PatientStatus, Person, Pysician	patientstatus, discharged, dischargeddate, location, treated, treatedbyteam, treatedInLastSixMonths, department, Emergency

points allow for $federatedqueryprocessing$ by querying a mediator which distributes subqueries to relevant data sources and integrates the results. We take the Bio2RDF project as an example. In Bio2RDF project data sources are exposed as DaaS services. RDF views are incorporated within service's WSDL files as annotations. Users can then issue SPARQL queries on a mediated ontology. Then, a mediator service uses WSDL files of those services for service selection.

Scientific Workflows and Online Analysis Pipelines. Analysis pipelines and workflows rely on downloading multiple datasets from public databases, copying and pasting from one web-based tool to another, integrating data, and transferring it into another tool. Scientific workflows have emerged as a paradigm for scientists to formalize and structure complex and distributed scientific processes to enable and accelerate scientific discoveries. Org4 (Fig 1.1,D) best describes this data access scenario.

Mashup Service APIs. Mashups are composed of easy to integrate data sources. A data source

Table 1.2: Selected data type properties from the ontologies in Table 1.1.

Ontology	DataType properties
NCBI Gene	Symbol, function, startPosition, endPosition, Chromosome
DrugBank	Drug Label (DL), Function, Dosage, Enzyme, Gene, Protien
PharmGKB	Gene, Symbol, Protien, Drug, Price (PR), variantLocation (SNP), disease (DI)
GeneCDS	GeneSymbol (GS), SNP, SPLTarget.
CDT	gene symbol, disease, chemical, function, location, participant,
E-Hip	PatientStatus (PS), Discharged (D), DischargedDate (DD), Location (L), Gender (G), Department (DE)
US Census	Age (A), Address (AD), Income(IN), Employer(EM), MaritalStatus(MS)

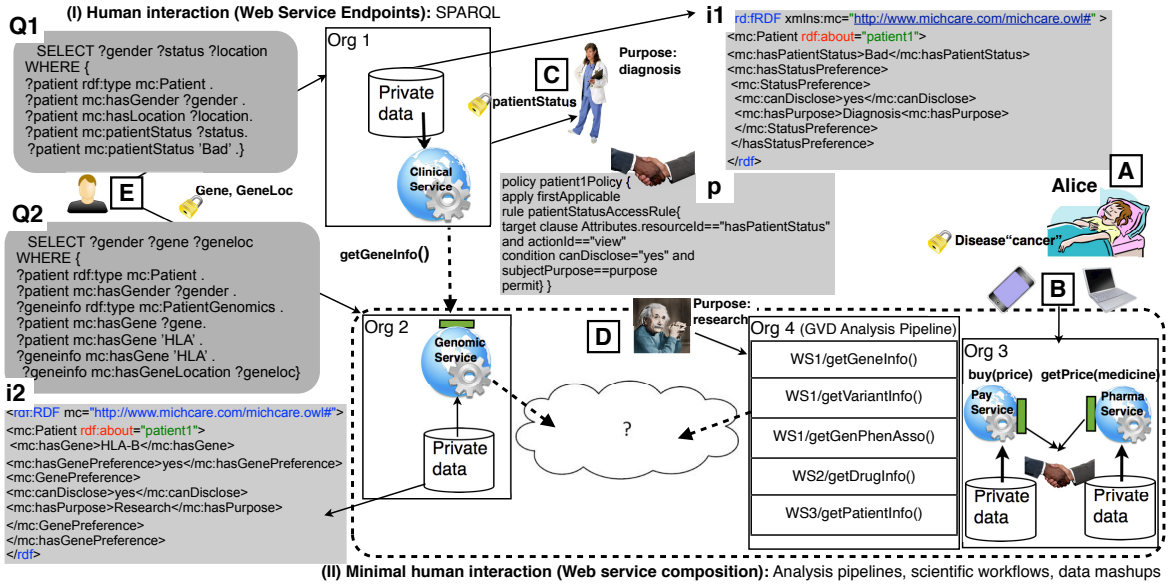


Figure 1.1: Different data access scenarios in Services-based interactions.

could be an in-house database, a web page, a web service, etc.. Using mashup technology, the mashup Web application receives a service request, dynamically determines the service providers, requests the required data through their APIs, integrates the collected data, and returns it to the original requester. A good example of this is BioCatalogue. BioCatalog provides a set of public RESTful endpoints that allow you to query the registry programmatically and integrate the data and functionality into your own scripts, workflows, apps, tools and mashups. Scientific workflows (e.g. taverna) are integrated with service registries, such as BioCatalog, and analysis pipelines, such as MyExperiment.

1.2 Dynamic Web Service Composition

In all scenarios mentioned above, dynamic Web service composition [73] may be involved, especially since the queried data may not necessarily get retrieved from a single Web service. Dynamic composition enables the specification of composite services without knowing a priori which

Web services will be actually used at run-time. Therefore, each composite service should have a composition plan CP. Fig. 1.2 depicts one CP, where nodes represent services and edges represent functional dependencies between services based on relations between their input and output parameters. Several composition plans $CP = \{CP_1, ..., CP_n\}$ can be returned by an execution engine to answer a query Q. In a CP, any service WS_1 which depends on another service WS_2 is considered a user of the data provided by WS_2. Thus, WS_1 can be viewed as a client and WS_2 can be viewed as a provider.

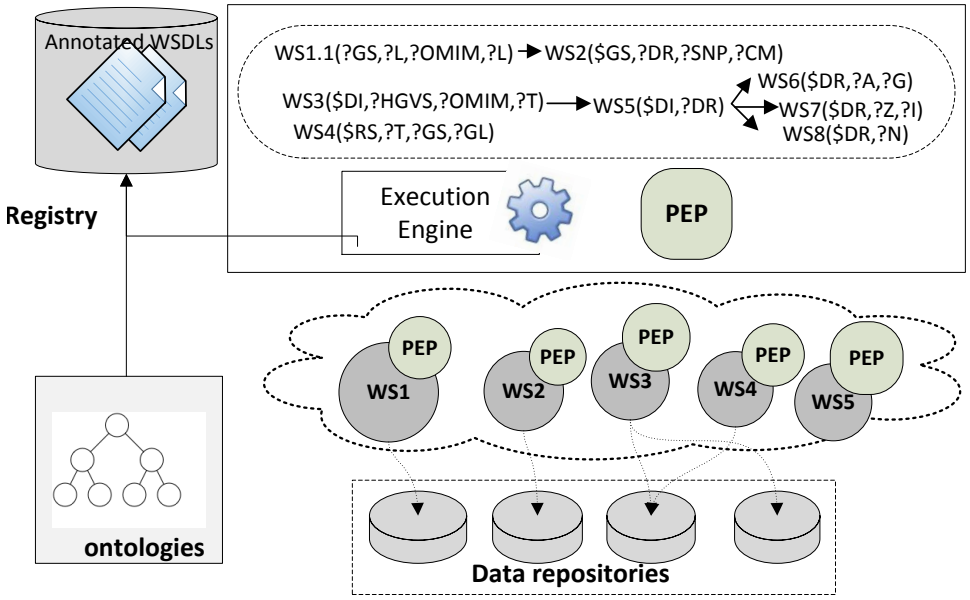


Figure 1.2: A sample composition.

For example, Org3 (Fig. 1.1) is composed of a payment service ($PayService$), which depends on a pharmacy service ($PharmaService$) to accomplish its task. Similarly, in SPARQL endpoints, a mediator service uses WSDL files of those services for service selection and CP formation. In all cases, a query gets decomposed into several sub queries if the data asked for in the query should be retrieved from multiple Web services (Fig. 1.3).

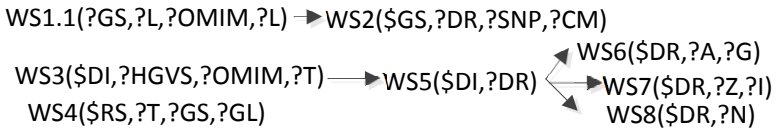


Figure 1.3: A sample composition plan that breaks a query down into subqueries against several services.

1.3 Private Dynamic Web Service Composition

Dynamic composition of services in those environments also requires appropriate privacy mechanisms to guarantee the confidentiality of all participants along the composition chain. Thus, before providing WS1 with a data item, WS2 should check the data owners privacy policy rule regarding that item. Checking for privacy rules should hold along the composition chain until all the data items requested in the original query initiated by the user are retrieved. Thus, privacy requirements should hold for all the dependencies among the services in CP, and our role becomes to validate CP in terms of privacy. If at least one dependency in CP violates a privacy rule, then CP is discarded from CP and the corresponding data item is discarded from the query results.

For instance, consider the query *Q="What are the ages, genders, zip, DNA, and income for patients infected with cancer and what are the prices paid by patients who reside in that area*. A subset of the services that might be involved in answering this query are depicted by the composition plan in Fig. 1.3, which takes the query $Q = \{A, G, Z, N\}$ and combines the results and returns the output. The data item DR (i.e. drug) is an input for WS_6, WS_7 and WS_8 and it is an output for WS_5.. Thus, WS_6, WS_7, and WS_8 depend on WS_5, for example, for providing DR.

Assume that *Actiq* (which is a cancer medication) is one of the very few medicines that costs $45.89. It is very easy for an adversary to use the above query to deduce that a patient has been diagnosed for cancer if they know how much they paid for a medicine (via a payment service) that is used for cancer. However, a smart privacy management engine should track the query history against all participating Web services to fulfill that query and use that history to learn more about the motives of the requester. If we treat each subquery aginst each sub Web service as a request and apply our inference algorithm to those requests.

A PEP at the composition orchestrator or mediator level as well as individual PEPs at component service levels can help propagate that query history information and infer context from the query history. Each component service may be in itself a composition of another group of services and so on. Since the inferred context is tied to the data resource, and since a data source can be shared by multiple services our approach works regardless of which Web services participated in the plan.

1.4 Problem Statement

The research problem discussed in this dissertation is motivated by adversarial data access scenarios that may take place in collaborative service-based data sharing environments. Namely, we study environments in which the participating services host different sets of data about the same users, identified by some common properties. We generalize two of the main privacy issues from which such environments may suffer:

- First, transforming such data sets into semantic data makes data linkage easier and machine processable.
- Second, *dynamic composition* of different data items (retrieved through participating Web services) may be misused by adversaries to reveal sensitive information, which was not deemed as such by the data owner at the time of data collection. Atomically, these data items may not reveal personally identifiable information, but linking those items may lead to unintended breach of privacy.

Thus, the user's consent that is statically defined in a privacy policy may not be enough for restricting data disclosure. We need a privacy management solution that is, *dynamic*, *context-sensitive*, and *semantic-based*.

1.5 Contributions To Research and Practice

This thesis makes the following contributions and focuses on dynamic privacy management in services based interactions. The thesis deemphasizes the role of applying cryptography and security techniques in achieving privacy, focusing instead on using data mining techniques to make privacy management smarter. I assume that the proposed techniques can be complemented by existing security and cryptography techniques, but these contributions are not in the scope of this thesis:

- We build a dynamic, context-aware, semantic-based privacy policy management framework on the top of the XACML reference architecture for policy-based access control. We evaluate implement the proposed framework using an existing deployment engine that incorporates the proposed context handling model into the privacy management module, and we incorporate the implemented framework into both a Web based and mobile based prototypes. We evaluate the feasibility of the implemented approach in two case studies in comparison to

similar existing works.

- We build an approach for private Web service selection that leverages the notion of K-Anonymity to the level of a Web service operation. We generalize the approach to the level of business process by solving the problem of protecting privacy in Web service outsourcing. We integrate the proposed model into an existing framework to achieve privacy at the level of the outsourced data, the outsourced operation, and the outsourced business process logic. We provide an implementation of our approach using an existing process management framework and provide an empirical evaluation of the feasibility of the approach.
- We introduce Pri-calculus, as an extension of the Pi-calculus with privacy level types; We define a data flow analysis for private service composition, including the lattice model, the abstract syntax, and dynamic semantics of our extended calculus; We also present a prototype implementation of our analysis as a privacy level annotations-based type checker and incorporate it into a composition engine;

1.6 Dissertation outline

The dissertation proposal is divided into six chapters. Chapter 2, introduces some background information. The chapter first introduces some mathematical concepts that we apply in this dissertation. It then describes a standard privacy policy language. Chapter 3, explains an approach for dynamic privacy management at the data access level. Chapters 4 and 5 explain two approaches for dynamic privacy management at the operation level. Chapter 6, discusses some related works. Finally, Chapter 7 concludes and highlights future research directions.

CHAPTER 2 BACKGROUND

This dissertation proposal utilizes several mathematical frameworks and techniques and this chapter highlights those techniques through an example. We also give a brief background about some Web service standards and privacy policy languages.

2.1 Information Theory

We briefly list basic definitions from information entropy theory followed by examples from our scenario.

2.1.1 Entropy

Entropy is a measure of uncertainty of a random variable [31]. The entropy $H(D_A)$ of a discrete random variable D_A is defined by:

$$H(D_A) = -\sum_{d_a \in D_A} Pr(d_a) log_2 Pr(d_a)$$

2.1.2 Joint Entropy and Conditional Entropy

The notion of entropy can be extended to two random variables. The joint entropy $H(D_A, D_B)$ of two discrete random variables D_A and D_B with joint probability $Pr(d_a, d_b)$ is defined as:

$$H(D_A, D_B) = -\sum_{d_a, d_b \in D_A, D_b} Pr(d_a, d_b) log_2 Pr(d_a, d_b)$$

Conditional entropy of one random variable given another is defined as:

$$H(D_A|D_B) = -\sum_{d_a, d_b \in D_A, D_b} Pr(d_a, d_b) log_2 Pr(d_a|d_b)$$

2.1.3 Data Diversity

Diversity of a set of data depends on the number of homogeneous groups of data and the proportion of attributes in each group. Entropy can be used as a measure of data diversity. According to Shannon: First, if there are multiple possible options which are equally likely, there is more uncertainty (monotonicity). Thus, the smaller the entropy, the fewer the number of different data items or the more regular the data items are. Second, if a data set is defined as the combination of

Table 2.1: A set of queries QS collected over several submissions.

phase	Q_i	P_i	D_i
I	0	Research	IncomeLevel,age
I	0	Diagnosis	name,testResult
I	0	Research	employer, incomeLevel
I	0	FederalTax	name,incomeLevel
II	1	Research	age,incomeLevel
II	0	Research	employer,incomeLevel
II	1	Diagnosis	name,testResult
II	2	Research	incomeLevel,testResult

several disjoint data sets, the entropy for them combined should be at least the weighted sum of the individual entropy values for the individual sets. For two data sets denoted by D_I and D_{II}, the overall entropy should be higher, or at least equal to the weighted entropy of the individual data sets involved:

$$H(D) = H(a,b) + aH(D_I) + bH(D_{II}), \text{ where } a = \frac{D_I}{D}, b = \frac{D_{II}}{D}$$

For example, we can use entropy as a measure of irregularity among a set of queries in a query space QS by determining the constant and varying attributes of QS. The attributes in our case are the query's purpose P and the set of data type properties D in the query and a combination of both. For instance, assume a set of queries that are gathered from several submissions (Table 2.1). In the following, we explain how we calculate the diversity based on each of these attributes:

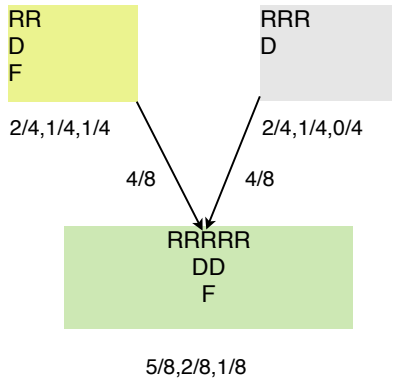


Figure 2.1: Diversity of combining queries from different phases. We assume the number of queries per phase is 4.

Diversity of queries by purpose P. Assume a set of queries gathered from several submissions (Table 2.1). We divide the queries into two phases assuming the number of queries per phase is 4. First, for the P attribute, in phase I, we have 4 queries two of which are for research purposes,

while the other two are for two different purposes so the initial entropy of QS with respect to attribute P is:

$$H_P(QS_I) = \frac{2}{4}log\frac{2}{4} + \frac{1}{4}log\frac{1}{4} + \frac{1}{4}log\frac{1}{4}$$

In phase II, two queries are for research purpose, one is for diagnosis purpose, and one is for federalTax purpose:

$$H_P(QS_{II}) = \frac{2}{4}log\frac{2}{4} + \frac{1}{4}log\frac{1}{4} + 0$$

If we combine both phases, the diversity increases (Fig. 2.1):

$$H_P(QS) = H(\frac{4}{8}, \frac{4}{8}) + \frac{4}{8}H(\frac{2}{4}, \frac{1}{4}, \frac{1}{4}) + \frac{4}{8}H(\frac{2}{4}, \frac{1}{4})$$

Diversity of queries by data D broken up by purpose P. Next, for each purpose, we calculate entropy for each subset of the data as broken up by that purpose. For example, our data suggest that of the queries that are for research purpose, one is asking for {income,age} while the other is asking for other data. For the other two purposes the entropy is 0.

$$H_{D|P=Research}(QS_I) = -\frac{1}{2}log(\frac{1}{2}) - (\frac{1}{2})log(\frac{1}{2})$$
$$H_{D|P=Other}(QS_I) = -(1)log(1) = 0$$

If we include the queries from phase II, three queries are for research purposes. Of these, two are asking for {age,income}. One is for diagnosis and it asks for {name,test}:

$$H_{D|P=Research}(QS_{II}) = -(\frac{2}{3})log(\frac{2}{3}) - (\frac{1}{3})log(\frac{1}{3})$$
$$H_{D|P=Diagnosis}(QS_{II}) = -(1)log(1) = 0$$

Then, we combine data from both phases $H_{D|P}(QS)$ for each purpose P and monitor the change in diversity.

Diversity of queries by Purpose P and Data D combined. Then, we consider both attributes

combined. Our collection shows that in phase I the requester's set of queries contain one query that is asking for {income level,age} for research purpose, one query asking for name and testResult for diagnosis purpose, one asking for employer and income for research purpose and one asking for {name,incomeLevel} for federalTax purpose. Thus, the initial entropy of QS is 1 which implies regularity of the query data:

$$H_{D,P}(QS_I) = 1$$

We calculate the entropy $H_{D,P}(QS_{II})$ by combining results from phase II. Then, we combine data from both phases $H(QS_{D,P})$ and monitor the change in diversity.

2.1.4 Information Gain

Information gain I is a measure of mutual information between two random variables D_A and D_B. In terms of entropy, mutual information is the reduction in the uncertainty of D_A due to the knowledge of D_B. Thus, I can be calculated based on entropy values H of each variable as follows:

$$I(D_A, D_B) = H(D_A) - H(D_A|D_B)$$
$$H(D_A) = -\sum_{d_a \in D_A} Pr(d_a) log_2 Pr(d_a)$$
$$H(D_A|D_B) = -\sum_{d_a, d_b \in D_A, D_b} Pr(d_a, d_b) log_2 Pr(d_a|d_b)$$

2.2 Data Mining

2.2.1 Naiive Bayes Calssification

To classify the queries in the above scenario as malicious or legitimate, we can use a classification model. Bayesian classification is appropriate for our purposes. We assume that the presence of one data type property in a classification is conditionally independent of another data type property. We make the same assumption regarding the purpose of a query. Based on this assumption, we construct a $Naive\ Bayesian\ Classification$ model. We convert a query Q_i into a $Bayesian\ Network\ BN$ (Fig. 2.2), where the root node represents the query's purpose P_i and the children represent the data type properties d_k in that query. In the rest of this dissertation, we treat the data type properties in a query as a set D_i. Based on our assumption, $Pr(D_i|C_j)$ is equivalent to

$Pr(d1, d2, .., d_k|C_j)$ which is equivalent to $Pr(d_1|C_j)Pr(d_2|C_j)...Pr(d_k|C_j)$.

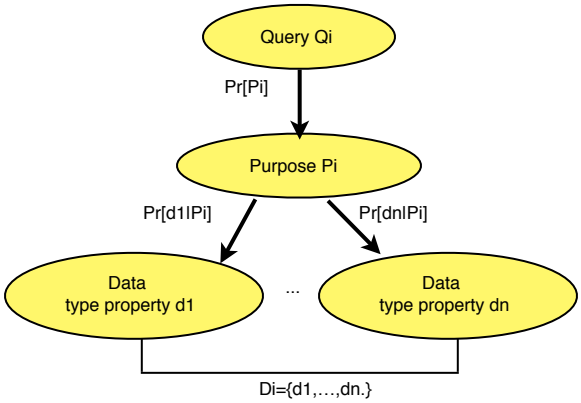


Figure 2.2: A bayesian network that reflects our query classification model. The line $D_i = d1, d2, ..., dk$ is just added for clarification purpose and is not part of the network structure.

We then apply the Naiive Bayesian learning algorithm [1]. The input to the learning algorithm is a set of labeled data (training data) and the instance to be classified (query) and the output is a classification C_j of that instance. For each query Q_i the learning algorithm is given the purpose P_i and the set of data type properties D_i. The parameters to be estimated are the purpose probabilities $Pr(P_i)$ and the conditional probabilities $Pr(D_i|P_i)$. For example, given the training data in Table 2.2, we wish to predict the class label of a newly submitted query Q_{i+1}:

$$Q_{i+1} = [C_{j+1} =?, P_{i+1} = Research, D_{i+1}],$$
$$\text{where } D_{i+1} = \{incomeLevel, age, employer\}$$

The prior probabilities can be computed based on the training samples. From the prior probabilities, we compute the posterior probabilities. The goal is to maximize $Pr(Cj|Pi, Di)Pr(Cj)$, for $j = \{malicious, legitimate\}$. First, we compute the prior probability $Pr(Cj)$ for each classification from the training set. Then, to compute $P(Cj|Pi, Di)P(Cj)$ we compute the conditional probabilities $Pr(Pi, Di|Cj)$ for all values of Pi and Di assuming conditional independence between the two random variables Pi and Di. We summarize the results in a conditional probability table CPT (Table 2.3). A CPT correspond directly to the learned parameters mentioned earlier. Based on the estimated parameters, the learning algorithm computes:

[1]The algorithm is implemented in several software packages including weka [44], matlab and orange

Table 2.2: Sample training data in the form of a set of queries data and previously determined classification

No.	Q_i	P_i	D_i	C_i
1	0	Research	Income,age	Legitimate
2	1	Research	age	Legitimate
3	0	Diagnosis	name	Malicious
4	0	Research	employer	Malicious
5	0	Diagnosis	name	Legitimate
6	1	Diagnosis	age	Malicious
7	0	Diagnosis	Testresult	Legitimate
8	0	Diagnosis	Testresult	Legitimate
9	0	Diagnosis	Testresult	Legitimate
10	0	Diagnosis	Testresult	Legitimate
11	**0**	**Research**	**Age,employer, incomelevel**	**?**

Table 2.3: Conditional Probability Table

P_i, D_i	C_i		**Marg. dist. for** P_i, D_i
	Malicious	**Legitimate**	
Research, incomeLevel	0	0	0
Research, name	0	0	0
Research, age	0	1/10	1/10
Research, employer	1/10	0	1/10
Research, incomeLevel,age	0	1/10	1/0
Diagnosis, incomeLevel	0	0	0
Diagnosis, name	1/10	1/10	2/10
Diagnosis, testResult	0	4/10	4/10
Diagnosis, age	1/10	0	1/10
Marg. dist. for C_i	3/10	7/10	**1**

$$Pr(Q_{i+1}|C_j)Pr(C_j), for j = \{malicious, legitimate\}$$

$$Q_{i+1} = (P_{i+1} = Research, D_{i+1})$$

, where $D_{i+1} = \{incomeLevel, age, employer\}$

If the probability of malicious, malicious is greater than legitimate, legitimate, then we deduce that the NBC predicts the new query Q_{i+1} to be malicious.

2.2.2 Feature Selection

Feature or attribute selection (significance) is a contrast data mining technique that assess the discriminative power of a certain feature or attribute in representing the entire data set. The tech-

nique is often used to enhance classification accuracy or speed up the model training. The metric can be measured in two ways: information measures including signal to noise, information gain ratio, etc. or statistical tests such as t-tests, Kolmogorov-Smirnov, Chi squared, etc.

In our case, for example, if the query is classified as malicious, feature selection can be used to filter out the subset of data type properties D'_j in the query that resulted in a malicious classification.

2.2.3 Hierarchical Clustering

Hierarchical Clustering is a technique to group a set of data into groups based on similar attributes [9]. The technique produces a set of nested clusters organized as a hierarchical tree, which is often visualized as a dendrogram (a tree like diagram that records the sequences of merges or splits) at each stage in the algorithm. The power of Hierarchical Clustering lies in the fact that it does not assume a particular number of clusters. Any desired number of clusters can be obtained by "cutting" the dendogram at the proper level.

Hierarchical clustering can be classified into two main techniques: Agglomerative and Divisive. Agglomerative clustering starts with the data points as individual clusters. Then, at each sage, merges the closest pair of clusters until only one cluster (or k clusters) are left. Divisive clustering, on the other hand, starts with one, all-inclusive cluster. Then, at each stage, splits a cluster until each cluster contains a data point (or there are k clusters). In this dissertation, we focus on Agglomerative clustering.

The key operation in hierarchical clustering is calculating the similarity or distance matrix, through which a merge or split decision is made. Different approaches to defining the distance between clusters distinguish the different algorithms. The widely studied algorithm in this category are the following: single linkage, complete linkage, average linkage among others. Single linkage algorithm computes similarity of two clusters based on the two most similar (closest) points in the different clusters. So, it is determined by one pair of points, i.e., by one link. Average linkage, on the other hand, is determined based on the average of pairwise similarity between points in the two clusters. We focus on these techniques for the purpose of this dissertation.

2.3 Privacy Policy Languages

Several privacy policy languages have been proposed, including P3P, Reni, etc. Our focus in this dissertation is on the Access Control Markup Language XACML.

2.3.1 XACML Context Handling

One of the most widely used privacy policy languages is XACML [91]. According to a standard XACML-based privacy policy management model, the organization hosting the Web service should define a Policy Administration Point (PAP), through which policies can be defined and deployed to a Policy Decision Point (PDP). Context handling in XACML is a protocol of communication between a PDP and a Policy Enforcement Point (PEP) (located either on the user agent side, the Web service side, or on a gateway between the user and the service). The PEP forms an XACML request and sends it to the PDP through the *Context Handler*, which collects initial attributes from the Policy Information Points (PIP). The PDP then uses those attributes to evaluate policies. The PDP requests additional attributes from the context handler as needed and finally returns a Permit or Deny decision to the PEP, which enforces the final decision.

2.3.2 XACML Static Privacy Policy Definitions

In XACML, a policy consists of a set of rules, each of which consists of a *subject*, *object*, *action*, and a set of optional *obligation*s. An XACML request consists of attributes including the *subject*, *object*, *action*, and *environment*. The PDP evaluates the request against the deployed policies in it's policy store based on the predefined rules. In case of "permit" decisions, the PEP performs obligations specified in a policy rule. We posit that we need a mechanism to dynamically identify the context of data usage and make a decision regarding data disclosure based on the inferred context at run-time. Context has been defined in the literature in terms of trust, affiliation, query history, temporal or spatial relationships [96]. Some of these solutions base access control decisions on static information, such as particular users or roles, and are therefore pre-determined. Recently, few researchers proposed solutions to dynamically handle context. However, the dynamicity of these solutions is not achieved at the privacy policy rule level. Moreover, these rules are not defined in semantic terms and do not infer context based on previous data usage behavior. We aim to extend the XACML PEP to be *dynamic*, *context-sensitive*, and *semantic-based*. Example of a policy with obligation is shown below:

Listing 2.1: A privacy policy that governs the usage of the instance in Listing 3.1 written in ALFA

```
namespace obligations{ obligation QRObligation = "edu:wayne:obligation:ch"}
policy patient1Policy {

rule patientStatusAccessRule{
      target clause Attributes.resourceId=="patientStatus" and actionId=="view
          "
      condition canDisclose="yes" and subjectPurpose==purpose
      permit}
on permit{
obligation obligations.QRObligation{
Attributes.resourceId = "hasContext"
}}}
```

XACML Requests. An XACML request consists of a set of attributes. An attribute can have one of three categories: resource, subject, and action. A request can also have a content element as we will explain later. Listing 2.2 depicts a simplified XACML request that corresponds to the SPARQL query in Listing 3.2.

Listing 2.2: An XACML request corresponding to Q_6

```
<Request>
<Attributes Category="resource">
<Attribute AttributeId="resource-id">
<AttributeValue>patientStatus</AttributeValue>
</Attribute>
<Attribute AttributeId="mc:canReleaseStatus">
<AttributeValue>yes</AttributeValue>
</Attributes>
<Attributes Category="access-subject">
<Attribute AttributeId="subject-purpose">
<AttributeValue>Diagnosis</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="action">
```

```
<Attribute AttributeId="action-id">
<AttributeValue>view</AttributeValue></Attribute>
</Attributes>
</Request>
```

Request handling in XACML is done in two different forms:

- In the first form, a PEP uses the Content element in XACML to send data directly to the XACML engine (PDP) together with the request and the PDP takes care of it. The PEP adds the retrieved data under the content element of the request and indicates the requested data as a resource attribute using XPath expressions. Finally, the PEP sends the request over to the PDP, who does policy evaluation by looking at the content element. The PDP extracts data from the content element using XPath. For example it can extract the $hasGene$ data from content element. In this case `AttributeSelector` elements are used to define the policies. Figures. 2.3 and 2.4 show a sample policy and it's corresponding request.
- In the second form a PEP parses the content of the matching data and extracts the values and sends them over to PDP. In this case, `AttributeDesignators` are used in XACML policy definitions to refer to data within files since the PDP can not parse the file content. This is called direct attribute matching, where the PEP uses a parser for fetching data from the data file then puts the desired attributes in the XACML request using the PEP client code. Figures 2.5 and 2.6 show a sample policy and it's corresponding request.

In both cases, the PEP creates a XACML request with the set of attributes of one or more category (subject, resource, action, environment). For example, we can define policies whose target will check if resource attribute $hasGene$ can be released and then there will be a permit effect rule which will check if resource attribute "$canReleaseGene$" is yes.

2.4 Private Data Publishing

2.4.1 K-Anonymity

K-Anonymity is an effective privacy requirement that has been first proposed by Samarati and Sweeny [92] to prevent linking an individual to a record in a data table through a quasi-identifier *QID*. A data table satisfies k-anonymity if every combination of values on a unique identifier is shared by at least k records in the table, where the identifier is a set of data items that could

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" xmlns:xacml="
   urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
PolicyId="policy1"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
   algorithm:first-applicable" Version="1.0">
<PolicyDefaults>
<XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
</PolicyDefaults>
<Target>
</Target>
<Rule RuleId="gender_disclosure_rule" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
//mc:Patient/mc:hasGene</AttributeValue>
<AttributeDesignator MustBePresent="false" Category="
   urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="
   urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3
   .org/2001/XMLSchema#string">
</AttributeDesignator>
</Match>
</AllOf>
</AnyOf>
</Target>
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
yes</AttributeValue>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
<AttributeSelector MustBePresent="false"
ContextSelectorId="urn:oasis:names:tc:xacml:3.0:content-selector" Category="
   urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
Path="//mc:Patient/mc:canReleaseGene/text()"
DataType="http://www.w3.org/2001/XMLSchema#string">
</AttributeSelector>
</Apply>
</Apply>
</Condition>
</Rule>
<Rule RuleId="rule2" Effect="Deny">
<Description>Deny rule</Description>
</Rule>
</Policy>
```

Figure 2.3: Policy defined using XACML AttributeSelectors.

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
   CombinedDecision="false"
ReturnPolicyIdList="false">
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource
   ">
<Content>
 <rdf:RDF xmlns:mc="http://localhost:9443/GenomicWS/michcare.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <mc:Patient rdf:about="http://localhost:9443/GenomicWS/michcare.owl#patient1"
    >
   <mc:hasIncome>50K</mc:hasIncome>
   <mc:hasLocation>UnitedStates</mc:hasLocation>
   <mc:canReleaseGene>yes</mc:canReleaseGene>
   <mc:hasGene>HLAB</mc:hasGene>
   <mc:hasRace>White</mc:hasRace>
   <mc:hasEducation>Bachelors</mc:hasEducation>
  </mc:Patient>
</rdf:RDF>
</Content>
<Attribute IncludeInResult="false"
AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
<AttributeValue
XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">
//mc:Patient/mc:hasGene
</AttributeValue>
</Attribute>
</Attributes>
</Request>
```

Figure 2.4: Request corresponding to the policy in Fig. 2.3. The matching instances should be appended to the request as content elements.

potentially identify a record in the table. Formally, a table is called K-Anonymous if for one record in the table that has some value QID, at least k-1 other records also have the value QID. In a K-Anonymous table, each record is indistinguishable from at least k-1 other records, with respect to QID. For example, Table 2.4 is 2-anonymous with respect to *QID*={*Race,Birth, Sex, ZIP*}.

Table 2.4: k-Anonymity for k=2 and QI={Race, Birth, Gender, ZIP}

Race	Birth	Gender	ZIP	disease
black	1965	m	0214*	short breath
black	1965	m	0214*	chest pain
White	1964	m	0213*	obesity
White	1964	m	0213*	depression
White	1964	m	0213*	cancer

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
PolicyId="policy2" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
   combining-algorithm:first-applicable" Version="1.0">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
hasGene</AttributeValue>
<AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0
   :resource:resource-id" Category="urn:oasis:names:tc:xacml:3.0:attribute-
   category:resource"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true">
</AttributeDesignator>
</Match>
</AllOf>
</AnyOf>
</Target>
<Rule Effect="Permit" RuleId="rule-1">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">yes</
   AttributeValue>
<AttributeDesignator AttributeId="mc:canReleaseGene"
Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false">
</AttributeDesignator>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
<Rule RuleId="rule2" Effect="Deny">
<Description>Deny rule</Description>
</Rule>
</Policy>
```

Figure 2.5: Policy defined using XACML AttributeDesignators.

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
   CombinedDecision="false"
ReturnPolicyIdList="false">
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource
   ">
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
   IncludeInResult="false">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
hasGene</AttributeValue>
</Attribute>
<Attribute AttributeId="mc:canReleaseGene" IncludeInResult="false">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
yes</AttributeValue>
</Attribute>
</Attributes>
</Request>
```

Figure 2.6: Request corresponding to the policy in Fig. 2.5. The matching instances do not have to be appended to the request as content elements. Rather, a direct attribute matching is performed.

CHAPTER 3 DYNAMIC PRIVACY AT THE MANUAL DATA QUERY LEVEL

Of the Web service data access patterns described in Section 1.1, this chapter focuses on manual data access via query submission and how privacy can be achieved in that case.

3.1 Privacy at The Data Level

All the scenarios mentioned above involve Web service operation invocation which takes data as input and provides other data as output. For example, a requester may ask for the Geneomic data of patients who have a specific employer, then he asks for demographic information of those patients. The request first gets directed to a lab WS with $hasGene$ as $D_i(input)$ and $hasDiagnosis$ as $D_i(output)$. It then goes to a demographics WS with $hasDiagnosis$ as $D_i(output)$ and $hasName$ as $D_i(input)$. Considering the analysis pipeline scenario , the following Web services are involved in the analysis process:

$$WS_1 = (Symbol, \{Location, OMIM, Symbol\}, getGeneInfo())$$

$$WS_1 = (rs, \{rs, location, gene\}, getSNPInfo())$$

$$WS_1 = (trait, \{trait, rsno., gene, location\}, getGenPhenAssoc())$$

$$WS_2 = (\{Drug, Gene, SNP, CoMedication\}, getDrugInfo())$$

$$WS_3 = (\{HGVS, Disease, OMIM, Date, Treatment, LabResult\}, getDiagnosis())$$

$$WS_4 = (\{Dosage, Drug, GeneDrugAsso, GenPhenAsso\}, getPharmInfo())$$

If we abstract away the operation names and input, output parameters, and focus on the data items that flow between the different services, we get the following queries:

$$Q_1 = (Research, \{Location, OMIM, Symbol\})$$

$$Q_2 = (Research, \{rsno., location, gene\})$$

$$Q_3 = (Research, \{Trait, rsno., gene, location\})$$

$$Q_4 = (Marketing, \{Drug, Gene, SNP, CoMedication\},)$$

$$Q_5 = (Diagnosis, \{HGVS, Disease, OMIM, VisitDate, Treatment, LabResult\}$$

$$Q_6 = (Marketing, \{Dosage, Drug, GeneDrugAsso, GenPhenAsso\})$$

3.2 Motivating Scenario

Data collection. Assume that Alice is a recently diagnosed 'cancer' patient, who uses a wearable device to monitor her status by instantly collecting her data and publishing it to a clinical service (Fig. 1.1, A). Alice also visits a Genomic lab (Fig. 1.1, Org2), which might also interact with her clinic. Alice conducts most of her purchases online using various devices (smart phone, laptop, etc) (Fig. 1.1,B) via an online service provider (Fig. 1.1, Org3) that is a composite service that dynamically discovers and selects its providers (e.g., a pharmacy service and a payment service) from a pool of competing services. Other organizations that might have Alice's data include research institutes, insurance companies, census databases, etc.. Thus, each of these organizations has a subset of Alice's data.

Listing 3.1: An ontology instance of a patient from our scenario with no context detected

```
<rdf:RDF xmlns:mc="http://www.michcare.com/MC.owl#">
<mc:Patient rdf:about="patient1">
  <mc:hasName>George</mc:hasName>
    <mc:hasNamePreference>
      <mc:NamePreference>
        <mc:canDisclose>No</mc:canDisclose>
        <mc:hasPurpose>None</mc:hasPurpose>
      </mc:NamePreference>
    </hasNamePreference>
 <mc:hasStatus>Bad</mc:hasStatus>
 <mc:hasStatusPreference>
      <mc:StatusPreference>
        <mc:canDisclose>yes</mc:canDisclose>
        <mc:hasPurpose>Research</mc:hasPurpose>
      </mc:StatusPreference>
    </hasStatusPreference>
</rdf>
```

Listing 3.2: A SPARQL query corresponding to the request in Listing 3.2

```
PREFIX mc: <http://192.168.0.8:9763/com.medi.sample.webapp-1.0.0/
    generatedpatient.owl#>
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?s ?l
WHERE {
?patient rdf:type mc:Patient .
?patient mc:hasLocation ?l.
?patient mc:patientStatus ?s.
?patient mc:patientStatus 'Bad' .
?patient mc:canReleaseStatus 'yes'}
```

Private Data Publishing. Each of the above mentioned organizations expose data as a Web service WS_i using a set of operations $Op_1, ...Op_j$ that get invoked upon Web service usage. For example, a physician can inquire about a patient's status (Fig. 1.1,C) and a researcher can use a Genetic Variation Detection (GVD) analysis pipeline (Fig. 1.1, D). Regardless of the data access technique used by a user, a query Q_j gets submitted to WS_i, which searches it's ontology-based repository and returns a set of data type properties D_j to the user. Assume that the set of all data type properties that a requester can search for are defined in a generic ontology. The ontology has a taxonomy for purposes $\boldsymbol{P}$ (e.g., $P = \{Research, Diagnosis, Marketing\}$) and another for data type properties $\boldsymbol{D}$ that can be obtained from several data repositories (e.g., Genomic, Drug, Pharmacy, Clinicial, and demographic). For simplicity, we refer to these properties as $\boldsymbol{D} = \{PS, D, DD, L, G, DE, GS, SNP, DI, DL, PR, A, AD, EM, IN, MS\}$ (Table 1.2).

To manage data privacy, WS_i defines a privacy policy for each patient instance i_j in it's repository, including Alice. Together with every instance, WS_i records patients predefined disclosure preferences over each data type property d_k in i_j, including whether d_k should be disclosed, and the purpose of disclosure P_j. For example, Alice's RDF file under a clinical service repository indicates that she has chosen to disclose the $hasPatientStatus$ data type property for $Diagnosis$ purposes (Fig. 1.1, i1). The usage governance of $hasPatientStatus$ is defined as a rule in Alice's policy file (written in ALFA [1]) (Fig. 1.1, p).

Adversarial Scenario. Assume that an adversary aims at linking genomic data of a victim with

Table 3.1: A subset of the Web services in CP

Service	**description**
WS1.1(?L,?OMIM,?GS)	returns gene location, OMIM, and gene symbol for all genes
WS1.2 ($RS,?RS,?L,?GS)	returns gene symbol, location, and RS no. for genes with "RS"
WS1.3($TR,?TR,?RS,?GS,?L)	returns trait, RS no., gene symbol, and gene location for a trait "TR"
WS2($GS,?DR,?SNP,?CM)	returns drug, SNP, and co medication associated with gene $GS
WS3($DI,?HGVS,?DI,?OMIM,?T)	returns HGVS, Disease, OMIM, and Treatment for a disease DI.
WS4(?DO,?DR,?GDA,?GPA)	returns dosage,drug,gene drug and genotype phenotype associations.
WS5($DI,?DR)	returns medicine for disease "DI"
WS6($DR,?A,?G)	returns Age and gender for patients who are covered for medicine "DR"
WS7.1($DR,?Z,?I)	returns zip and income for patients who ordered medicine "DR"
WS7.2($DR,?P)	returns price for medicine "DR"
WS8($DR,?N)	returns DNA for patients who has subscriptions for medicine "DR"
WS9($Z,?P)	returns price "p" for orders from patients who live in zip "Z"

other clinical, pharmacy, and demographic data. He may ask for *"the ages, genders, zip, genomic data, and income for cancer patients"*. He can also use his background knowledge. For instance, if he knows that *Actiq* is one of the very few cancer medications that costs $45.89, he can craft a query to deduce that a patient has been diagnosed for cancer if he knows how much he paid for a medicine that is used for cancer. He submits a query to know *"the price ranges paid for certain medicines by patients who reside in a certain area"*. Alternatively, he asks for *"Gene variation data of patients within an area and the genotype phynotype associations"* to get to a certain disease then link that with other demographic data. To accomplish his task he submits queries (e.g., Q_1 and Q_2 in Fig. 1.1, E) or invokes operations on several data services (Fig 1.1, Org2, Org3, ...) and analysis pipelines (Fig 1.1, Org4). A subset of the services that might be involved are shown in Table 3.1.

Several composition plans $CP = \{CP_1, ..., CP_n\}$ can be returned by the execution engine to answer the adversary queries. Fig. 4.4 depicts one possible CP. The CP takes a query (e.g., $Q = \{A, G, Z, N, I\}$) and combines the results and returns the output. The data item s (e.g. medicine) is an input parameter for WS_6, WS_7 and WS_8, and an output for WS_5. Thus, WS_6,

WS_7 and WS_8 depend on WS_5 for providing DR. Thus, before providing WS_6 with DR, WS_5 should check the data owner's privacy policy rule regarding that item. Checking for privacy rules should hold along the composition chain until all the data items requested in the original query initiated by the user are retrieved. The privacy management engine should validate CP in terms of privacy, and if at least one dependency in CP violates a privacy rule, then CP should be discarded from CP and the corresponding data item should be discarded from the query results.

Table 3.2: Sample queries from our adversarial scenario

Q_j	P_j	D_j
1	Research	**PS**,DD,SNP,**DI**,**A**,IN
2	Research	**D**,**L**,G,DE,**DI**,DL,PR,**A**,EM,IN,**MS**
3	Research	**L**,**DI**,**MS**
4	Marketing	D
5	Research	D
6	Research	PS
7	Diagnosis	G
8	Research	PS,D,G
9	Diagnosis	PS,D
10	Diagnosis	**PS**,**D**,**L**,**MS**

3.3 Why Context Matters?

For illustration purposes we abstract away the operation names, input (denoted by $), and output (denoted by ?) parameters in Table 3.1, and represent the data type properties that flow between the different services as queries. Table 3.2 shows a subset of those queries. Without context information, the data type properties requested by the adversary will be disclosed as long as the static privacy policy rules tied to those properties match the purpose of usage. For example, upon receiving Q_{10} (Table 3.2), the Web service looks up its repository for a set of matching instances I_{10}=$\{i_{101}, i_{102}, i_{103}, ..., i_{10k}\}$ (including Alice). Then, for each instance in I_{10} it checks the policy rules tied to each of the data type properties in Q_{10}. In the case of Alice, the $hasPatientStatus$ property can be disclosed for the purpose indicated in Q_{10}. Thus, the adversary can continue to submit queries asking for more data type properties by repeating some of these properties in subsequent queries, with the goal of linking them to other properties available from other data sets (e.g. $hasAddress$). For example, in Q_1 he asks for D_1=$\{$**PS**, DD, SNP, **DI**, **A**, $IN\}$ and indicates the purpose as P_1=$\{Research\}$. Then, in Q_2 he repeats some of the data type properties as well as the purpose (D_2=$\{$**D**, **L**, G, DE, **DI**, DL, PR, **A**, EM, IN, **MS**$\}$ and indicates the

purpose as P_2=$\{Research\}$). In Q_3 he repeats some of the data type properties in the previous queries as well as the purpose (D_3=$\{$**L**, **DI**, **MS**$\}$, P_3=$\{Research\}$). Later, in Q_{10} he repeats the data type properties in the previous queries but with a different purpose (D_{10}=$\{$**PS**, **D**, **L**, **MS**$\}$, P_{10}=$\{Diagnosis\}$).

To avoid the above scenario, we need to make WS_i *context-sensitive* by tracking the query history against all participating Web services, including each subquery against each sub service, and use that history to infer context. A PEP at the composition orchestrator level as well as individual PEPs at component service levels can then help propagate that query history information. Each component service may be in itself a composite service. Since the inferred context is tied to the data resource, and since a data source can be shared by multiple services our approach works regardless of which Web services participated in a CP.

Our definition of context is inspired by the above mentioned adversarial query behavior. First, our intuition is that a learning engine can be beneficial for the Web service by inferring the context of a query based on previous queries to make the inferred context a function of prior contexts. The Bayesian learning and updating mechanism helps achieve this capability. Second, we argue that an adversary is likely to submit several queries asking for data items that are not in themselves sensitive but may reveal, when combined, information about predetermined sensitive data items. Conditional Entropy helps achieve this goal. Third, we argue that an adversary may ask for data in subsequent phases seeking more sensitive data in each phase. In an initial phase, the requester may submit initial exploratory queries that do not explicitly ask for sensitive data. The purpose of those queries is to get an overall view of the data. For example, an adversary can first ask for $hasPatientStatus$ of all patients that are within some $hasAge$ range for $Research$ purposes. In later phases, he may look for patients who have been diagnosed for a certain disease for $Diagnosis$ purpose. To this effect, he changes the purpose of the query. Therefore, in each phase some query attributes are expected to change abnormally. So, we use Data Diversity as an indicator of the difficulty in identifying sensitive data. We combine, in a novel way, existing techniques from the fields of probability theory and information theory. We Next, we present our solution by defining the system architecture, our notion of context, and how we incorporate it into dynamic privacy management.

3.4 Architecture

We build a dynamic, semantic-based privacy policy management framework on the top of the XACML reference architecture for policy-based access control (Fig. 3.1). According to a standard XACML engine, whenever a user submits a query, the query first goes to the *PEP*, which wraps it into an XACML request and forwards the request to the *PDP*, which communicates with the *PIP* to fetch the required attributes. For example, Listing 2.2 shows the XACML request that corresponds to Q_6 (Table 3.2).

Listing 3.3: An XACML request corresponding to Q_6

```
<Request>
<Attributes Category="resource">
<Attribute AttributeId="resource-id">
<AttributeValue>patientStatus</AttributeValue>
</Attribute>
<Attribute AttributeId="mc:canReleaseStatus">
<AttributeValue>yes</AttributeValue>
</Attributes>
<Attributes Category="access-subject">
<Attribute AttributeId="subject-purpose">
<AttributeValue>Diagnosis</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="action">
<Attribute AttributeId="action-id">
<AttributeValue>view</AttributeValue></Attribute>
</Attributes>
</Request>
```

System components. In our system, the *PIP* communicates with the *Semantic Handler (SH)*, which looks up the required attributes in the service's repository (More details in Section. 3.5.3). The PDP then uses the attribute values to evaluate the request. If a permit decision is returned, the PDP consults the semantic handler (via the PIP) for previously recorded context of the matching data instances. The retrieved context is considered as a resource bag of context elements. The PDP

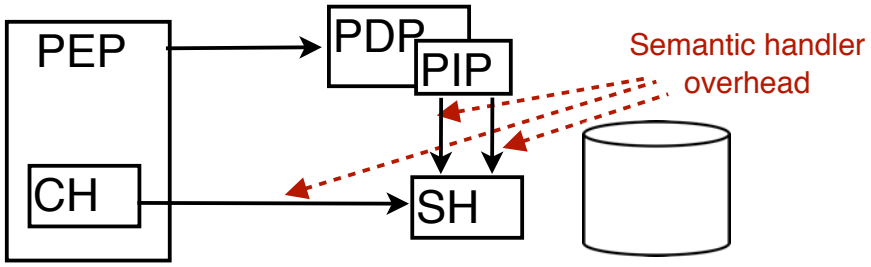


Figure 3.1: Dynamic Semantic-based Privacy Management.

then wraps the context bag as an XACML obligation element and sends it over to the PEP together with the obligation logic to be performed (i.e., handle context).

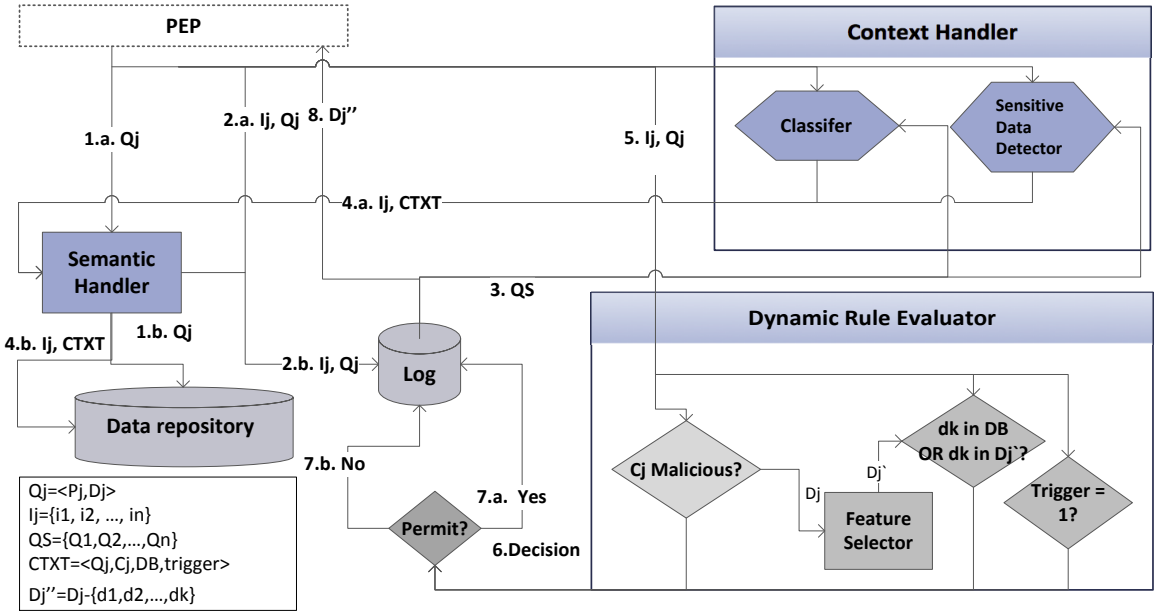


Figure 3.2: Approach.

The PEP uses the obligation to perform further check by communicating with the *Semantic Handler* (Fig. 3.2, 1). The Semantic handler passes the set of instances I_j that match the query together with the query Q_j to the *Context Handler* (Fig. 3.2, 2). The context handler consists of two sub components: the *Classifier*, which dynamically classifies a query as being potentially $malicious$ or $legitimate$, and the *Sensitive Data Detector*, which dynamically determines the subset of data type properties in a query that could potentially be sensitive (More details in Section. 3.5.2). The PEP uses the context $CTXT$ (inferred by the context handler) to update the context of each instance i_k in the set I_j that matches Q_j via the *Semantic Handler* (Fig. 3.2, 4). The PEP then uses $CTXT$ to make the final decision through the *Dynamic Rule Evaluator*

(Fig. 3.2, 5) (More details in Section 3.5.4). Finally, the PEP then sends the final response back to the requester.

3.5 Formal Model Definition

3.5.1 Dynamic Privacy Policy Rule Definition.

We define a privacy policy of an instance as a set of rules. A privacy rule consists of the following:

- Property: the data type property of a data owner for which the rule is defined.
- Condition: the condition(s) that must be satisfied for a data type property before access can be granted. A condition is expressed using concepts and relationships from the ontology. Namely, the patient preference regarding the disclosure of a data type property. A patient Preference includes the following: Disclosure, which is a boolean property $disclosure$ indicating whether a data type property d_k can be disclosed. Purpose, which limits the purpose of usage P_i for d_k.
- Obligation: We define obligation elements for each policy rule to be dynamically fulfilled only when a rule evaluation yields a "permit" decision. PDP uses Obligations to tell the PEP to not only rely on statically defined rules by the data owner, but to further apply the context inference algorithm.

Context: The inferred context consists of previous requester's behavior, including previous queries (Q_j), previous classifications (C_j), previous relatively sensitive sets (D_B), and previous diversity triggers.

- Q_j: the submitted query. This query is fed into the context handler to infer the next context.
- C_j: is the query classification. This item helps to check if a data type property d_j resulted in a malicious classification of the submitted query.
- D_B: this is the detected set of sensitive data type properties relative to the requested set D_j. It helps to check if any of the data type properties d_j in D_j is included in D_B.
- trigger: this helps to check if d_j caused irregularity in the query diversity values. In summary:

$$Policy = (P_{id}, Rule_1, Rule_2, ..., Rule_n)$$

$$Rule = (Property, Condition, Obligation)$$

$$Condition = (disclosure, purpose)$$

$$Obligation = (O_{id}, Context_1, Context_2, ..., Context_n)$$

$$Context = (Q_j, C_j, D_B, trigger)$$

3.5.2 Context Handling

We express a query Q_j as a tuple $\langle P_j, D_j \rangle$ that consists of the purpose P_j and the set D_j (Table 3.3). We represent P_j by a numerical value, and represent the set D_j by a vector of binary values, where 1 indicates that d_k appears in the query and 0 indicates that it does not.

Table 3.3: Queries from Table 3.2 as tuples.

Qj	**Pj**	**Dj**															
		PS	**D**	**DD**	**L**	**G**	**DE**	**GS**	**SNP**	**DI**	**DL**	**PR**	**A**	**AD**	**EM**	**IN**	**MS**
1	1.0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	1	0
2	1.0	0	1	0	1	1	1	0	0	1	1	1	1	0	1	1	1
3	1.0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
4	3.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	2.0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
8	1.0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
9	2.0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	2.0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1

Algorithm 1 summarizes our context inference algorithm. We break our solution to context inference into the following two sub problems:

Query Classification. For classifying queries we use the Naive Bayesian learning (NBL) algorithm. The input to the learning algorithm is the query space QS and the output is a classification C_j. We assume that the presence of one data type property in a classification is conditionally independent of another data type property. We also assume that the data type properties asked for in a query are dependent on a query's purpose. Based on that, we construct a *Naive Bayesian Classification* model by converting a query Q_i into a *Bayesian Network*, where the root node represents a

query's purpose P_i and the children represent data type properties $d_1, ..., d_k$. In the rest of this thesis, we treat the data type properties in a query as a set D_i. So, based on our definition, $Pr(D_i|C_j)$ is equivalent to $Pr(d1, d2, .., d_k|C_j)$ which is equivalent to $Pr(d_1|C_j)Pr(d_2|C_j)...Pr(d_k|C_j)$. For each query Q_i the learning algorithm is given the purpose P_i and the set of data type properties D_i. The parameters to be estimated are the purpose probabilities $Pr(P_i)$ and the conditional probabilities $Pr(D_i|P_i)$. Therefore, to predict the class label C_{j+1} for a newly submitted query Q_{i+1} with purpose P_{i+1} and set of data type properties D_{i+1}, the NBL algorithm computes $Pr(P_{i+1}, D_{i+1}|C_j)Pr(C_j)$, for $j = \{malicious, legitimate\}$ based on the estimated parameters from the training data.

Sensitive Data Detection. Our goal for sensitive data detection is to determine the set of data type properties in a query that could potentially be sensitive, even though those properties have not been deemed sensitive at the time of data collection. This problem reduces to two sub-problems:

Relative Sensitivity of a Set of Data Type Properties: Algorithm 2 summarizes the relative sensitivity algorithm. We apply conditional entropy to measure the relative sensitivity of a set of data type properties D_i that is asked for in a newly submitted query with respect to two things. First, users are often asked to make privacy decisions regarding their sensitive data (e.g., $Name$) at the time of data collection. Let D_A be the set of predetermined sensitive data type properties. We apply conditional entropy to measure the relative sensitivity of D_i with respect to D_A. Second, we measure the relative sensitivity of D_i with respect to all sets of data type properties $D_1, ..., D_k$ in the previously submitted queries in QS. In both cases, we use the notion of information gain as a measure of the mutual information between two random variables. We define the information gain $I(D_A, D_i)$ for D_i with respect to D_A, as the reduction in uncertainty about the value of D_A

Algorithm 1 Context Inference Algorithm

1: Input: QS, Q_j
2: Output: $CTXT$
3: $C_j \leftarrow$ QUERYCLASSIFICATION(QS, Q_j)
4: $D_B \leftarrow$ RELATIVESENSITIVITY(QS, Q_j, t, D_A)
5: **if** $i \mod M$ equals 0 **then**
6: $\quad trigger \leftarrow$ QUERYDIVERSITY(QS, t, M)
7: **end if**
8: $CTXT \leftarrow C_j \cup D_B \cup trigger$

when the requester knows the value of D_i. Formally:

$$I(D_A, D_i) = H(D_A) - H(D_A|D_i)$$
$$H(D_A) = -\sum_{d_a \in D_A} Pr(d_a) log_2 Pr(d_a)$$
$$H(D_A|D_i) = -\sum_{d_a, d_i \in D_A, D_i} Pr(d_a, d_i) log_2 Pr(d_a|d_i)$$

We apply the same formulas above to measure $I(D_k, D_i)$. The relative sensitivity algorithm first computes the information gain between the sets D_i and D_A. It then computes the information gain between D_i and each set of data type properties $D_1, ..., D_k$ in the set of previously submitted queries QS. If either case results in an information gain that is higher than a threshold t, the algorithm distills the data type properties in D_i that caused the highest information gain (D_{sig}). The resulting data type properties (D_{sig}) are then added to the subset of relatively sensitive data D_B.

Algorithm 2 Relative Sensitivity Algorithm

```
input: QS, Q_j, t, D_A
output: D_B
D_B ← φ
if I(D_j, D_A) ≥ t then
    D_sig ← SIGNIFICANTSUBSET(D_j, D_A)
    D_B ← D_B ∪ D_sig
end if
for each Q_k in QS do
    if I(D_j, D_k) ≥ t then
        D_sig ← SIGNIFICANTSUBSET(D_j, D_k)
        D_B ← D_B ∪ D_sig
    end if
end for
```

To get the most significant subset D_{sig}, we iterated through each set, and moved one element at a tiem from each set. We then calculated the gain of the new sets and observed the difference from the principal gain. We did this step for all elements in each set and observed which element(s) (when removed) affected the gain the most. This metric was justified since information gain ignores the population size, so it is valid to directly compare the information gain computed from a vector of n elements with another of $n - 1$ elements. Other methods that could be applied to find the most interacting subsets between D_i and D_A or D_k. For example, we could calculate the Singular

Vector Decomposition (SVD) of the conditional probability matrix (*CPM*) of D_i and D_k or D_A, respectively. The singular vectors with the largest values indicate which subset of D_i and D_k or D_A interact the most with each other.

Table 3.4 shows an example of applying the relative sensitivity algorithm to a set of queries from three iterations. In the in the 3rd iteration, the query {`Marketing, yes, yes, yes, yes, yes, yes, yes, yes, no, no, yes, yes, no, yes, no`} was indicated as being related to both D_A and to the set of data type properties in the query {`Marketing, yes, no, yes, no, yes, yes, no, yes, no, no, yes, yes, no, yes, no`} that was submitted previously in the same iteration. In the 4th iteration, the query {`Diagnosis, yes, no, yes, no, yes, no, yes, yes, yes, yes, no, no, no, no, no`} was indicated as being related to the query {`Research, yes, yes, yes, yes, no, no, no, no, no, no, no, no, no, no, no`} from iteration 3.

Data diversity for a set of queries. The diversity detection part of the sensitive data detector provides an extra check for data sensitivity. The query diversity algorithm calculates the entropy for each criterion in a and creates a map of entropy values for each phase. It then uses the resulting phase diversity map to monitor the change in diversity between phases by comparing the change to a threshold t. The query diversity algorithm takes the query space QS, the threshold t, and the number of queries M to consider in each phase as inputs and returns a boolean value to the context inference algorithm indicating whether there is an attempt to breach sensitive data in the recently submitted query.

Algorithm 3 summarizes the query diversity algorithm. We use the notion of joint entropy as a measure of data diversity. The diversity of a set of data depends on the number of homogeneous groups of data and the proportion of attributes in each group. The data set in our case is a set of submitted queries QS. Our desired metric shares some properties that Shannon sought in his measure of information uncertainty [89]. First, if there are multiple possible options which are equally likely, there is more uncertainty. Thus, the smaller the entropy, the fewer the number of different queries or the more regular the queries are. Second, if a data set is defined as the

combination of several disjoint data sets, the entropy for them combined should be at least the weighted sum of the individual entropy values for the individual sets. In our case, for a query set QS composed of query subsets QS_I and QS_{II} submitted in two phases, the overall entropy should be higher, or at least equal to the weighted entropy of the query sets involved. Formally:

$$H(QS) = H(x, y) + xH(QS_I) + yH(QS_{II}),\ \text{s.t}\ x = \frac{QS_I}{QS}, y = \frac{QS_{II}}{QS}$$

We use the above formula to measure the change in diversity among a set of queries by determining the constant and varying attributes of QS assuming all queries are submitted by the same source. The attributes in our case are the purpose P_i and the set D_i. In each phase either of these attributes is expected to change abnormally. To measure this change we track the entropy for both query attributes. Formally, for a query set QS we calculate the entropy H for each group of homogeneous queries. We determine the homogeneity of a group of queries based on the following criteria:

- The purpose regardless of the data (P);
- The data given a purpose ($D|P$);
- Both attributes combined (PD).

Algorithm 3 Query Diversity Algorithm

```
input QS, t, M
output: trigger
QS_selected ← φ, QS_previous ← φ, count ← 0, phase ← 1
while count < size(QS) do
    for each Q_k in QS such that k ≤ M do
        QS_selected ← QS_selected ∪ Q_k
        k ← k + 1
    end for
    Calculate diversity maps H_P, H_PD, H_P|D
    Update phase diversity map H_phase
    Update QS_previous
    phase ← phase + 1
    count ← count + M
end while
if DIVERSITYCHANGEDETECTION(H_phase) ≥ t then
    trigger ← 1
end if
```

We calculate the entropy for each criteria *a* as:

$$H_a(QS) = -\sum_{i=0}^{n} Pr(a_i) log_2 Pr(a_i) \qquad \text{where } a = \{P, D|P, PD\}$$

3.5.3 Semantic Handling

For dynamic rule evaluation the semantic handler gets invoked in two cases.

Updating Instance Context. The semantic handler interprets the submitted query Q_j as a SPARQL query and runs it be the RDF repository. If Q_i matches a set of instances I_j, the semantic handler keeps a log of I_j (Fig. 3.2). After inferring the context, the Web service PEP updates the context block of each matching instance in it's repository (Fig. 3.2).

Checking updated context. If a query Q_i matches an instance, the PDP first checks the policy rules that govern each of the data type properties in D_i of that instance to see if the purpose P_i of the query matches the purpose indicated in each rule. If any of the data type properties d_i in D_i does not match any of the rule conditions, a *"deny"* response is returned and the corresponding data type property will not be disclosed. However, if a *"permit"* response is returned, the PDP consults the semantic handler (via the PIP) to retrieve the contents of the $hasContext$ data type property of each matching instance i_k in I_j. The result is returned as a bag of context elements that are added as attribute assignments in the obligations stated in the policy rule. The PDP sends the response together with the obligations over to the PEP. The PEP performs the obligation by iterating through the context block of each of the data type properties in each of the matching instances and checking the components of a context block. For example, suppose that the set of instances that match a query Q_1 is denoted by $I_1 = i11, i1n$. If we take instance $i11 =< d1, CTXT1, d2, .., dn, CTXTn >$ as an example, it consists of several data type properties each of which has a set of contexts. For example, the context set associated with $d1$ is $CTXT1 = ctxt1, .., ctxtn$, where $ctxt1 = (Q1, C1, DB1, trigger1)$ and $ctxt2 = (Q2, C2, DB2, trigger2)$, etc.. So if a newly submitted query Q_2 asks for $d1(i.e., Q2 =< P2, D2 = d1, d2, , dn >$, the algorithm checks the previously submitted queries $Q1, , Qn$, that also asked for d1, which are stored in the context block CTXT1 of d1, and that is done for every dk that appears in a newly submitted query.

First, the PEP checks if any of the data type properties d_i in D_i is included in a previously detected relatively sensitive data set D_B. It then checks if d_i has resulted in the classification of a malicious query (C_i). Finally, it checks if d_i caused irregularity in the query diversity values (trigger). If at least one match is found, the PEP rejects the disclosure of d_i. For example, the policy in Figure 1.1 permits the disclosure of the $PatientStatus$ property for the purpose stated by the requester. However, a previously inferred context indicates that $PatientStatus$ has been marked to be potentially sensitive as indicated in the DB part. Thus, the PEP denies access to $PatientStatus$.

3.5.4 Dynamic Rule Evaluation

In this section, we explain how the inferred context can be used to achieve dynamicity at the rule level. After receiving the obligations from the PDP, the PEP uses the set I_j and the query Q_j to perform *Dynamic Rule Evaluation* (Fig. 3.2, 5). Algorithm 4 explains the dynamic rule evaluation algorithm. First, if the query is classified as malicious, a feature selection is used to filter out the subset of data type properties D'_j in the query that resulted in such a classification. Second, a check is made to see if any of the data type properties d_k of D_i is included in a previously detected relatively sensitive data set D_B. Finally, the diversity trigger is checked to determine if there has been irregularity in the query sets due to the newly submitted query. If any data type property d_k requested in the query is either sensitive, relatively sensitive, resulted in a malicious classification, or caused irregularity in the query set, the set D_j in the query Q_j is revised to exclude d_k and the new data set D''_j is returned to the requester (Fig. 3.2, 8).

3.6 Prototype Implementation

In developing our privacy management framework, we took into account the technical implications that we need to be aware of in Services based environments. Namely, Web browsing entails that privacy protection need to be performed while the individual is on-line or when a developer is using a Web service based API, etc. Also, we were aware of existing standards such as privacy policy languages, etc. Therefore, we developed an XACML-compliant Web sites driven by PEP-enabled Web services as well as end user analysis tools that are compliant with those Web site policies and users privacy preferences. This way those sites can advise the user if there are potential privacy issues.

Algorithm 4 Dynamic Rule Evaluation Algorithm

```
1: Input: Q_j, I
2: Output: Q'_j
3: for each I_k in I do
4:     for each R_k in I_k do
5:         if P_k not equals P_j then
6:             D_j ← D_j \ d_k
7:         end if
8:         for each CTXT_k in CTXT do
9:             if C_k equals malicious then
10:                D_Selected ← FEATURESELECTION(D_j)
11:                for each d_s in D_Selected do
12:                    D'_j ← D_j \ d_s
13:                end for
14:            end if
15:            for each d_b in D_B do
16:                D'_j ← D_j \ d_b
17:            end for
18:        end for
19:    end for
20: end for
```

The privacy management framework can be developed in several ways. It can be developed as a downloadable plug-in for all service providers. So, an organization does not need to implement its own privacy management component. Alternatively, it can be implemented as yet another cloud service that is operated by an independent organization.

3.6.1 Extending XACML PEP

We implemented the proposed solution in Java (Fig. 3.3) with the following main components:

Context handler: We implemented the classifier component using the Weka API [44] and the query diversity and relative sensitivity components using the JavaMI API [24]. For relative sensitivity, we used an implementation of the Chi-Squared test [74] to measure the significance of the mutual information between two sets of data type properties with an alpha level of 0.05. For query diversity, we chose an M value of 5 (as detailed in Algorithm 3).

PEP: We used the WSO2 Identity Server 4.5 (WSO2 IS) [7] as our XACML engine. WSO2 IS acts as combination of PDP , PAP and PIP components. The PIP uses an LDAP-based user store embedded with the server. We implemented two forms of request handling in our PEP client:

- In the first implementation our PEP uses the Content element in XACML to send RDF data directly to the XACML engine (PDP) together with the request and the PDP takes

care of it. The PEP adds the RDF data retrieved from running the SPARQL query under the content element of the request and indicates the requested data as a resource attribute using XPath expressions. Finally, the PEP sends the request over to WSO2 PDP, who does policy evaluation by looking at the content element. The PDP extracts data from the content element using XPath. For example it can extract the hasGene data from content element. In this case we used AttributeSelector elements to define the policies. Figures. 2.3 and 2.4 show a sample policy and it's corresponding request.

- In the second implementation our PEP parses the RDF content of the matching instances and extracts the values and sends them over to PDP. Therefore, we used AttributeDesignators in XACML policy definitions to refer to data within the RDF files since the PDP can not parse an RDF content. In this case, we designed our PEP based on direct attribute matching, where the PEP uses an RDF parser for fetching data from the RDF file and then puts the desired attributes in the XACML request using the PEP client code. Figures 2.5 and 2.6 show a sample policy and it's corresponding request.

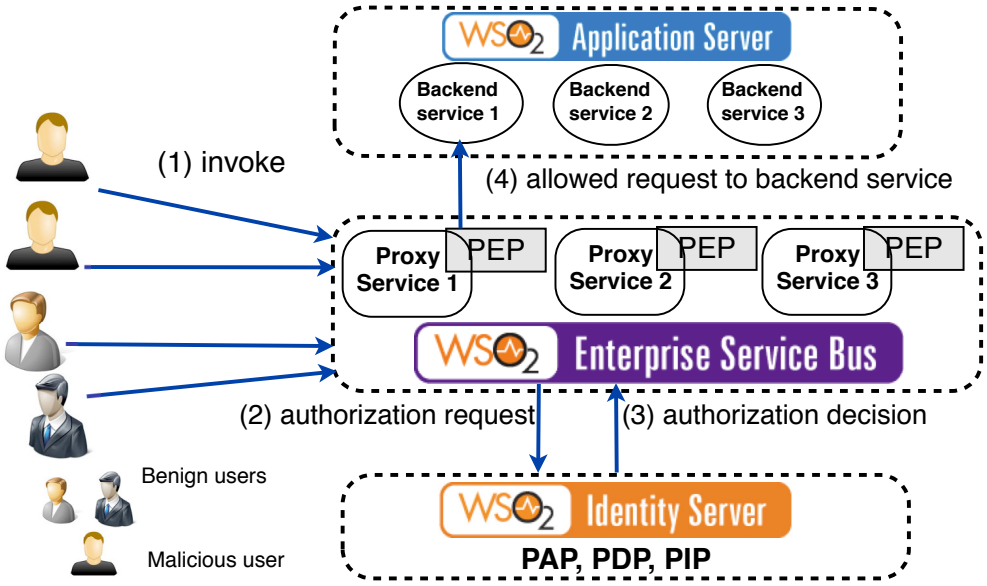


Figure 3.3: WSO2 based implementation of our framework

In both cases, the PEP creates a XACML request with the set of attributes of one or more category (subject, resource, action, environment). For example, we defined policies whose target will check if resource attribute hasGene can be released and then there will be a permit effect rule which will check if resource attribute "canReleaseGene" is yes. In our implementation the

PDP makes a decision based on the XACML policies stored for each instance, which also take the updated context into account and the PEP enforces the final decision. Based on that final decision the PEP calls the appropriate Web service operation, which in turn generates a SPARQL query against an RDF file.

Web Portal: We built a portal for health data inquiry. Since the PDP functionality of the WSO2 IS is exposed as a web service, our Portal functioned as a PEP. We used the WSO2 PEP agent library that provides a client side API to communicate with the WSO2 IS PDP. For request handling, we used `AttributeDesignators` in XACML policy definitions to refer to data within the RDF files since this technique is faster and more reusable than using `AttributeSelectors` with XPath expressions. The former technique is based on direct attribute matching, where the PEP uses an RDF parser for fetching data from the RDF file and then puts the desired attributes in the XACML request.

Web Services: We implemented five Web services which expose a set of operations to retrieve the patient's data. Each service provides an end point to query data. We used the WSO2 Application Server 4.1 to host the Web services and the portal.

3.6.2 MobiDyc: Private Mobile Based Cloud Based Framework

With the emergence of the Participatory Sensing paradigm [33], and the widespread use of mobile phones, users can now share their data, e.g., health data. The success of cloud service models increased the adoption of mobile healthcare applications [4, 5], which instantly record and analyze patients data. Mobile applications collect data from ubiquitous devices and combine it with other data about users for different purposes. Atomically, these data sources may not reveal personally identifiable information for individuals, but linking a number of distributed sources may lead to unintended consequences and breach of privacy. A malicious request, for instance, can benefit from combining atomic data items even if it claims a purpose that complies with a patient's privacy preference for each of the atomic items. Thus, the patient's consent and his privacy policy at the time of data collection may not be enough for data disclosure. According to governmental reports, around eight million records of patient's health data was leaked in the past few years [79]. Therefore, in order to encourage users to share their data we need to provide them

with privacy-enabled infrastructures.

This section contributes a privacy framework for mobile health care applications with support for dynamic privacy management of health data sharing. We provide an implementation of our approach that builds on top of the Google App Engine cloud platform.

Architecture We build a semantic privacy policy management framework on the top of the Google App Engine reference architecture for Mobile App development and the XACML reference architectures for policy-based access control. Fig. 3.4 illustrates the main components of our system. The data generated by the users is eventually stored and managed by the GAE data store. The data store also stores the users data and associated access policies, and the later are deployed in the *PDP*. A requester using our system uses his mobile device to query data. The application forwards the request to the *PEP* component, which forwards the request to the *PDP*, which retrieves the policies from the data store and evaluates them. In case of a *Permit* decision, a set of obligations is sent to the *PEP* for further check. The PEP then communicates with the *Semantic Handler* (**SH**), which interprets the request as a SPARQL query Q_j and looks up attributes in the service's ontology-based repository and passes the set of instances I that match the query together with the query Q_j to the *Context Handler* (**CH**). The *Context Handler* consists of two sub components. The *Classifier*, which dynamically classifies a query as being potentially $malicious$ or $legitimate$, and the *Sensitive Data Detector*, which dynamically determines the subset of data type properties in a query that could potentially be sensitive. WS_i uses the context $CTXT$ inferred by it's sub components to update the context of each instance in I. The PEP uses $CTXT$ to make the final decision by performing *Dynamic Rule Check* (**DRC**). The PEP then notifies the PDP, which looks up the updated rule context and sends the response back to the PEP.

Implementation We programmed all projects in Java using Eclipse 3.4 and instrumented it with the Google plugin for Eclipse and the Android Development Tools. We detail the main parts of our project below:

- *Context Handler:* we implemented the classifier component using the Weka API [44] and the query diversity and relative sensitivity components using the JavaMI API [24]. For relative sensitivity, we used the Chi-Squared test to measure the significance of the mutual informa-

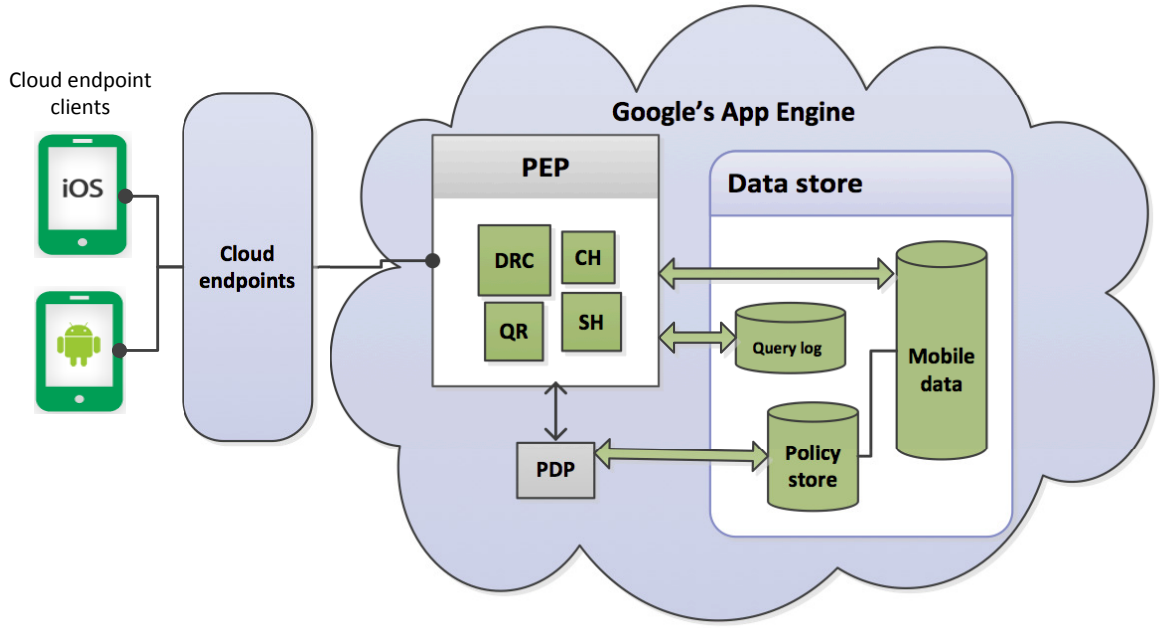


Figure 3.4: Dynamic privacy policy management solution.

tion between two sets of data type properties with an alpha level of 0.05. For query diversity, we chose an M value of 5.

- *PEP Agent* We implemented a PEP agent client using the SunXACML engine [3], and we incorporated our context handler implementation into the PEP implementation. In XACML, a policy rule does not control data retrieval. For policy enforcement, we used XACML obligations at the PEP level to ensure only desirable data type attributes are returned to the user. We define our own obligation for query rewriting (`QRObligation`) by extending the `Obligation` class. We execute an instance of `QRObligation` at the PEP in case of *permit* decisions by calling the `evaluate()` method. The method reads the rewritten query Q'_j returned by the QueryRewriting algorithm and returns the subset D'_j of the requested set D_j to the user.
- *MobiDyc* is the Android client through which the user can query data (Fig. 3.5). The application allows users to retrieve health data stored in the cloud through an Android mobile device or emulator. The client communicates with the App Engine in the background to gather stored information requested by the user. We created an object-relational mapping interface which interacts with the backend data repository to enable users to query data using their mobile devices. Each user query gets inserted to the backend data store.

- The *MobiDyc-AppEngine* is the backend project through which we provide the service in the cloud. We implemented a service that exposes patients data as operations and we deployed our services as backends to the GAE repository.

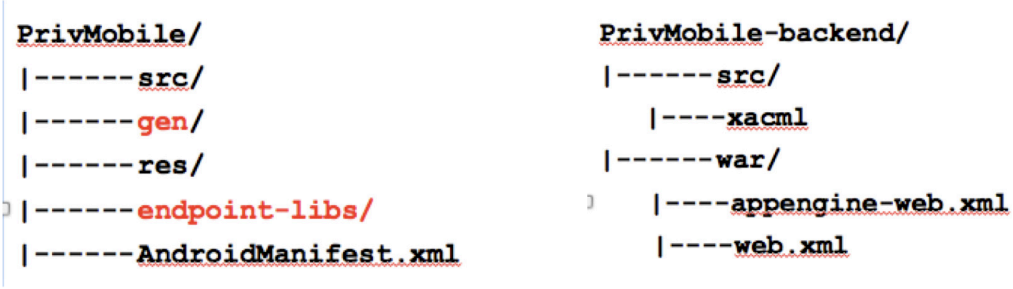


Figure 3.5: Dynamic privacy policy management solution.

Table 3.4: Results from different components for the iterations 3, 4, and 5.

	3		4		5	
Q	**Cluster**	**Rel. Sen. data detected? (DB)**	**Class**	**Rel. Sen. data detected? (DB)**	**Class**	**Rel. Sen. data detected?**
1	1	no	malicious	yes (Research, no, no, yes, yes, yes, no, no, no, no, no, no, no, no, no)	malicious	no
2	1	yes (Research, no, no, yes, yes, yes, no, no, no, no, no, no, no, no, no)	legitimate	yes (Research, no, no, yes, yes, yes, no, no, no, no, no, no, no, no, no)	legitimate	yes (Research, no, no, yes, yes, no, no, no, no, no, yes, yes, no, no, no) p¡¡0.05, IGain: 0.49
3	1	yes (Research, no, yes, yes, yes, no, no, no, no, no, no, no, no, no, no)	legitimate	yes (Research, no, yes, yes, yes, no, no, no, no, no, no, no, no, no, no)	legitimate	yes (Diagnosis, yes, no, yes, no, yes, no, yes, yes, yes, yes, no, no, no, no, no) p 0.02, IGain: 0.31
4	0	no **(conflict)**	legitimate	yes (1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0) **Highest IGain of 1.**	legitimate	yes
5	0	yes (Research, no, no, yes, yes, no, no, no, no, no, yes, yes, no, no, no)	legitimate	no	legitimate	no
6	0	no **(conflict)**	legitimate	no	legitimate	yes
7	1	no	legitimate	yes (Research, no, no, yes, yes, no, no, no, no, no, yes, yes, no, no, no)	legitimate	yes
8	1	no	legitimate	yes (Diagnosis, yes, no, yes, no, yes, no, yes, yes, yes, yes, no, no, no, no, no) p 0.02, IGain: 0.31	legitimate	yes
9	1	no	malicious	yes (Research, no, no, yes, yes, no, no, no, no, no, no, yes, yes, no, no, no) **p¡¡0.05, IGain of 0.8**	malicious	yes
10	1	yes (DA)	malicious	no **(conflict)**	malicious	yes

Table 3.5: Web services implemented in our experiment

Service	**Dinput**	**Doutput**	**Op**	**Description**
WSGenomic	hasGene	{hasGeneLocation, hasOMIM, hasGene, hasGender, hasAge}	getGeneInfo(x)	Returns detailed gene info of patients who have gene x
	hasRS	{hasRS, hasSNPLocation, hasGene, hasGender, hasAge}	getSNPInfo(x)	Returns detailed SNP info of patients who have RS x
	hasTrait	{hasTrait, hasRS, hasGene, hasRSLocation, hasGender, hasAge}	getGenPhenAssoc(x)	Returns Genotype Phenotype associations of patients who have trait x
WSDrug	hasDrug	{hasDrug, hasGene, hasSNP, hasCoMedication, hasGenderhasAge}	getDrugInfo(x)	Returns Drug info of Drug x
WSClinical	hasTrait	{hasHGVS, hasDisease, hasOMIM, hasDate, hasTreatment, hasTestResult, hasGender, hasAge}	getDiagnosis(x)	returns clinical details of trait x
WSPharma	hasDrug	{Dosage, hasDrug, hasGeneDrugAsso, hasGenPhenAsso, hasGender, hasAge}	getPharmInfo(x)	returns pharamy info about a drug x
WSClinical	{hasTrait, hasDrug, hasT}	{hasName, hasAddress, hasGender, hasAge}	getClinicalPersonalInfo(x,y,z)	
WSDemog	{hasAge, hasGender}	{hasName, hasAddress, hasGender, hasAge}	getPersonalInfo(x,y)	

3.7 Evaluation

We conducted two case studies using two XACML engines and two performance testing frameworks, as detailed in the following:

3.7.1 Case Study 1: Bio2RDF

The purpose of the first study is to evaluate the performance of our context handler and the overall performance of our PEP implementation with the context handler included.

Study Setup. In this section we discuss study setup.

Environment. We ran our tests on a 2GHz Intel Dual-Core i7, 8GB RAM, 64bit OS X 10.8 Mac machine. We used one WSO2 IS instance with a policy store which is running on the same machine. We ran Jmeter on the same machine with a memory foot print of -Xms1024m -Xmx2048m -XX:MaxPermSize=1024m. To measure scalability, we used a 2.83GHz Intel Quad-Core CPU, 8GB RAM, 32bit Windows Server 2007 machine.

Data sets. As for the request sets, we generated synthetic sets to simulate practical cases in which one data type property appears repeatedly in different requests. To generate a set of n requests we first generated k core requests, with m data type properties each, such that all properties within a single request, as well as across multiple requests are completely different. Then, we permuted the $n - k$ remaining requests from the k previous ones. For the instance sets, we created RDF files using concepts from several ontologies of the Bio2RDF project (Table 1.2), including *NCBI-Gene*,*PharmKGB*, *DrugBank*, *CDT*, and *GeneCDS* available from [22]. We created an adversarial model through a set of queries modeled on the data type properties in Table 1.2 and the purposes in Table 3.2. For the policy sets, we generated several policies using an ALFA-generated core policy template. We feed the policy generation code with the set of data type properties in Table 1.2 and the purposes in Table 3.2.

Context Handler Evaluation. We evaluated each of the context handler components, detailed as under:

Clusterer. To build our classification model we needed a set of labeled queries. Since we did not have labeled queries, we applied clustering to an initial set of queries. Since our query data consists mostly of binary attributes, and since we want to measure similarities between queries based on

the '1' value of the query attributes, hierarchical clustering is most suitable in our case. To cluster the first set of queries we applied four configurations of Agglomerative Hierarchical Clustering as implemented in Matlab: Single linkage/Jaccard Coefficient (SLJ), Average linkage/ Jaccard Coefficient (ALJ), Single linkage/Euclidean distance (SLE), and Average linkage/Euclidean distance (ALE). We used the query sets from the first three iterations to evaluate our clustering model. We first performed clustering based on the first set of queries (training set). For the validation stage we used the second set of queries (validation data) and we evaluated the clustering model based on the previous clusters. We compared the robustness of several clustering algorithms by means of an ROC curve by comparing a partition QS_c of the query set QS obtained by the clustering algorithm to a true partition QS_t labeled based on our knowledge of the queries. The ROC curve indicates that ALE clustering is the most robust (Fig. 3.6, a). Finally, we used another set of queries (test data) to perform the actual clustering based on unseen labels.

Classifier. We used the clustering results as labels to train our classification model. The results of running the NB classification model on the query sets from the third, fourth, and fifth iterations indicate that 60% of the queries in the third and fourth iterations, and 80% of the queries in the fifth iteration were labeled as malicious while the rest were labeled as legitimate.

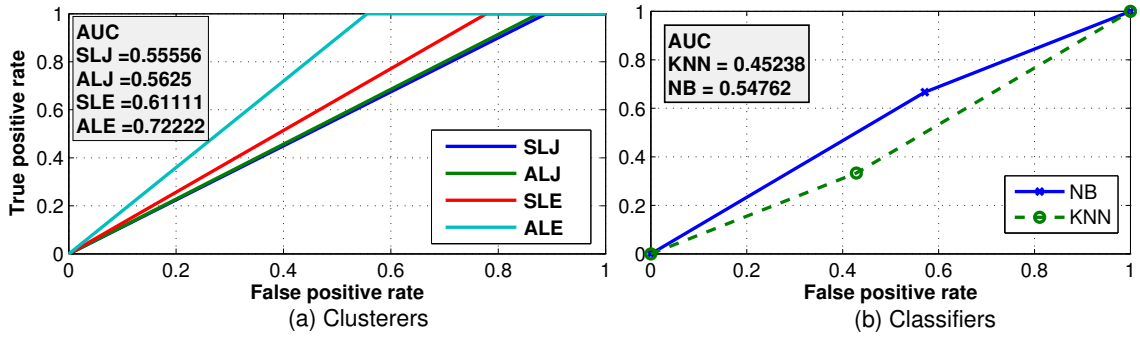


Figure 3.6: ROC curve comparison

Relative Sensitivity. The relative sensitivity results agreed with the classification results 60% of the time for the third iteration, 30% of the time for the fourth iteration, and 40% of the time for the fifth iteration. Moreover, the relative sensitivity results detected that 80% of the queries in the third iteration contained a subset of data type properties that is relatively sensitive. In the fourth iteration, only 20% of queries contained relatively sensitive data. In both cases, half of the queries

contained data that was sensitive relative to D_A while the rest contained data that was sensitive relative to some other set D_k in a previously submitted query. 40% queries in the fifth iteration contained relatively sensitive data, most of which contained data sensitive relative to D_A.

Table 3.6: Diversity handler results for first three iterations.

Iteration1					Iteration2					Iteration3				
Ph	**HP**	**HPD**	**HDIP**		**Ph**	**HP**	**HPD**	**HDIP**		**Ph**	**HP**	**HPD**	**HDIP**	
			P1	1.58				P1	0.0				P1	1.6
1	0.97	1.92	P2	0.0	3	1.5	2.32	P2	1.0				P2	0.0
			P3	0.0				P3	1.0				P3	0.0
			P1	0.0	H	2.0	3.22	P1	1.14	H	2.08	4.1	P1	1.5
2	0.0	0.72	P2	0.0				P2	1.53				P2	1.56
			P3	0.0				P3	0.13				P3	1.0
			P1	0.474		1.37	2.32	P1	0.0	6	1.44	3.32	P1	1.6
H	1.48	2.32	P2	0.86				P2	1.6				P2	0.0
			P3	0.0				P3	0.0				P3	0.0
					H	2.15	3.64	P1	1.1	H	2.08	3.28	P1	1.8
								P2	1.6				P2	2.0
								P3	1.0				P3	1.1

Diversity. Table 3.6 provides details of query diversity results for each phase of the third, fourth, and fifth iterations. The table shows both entropy values per phase and diversity changes between phases. We focus on the cases where the entropy values are 0 which suggest that all queries had the same value for an attribute and the cases where the entropy is 1 which suggest that the queries had equal number of each attribute value. For example, entropy values for the purpose attribute (**HP**) in the second phase of the third iteration match the relative sensitivity results for the last four queries of the third iteration. The diversity results indicate that in the third iteration, the purposes of the submitted queries were equally likely for phase one while in phase two all submitted queries were for the same purpose. The diversity in purposes increased until phase 4 when it remained around 2. For the purpose and data combined (**HPD**), the results did not indicate interesting entropy values or significant increase or decrease in diversity. A closer look at the entropy results for the data per purpose (**HPID**) suggest that for most of the purposes in the third iteration the entropy per phase is 0 and so is the diversity which indicate similarities among queries in the initial phases which match the results from the other components.

PEP Evaluation.

Jmeter Test Plan. We used Apache Jmeter 2.8 and we configured it to simulate 100 concurrent

	3		4		5	
Q	**Cluster**	**Rel. Sen. data detected? (DB)**	**Class**	**Rel. Sen. data detected? (DB)**	**Class**	**Rel. Sen. data detected?**
1	1	no	malicious	yes	malicious	no
2	1	yes	legitimate	yes	legitimate	yes $p \leq$ 0.05,IGain: 0.49
3	1	yes	legitimate	yes	legitimate	yes p 0.02, IGain: 0.31
4	0	no **(conflict)**	legitimate	yes **Highest IGain of 1.**	legitimate	yes
5	0	yes	legitimate	no	legitimate	no
6	0	no **(conflict)**	legitimate	no	legitimate	yes
7	1	no	legitimate	yes	legitimate	yes
8	1	no	legitimate	yes	legitimate	yes
9	1	no	malicious	yes **$p \leq$ 0.05,IGain of 0.8**	malicious	yes
10	1	yes (DA)	malicious	no **(conflict)**	malicious	yes

users, starting a new thread and sending parallel SOAP/XML-RPC Requests every 30 seconds (by setting the ramp up period to 3000 seconds) for varying number of requests. Since the WSO2 IS PDP is exposed via a Web service, we configured Jmeter to call that service API. We imported XACML policies associated with RDF instances into the PDP policy store using a tool that automates the WSO2 IS `EntitlementPolicyAdminService` API. Since the PDP uses caching to improve the performance, each time we ran the tests we loaded the PDP with different XACML requests stored in a CSV file. We used two transaction controllers. One (PEP+CH) had three Java Request samplers to test our implementation of the different context handler components (Table. 3.7), while the other (PEP) had a SOAP/XML-RPC sampler to test XACML request evaluation by comparing them to the 10 published policies.

In the test plan of our context handler we used two transaction controllers. One controller had three Jmeter Java Request samplers to test our implementation of the different context handler components (Table. 3.7). For each component we implemented a Java class that implements the `JavaSamplerClient` interface then we configured the JavaRequest sampler in Jmeter to test our classes. The other controller had both the first controller and a SOAP/XML-RPC sampler to

test XACML request evaluation. We created a similar version of our test plan but with the Web portal included. We created an HTTPS Request sampler to record user navigations through our Web portal, which in turn communicates with our PEP agent client. To avoid the overhead created through navigating through the Web portal, we show the results from testing only the context handler Java code.

Table 3.7: Setup for Jmeter test plan

XACML version	3.0
XACML Engine	WSO2 IS
PIP attribute store	LDAP store
Threads (Users)	100
Iterations (no. of test runs)	5, 10, 20
Ramp-up time	3000 sec
Transport	SOAP over HTTPS
Sampler(s)	Soap/XML-RPC and Java Request
Policies (Instances)	10
Requests	10, 50, 100
Data type properties/request	14, 28

Metrics. We defined different metrics and, for each implementation, we measured the average evaluation time over different runs. The evaluation time measures the elapsed time from the moment a given request is sent to the server until the moment the last bit of information has returned to the client. Our metrics are:

- Cost of context inference (CCI). This metric measures the cost incurred in the context inference phase relative to the total evaluation time.
- Accuracy (ACC). This metric measures the percentage of permit decisions vs. deny decisions in our implementation compared to that of the standard implementation provided by WSO2 IS.
- Scalability (SCA). We define the scalability point as the minimum number of concurrent users from which any increase no longer increases the Throughput per second.

Results. We present results for PEP evaluation time for each of the metrics mentioned above.

- CCI. The results of running the test plan for 5, 10, and 20 iterations indicate that the difference in average evaluation time between the two implementations did not change significantly (Fig. 3.7). Also, the overhead introduced by the context handler is not significant.

The evaluation time varies between the different context handler components. The relative sensitivity component has the largest evaluation time followed by the diversity handler then the classifier. The PEP slightly outperforms the PEP+CH implementation at the cost of producing less accurate decisions as the ACC metric indicates.

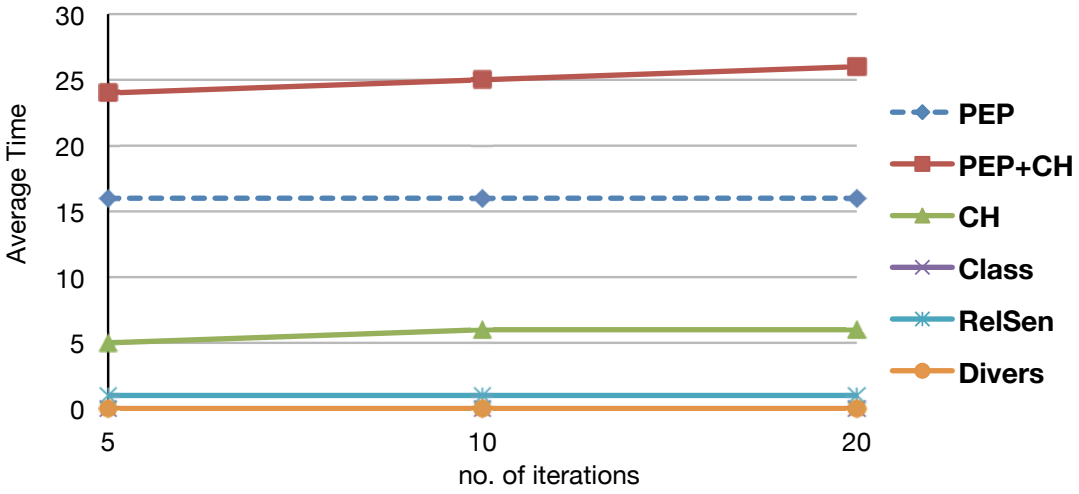


Figure 3.7: Cost of Context Inference over several test runs.

- ACC. The results for different number of requests for a fixed number of 14 data type properties and 3 purposes (Fig. 3.8) show that, on average, the percentage of the decisions that came as permit using a standard PEP is always higher than that of PEP with our context handler incorporated. Same is observed for different number of data type properties for a fixed number of 100 requests and 3 purposes (Fig. 3.8). This metric justifies the results obtained from the CCI metric above.

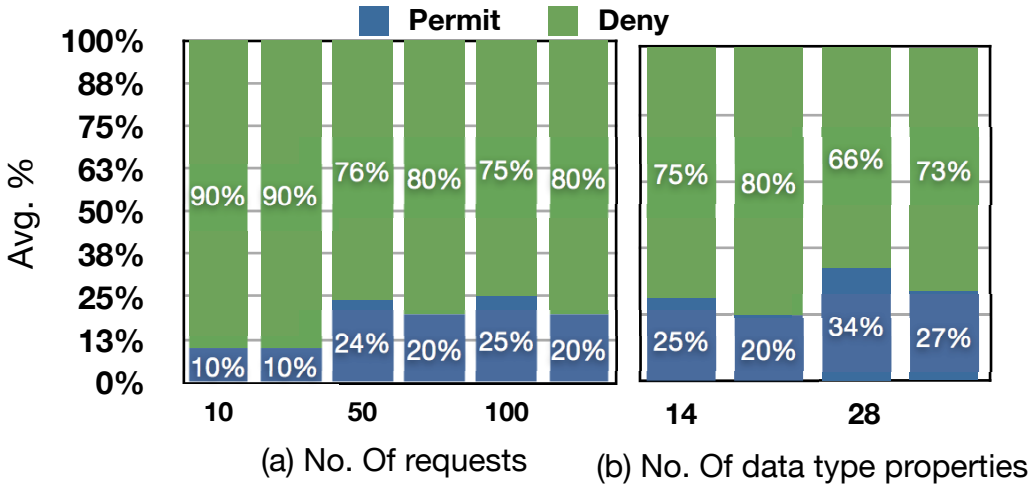


Figure 3.8: PEP Accuracy for PEP (left) vs. PEP+CH (right)

- SCA. To measure scalability, we ran one test on a local machine with the specifications provided in Sec. 3.7.1 and another on a cloud version of Jmeter. In the first case, we scheduled our load test to start a new thread every 30 seconds. After 30 min of running the test, we found that the throughput increased until it reached about 70 requests per second (50 on average) with 53 active users. It then hardly increased as more users were added with an increasing error rate. In the second case, we scheduled the test with 50 users (due to Jmeter limitations) where a new thread starts every 2 seconds. We found that the throughput increased until it reached more than 50 hits/sec on average with all 50 users active (Fig. 3.9).

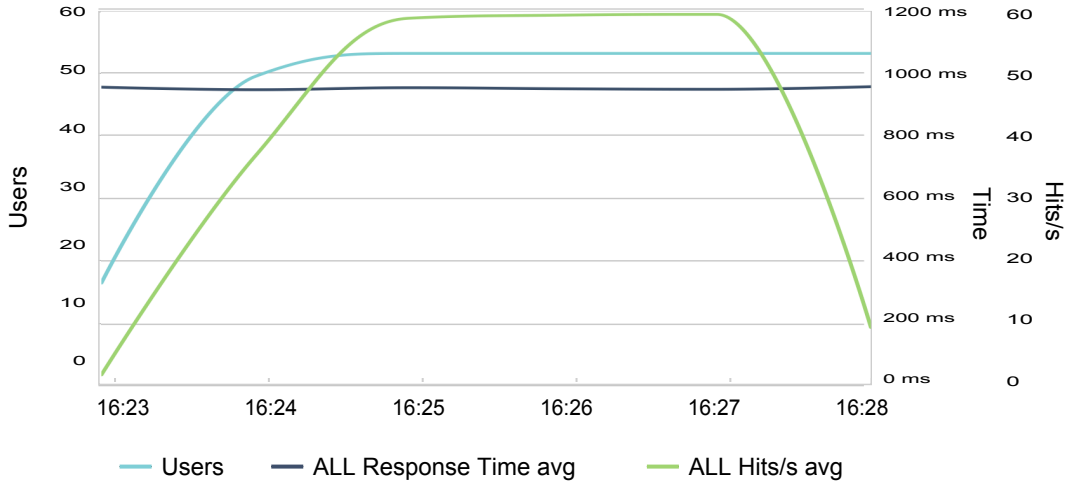


Figure 3.9: Scalability

3.7.2 Case Study 2: E-HIP

In this study, we compared our PEP implementation to a similar work, in Decat et al [36]. We used the Home Patient Monitoring System (HPMS) available from E-HIP [6] as our subject system. The E-HIP project provides an information platform for sharing patient's medical data across several healthcare providers including hospitals, general practioners, and screening centers.

Study Setup. In this section we discuss study setup.

Environment. We used the same environment settings used in the first study (Section. 3.7.1). We used an optimized version of SunXACML [3] to implement our PEP. We incorporated the context handler into the PEP implementation. For the PIP module, we used a JDBC-based attribute store using a MySQL database in the case of Decat et al. and an ontology-based repository in our case. As a PIP attribute store we used a MySQL database running on the same machine.

Data sets. We used the request generation technique used in the first study to generate XACML requests. As for the instance sets, we generated RDF files. For requests, instances, and policies we used the classes and data type properties from the E-HIP project (Table 1.2) and the 3 purposes shown in Table 3.2. For the policy set, we used a realistic policy set provided by the Ehip project, which consists of 19 atomic policies. The policies are designed to require 30 attributes in total. Since our approach requires fixed query sizes, we chose 16 of those attributes to represent the resource attributes. Of the 30 attributes, 8 are considered sensitive, so we chose those as our initial set of sensitive data D_A and assigned the remaining 8 properties 0 values.

Java Metrics Test Plans. For performance testing we used the Java Metrics library. Table 3.8 shows the test plan.

Table 3.8: Setup for Java metrics library test plan.

Subject system	E-HIP HPMS
XACML version	2.0
Environment	2GHz Intel Dual-Core i7 Mac, 8GB RAM, 64bit OS X 10.8.
XACML Engine	SunXACML
PIP attribute store	MySQL database
Threads (Users)	1, 2, 4, 8, 16, 32
Warm-up runs	10
Iterations (no. of test runs)	100
Policies (instances)	19 policies
Requests	3
Data type properties/ request	3, 6, 11

Metrics. We compared the average evaluation time and average throughput of both implementations (SunXACML, SunXACML+CH) for each of the metrics defined in Sec. 3.7.1. The throughput measures either the number of requests per second that are sent to the PEP or the number of classified queries that are written to an output file during a test. For this study, we measured the following additional metrics:

- Dimensionality (DIM). In this metric, we record the time and the throughput for varying input sizes. Namely, the number of requests and the number of data type properties in a request.
- Overhead Cost of Semantic Handler (OCSH). Based on our discussion of obligations above,

the overhead incurred by the Semantic handler (SH) is due to three steps: retrieving the required data type properties in the request (e.g. hasPurpose); retrieving the Context for the obligation (PDP, SH); and retrieving the instances that match the query (PEP, SH) (Fig. 3.1). This measure reflects the attribute fetch time of the PIP, which is in our case the overhead added by searching through the RDF repositories for matching instances. It measures the overhead of PEP parsing RDF repositories looking for attributes and generating requests from them.

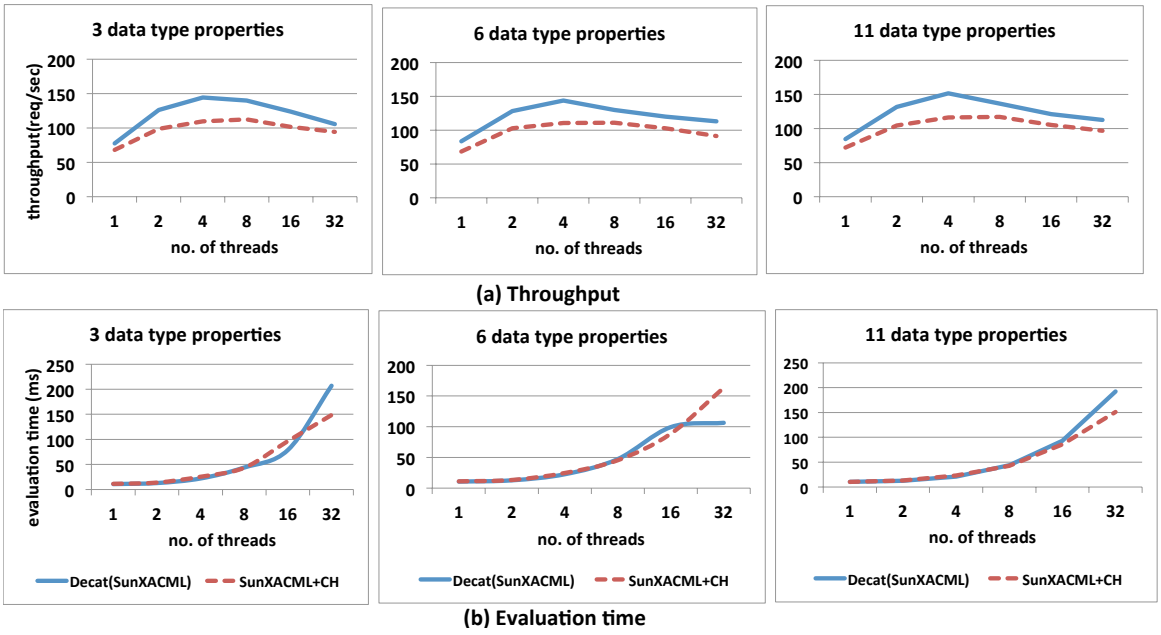


Figure 3.10: PEP performance for different number of data type properties for different threads

Results. We present results of comparing our implementation of PEP (SunXACML+CH) to that of Decat et al. (SunXACML) for each of the above mentioned metrics.

- CCI. The results of both the evaluation time and throughput (Fig. 3.10) indicate that the overhead introduced by the context handler is not significant.
- DIM. Fig. 3.10 illustrate that the average evaluation time and throughput of our PEP implementation did not change significantly between our implementation and that of Decat et al.
- OCSH. Fig. 3.11 shows the total evaluation time, the portion of time spent in fetching attributes, the portion spent on processing the rules of our PEP implementation compared to that of Decat et al. It also illustrates the evaluation times in terms of the number of data type

properties. The total evaluation time grows linearly and so does the overhead of fetching attributes as the number of data type properties increases. The figure also confirms that the overhead of processing requests through context inference (CCI metric) is not significant compared to Decat et al.

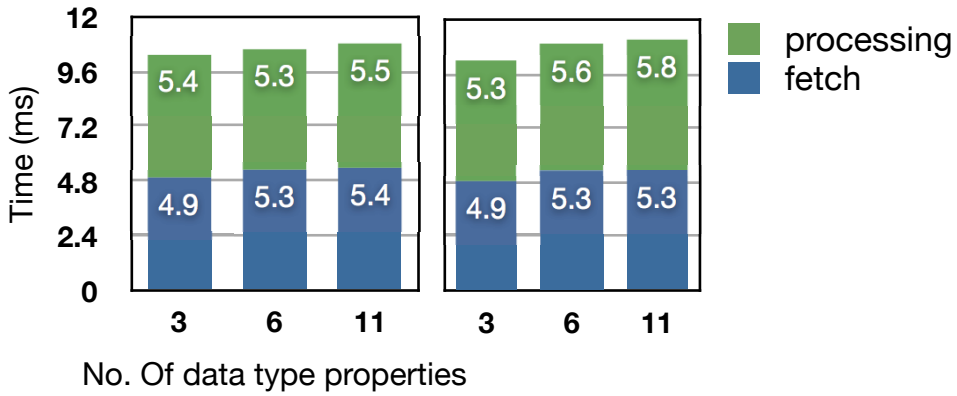


Figure 3.11: Attribute fetch time in Decat et al (left) compared to our implementation (right)

- ACC. Fig. 3.12 illustrates the accuracy of our implementation (PEP+CH) compared to that of Decat et al. (PEP) for different number of requests. The results indicate that our implementation filters out more denied queries compared to Decat et al.

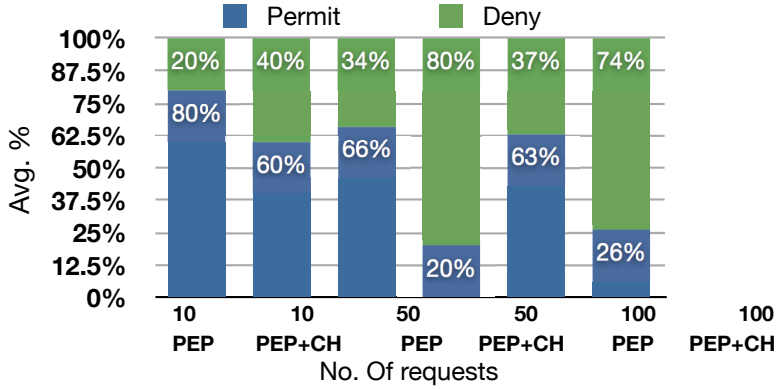


Figure 3.12: Accuracy of Decat et al. (left) compared to our implementation (right) for different number of requests

3.7.3 Case Study 3: Comparison to Dynamic Data Publishing

We compare our approach to two representative algorithms for K-Anonymity and HDB. Namely, the K-Anonymity algorithm in [60] and the HDB algorithm in [58]. Since none of the algorithms add context to the data we cannot compare to our context inference algorithm. The candidate algorithm for comparison becomes the dynamic rule evaluation algorithm (Algorithm 4). Some of the parameters that vary between the three algorithms are the the data set size

($|Ij|$) (the size of the original table $|T|$ in their case), the number of attributes in a query $|Dj|$ ($dimension(QID)$ in [60] and number of columns $|C|$ in [58]), the number of rules $|Rj|$ (for each dk in Dj of an instance i in Ij), which is the closest equivalent to the size of the choices table (choice %) in [58], and k. The only common parameter among all algorithms is $|Ij|$. However, the results cannot be reproduced entirely unless we use an environment similar to theirs (which makes a precise comparison hard).

In terms of metrics we can compare our algorithm to HDB in terms of execution time. Since Mondrian is a fast algorithm (nlogn) we can focus on the utility loss metric.

Execution time. A major limiting factor in HDB can be attributed to the cost incurred by translating policy files into choice tables and storing them in the database, rewriting queries, and fetching the privacy meta-data. Thus, we expect an overhead cost to be incurred by those modules according to the worst case conditions reported in [58]. We can estimate the overhead cost of privacy translation and choice retrieval based on their experiments. We expect the results to vary based on the storage method of the choice table (CPU processing time in case of internal storage of choices as case statements or I/O time in case of external storage of choices), the percentage of attributes (columns) on which a user chooses to add a restriction (size of a choice table), and the disclosure model (query vs. table semantics). None of which applies in our case.

The only overhead cost that we expect by running our Dynamic Rule Evaluation algorithm is the overhead cost added by the semantic handler which we reported in our experiments. In addition to $|Ij|$,$|Dj|$,$|Rj|$, our algorithm further relies on the size of the context block $|CTXTj|$ for each attribute dk of each instance in the set Ij.

Utility loss. As a measure of quality in terms of query answerability, the discernibility metric has been applied to Mondrian. We expect that for a large value of k the utility loss will decrease when $|QID|$ (Dj in our case) increases and we expect it to increase when k increases for a fixed $|QID|$.This measure does not apply in our case, so a better measure need to be defined and applied to both algorithms for precise comparison.

CHAPTER 4 K-ANONYMITY BASED PRIVATE WEB SERVICE SELECTION

Unlike Chapter 3, this chapter focuses on how we solve privacy issues in automatic or semi-automatic data access environments as described in Section 1.1. In service oriented environments, interaction occurs between two entities: clients and providers. Clients submit input data to invoke providers operations, who in turn return output data. Privacy can be achieved on input data, output data, or operation invocation. To guarantee privacy of operation invocation, it is essential to know if there is compatibility between a client's privacy requirements and the Web service privacy policies before the client invokes the operation. The client can then use the results of such a comparison to decide whether to invoke the operation. In this chapter, we focus on privacy at the Web service operation invocation level, then we generalize the approach to the business process level.

4.1 Motivating Scenario

Assume a collaborative Web service environment that consists of five web services: $WS_1, WS_2, WS_3, WS_4 and WS_5$, each of which has a set of operations $Op_1, ..., Op_n$. According to a naiive Web service conversation model, an initially invoked operation initiates a chain of subsequent operation invocations according to a configuration defined in a WSCL file (e.g., Listing 4.1).

Listing 4.1: A 5-Operation WSCL Definition. Interactions show all operations and Transitions show which operation calls which other operation

```
<ConversationTransitions>
  <Transition>
    <SourceInteraction href="getLab1Results"/>
    <DestinationInteraction href="getLab3Results"/>
  </Transition>
  <Transition>
    <SourceInteraction href="getLab2Results"/>
    <DestinationInteraction href="getLab3Results"/>
  </Transition>
  <Transition>
    <SourceInteraction href="getLab3Results"/>
    <DestinationInteraction href="getLab4Results"/>
  </Transition>
```

```
  <Transition>
    <SourceInteraction href="getLab5Results"/>
    <DestinationInteraction href="getLab4Results"/>
  </Transition>
</ConversationTransitions>
```

Fig. 4.1 depicts an abstract choreography of the conversations defined in Listing 4.1. Based on the figure, we have the following orders of operation invocations:

$$\begin{aligned} WS1/OP1 \rightarrow WS3/OP3 \rightarrow WS4/OP4, \\ WS2/OP2 \rightarrow WS3/OP3 \rightarrow WS4/OP4, \\ WS3/OP3 \rightarrow WS4/OP4, \\ WS5/OP5 \rightarrow WS4/OP4, \\ WS4/OP4 \end{aligned}$$

Given all possible end to end routes, for each possible operation invocation, there may exist one or more downstream operations along the route to the operation destination. For example, for the route $1 \rightarrow 4$, Op_1 has one downstream operation Op_3, which in turn has one downstream operation Op_4. Therefore, invoking Op_4 indicates that Op_3 was invoked, and thus that Op_1 was invoked. If we assume that Op_1 is a private resource, the above mentioned invocation scenario is not private, since for the source operation Op_1 there are at least 2 operations leading to the downstream operation Op_3 which it invokes. Similarly, there are 4 operations leading to the downstream operation Op_4 which it invokes. For example, if a requester Bob wants to detect a disease with which a patient Alice was diagnosed, he can use a service composition chain which involves the operation sequence getPrescriptionPrice(patientid) on $WS_{clinical}$, getPrice(medicine) on WS_{drug}, getMedicine(disease) on $WS_{drugPhenoAssoc}$, etc. Since some diseases can be inferred form the price of their medicine (e.g., cancer), the operation getPrescriptionPrice(patientid) can be considered a private resource, and knowing that the operation getPrice(medicine) has been invoked can reveal that getPrescriptionPrice has been invoked. Thus, a standard WSCL configuration does not guarantee the privacy of the data hosted by a Web service neither at the operation level nor the data

(operation input and output parameters) level. It is crucial to privacy to know the extent to which the invoking an operation can be inferred if one knows that a downstream operation was invoked. We propose an approach that takes into account k-Anonymity at the operation level in determining compatibility within privacy frameworks.

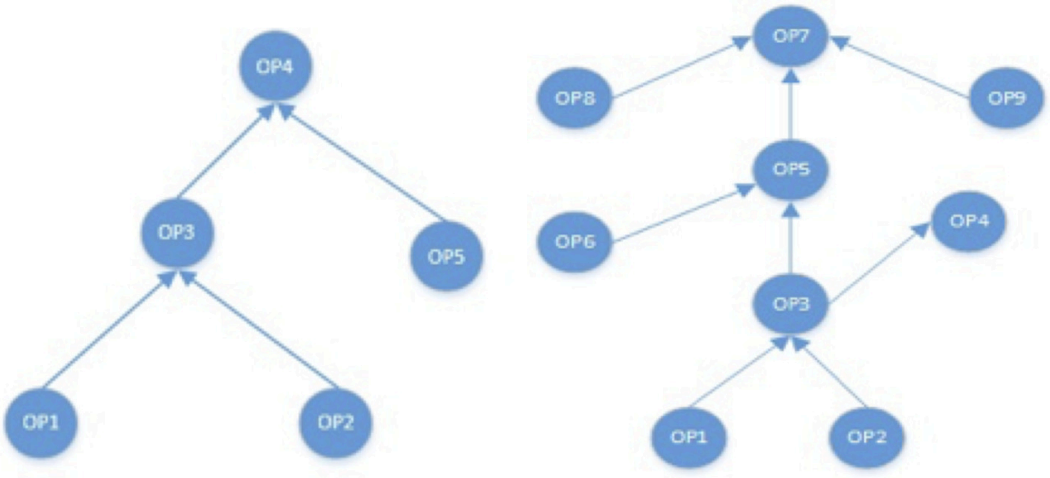


Figure 4.1: A choreography of a 5-operation (left) and 9-operation (right) invocation configurations.

4.2 Web Service Composition Standards

Two main composition models can be applied to obtain value-added services: orchestration (e.g., BPEL) and choreography (e.g., WSCL). In the orchestration model there is a centralized orchestrator that manages the invocations, while in the model it is more of a decentralized style and each Web service has it's own WSCL definition. Fig. 4.2 depicts a model that combines both models.

4.2.1 Web Service Conversation Language (WSCL)

WSCL definitions defined and accessed on the Web service side reflect the relationships between web service operations; by indicating which operations call which other operations. For instance, in a composition consisting of nine Web services: $WS_1, \ldots WS_9$, and a set of operations $Op_1, ..., Op_9$ corresponding to each service, an initially invoked operation initiates a chain of subsequent operation invocations according to a configuration defined in a 9-Operation WSCL Definition (Listing. 4.2). The WSCL definition specifies interactions (all possible operations) as well as transitions (which operation calls which other operation). Fig. 4.2 depicts an abstract choreography of the conversations defined in a 5-Operation and a 9-Operation WSCL models employed by the composite services CS_2 and CS_3, respectively.

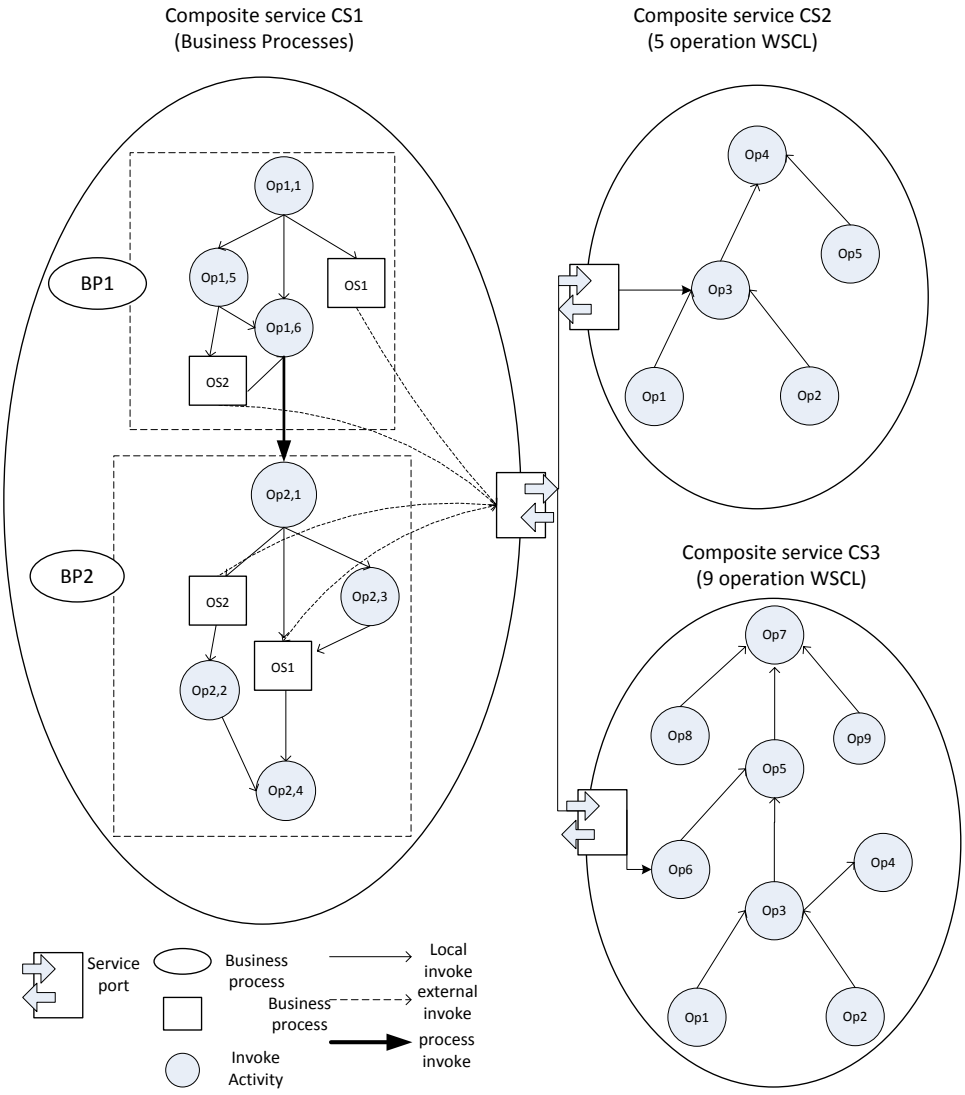


Figure 4.2: A hybrid composition model using BPEL orchestration and WSCL choreography.

Listing 4.2: A 9-Operation WSCL Definition. Interactions show all operations and Transitions show which operation calls which other operation

```
<ConversationTransitions>
  <Transition>
    <SourceInteraction href="Op1"/>
    <DestinationInteraction href="Op3"/>
  </Transition>
  <Transition>
    <SourceInteraction href="Op2"/>
    <DestinationInteraction href="Op3"/>
  </Transition>
  <Transition>
```

```
    <SourceInteraction href="Op3"/>
    <DestinationInteraction href="Op4"/>
   </Transition>
    <Transition>
    <SourceInteraction href="Op3"/>
    <DestinationInteraction href="Op5"/>
   </Transition>
 <Transition>
    <SourceInteraction href="Op6"/>
    <DestinationInteraction href="Op5"/>
   </Transition>
   <Transition>
    <SourceInteraction href="Op5"/>
    <DestinationInteraction href="Op7"/>
   </Transition>
    <Transition>
    <SourceInteraction href="Op8"/>
    <DestinationInteraction href="Op7"/>
   </Transition>
    <Transition>
    <SourceInteraction href="Op9"/>
    <DestinationInteraction href="Op7"/>
   </Transition>
 </ConversationTransitions>
```

4.2.2 Business Process Execution Language (BPEL)

In BPEL, the composition result is called a *process* (Listing. 4.3, participating services are *partners*, and message exchange or intermediate result transformation is called an *activity*. BPEL also defines the data used by a process as *variables*. A process thus consists of a set of activities and interacts with external partner services through WSDL interfaces. Business processes can be either *Executable business processes*, which model actual behavior of a service in a business interaction and business protocols, which describe process logic in terms of message exchange between the interacting services, without exposing their implementation details. Fig. 4.2 depicts an abstract

orchestration of two processes in a composite service CS_1.

Each step of a process is represented by an *activity*, which is implemented as either an outbound web service invocation on an external partner service or as an inbound web service request. According to BPEL data model, a `process` consists of a sequence of activities. Those `activities` represent units of processing (e.g. receive and reply) and show what the process actually does. External partner services are defined by WSDL interfaces. Activities map to operations defined in WSDL files. Some of these activities are invocations to external Web services. In BPEL these external services are called `partnerlinks`, and represent the web services with which the process (composite service) interacts. If an activity involves invoking an operation defined in a WSDL of an external service, the operation is considered an *outsourced* operation. For instance, the invocation $Op_{2,1} \rightarrow OS_1$ illustrates an outsourced communication link between CS_1 and CS_3. A process in BPEL can communicate with either a Web service or with another BPEL process. For instance, the invocation $Op_{1,6} \rightarrow Op_{2,1}$ illustrates a communication link between BP_1 and BP_2.

The web service operations exposed by the process can aggregate other web services in performing the business tasks associated with the process. Thus, the flow of the process includes a set of Web service interactions between the process itself, internal Web services, and external partner Web services. BPEL processes interact with WSDL services exposed by business partners. Thus, we distinguish between interfaces exposed by the BPEL process and interfaces consumed by the BPEL process. The later may involve *outsourcing*. Composition in BPEL is recursive; it consists of structured activities, which can be nested. For instance, activities contained in a flow are executed in parallel, partially ordered through control structure activities contained in a sequence are performed sequentially in lexical order. In an if then else, exactly one branch of activity is selected from a set of choices.

Listing 4.3: BPEL definition from our scenario

```
<process name="outsourcedOperationCallScenario">
<partnerLinks>
<partnerLink name=client/>
<partnerLink name=serviceA/>
```

```
<partnerLink name=serviceB/>
<partnerLink name=serviceC/>
</partnerLinks>
<variables>
<variable name=processInput/>
<variable name=AInput/>
<variable name=AOutput/>
<variable name=BCInput/>
<variable name=BOutput/>
<variable name=COutput/>
<variable name=processOutput/>
</variables>
<sequence>
<receive name=receiveInput variable=
input/>
<assign><copy>
<from variable=processInput/>
<to variable=AInput/>
</copy></assign>
<scope>
<sequence>
<invoke name=invokeA partner-
Link=serviceA
inputVariable=AInput output-
Variable=AOutput/>
</sequence>
</scope>
<assign><copy>
<from variable=AOutput/>
<to variable=BCInput/>
</copy></assign>
<flow>
<sequence>
<invoke name=invokeB partner-Link=serviceB
```

```
inputVariable=BCInput/>
<receive name=receive_invokeB
partnerLink=serviceB
variable=BOutput/>
</sequence>
<sequence>
<invoke name=invokeC partner-Link=serviceC
inputVariable=BCInput/>
<receive name=receive_invokeC
partnerLink=serviceC
variable=COutput/>
</sequence>
</flow>
<invoke name=reply
partnerLink=client
inputVariable=processOutput/>
</sequence>
</process>
```

4.3 Private Web Service Selection Model

In this section, we present the privacy management framework proposed in [94, 95], and how we extend it using K-Anonymity. The model accounts for privacy at both the data (input and output) and operation usage levels. We also provided a matching protocol and a negotiation model to resolve incompatibility between client's requirements and provider's policies. We briefly describe the model and we refer the reader to [94, 95] for further details.

According to the model, a provider WS defines a privacy policy PP^{WS}. For each provider WS, a client C defines privacy requirements $PR^{C/WS}$ regarding WS input output and operations. C may demand full compatibility between PP^{WS} and $PR^{C/WS}$ or a partial compatibility with certain threshold. In case of incompatibility, C and WS have two options: either discontinue the interaction or initiate a negotiation process to reconcile privacy policies and requirements. The model consists of the following concepts:

- Resource: we refer to private information as resources, including input parameters submitted

Table 4.1: Two business processes (BP1 is from Org1's perspective, BP2 is from a competitor's perspective) and constituent activities.

CS	BP	Activity	flow type	WS	Operation	input	output
1	1	receive		client	Op1,1		
1	1	invoke AB: performed by providers A,B	PAR	ServiceA	Op1,5	processInput/AInput	Aoutput
1	1	invoke AB: performed by providers A,B	PAR	ServiceB	OS1	processInput/BInput	BOutput
1	1	invoke C: performed by provider C	SEQ	ServiceC	OS2	Aoutput	
1	2	receive		client	Op2,1		
1	2	invoke DBE: performed by providers DBE	PAR	ServiceD	Op2,2	processInput/DInput	Doutput
1	2	invoke DBE: performed by providers DBE	PAR	ServiceB	OS1	processInput/BInput	BOutput
1	2	invoke DBE: performed by providers DBE	PAR	ServiceE	Op2,3	processInput/EInput	EOutput
1	2	invoke F: performed by provider F	SEQ	ServiceF	Op2,4	BOutput	FOutput
1	2	invoke C: performed by provider C	SEQ	ServiceC	OS2	FOutput	COutput

to providers, the fact that a client invoked an operation, or the output data. The type of resource is determined using the notion of a privacy level.

- Privacy level: resources can be either data or operations that the Web service invokes to retrieve such data. We handle both cases. Privacy at the data level handles handles privacy policies or requirements imposed on data resource that a client and a provider exchange, whereas privacy at the operation level handles privacy policies or requirements imposed on the invoked operation. Data resources can be operation input and output parameters defined by a service (i.e., WSDL). For instance, an input parameter to an operation `getLabResults` could be `patient_id` and the output could be `test_results` (Fig. 4.3). `patient_id` and `test_results` are data resources whereas the operation `getLabResults` is an operation resource. The operation `getLabResults` can be considered a private resource by

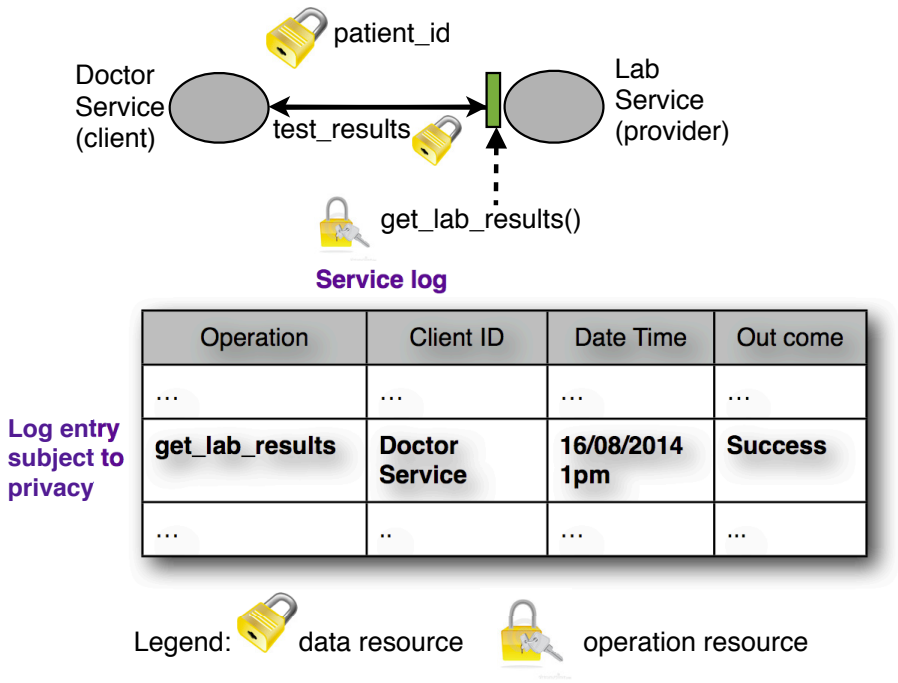


Figure 4.3: Web service privacy model.

the organization hosting the Web service that defines that operation (e.g., a hospital), especially when disclosing the invocation of this operation to other organizations (e.g., research institutes) can cause liability issues for the hospital.

- Privacy Rule: We define a rule R_i as a tuple $\langle T_i, L_i, D_i, S_i \rangle$, where T_i is the topic, and it can have any of the values $\{Purpose, Retention, Recipient, KAnon\}$, L_i is the privacy level and it takes the values {*data, operation*}, D_i is the domain and contains the possible values that can be taken by a topic $\{no - retention, indefinitely, stated - purpose\ , public, government, research, federaltax, same, and\ otherservices\}$, and S_i is the scope of a rule and defines the granularity of the resource that is subject to privacy constraints, and it can be any of the values $\{total, partial, GTE1, GTE2, GTE3, GTE4, GTE5, GTE6, GTE7, GTE8, GTE9, \ GTE10, INFINITE\}$. The values $GTEx$ are only assigned in cases where the topic is indicated as KAnon, and they indicate that the KAnonymity value should be at least as large as this value and *INFINITE* means that the operation does not invoke any other operations (there are no downstream operations). In the case of data rules (i.e., $L_i = data$), we consider data resources as atomic. Hence, the only scope value allowed is total. Partial scope may be assigned only to operations (i.e.,

$L_i = operation$) and complex data resources (e.g., a data structure). If an operation resource is assigned a total scope for a given rule, then the whole entry of that operation in the service log is private. An example privacy rule is $R1 = \langle T1 = Recipient, L1 = Data, D1 = \{public, government, federaltax\}, S1 = \{total\}\rangle$

- Privacy Assertion: A privacy assertion $A(Ri, rs)$ is the application of a rule $Ri = \langle Ti, Li, Di, Si\rangle$ on a resource rs, where rs has a level Li, domain Di, and scope Si. A privacy assertion $A(Ri, rs)$ on a resource rs determines the granularity of a resource that is subject to privacy. The granularity g of a resource is determined by the scope Si of a rule. An assertion on rs according to $Ri = \langle Ti, Li, Di, Si\rangle$ is defined by the pair $\langle pf, g\rangle; pf = d_i \wedge \cdots \wedge d_j$ where $d_i, \ldots, d_j \in Di; g \in Si$. The assertion $A_1(R_1, patient\_id) = (government \wedge research)$, for instance states that $patientid$ can be shared with government and research institutes.
- Privacy Policy: a privacy policy PP_i of a service provider WS_i is a set of assertions that the service specifies on the resources that it hosts. For instance, for a Lab service $PP_{LabService} = \{A_1(R_1, patient\_id) = (government \wedge research, total), A2(R_2, getLabResult()) = (federaltax \wedge research, total)\}$.
- Privacy Requirement: clients perceive privacy not only on resources but also on services. For each service WS, a client C defines a privacy requirement $PR^{C/WS}$ stating C's assertions about resources provided by WS. $PR^{C/WS}$ assertions describe two requirements. First, C's expectation about how WS will treat the privacy of resources (noted as $A(Ri, rs^E)$) (e.g., patient_id). Second, C's practices regarding how C treats the privacy of output data returned by WS (noted as $A(Ri, rs^P)$) (e.g. $test\_result$). Clients may unequally value the assertions specified in $PR^{C/WS}$ by assigning a weight w_j to each $A(Ri, rs)$ in $PR^{C/WS}$. The weight is a decimal value between 0 and 1 that determines the importance of the corresponding assertion. The total of weights assigned to all assertions is equal to 1. We give clients the possibility to control their privacy requirements by associating a mandatory attribute m_j to each assertion $(A_j(R_i, rs_k), w_j, m_j)$ in $PR^{C/WS}$.

Subsumption Matching Protocol. We use the notion of privacy subsumption defined in [94] to

Table 4.2: Domain subsumption relationships

subsumed domain	subsuming domain
government	public
research	public
federal tax	public
federal tax	government

capture the semantic relationship among domain values. For a set of domains D_i={d_i,d_j}, d_i is subsumed by d_j iff d_j is more general than d_i. For instance, in the set of domains D_i={public, government, federal tax, research} the value public is more general than the other values in D_i. Subsumption can also be generalized to assertions [94]. Two Assertions $A(Ri, rs) = (pf, g)$ and $A(Ri, rs) = (pf, g)$ are compatible if their is a subsumption between them. Namely, A' is subsumed by A ($A \sqsubseteq A$) , if $Ri = Ri$, $rs = rs$, $g = g$, and $pf \Rightarrow pf$.

Privacy Preserving Composition Plan. In a service composition plan CP, any service WS which depends on another service WS' is considered a consumer of the data provided by WS'. Thus, WS can be viewed as a client and WS' can be viewed as a provider. We have previously proposed a Privacy Compatibility Matching PCM algorithm to check the compatibility between the PR of WS and PP of WS'. The algorithm considers a CP as privacy compatible if the privacy compatibility is fully satisfied for all the dependencies among the services in CP. If at least one dependency in CP has an incompatible assertion then CP violates privacy and is discarded from CP. Fig. 4.4 depicts a sample CP. The data resource drug DR is an input parameter to WS_6, WS_7 and WS_8, and an output for WS_5. Thus, those services depend on WS_5 for providing DR.

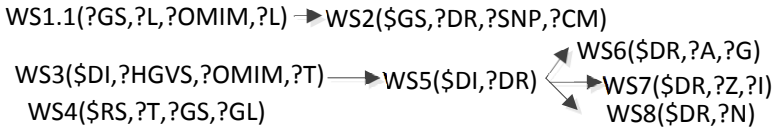


Figure 4.4: A sample CP with dependencies.

Privacy Compatibility Matching (PCM) Algorithm. Upon request of a resource rs, the PCM algorithm extracts the resource rs and the assertions associated with that resource. For full compatibility, the algorithm requires that all assertions A in both PR^{WS} and $PP^{WS'}$ are fully compatible. For each $A \in PR^{WS}$ and $A' \in PP^{WS'}$, the algorithm checks if A subsumes A' and returns the set InC of incompatible assertion pairs . The algorithm follows the subsumption pro-

tocol described above. For instance, for the composition in Fig. 4.4, the PCM algorithm checks the compatibility of the assertions defined on resource $rs = DR$ in PR_{WS5} and PP_{WS6} etc. If $PR_{WS5} = (A1(R1, DR) = hospital)$ and $PP_{WS6} = (A'1(R1, DR) = researchLab)$, then the two assertions are incompatible since hospital is not $\sqsubseteq$ researchLab. Hence, PCM adds both assertions to the set of incompatible assertions (i.e., $InC = (A1, A1')$).

The above check works in the case of total matching. However, some clients may be willing to use a service even if some of their privacy constraints are not satisfied. To that end, the PCM also enables partial matching by incorporating the notion of a privacy matching degree. The privacy matching degree estimates the ratio of client assertions ($PR^{C/WS}$) that match the service assertions (PP_WS). We refer to $M \subset PR^{C/WS}$ as the set of all such $PR^{C/WS}$ assertions. The privacy degree is mathematically defined as:

$$Degree(PR^{C/WS}, PP_{WS}) = \sum w_j \forall (A_j(R_i, rs_k), w_j, m_j) \in M$$

The threshold τ is provided by a service client and illustrates the minimum value allowed for a matching degree. The PCM determines that $PR^{C/WS}$ and PP_WS are compatible if any of the following holds:

- The privacy matching degree is above the threshold set by C: $Degree(PR^{C/WS}, PP_{WS}) \geq \tau$.
- Every non-matched $PR^{C/WS}$ assertion is optional: $\forall (A_j(R_i, rs_k), w_j, m_j) \in (PR^{C/WS} - M) : m_j =$ "$False$"

4.4 K-Anonymity Based Privacy Compatibility Matching (KPCM)

We extend our Privacy Compatibility Matching (PCM) algorithm to also incorporate k-anonymity check. Motivated by the notion of K-Anonymity at the relational data table level (Section 2.4.1), we leverage that notion to a Web service operation invocation level. Our architecture is shown in Fig. 4.5.

K-Anonymity at the Operation level: K-Anonymity beyond relational data. Since QIDs can be indicators of a data owner's identity (Section 2.4.1, downstream operation invocations can be indicators of source operation invocations. Analogous to linkability through QIDs, we use destination operation invocations as a basis for calculating K-Anonymity scores. We leverage the

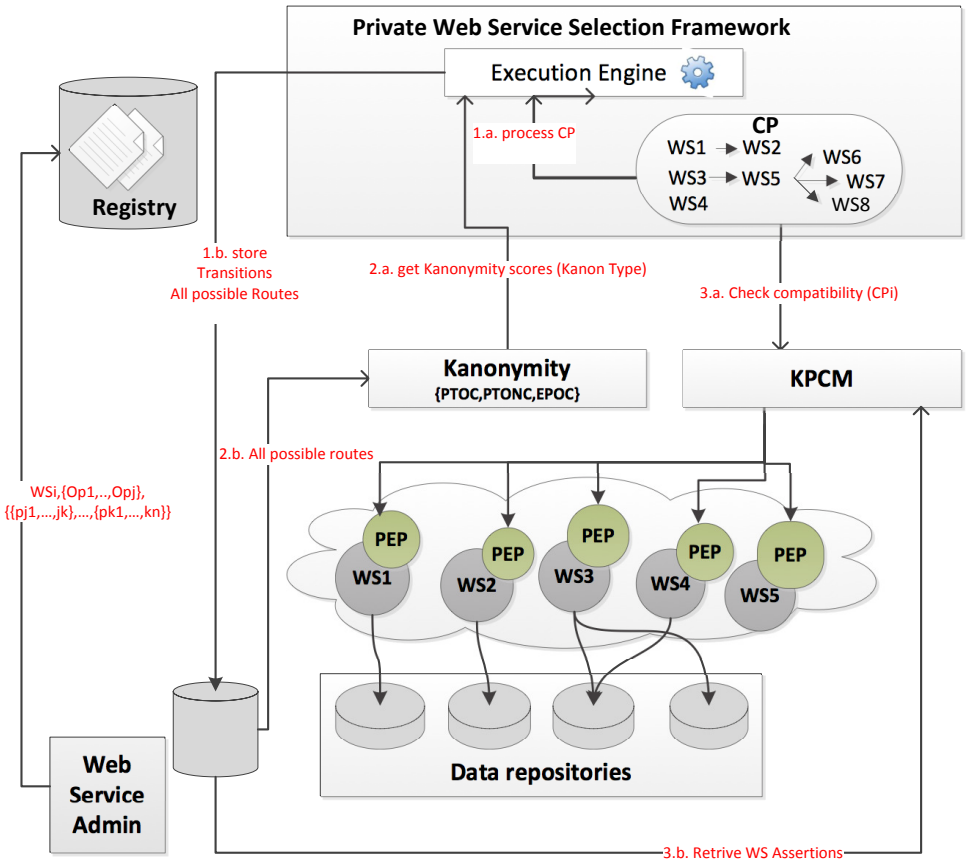


Figure 4.5: KPCM Architecture.

notion of k-anonymity to the level of Web service operation. We define k-Anonymity of an operation as the value that determines the extent to which an operation invocation can be inferred if one knows that a downstream operation was invoked. To determine k-anonymity counts, we use operation invocation route configuration stored in Web Services Conversation Language (WSCL) definitions.

Definition *Let $G = \{Op_1, ..., Op_n\}$ be a WSCL graph of n operations. For each source operation $Op_i \in G$, there exists a k-anonymity score k if, for each destination (downstream operation), there are at least k occurrences of that operation. Op_i is called k-Anonymous .*

To cover all possible invocation scenarios we take into account the cases where all operations can be invoked (including those that are invoked in the middle), the case where all operations can be invoked (excluding middle operations), and the case where only end point operations can be invoked. Next, we explain how we determine K-Anonymity values.

Table 4.3: Transisitions (left) and all possible routes (right)

Opsrc	Opdst
getLab1Results	getLab3Results
getLab2Results	getLab3Results
getLab3Results	getLab4Results
getLab5Results	getLab4Results

Opsrc	Opdst
getLab1Results	getLab3Results
getLab1Results	getLab4Results
getLab2Results	getLab3Results
getLab2Results	getLab4Results
getLab3Results	getLab4Results
getLab5Results	getLab4Results

Table 4.4: KAnonymity counts for All WSCL Methods can Be called (Pass-Through Methods Counted)

Source	Destination	KAnon	Scope
getLab1Results	getLab3Results	2	GTE2
getLab1Results	getLab4Results	4	GTE4
getLab2Results	getLab3Results	2	GTE2
getLab2Results	getLab4Results	4	GTE4
getLab3Results	getLab4Results	2	GTE2
getLab5Results	getLab4Results	4	GTE4

Calculating k-Anonymity Values. K-Anonymity values are derived based on both WSCL definitions and K-Anonymity types.

- **WSCL Definition Selection.** First, the transitions are extracted from the WSCL file (e.g., Table 4.3). Then, for each source and destination combination a row is added to the table WSCL_Transitions. Then, the same information is put onto table WSCL_Transition_AllPossibleRoutes. Next, we add additional rows that represent the cases where the source of one row is equal to the destination of another row. For instance, if the transitions 1 $\rightarrow$ 3, 3 $\rightarrow$ 4, and 4 $\rightarrow$ 5, are found, then 1$\rightarrow$5 is considered as an end-to-end route. Therefore, a row with the source operation 1 and the destination operation 5 is added to the table. Table 4.3 indicates all the possible end to end routes derived from the transitions shown in Table 4.3.
- **Determine K-Anonymity Type.** The user then selects the desired k-Anonymity type from the following options:

  PTOC - All WSCL Operations Can Be Called (Pass-Through Operations Counted). For this K-Anonymity type, we first classify the end-to-end routes derived in Table 4.3 by the number of nodes that they have in the middle (middlemen). Following our running example, the route 1 $\rightarrow$ 5 consists of three sub routes 1 $\rightarrow$ 3, 3 $\rightarrow$ 4, and 4 $\rightarrow$ 5. In this case, there are two middlemen: Op_3 and Op_4. Then for each case, we get a count of all the routes leading

to the destination of the source operation that we are invoking (count in Table 4.10). Then we subtract from this count the number of routes that lead to the source operation that we are invoking (KAnon1 in Table 4.10). For example, from the perspective of Op_3, when we calculate k-Anonymity for an operation to which it leads, we subtract the count of operations leading to Op_3 (i.e. Op_1 and Op_2) since we need k-Anonymity from the perspective of Op_3. For the 5-Operation WSCL file, these operations can be called: 1, 2, 3, 4 and 5 and the operations that we pass through are part of the k-Anonymity count. Table 4.4 expands upon the information in table 4.3, indicating all possible end-to-end routes and k-Anonymity values.

Table 4.5: WS Privacy Ruleset Items derived from Table 4.4

WS	rule	item	topic	level	domain	scope
1	3	10	k-Anon	operation	other services	GTE2
2	3	10	k-Anon	operation	other services	GTE2
3	3	10	k-Anon	operation	other services	GTE2
4	3	28	k-Anon	operation	other services	INFINITE
5	3	11	k-Anon	operation	other services	GTE4

The rows in Table 4.4 are then added to set of WS Privacy Ruleset Items in order to be used in Web service assertions later on (Table 4.5). For each source Web service operation we choose the rows with smallest GTE values, which define the minimum KAnonymity Privacy Policy Rule Item for each operation. Since these represent KAnonymity rules, we indicate the topic as "K-Anonymity", the level as "operation", and the domain as "other Web services".

Table 4.6: KAnonymity counts for All WSCL Operations can Be called (Pass-Through Operations Not Counted)

source	**destination**	**KAnon**	**scope**
getLab1Results	getLab3Results	2	GTE2
getLab1Results	getLab4Results	3	GTE3
getLab2Results	getLab3Results	2	GTE2
getLab2Results	getLab4Results	3	GTE3
getLab3Results	getLab4Results	2	GTE2
getLab5Results	getLab4Results	3	GTE3

PTONC- All WSCL Operations Can Be Called (Pass-Through Operations Not Counted). For this K-Anonymity type, we start with the set of rows resulting from the PTOC stage (Table 4.4). Then, as an additional step, we subtract out the number of middlemen from

the k-Anonymity count for the initial operation (source operation) for each route. For the 5-Operation WSCL file, these operations can be called: 1, 2, 3, 4 and 5, and the operations that we pass through are not part of the k-Anonymity count. Table 4.6 expands upon the information in table 4.3, indicating all possible end-to-end routes and k-Anonymity values. Table 4.6 represents the set of WS Privacy Ruleset Items that will be used in Web service assertions.

Table 4.7: WS Privacy Ruleset Items derived from Table 4.6

WS	rule	item	topic	level	domain	scope
1	3	10	k-Anon	operation	other services	GTE2
2	3	10	k-Anon	operation	other services	GTE2
3	3	10	k-Anon	operation	other services	GTE2
4	3	28	k-Anon	operation	other services	INFINITE
5	3	11	k-Anon	operation	other services	GTE3

Table 4.8: KAnonymity counts-Only Endpoint Operations can Be called

source	**destination**	**KAnon**	**scope**
getLab1Results	getLab3Results	2	GTE2
getLab1Results	getLab4Results	3	GTE3
getLab2Results	getLab3Results	2	GTE2
getLab2Results	getLab4Results	3	GTE3
getLab5Results	getLab4Results	3	GTE3

EPOC- Only Endpoint WSCL Operations Can Be Called. This k-Anonymity type is a subset of the above case (PTONC), with k-Anonymity values for only endpoint operations. For the 5-Operation WSCL file, these operations can be called: 1, 2, and 5, and operations that we pass through are not part of the k-Anonymity count. Table 4.8 expands upon the information in table 4.3, indicating all possible end-to-end routes and k-Anonymity values. Table 4.8 represents the set of WS Privacy Ruleset Items that will be used in Web service assertions.

Table 4.9: WS Privacy Ruleset Items derived from Table 4.8

WS	rule	rule item	topic	level	domain	scope
1	rule 3	10	k-Anonymity	operation	other services	GTE2
2	rule 3	10	k-Anonymity	operation	other services	GTE2
5	rule 3	11	k-Anonymity	operation	other services	GTE3

4.4.1 K-Anonymity Based PCM (KPCM)

Following the above scenario, before a Web service operation can be called (e.g., `getLab1Results()` on WS1), the KPCM algorithm is invoked (Algorithm. 5). The KPCM

algorithm takes as input a Web service WS_i and an operation (Op_j). The algorithm first retrieves the client assertions array $PR^{C/WS}$ from the client database. A call is then made to retrieve the Web service assertions array PP^{WS} for the specified operation. Then the system goes through each assertion in both arrays looking for mismatches. The matching check occurs on mandatory items in each assertion A including the following:

- Resource R: Client PR Practices and WS PP Expectations are treated as special cases.
- Rule Topic, Level, Scope: As defined in Client PR and Web service PP; need full match. If scope is "k-Anonymity" a special check is performed by calling the $K - AnonymityCheck$ algorithm. The $KPCM$ algorithm extracts the WSCL transitions table TR from the WSCL file (e.g., Table 4.3) and passes it to the $KAnonymityCheck$ algorithm.
- Domain: a check is performed per propositional formula. For the domain, a full, partial or no match may be required. If there is no direct match, the algorithm checks for a subsumption match by calling the $SubsumptionMatchProtocol$ algorithm.

Algorithm 5 KPCM

```
1: input WS_i, Op_j
2: output: match
3: PR ← φ, PP ← φ, A' ← φ, compatible ← true
4: PR ←GETPRIVACYREQUIREMENTARRAY
5: PP ←GETPRIVACYPOLICYARRAY(WS_i, Op_j)
6: for each A_i in PR do
7:     for each A_j in PP do
8:         if M_i then
9:             if A_i[rs] == A_j[rs] & A_i[R][T] == A_j[R][T] & A_i[R][L] == A_j[R][L] & A_i[R][S] == A_j[R][S] then
10:                if A_i[R][S] =' KAnonymity' then
11:                    TR ← GETWSCLTRANSACTIONS
12:                    KANONYMITYCHECK(TR, type)
13:                end if
14:                if A_i[D]! = A_j[D] then
15:                    SUBSUMPTIONMATCHPROTOCOL(A_i, A_j)
16:                end if
17:            end if
18:        end if
19:    end for
20: end for
21: return compatible
```

If a mismatch is found, a false compatibility trigger is returned together with the set of assertions A' that are incompatible. If there is compatibility, the desired web service operation is called. For

Table 4.10: K-Anonymity values for the WSCL configuration in Table 4.3

WS	Opsrc	Opdes	midmen	count	KAnon1	KAnon2	KAnon3
1	1	3	0	2	2	2	2
1	1	4	1	4	4	3	3
2	2	3	0	2	2	2	2
2	2	4	1	4	4	3	3
3	3	4	0	4	2	2	0
5	5	4	0	4	4	4	4

example, a service privacy policy array may include the assertions:

$$PP^{Service_{lab}} = A_1(R_1, patient_id), A_2(R_2, getLabResult)$$

, where $A_1(R_1, patient_id) = (gov \wedge research, total)$,

and $A_2(R_2, getLabResult) = (fedtax \wedge research, total)$

The *KAnonymityCheck* algorithm (Algorithm 6) takes as input a WSCL transitions table ($TR[Op_1, ..., Opn]$) and a previously configured kanonymity type ($PTMC, PTMNC, EPMC$). The algorithms first generates the end-to-end routes table (TR_{routes}). To generate TR_{routes} the algorithm copies all the rows from TR to TR_{routes}. It then adds additional rows that represent the cases where the source of one row is equal to the destination of another row (Table 4.3). Then, for each row in TR_{routes}, it calculates the midmen, the count, and based on the selected type, it generates the kanon count and appends the newly created row to a new table TR_{KAnon} with k-anonymity counts included. The algorithm returns the table TR_{KAnon}.

4.5 Private Web Service Outsourcing Approach

In this section, we present our approach, including the system architecture (Section 4.5.2). We also provide two definitions of K-Anonymity: at the operation level (Section 4.5.3) and at the business process level (Section 4.5.4) and the K-Anonymity score calculation in each case. Then, we introduce uncertain operation invocation precedence graphs in Section 4.5.9, the different K-Anonymity types in Section 4.5.7, and the K-Anonymity check algorithm. Finally, we explain how all of that fits into our model for private service outsourcing.

4.5.1 Running Example

We consider the case of a composite service CS in which the invocation of an operation may result in the execution of part of a process that includes various activities. We look into several

Algorithm 6 KAnonymityCheck

```
1: input TR, type
2: output: TR_KAnon
3: row ← φ, TR_routes ← φ, TR_KAnon ← φ, count ← 0, KAnon ← 0, scope ← INFINITE
4: TR_routes ← TR_routes ∪ TR
5: for each row_i in TR do
6:     for each row_j in TR do
7:         if row_i[Op_dst] == row_j[Op_src] then
8:             row ← (row_i[Op_src], row_j[Op_dst])
9:             TR_routes ← TR_routes ∪ row
10:        end if
11:    end for
12: end for
13: for each row_i in TR_routes do
14:    for each row_j in TR_routes do
15:        if row_i[Op_dst] == row_j[Op_dst] then
16:            count ← count + 1
17:        end if
18:        if type == PTOC then
19:            KAnon ← count
20:        end if
21:        if type == PTONC then
22:            KAnon ← count − midmen
23:        end if
24:        if type == EPOC then
25:            KAnon ← count − midmen
26:        end if
27:        Scope ← KAnon
28:        row ← row ∪ KAnon ∪ Scope
29:        TR_KAnon ← TR_KAnon ∪ row
30:    end for
31: end for
32: return TR_KAnon
```

outsourcing instances from the perspective of one process as well as different process within CS. Assume that, given that a process BP_i has outsourced to a provider WS_i as part of a business process execution route, an adversary would like to guess (with probability p) how likely that the same outsourcing had occurred in other processes or other routes within the same process, and to which providers a process has outsourced.

We look into the cases where a process may outsource to the same provider or to different providers. We also look into the cases where an outsourcing occurred in two routes in two processes or two routes within the same process. A process may or may not go down a certain route (due to a conditional activity for example), but in each route it may outsource certain operations. In each process, we classify operations as either outsourced or inhouse. Fig 4.2 depicts a composite service CS_1 that uses an orchestration model of two business processes. The invocation

graph shows that both processes outsource to similar services. The problem occurs when different competitors, attempt to monitor the outsourcing behavior of each other and infer to whom other competitors are outsourcing or whom they are partnering with.

Listing 4.3 shows a BPEL configuration representing a scenario in which there are three Web service providers, A, B, and C that outsource to a client, who requests access to a customers data from each service in order to accomplish subsequent outsourced activities. Table. 4.1 explains each activity. One practical example of this could be several telemarketing companies (clients) requesting customer data access from several bank services. The example below shows one process and one client.

4.5.2 System Architecture

The proposed approach can be utilized either by developers and service administrators at design time or by the composition engine at runtime (Fig. 4.6). At design time, a developer designs an application that involves composition. At deploy time, the service admins implement the Web service logic and deploys it to the registry. They also specify the different parameters used by the K-Anonymity check algorithm, including K-Anonymity types. K-Anonymity types account for three different possible invocation scenarios on an operation invocation precedence graph. Namely, *PTOC* (All Operations Can Be Called (Pass-Through Operations Counted), *PTONC* (All Operations Can Be Called (Pass-Through Operations Not Counted)), and *EPOC* (Only Endpoint Operations Can Be Called) as we shall explain later.

At run-time, the composition engine parses BPEL or WSCL definitions for operation invocations depending on the composition style applied. Since in BPEL activities represent units of processing, the algorithm parses the BPEL definition and instead of getting all the operations it tracks the *invoke* and *receive* activities. Different processes interact with each other via partner link ports. It tracks partner links as indicators of the WSDL services with which processes interact and may involve outsourcing. It relies on language constructs at an individual process level to handle all possible scenarios. It parses Process execution flows of each process of CS looking for operation invocations including both *inhouse* and *outsourced* ones. It differentiates between *inhouse* and *outsourced* operations using *partner links*.

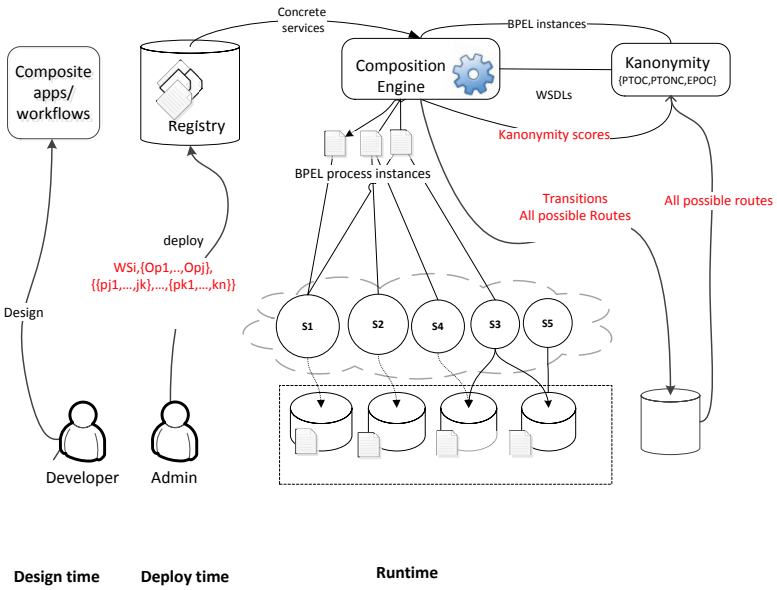


Figure 4.6: A framework for private Web service outsourcing.

4.5.3 K-Anonymity At the Operation Invocation Level (Revisited)

We leverage the notion of k-anonymity to the level of Web service operation. We define k-Anonymity of an operation as the value that determines the extent to which an operation invocation can be inferred if one knows that a downstream operation was invoked. To determine k-anonymity scores, we use operation invocation route configuration stored in Operation Invocation Precedence Graph (OIPG) definitions (e.g., WSCL).

Definition *Let $G = (Op, TR)$ be be a directed operation invocation precedence graph, OIPG, that consists of n operations (vertices), Op, and m transitions (edges), TR. Each vertex is an operation $\in Op$ and each edge is a pair of source/destination operations (Op_i, Op_j) where $Op_i, Op_j \in Op$. For each source operation $Op_i \in TR$, there exists a k-anonymity score k if, for each destination (downstream) operation Op_j of Op_i, there are at least another $k-1$ occurrences of that operation. Op_i is called k-Anonymous with respect to Op_j.*

Definition *An end point operation in an OIPG is an operation that is not invoked by any other operations or that does not invoke a downstream operation.*

4.5.4 K-Anonymity At the Business Process Level

We define K-Anonymity as the extent to which outsourcing an operation (by a service) in one process (route) can be inferred if one knows that an outsourcing of that operation has occurred in another process. Formally:

Definition *Let G be a graph with several components $G_i, \ldots, G_n$ corresponding to a set of business processes $BP = \{BP_1, \ldots, BP_n\}$ in a composite service CS. Each component G_i consists of a subset of invoke activities corresponding to inhouse Web service operations $Op = \{Op_{i,1}, \ldots, Op_{i,m}\}$ in BP_i, and another that corresponds to outsourced operations $OS = \{OS_1, \ldots, OS_w\}$ in BP_i. For each process $BP_i \in BP$ there exists a k-anonymity score k with respect to an outsourced operation OS_w, if for that outsourced operation there exists at least $k-1$ other outsourcing occurrences of that operation. There exists also a score k with respect to an inhouse operation $Op_{i,j}$, if for that operation there exists at least another $k-1$ occurrences of that operation. BP_i is called k-Anonymous with respect to OS_w or $Op_{i,j}$.*

4.5.5 K-Anonymity Score calculation for Inhouse Operations

To determine k-anonymity scores, we do the following:

Table 4.11: TR table (left) from the graph in Fig. 4.2 and $T_{EERouts}$ table derived from it.

Opi	Opj
Op1	Op3
Op2	Op3
Op3	Op4
Op3	Op5
Op6	Op5
Op5	Op7
Op8	Op7
Op9	Op7

Opi	Opj
Op1	Op3
Op2	Op3
Op3	Op4
Op3	Op5
Op6	Op5
Op5	Op7
Op8	Op7
Op9	Op7
Op1	Op5
Op2	Op5
Op2	Op5
Op3	Op7
Op6	Op7
Op6	Op7
Op6	Op7
Op6	Op7

- For each CS, for each BP, the algorithm generates source/destination operation transitions by parsing transitions in WSCL files (Table 4.11) or invoke activities in execution flows defined

in BPEL files (Table 4.12).

- From those transition tables, the algorithm generates a table of all possible end-to-end routes $T_{EERouts}$ (Tables 4.11 and 4.12).
- Each end-to-end route is then classified based on the number of intermediate operations along each source-destination path. Classes could be 0, 1, 2, 3, 4, etc. depending on the number of intermediate operations (column *midmen*, Table A.3). For example, in the 9-Operation WSCL graph in Fig. 4.2, the route $Op_1 \rightarrow Op_7$ consists of three sub routes $Op_1 \rightarrow Op_3$, $Op_3 \rightarrow Op_5$, and $Op_5 \rightarrow Op_7$. Thus, there are two middlemen: Op_3 and Op_5. So, this route falls under the 2-middlemen class.
- Then for each source/destination operation transition $Op_i \rightarrow Op_j$, the algorithm calculates a count of all the routes leading to the destination Op_j of the source operation being invoked ($|T_{Op_{jRouts}}|$). For example, for $Op_{1,1} \rightarrow Op_{1,5}$ in Fig. 4.2, $|T_{Op_{jRouts}}|$ for $Op_{1,5}$ is 1 (column $|T_{Op_{jRouts}}|$, Table A.3).
- To find the K-Anonymity score k, the algorithm subtracts from $|T_{Op_{jRouts}}|$ the number of routes that lead to the destination through that source operation $|T_{Op_{iRouts}}|$.

Algorithm 7 MidmenClassSupression

```
1: /*search for the row in EERoutes that has midmenOp as source and Opj as destination and they have a direct link*/
2: for each row_i in TR_EERoutes do
3:     if row_i[midmen] > 0 then
4:         for each op in row_i[midmenOp] do
5:             for each row_j in TR_EERoutes do
6:                 if op == row_j[Op_i] AND row_i[Op_j] == row_j[Op_j] AND row_j[midmen] == 0 then
7:                     /*mark that row for supression*/
8:                     row_j[Op_i] =' S'
9:                     row_j[Op_j] =' S'
10:                end if
11:            end for
12:        end for
13:    end if
14: end for
```

4.5.6 K-Anonymity Score Calculation for Outsourced Operations

To calculate K-Anonymity scores for outsourced operations, we take the Business process BP_i into account, in selecting the end-to-end routes to consider for the K-Anonymity score calculation of an *outsourced* operation. The reason for this is that an outsourced operation is common between the two processes, which may produce excessively more routes (thus introduce negative number of

OIPG				OIPG with 1 midman supression	
Opi	Opj	midmen	midman Op	Opi	Opj
1	3	0		1	3
1	4	1	3	1	4
1	5	1	3	1	5
1	7	2	3,5	1	7
2	3	0		2	3
2	4	1	3	2	4
2	5	1	3	2	5
2	7	2	3,5	2	7
3	4	0		S	S
3	5	0		S	S
3	7	1	5	3	7

routes). An inhouse operation (e.g., $Op_{1,1}$), on the other hand, is local to the current process (e.g., BP_1). For each outsourced operation, the score is k if there are at least $k-1$ routes pointing to that operation in another process (k occurrences of that operation). We generate two counts: one count from the perspective of OS_i as a destination and one from the perspective of OS_i as a source. Table 4.14 explains how K-Anonymity scores are calculated.

In our running example (Table 4.1), an competitor may be interested in the possibility that a competitor service provider had outsourced operation OS_2. They can use the fact that the provider had invoked $Op_{2,1}$ to infer that the competitor outsourced OS_2. In this case, the operation OS_2 is *2-Anonymous*, since it could have been invoked through at least two other routes, which is an indicator that at least two other providers have outsourced that operation.

4.5.7 K-Anonymity Types

In this section, we describe the three K-Anonymity types used by the algorithm to generate K-Anonymous (Uncertain) operation invocation precedence graphs, and the K scores or uncertainties associated with each. We apply the same calculations to both operation level and business process level K-Anonymity.

PTOC - All Operations Can Be Called (Pass-Through Operations Counted). For this K-

Table 4.12: Transitions table (left) from the graph in Fig. 4.2 and $T_{EERouts}$ table derived from it.

CS	BP	Opi	Opj
1	1	1,1	1,5
1	1	1,1	OS1
1	1	1,1	1,6
1	1	1,5	OS2
1	1	1,5	1,6
1	1	OS2	1,6
1	1	1,6	2,1
1	2	2,1	OS2
1	2	2,1	2,3
1	2	2,1	OS1
1	2	OS1	2,4
1	2	OS2	2,2
1	2	2,2	2,4
1	2	2,3	OS1

CS	BP	OPi	Opj
1	1	1,1	1,5
1	1	1,1	OS1
1	1	1,1	1,6
1	1	1,5	OS2
1	1	1,5	1,6
1	1	OS2	1,6
1	1	1,5	1,6
1	2	2,1	OS2
1	2	2,1	2,3
1	2	2,1	OS1
1	2	OS1	2,4
1	2	OS2	2,2
1	2	2,2	2,4
1	2	2,3	OS1
1	1	1,1	OS2
1	1	1,1	OS2
1	1	1,1	OS2
1	1	1,1	1,6
1	2	2,1	2,4
1	2	2,1	2,4
1	2	2,1	2,4
1	2	2,1	OS1

Anonymity type, we consider all operation invocations, including the intermediate operations that lead to the destination (i.e., middlemen). For the 9-Operation WSCL file, the following operations can be called: $Op_1, Op_2, Op_3, Op_4, Op_5, Op_6, Op_7, Op_8,$ and Op_9 and the operations that we pass through are part of the k-Anonymity score (column k_{PTOC}, Table 4.13). Similarly, for the BPEL in Fig. 4.2, the PTOC k scores are shown in Table A.3.

PTONC- All Operations Can Be Called (Pass-Through Operations Not Counted). For this K-Anonymity type, we start with the set of rows resulting from the PTOC stage (k_{PTOC}, Table 4.13 or k_{PTOC}, Table A.3). Then, as an additional step, we subtract the number of middlemen from the k-Anonymity score for the source operation Op_i for each route. The number of middlemen corresponds to the intermediary nodes that lead from Op_i to Op_j, and does not include the intermediary nodes that lead to Op_j through other source operations. For the 9-Operation WSCL file, the following operations can be called: $Op_1, Op_2, Op_3, Op_4, Op_5, Op_6, Op_7, Op_8,$ and Op_9, and the operations that we pass through are not part of the k-Anonymity score (column k_{PTONC}, Table 4.13). For example, in calculating k score for Op_1 in the invocation $Op_1 \rightarrow Op_4$, we exclude the route that passes through Op_3 from Op_1 but not the route that passes through Op_3 from Op_2.

Table 4.13: K-Anonymity scores derived from the WSCL configuration in Table 4.11

Opi	Opj	midmen	$\|T_{Op_{iRouts}}\|$	$\|T_{Op_{jRouts}}\|$	k_{PTOC}	k_{PTONC}	k_{OEOC}
1	3	0	0	2	2	2	2
1	4	1	0	3	3	2	2
1	5	1	0	4	4	3	3
1	7	2	0	7	7	5	5
2	3	0	0	2	2	2	2
2	4	1	0	3	3	2	2
2	5	1	0	4	4	3	3
2	7	2	0	7	7	5	5
3	4	0	2	3	1	1	-
3	5	0	2	4	2	2	-
3	7	1	2	7	5	4	-
5	7	0	4	7	3	3	-
6	5	0	0	4	4	4	4
6	7	1	0	7	7	6	6
8	7	0	0	7	7	7	7
9	7	0	0	7	7	7	7

Similarly, for the BPEL in Fig. 4.2, k_{PTONC} scores are shown in Table A.3.

EPOC- Only Endpoint Operations Can Be Called. This k-Anonymity type is a subset of the above case (PTONC), with k-Anonymity values for only endpoint operations. For the 9-Operation WSCL transactions in Fig. 4.2, only the following operations can be called: $Op_1, Op_2, Op_4, Op_6, Op_8,$ and Op_9, and operations that we pass through are not part of the k-Anonymity score (column k EPOC, Table 4.13). Similarly, for the BPEL in Fig. 4.2, only $Op_{1,1}, Op_{1,6}$, and OS_1 in BP_1 and $Op_{2,1}, Op_{2,4}$ in BP_2 can be invoked. The k_{EPOC} scores are shown in Table A.3.

4.5.8 K-Anonymity as a Measure of Uncertainty

We use k-Anonymity as a measure of uncertainty. The higher the K-Anonymity score the higher the uncertainty and the more private the invocation. Recall from our scenario, since some diseases can be inferred form the price of their medicine (e.g., cancer), the operation $getPrice(medicine)$ can be considered a private resource, since knowing that this operation has been invoked can reveal that $getMedicine()$ has been invoked. Similarly, $getMedicine()$ can reveal that $getDisease()$ has been invoked (Fig. 4.7). However, if at least one more operation $getMedicine()$ has been invoked, the probability of knowing that $getMedicine$ has been invoked through $getDisease()$ becomes 50%, which increases the uncertainty.

Definition A transition table TR consists of a multiset of source/destination pairs. An equivalence

Table 4.14: K-Anonymity scores derived from the BPEL configuration in Table 4.12

CS	BP	Op_i	OpType	Op_j	OpType	midmen	$\|T_{Op_{jRouts}}\|$	$\|T_{Op_{iRouts}}\|$	PTOC k	PTONC k	OEOC k
1	1	1,1	inhouse	1,5	inhouse	0	1	0	1	1	1
1	1	1,1	inhouse	OS1	outsourced	0	4	0	4	4	4
1	1	1,1	inhouse	1,6	inhouse	0	4	0	4	4	4
1	1	1,5	inhouse	OS2	outsourced	0	5	1	4	4	-
1	1	1,5	inhouse	1,6	inhouse	0	4	1	3	3	-
1	1	OS2	outsourced	1,6	inhouse	0	4	5			
1	1	1,5	inhouse	1,6	inhouse	0	4	1	3	3	-
1	2	2,1	inhouse	OS2	outsourced	0	5	0	5	5	5
1	2	2,1	inhouse	2,3	inhouse	0	1	0	1	1	1
1	2	2,1	inhouse	OS1	outsourced	0	4	0	4	4	4
1	2	OS1	outsourced	2,4	inhouse	0	5	4	1	1	1
1	2	OS2	outsourced	2,2	inhouse	0	1	5			
1	2	2,2	inhouse	2,4	inhouse	1	5	1	4	3	-
1	2	2,3	inhouse	OS1	outsourced	1	4	1	3	2	-
1	1	1,1	inhouse	OS2	outsourced	1	5	0	5	4	4
1	1	1,1	inhouse	OS2	outsourced	2	5	0	5	3	3
1	1	1,1	inhouse	OS2	outsourced	1	5	0	5	4	4
1	1	1,1	inhouse	1,6	inhouse	1	6	0	6	5	5
1	1	1,1	inhouse	1,6	inhouse	2	6	0	6	4	-
1	2	2,1	inhouse	2,4	inhouse	1	5	0	5	4	4
1	2	2,1	inhouse	2,4	inhouse	2	5	0	5	3	3
1	2	2,1	inhouse	2,4	inhouse	2	5	0	5	3	3
1	2	2,1	inhouse	OS1	inhouse	1	4	0	4	3	3

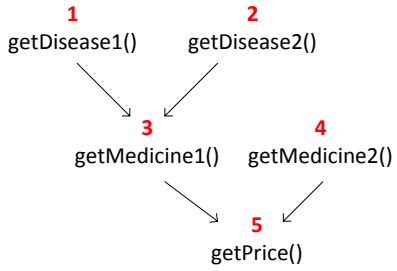
class for TR with respect to Op_j is the set of all pairs in TR containing identical values Op_j.

Definition K-Anonymity property states that Table TR is k-anonymous with respect to Op_i, Op_j if every unique pair (Op_j) in TR occurs at least k times. That is, the size of each equivalence class in TR with respect to Op_j is at least k.

We explain how the algorithm generates a probability (uncertainty) table depending on the k score for each operation. We can group the Op_j invocations in $TR_{EERoutes}$ table into equivalence classes based on the source operation Op_i. Then, we calculate the probability of a destination operation Op_j leading to (can be an indicator of) Op_i based on the other operations that also lead to Op_i. Thus, from the perspective of the i^{th} operation in $T_{EERoutes}$, the probability $Pr(Op_j \rightarrow Op_i)$ can be calculated as follows:

$$Pr(Op_j \rightarrow Op_i) = 1/|T_{Op_j \rightarrow Op_i}|$$

where $T_{Op_j \rightarrow Op_i}$ is the table of only the routes that has Op_i as their source operation. For instance, in Table 4.12, $Pr(Op_j = 1,5 \rightarrow Op_i = 1,1) = 1/7$ (14%) since there are 7 routes of it's equivalence class 1,1. Similarly, $Pr(Op_j = OS_1 \rightarrow Op_i = 1,1) = 1/7$, $Pr(Op_j = OS_2 \rightarrow$



Opsrc	Opdst
getDisease1()	getMedicine1()
getMedicine1()	getPrice()
getDisease2()	getMedicine1()
getMedicine2()	getPrice()
getDisease1()	getPrice()
getDisease2()	getPrice()

Opsrc	Opds	#routes to Opdst	#routes to Opsrc	Kanon	uncertainty	midmen
1	3	2	0	2	50%	0
1	5	4	0	4	25%	1
2	3	2	0	2	50%	0
2	5	4	0	4	25%	1
3	5	4	2	2	50%	0
4	5	4	0	4	25%	0

Figure 4.7: K-Anonymity as uncertainty from our scenario.

$Op_i = 1,1) = 3/7$, and $Pr(Op_j = 1,6 \rightarrow Op_i = 1,1) = 2/7$. Thus, for any equivalence class Op_i, we can generalize the probability of knowing that Op_i was invoked given that an operation Op_{dst} has been invoked to the formula:

$$Pr(Op_{dst} \rightarrow Op_{src}) = \frac{1}{n} \times \sum_{s=1}^{n} Pr(Op_{dst} \rightarrow Op_i), \text{ where n is the size of the equivalence class}$$

The resulting probability values represent uncertainty in detecting that an invocation occurred in a route given that an invocation had occurred in another route (Table A.3). If an invocation involves a single service/process invocation, an adversary can deduce with certainty 100% that an invocation or an outsourcing had occurred. However, if at least two processes are involved, then the probability will be 50%, and the uncertainty increases.

4.5.9 Anonymous (Uncertain) Operation Invocation Precedence Graph

We introduce the notion of an operation precedence graph as a generic abstraction for modeling relationships between Web service operation invocations in standards like BPEL and WSCL. We add the calculated uncertainty values as labels on the precedence graph edges to produce an *uncertain precedence graph*. Fig. 4.8 shows the precedence graph for the 9-Operation WSCL with uncertainty values added to edges. Figures. 4.9, 4.10, and 4.11 depict the precedence graph from our running example corresponding to the PTOC, PTONC, and OEOC K-Anonymity types, respectively. The graph includes uncertainty values on edges. For instance, the operation $OS2$ is $2 - Anonymous$, so this leaves the adversary with 50% uncertainty (two outsourcing invocations)

Table 4.15: Different Equivalence classes in $TR_{EERoutes}$ and their sizes (Derived from Table A.3).

equivalence class		size
Opi	**Opj**	
Op1	Op3	2
Op1	Op5	
Op2	Op5	3
Op2	Op5	
Op2	Op3	
Op3	Op7	3
Op3	Op4	
Op3	Op5	
Op5	Op7	1
Op6	Op7	5
Op6	Op7	
Op6	Op7	
Op6	Op7	
Op6	Op5	
Op8	Op7	1
Op9	Op7	1

Table 4.16: Routes in the $Op_i = 1, 1$ Equivalence class (derived from Table 4.14)

CS	BP	Op_i	Op_j
1	1	1,1	1,5
1	1	1,1	OS1
1	1	1,1	1,6
1	1	1,1	OS2
1	1	1,1	OS2
1	1	1,1	OS2
1	1	1,1	1,6

about the exact same process that outsourced the operation.

The uncertainty values on the precedence graph introduce several observations:

Observation 1 *The uncertainty decreases by excluding the routes leading to the source operation for which we calculate the K-Anonymity score. Since we calculate K-Anonymity from the perspective of the an outsourced operation by counting the number of other routes that lead to it's destination, we subtract the number of those routes that lead to the operation itself. For example, by subtracting the two routes that lead to the outsourced operation $OS2$ (Table A.3), the score*

Table 4.17: Routes in the OS_1 Equivalence class (derived from Table 4.14).

CS	BP	Op_i	Op_j
1	1	1,1	OS1
1	2	2,1	OS1
1	2	2,3	OS1
1	2	2,1	OS1

Table 4.18: Uncertainty probabilities derived from the End-To-End Routes table $T_{EERouts}$ (Table 4.12)

CS	BP	Op_i	Op_j	$\lvert T_{Op_j \rightarrow Op_i} \rvert$	$Pr(Op_j \rightarrow Op_i)$	$Pr(Op_{dst} \rightarrow Op_i)$
1	1	1,1	1,5	7	1/7	7/49
1	1	1,1	OS1	7	1/7	7/49
1	1	1,1	1,6	7	1/7	7/49
1	1	1,5	OS2	2	1/3	3/9
1	1	1,5	1,6	2	1/3	3/9
1	1	OS2	1,6	2	1/2	2/4
1	1	1,5	1,6	2	1/3	3/9
1	2	2,1	OS2	3	1/3	7/21
1	2	2,1	2,3	3	1/3	7/21
1	2	2,1	OS1	3	1/3	7/21
1	2	OS1	2,4	1	1	1
1	2	OS2	2,2	2	1/2	2/4
1	2	2,2	2,4	1	1	1
1	2	2,3	OS1	1	1	1
1	1	1,1	OS2	7	1/7	7/49
1	1	1,1	OS2	7	1/7	7/49
1	1	1,1	OS2	7	1/7	7/49
1	1	1,1	1,6	7	1/7	7/49
1	2	2,1	2,4	3	1/3	7/21
1	2	2,1	2,4	3	1/3	7/21
1	2	2,1	2,4	3	1/3	7/21
1	2	2,1	OS1	3	1/3	7/21

reduces from 4 to 2 which decreases the uncertainty from 25% to 50% thus increases the privacy risk.

Observation 2 *The uncertainty varies from route to another. For instance, in Table A.3, operation OS_1 has three scores corresponding to operations $Op_{1,1}$, $Op_{2,1}$, and $Op_{2,3}$. In particular, it is 4-Anonymous (25% uncertainty) with respect to operations $Op_{1,1}$ and $Op_{2,1}$ (since at least four other routes exist), while it is 3-Anonymous (33% uncertainty) with respect to $Op_{2,3}$. So, the route that passes through $Op_{2,3}$ is more private. The algorithm uses the lower uncertainty percentage as a threshold to guarantee a private composition. The notion of K-Anonymity threshold is explained*

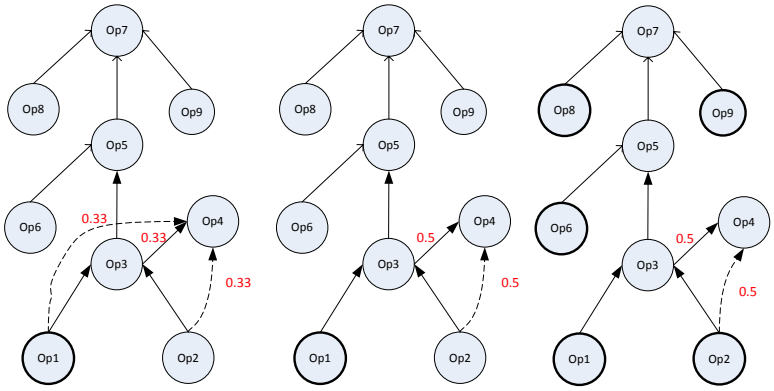


Figure 4.8: Operation precedence graphs for the three K-Anonymity types with uncertainties added to edge labels.

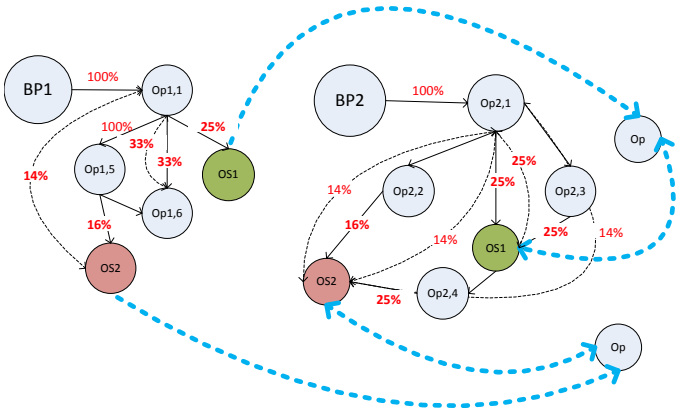


Figure 4.9: Operation precedence graph (PTOC) with uncertainties added to edge labels.

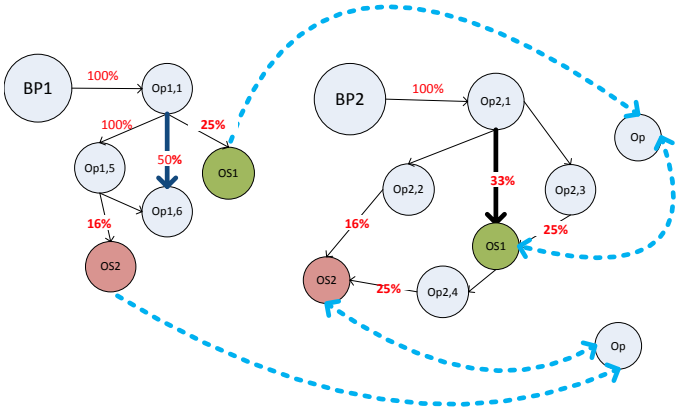


Figure 4.10: Operation precedence graph (PTONC) with uncertainties added to edge labels.

in detail in Sec. 4.5.11.

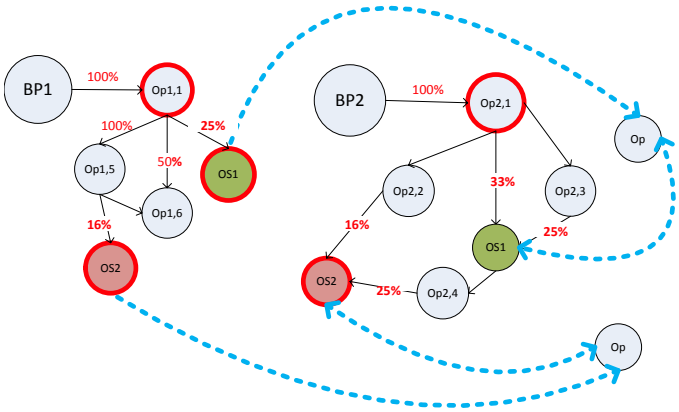


Figure 4.11: Operation precedence graph (OEOC) with uncertainties added to edge labels.

Observation 3 *The uncertainty value depends on the K-Anonymity type.*

For instance, the graphs in Figures. 4.9, 4.10, and 4.11 depict that the uncertainty decreases as we move from the K-Anonymity type PTOC to PTONC and from PTONC to OEOC. The reason for this is that when we move from PTOC to PTONC we limit the invocations to only direct invoca-

tions, so we exclude some edges (e.g., the indirect edge from $Op_{1,1} \rightarrow Op_{1,6}$). The uncertainty calculations exclude the routes that go through the middlemen that lead from Op_i to Op_j, and includes only the routes that pass through the middlemen that lead to Op_j through other source operations. For instance in the precedence graph of the 9-Operation WSCL (Fig. 4.8), the uncertainty for the invocation $Op_1 \rightarrow Op_4$ decreases (33% to 50%) since it excludes the route that passes through Op_3 from Op_1 and considers only the indirect route that passes through Op_3 from Op_2 and the direct route $Op_3 \rightarrow Op_4$. Same thing applies to the graphs corresponding to the BPEL graphs. Fig. 4.10 highlights the invocation edges that are included in the uncertainty value calculation for some operation pairs (e.g., $Op_{1,1} \rightarrow Op_{1,6}$). Also, when we move from PTONC to OEOC we get limit the number of operations that can be invoked, so we exclude some operations. Fig. 4.11 highlights the operation nodes that are included in the uncertainty value calculation.

4.5.10 K-Anonymity Check Algorithm (Revisited)

For a composite service CS_i in which an invocation Op_0 initiates a business process BP_j, the K-Anonymity check algorithm (Algorithm 8) generates a transitions table TR from the execution flow in BP_j. Then, it traverses the resulting table to produce $TR_{EERoutes}$. The algorithm initializes k score of all operations to 0 and the corresponding thresholds to ∞. Then, it iterates the $TR_{EERoutes}$ table, and for each operation the algorithm calculates k. It then selects the minimum value of k as a threshold.

Table 4.19: Example of K-Anonymity rules that can be applied to Web service definitions.

WSi	BPi	Opsrc	Opdst	input	output	KAnon type	level	operation type	Threshold
CS1	1	Op1,1	OS1	d1	d2,d3	PLOC	process	outsourced	GTE2
CS1	1	Op1,2	Op1,4	d2	d5	PTONC	process	in house	GTE2
CS1	1	Op1,3	Op1,1	d6	d7	EPOC	process	in house	GTE2
CS1	2	Op2,4	Op2,2	d1	d2	EPOC	process	in house	∞
CS1	2	Op2,5	Op2,1	d7	d8	PTOC	process	in house	GTE4
CS2	-	Op1	Op2	d1	d5,d6	PLOC	operation	-	GTE3
CS2	-	Op2	-	d2,d8	d3,d4	PTONC	operation	-	GTE2
CS2	-	Op3	Op1	d1,d2,d4	d9,d3	EPOC	operation	-	GTE2
CS3	-	Op1	Op2	d1	d4,d7	-	data	-	-
CS3	-	Op4	Op2	-	d1,d2,d3	-	data	-	-

4.5.11 Private Web Service Outsourcing Model

After deriving k-anonymity scores, we can incorporate them into rules that govern the usage of business processes or operations for each composite service in the system. Table 4.19 shows a sample set of rules. According to our model, we define the following:

- Privacy level: resources could be data (BPEL variables or operation parameters), BPEL business processes, or WSCL operations.
- Operation type: in addition to privacy level being operation, we add the operation type as $\{$*outsourced,inhouse*$\}$ to indicate whether the invoked operation is local to the Web service or is an outsourced operation invoked on an external Web service.
- K-Anonymity threshold: Unlike traditional K-Anonymity, our approach accounts for all possible invocation routes, so one operation can have two different K-Anonymity scores with respect to two different source (invoking) operations along a certain route. Moreover, those scores vary within the same business process as well as across different processes. For instance, in the BPEL example in Table A.3, OS_1 has two scores corresponding to operations $Op_{1,1}$, $Op_{2,1}$, and $Op_{2,3}$. In particular, it is 4 anonymous with respect to operations $Op_{1,1}$ and $Op_{2,1}$ since at least four other routes exist, while it is 3 anonymous with respect to $Op_{2,3}$. Thus, our algorithm traverses the scores and determines a threshold value based $GTEX$ based on the minimum score $min(k)$. Thus, a service admin can choose among the set $\{GTE1, GTE2, GTE3,\ GTE4, GTE5, GTE6, GTE7, GTE8, GTE9,\ GTE10, \infty\}$. Thresholds indicate that the K-Anonymity value should be at least as large as this value. ∞ means that the operation does not invoke any downstream operations.

4.6 Prototype Implementation

The proposed K-Anonymity check algorithm can be incorporated into any service composition middleware. It could be integrated either in a composition development environment to assist developers during process design by performing compile-time process validation or in a run-time environment (*composition engine*) to perform run-time validation during process execution. We provide two implementations of our framework.

4.6.1 Design-time Validation System

We implemented the first system in Microsoft Visual C sharp using Visual Studio 2010. We used ASP.Net for our web application and Windows Communication Foundation (WCF) for Web service implementation. We used ASP.Net Development Server to deploy our Web application and Web services and we used SQL Server 2008 to store our database. Both SOAP and WSDL protocols were used. The prototype consists of the following subviews:

Main console is where the K-Anonymity check is performed. Through this view the user selects a Composite service, then all the business process that comprise the composition in that service are displayed. From those processes, the user can execute a composition. The system populates the K-Anonymity scores table for each operation. The results of the K-Anonymity check as well as the results of the composition are then displayed. For each operation in the table, the user is presented with information on whether there is an indicator of outsourcing in other routes. It also indicates the uncertainty.

Admin console on the Web service side allows the user to select a BPEL file and a k-Anonymity type. The file is then parsed and used together with the K-Anonymity type to calculate k-Anonymity scores for each of the defined operations. The system displays the results as well as the precedence graph of operation invocations.

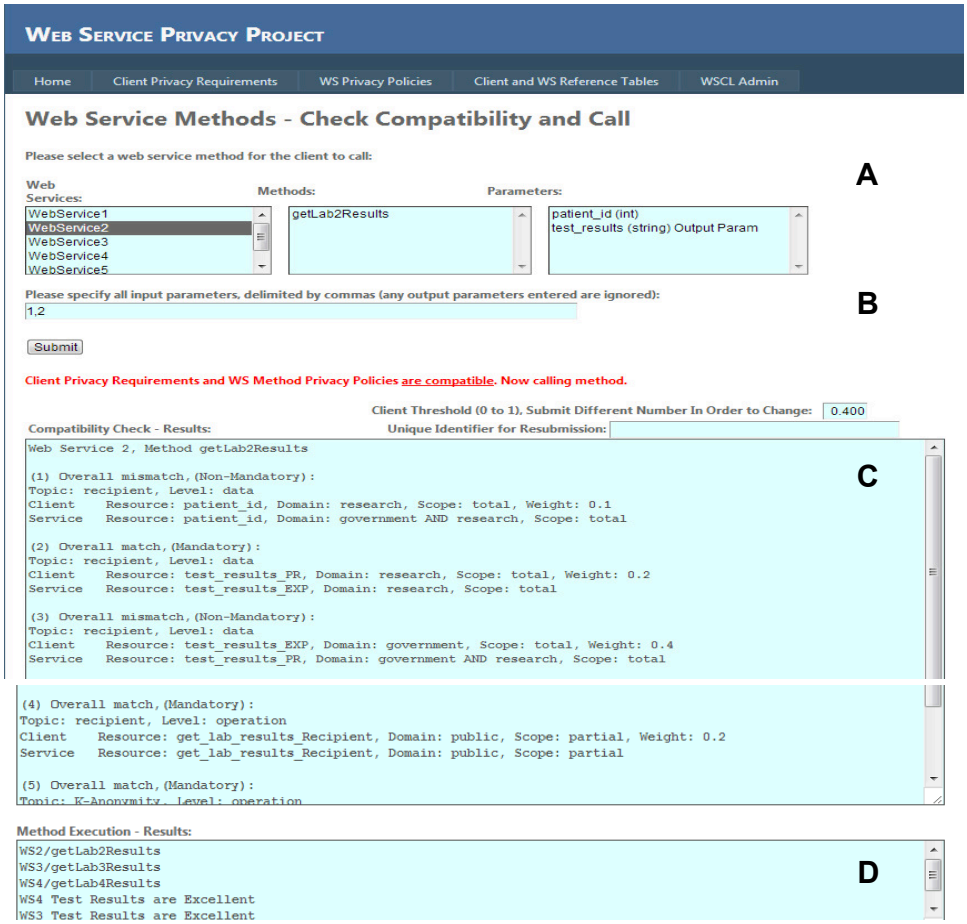


Figure 4.12: Prototype main view

Home (default) view is where compatibility check between PR and PP is performed. Through this

view the user selects a Web service, then all the operations for that service are displayed along with the corresponding parameters (Fig. 4.12, A). The user then enters the comma-delimited parameters and clicks submit (Fig. 4.12, B). Then the system populates client assertions array from the client database as well as the Web service assertions array for the specified operation. Then the system goes through each item in both arrays, looking for matches. The results of the compatibility check (Fig. 4.12, C) as well as the results of invoking the selected operation (Fig. 4.12, D) are then displayed. For each client item, the user is presented with information on whether there is an overall match or overall mismatch, and whether the matching is mandatory or non-mandatory; topic, level, resource, domain and scope, and client-specified weight are also shown. It also indicates whether simple or complex subsumption match was performed depending on the domain. Special output Messages from compatibility checks include:

(a) k-Anonymity scopes are identical (match)

```
(5) Overall match,(Mandatory):
Topic: K-Anonymity, Level: operation
Client    Resource: getLabResults_KAnon, Domain: other web services, Scope: GTE2 (K-Anon >
Service   Resource: getLabResults_KAnon, Domain: other web services, Scope: GTE2 (K-Anon >
```

(b) **Client k-Anonymity scope is less than WS (match)**

```
(5) Overall match,(Mandatory):
Topic: K-Anonymity, Level: operation
Client    Resource: getLabResults_KAnon, Domain: other web services, Scope: GTE1 (K-Anon >
Service   Resource: getLabResults_KAnon, Domain: other web services, Scope: GTE2 (K-Anon >
```

(c) **Client k-Anonymity scope is greater than WS (mismatch)**

```
(5) Overall mismatch,(Mandatory):
Topic: K-Anonymity, Level: operation
Client    Resource: getLabResults_KAnon, Domain: other web services, Scope: GTE4 (K-Anon >
Service   Resource: getLabResults_KAnon, Domain: other web services, Scope: GTE2 (K-Anon >
```

Figure 4.13: K-Anonymity compatibility messages

- K-Anonymity compatibility checks: in this case, three results can be returned depending on the WS and client scope values (Fig. 4.13). A match is returned if k-Anonymity scopes are identical or if a client's k-Anonymity scope is less than that of WS. A mismatch is returned if client's k-Anonymity scope is greater than that of WS.
- Threshold compatibility checks: if the Web service privacy matching degree is less than the threshold specified by the client, a mismatch is reported. In this case, the user is prompted to resubmit with three options. First, is to make a change in the area(s) of incompatibility

(i.e., client side assertions or threshold level) and resubmit without determining the GUID "Unique Identifier for Resubmission" value. This tells the system that the incentive is not desired. For example, the user can change the client threshold amount by entering a value in the field "Client Threshold (0 to 1)". Second, is to make a change in the area(s) of incompatibility, provide the GUID, and resubmit. This tells the system that the incentive is desired. Third, is to leave the areas of incompatibility unchanged, provide the GUID and submit in order to be presented with a different incentive retrieved from the Web service database. If incompatible again, another offer is made. The user can continue to resubmit until all incentives stored in the WS database are exhausted and no more offers are made.

WSCL Admin view on the Web service side allows the user to select a WSCL file and a k-Anonymity type (Fig. 4.14, A). The WSCL file is then parsed and used together with the K-Anonymity type to calculate k-Anonymity scores for each of the defined operations. The system displays the results as well as a choreography of operation interactions and transitions (Fig. 4.14, B).

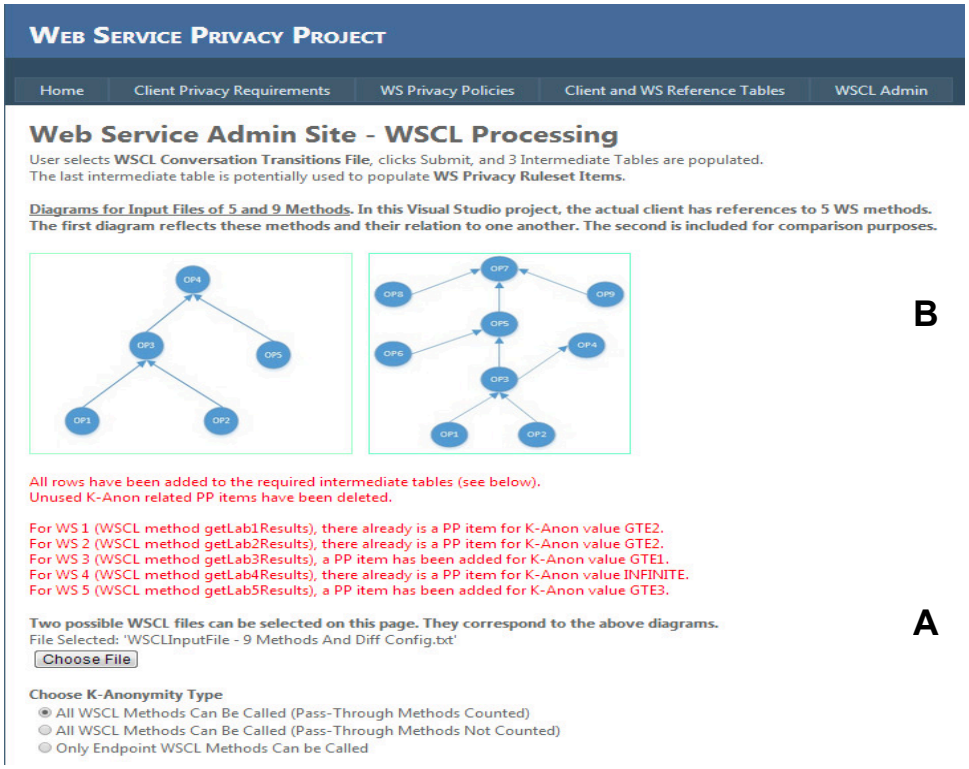


Figure 4.14: WSCL admin view

Table 4.20: Examples of Web Service and Client Assertions. W and M are only in the case of client assertions.

WS	**operation**	**resource**	W^*	M^*	**R**	**T**	**L**	**D**	**S**
1	getLab1Res	get_lab_results	0.1	No	1	recipient	data	public	total
1	getLab1Res	get_lab_results	0.1	Yes	3	K-Anon	op	services	GTE2
1	getLab1Res	patient_id	0.1	Yes	1	recipient	data	govrnmt	total
1	getLab1Res	test_results_EXP	0.1	No	1	recipient	data	govrnmt	total
1	getLab1Res	test_results_PR	0.1	Yes	1	recipient	data	research	total
2	getLab2Res	get_lab_results	0.2	yes	2	recipient	op	public	partial
2	getLab2Res	get_lab_results	0.1	yes	3	K-Anon	op	services	GTE1
3	getLab3Res	get_lab_results	0.2	yes	2	recipient	op	public	partial
3	getLab3Res	get_lab_results	0.1	yes	3	K-Anon	op	services	GTE4

Client and Web Service Reference View serves as a client and WS assertion configuration panel. Each Web service provides the user with the ability to select from a set of preferences whenever they want to add a *WS PP Assertion* or a *Client PR Assertion* (which is identical to a WS assertion except for the mandatory flag M and the weight W) (Table 4.20). The newly added row then becomes available in the lists of client PR and WS PP views .

Client Privacy Requirements (PR) view is located on the client side. Using this view, a user first selects from the full set of privacy rule constituent items that can be assigned to a client, assuming these choices have been added via the Client and WS Preference view. When an individual item is chosen, it is added to the list of entitled privacy rule items for that client. Items on this list are now available to be used in client assertions (PRs). One can add to the list of Client Assertions by selecting a client privacy rule item, the Web service, and the desired operation. The set of client assertions is later compared with Assertions for WS via the Home (default) view for compatibility check.

WS Privacy Policies (PP) view is located on the Web service side and the client obtains access to it by invoking an operation on the selected Web service. This view displays a list of PP items for a WS. This way, the k-Anonymity information reflected in the WSCL file is made available to be used for defining WS PP assertions.

4.6.2 Run-time Verification System

We implemented the second system in Java using Eclipse and SQL server as our database server. Since most of the K-Anonymity check code is implemented as stored procedures we invoked those procedures from our code. We incorporated the K-Anonymity check algorithm into the composi-

tion middleware provided by the WSO2 business process server (WSO2 BPS) [7]. WSO2 BPS provides a comprehensive web-based console to manage, deploy, view and execute processes within a single server instance. WSO2 BPS implements an Apache ODE-powered BPEL engine. WSO2 BPS supports the BPWS4J [34] implementation of the IBM, Microsoft and BEA BPEL4WS specification. The BPWS4J platform is an Eclipse plug-in that consists of an engine and an editor. The BPWS4J engine takes the BPEL document for each process to be executed, a WSDL description of the interface that the process presents to the external clients or service partners (without binding information), and several WSDL documents of the partner services with which the process may interact. We extended the RESTful management APIs to include the K-Anonymity checks.

4.7 Experiments

4.7.1 Environment

We conducted the experiments using a 2.83GHz Intel Quad-Core CPU, 8GB RAM, 32bit Windows Server 2007 machine. We ran Jmeter on the same machine with a memory foot print of -Xms1024m -Xmx2048m -XX:MaxPermSize=1024m. We used an SQL server 2008 instance and a WSO2 BPS server 3.5 instance running on the same machine.

4.7.2 Jmeter Test Plan

We configured Apache Jmeter 2.8 with threads to simulate 100 concurrent users, starting a new thread and sending parallel XML documents (WSCLs and BPELs) every 30 seconds (by setting the ramp up period to 3000 seconds) for varying number of documents. Since the WSO2 BPS is exposed via a Web service, we configured Jmeter to call that service API. We imported 30 BPEL files associated with 10 services into the BPEL store using the WSO2 BPS API. Since BPS uses caching to improve the performance, each time we ran the tests we loaded BPS with different BPELs stored in a CSV file. We used a transaction controller with a SOAP/XML-RPC sampler to test WSCL and BPEL K-Anonymous graph generation (Table. 4.21).

4.7.3 Data Set

We investigated 2507 real world services from the QWS data set [13]. Since the dataset does not take service composition into account, we generated random service compositions from the available services. We used the BPELUnit Framework [72] to generate BPEL compositions while we implemented our own WSCL generator. We generated synthetic compositions to simulate cases

Table 4.21: Jmeter test plan setup

BPEL version	3.0
BPEL Engine	WSO2 BPS
Database	SQL server
Threads (Users)	100
Iterations (no. of test runs)	10
Ramp-up time	3000 sec
Transport	SOAP over HTTPS
Sampler(s)	Soap/XML-RPC and HTTP Request
Services	10
BPEL files	30
Operations/BPEL	20

in which an operation appears as both a source and destination along a certain route. To generate a set of n composition documents (BPEL and WSCL) we first generated k core documents, with m operations each, such that all operations within a single document, as well as across multiple documents are different. Then, we permuted the $n - k$ remaining documents from the k previous ones. We generated 30 different WSCL documents ($WSCL_{01}, \ldots, WSCL_{30}$, Table A.4), each of which defines operations identified by 20 collaborating services. We generated 30 BPEL documents ($BPEL_1, \ldots, BPEL_{30}$, Table A.5), each of which defines a process identified by 15 inhouse, 5 outsourced operation invocations, and 10 variables from several collaborating partner services from the QWS service repository. Each generated BPEL document started with a *receive* activity and ended with a *reply* activity and was generated using a nested structure of *flow* and *sequence* activities, each of which included a set of random number of *invoke* activities. The services and their operations were randomly selected such that there is overlap between the services used in each document. The parameters of each generated document are shown in Tables A.4 and A.5.

4.7.4 Variables and Metrics

The performance of our K-Anonymity check algorithm depends on the size of the composition. We measure the size of the composition in terms of both number of Web services involved in a composition $|WS|$ which is proportional to the number of operations $|Op|$. We also measured the size of the composition in terms of the number of operation invocations $|Transitions|$ and the total possible routes $|TEERoutes|$. We analyzed the evaluation *time*, *throughput*, and *scalability* of the K-Anonymity check algorithm as the above variables increase and we compared the results among the different K-Anonymity types (PTOC, PTONC, and OEOC).

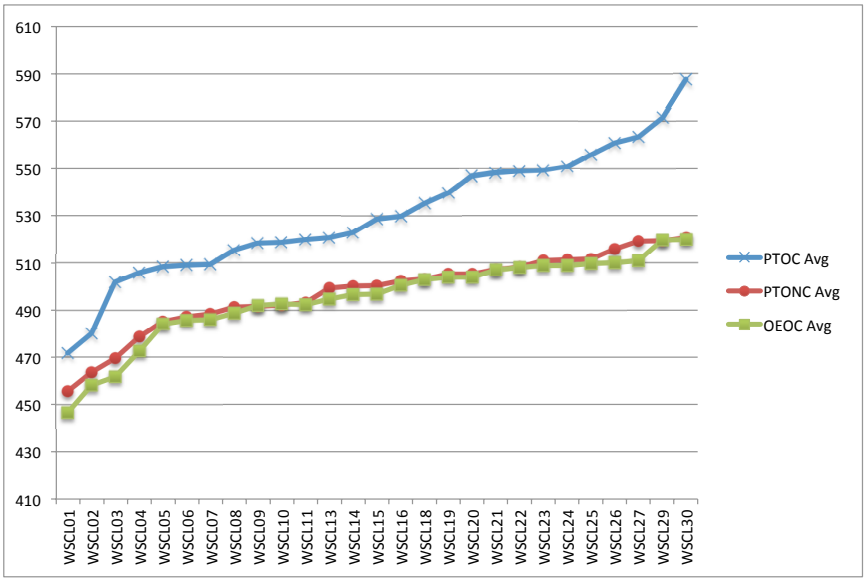


Figure 4.15: Evaluation time of different WSCL compositions

4.7.5 Results

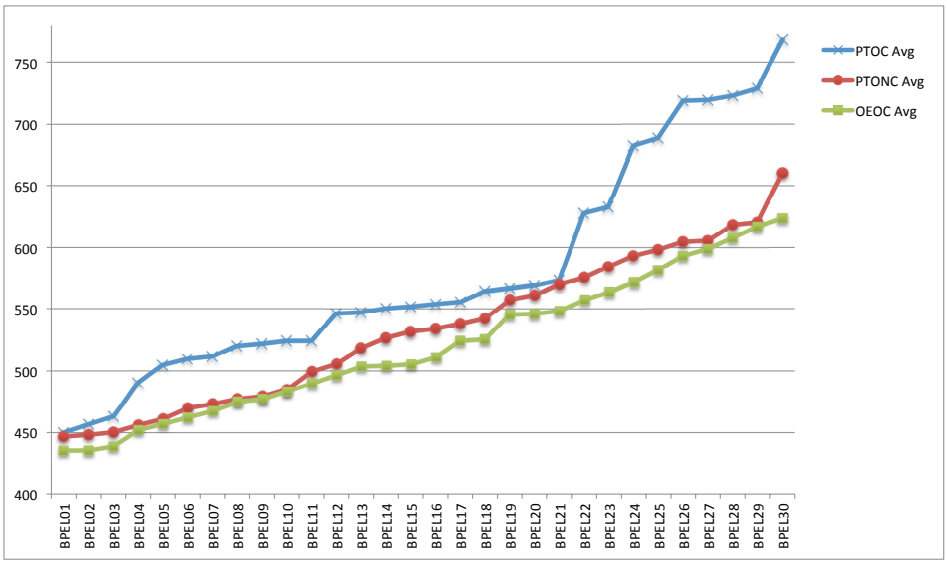


Figure 4.16: Performance results of different BPEL compositions

K-Anonymity check time. The K-Anonymity check algorithm (algorithm 8) depends on $|TR|$ (N) and $|T_{EERoutes}|$ (M). The time complexity is $O(M^2)$ since $M \geq N$. The results show that the time grows linearly with the number of routes regardless of the size of the composition ($|Op|$, which is equivalent to the number of services as per our assumption). for both WSCL (Fig. 4.15) and BPEL (Fig. 4.16) compositions. Table. A.1 illustrates that the time different is due to the number of routes that have to be traversed by the algorithm in each case.

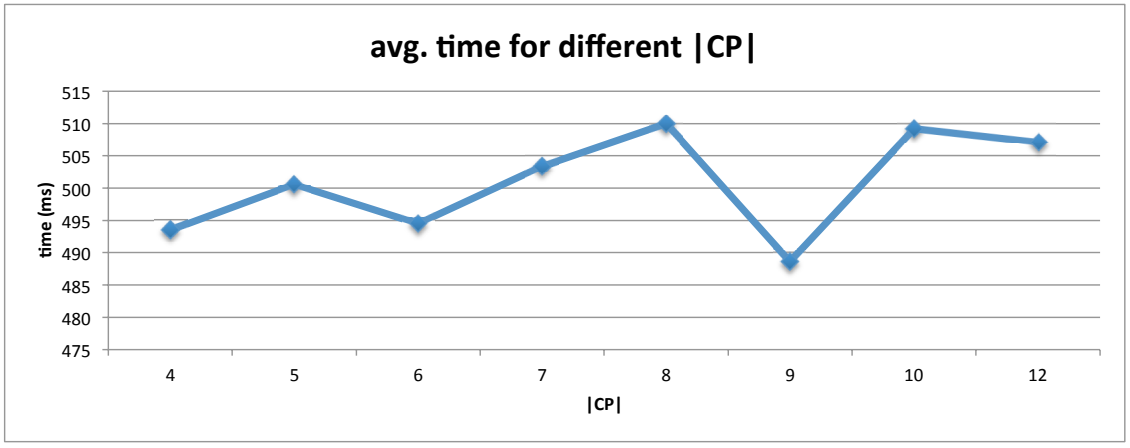


Figure 4.17: Performance results of different CP sizes.

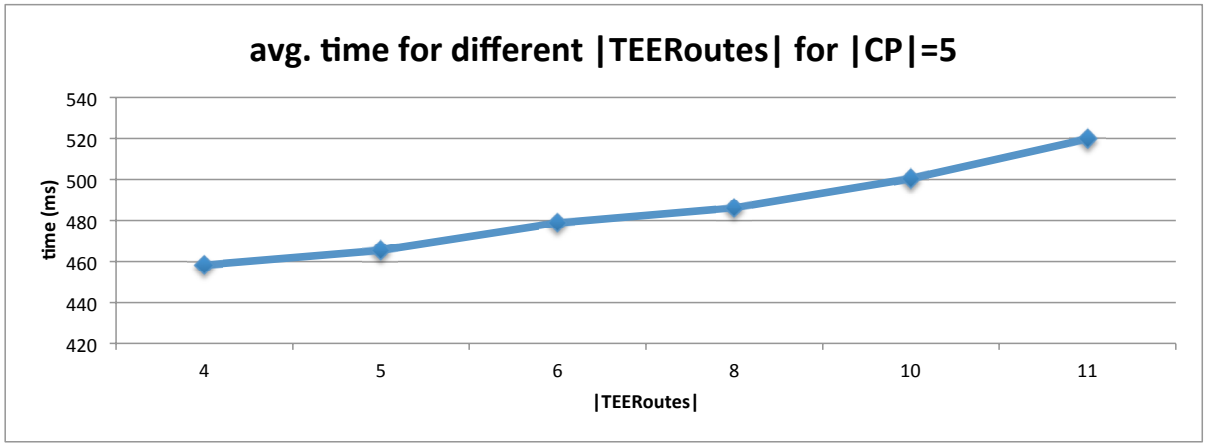


Figure 4.18: Performance results of different $TR_EERoutes$ table sizes for a fixed CP of size 5.

Privacy compatibility check time. We evaluated the feasibility of our approach with k-anonymity included (KPCM). The performance of the $KPCM$ algorithm depends on the total number of assertions that must be compared and the size of the composition plan CP. The number of comparisons among a PR set containing N assertions and a PP set containing M assertions in a composition plan CP of size $|CP|$ is equal to N x M. We analyzed the scalability of our $KPCM$ as the size of PR and PP increases in terms of both the number of resources and number of assertions as well as the size of CP. We generated several composition plans in which we varied the number of services ($|CP|$). We measured the performance of KPCM as the composition CP size increases in terms of the number of services. We also measured how the time changes as the size of PR and PP increase in terms of the number of assertions $|A|$.

The time requirement grows almost linearly with the number of services, and we expect the KPCM to scale well on larger PR and PP sets. The reported time is the time to check the pri-

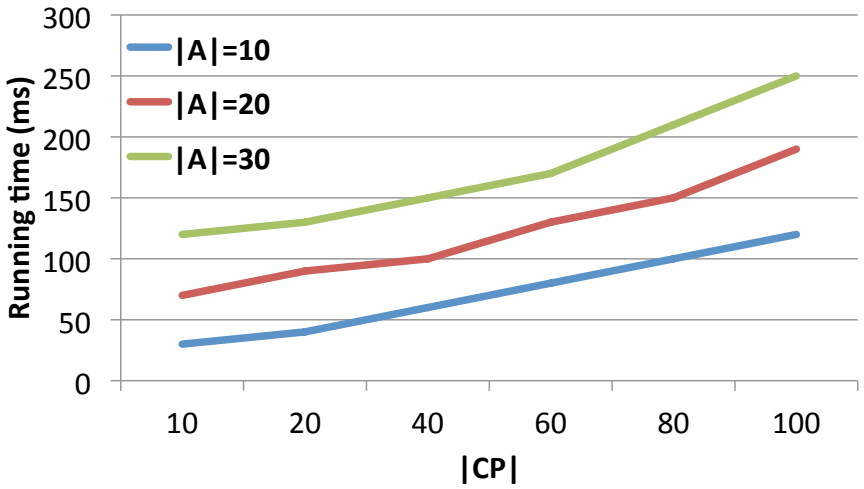


Figure 4.19: KPCM Performance

vacy compatibility of PR and PP of all services in CP. The running time of the algorithm has slightly increased from 135ms for $|A|$=10 to 250ms for $|A|$=30, when 100 services were processed (Fig. 4.19).

4.8 Conclusion

WSCL and BPEL definitions can play a very important role in determining privacy compatibility between a client and a Web service at the operation invocation and business process levels. We presented formal definitions and prototype implementations of two approaches: one that integrates k-Anonymity scores calculated from Web service compositions defined in WSCL and BPEL into a privacy management framework.

Algorithm 8 KAnonymityCheck

input $CS_i, BP_j, Op_0, Otype, Ktype$
output: TR_{KAnon}
$TR \leftarrow$ GETTRANSACTIONS(BP_j)
$row \leftarrow \phi, T_{EERouts} \leftarrow \phi, T_{KAnon} \leftarrow \phi, T_{Opiroutes} \leftarrow 0, T_{Opjroutes} \leftarrow 0, count \leftarrow 0, k \leftarrow 0, K_\tau \leftarrow \infty$
/*get all end to end routes*/
$T_{EERouts} \leftarrow T_{EERouts} \cup TR$
for each row_i in TR **do**
    for each row_j in TR **do**
        if $row_i[Op_{dst}] == row_j[Op_{src}]$ **then**
            $midmen \leftarrow midmen + 1$
            $row \leftarrow (row_i[Op_{src}], row_j[Op_{dst}])$
            $T_{EERouts} \leftarrow T_{EERouts} \cup row$
        end if
    end for
    $T_{KAnon} \leftarrow T_{KAnon} \cup row_i \cup midmen$
end for
for each row_i in T_{KAnon} **do**
    for each row_j in T_{KAnon} **do**
        if $row_i[Op_{dst}] == row_j[Op_{dst}]$ **then**
            $T_{Opjroutes} \leftarrow T_{Opjroutes} + 1$
        end if
    end for
    $row_i \leftarrow row_i \cup T_{Opjroutes}$
end for
for each row_i in T_{KAnon} **do**
    for each row_j in T_{KAnon} **do**
        if $row_i[Op_{src}] == row_j[Op_{src}]$ **then**
            $T_{Opiroutes} \leftarrow T_{Opiroutes} + 1$
        end if
    end for
    $row_i \leftarrow row_i \cup T_{Opiroutes}$
end for
for each row_i in T_{KAnon} **do**
    for each row_j in T_{KAnon} **do**
        $count \leftarrow row_i[T_{Opjroutes}] - row_i[T_{Opiroutes}]$
        $row_i \leftarrow row_i \cup count$
        if $Ktype == PTOC$ **then**
            $k \leftarrow count$
            $row_i \leftarrow row_i \cup k$
        end if
        if $Ktype == PTONC$ **then**
            $k \leftarrow count - row_i[midmen]$
            $row_i \leftarrow row_i \cup k$
        end if
        if $Ktype == EPOC$ **then**
            $k \leftarrow count - row_i[midmen]$
            $row_i \leftarrow row_i \cup k$
        end if
    end for
end for
return T_{KAnon}

CHAPTER 5 PRIVACY FLOW ANALYSIS

In services-based data analytics environments, multiple services may be dynamically composed to enable scientific data discovery. This composition requires appropriate privacy mechanisms to guarantee the confidentiality of all participants along the composition chain. Since services on the Web interact through process executions that trigger operation invocations by providing input data and retrieving output data, a mechanism is needed to ensure that the data privacy is maintained as it flows between those services. To this end, we propose a static privacy flow analysis between the composition plan generation and its execution phases, that uses dataflow concepts from programming languages. We formally define our static analysis using *Pri*-calculus, which extends the *Pi*-calculus for process algebra with privacy level types for private flow enforcement.

Using cloud-based services for data analytics makes them increasingly data-intensive. Maintaining privacy in such environments is a challenging task, since that data could potentially be distributed among several organizations, some of which manages the data access and usage via a service. Privacy of health science data is of a major importance. For instance, scientific workflows are formalizations of complex scientific processes distributed among several services that need to be dynamically composed to enable scientific discoveries (data search and analysis tasks). Composition of such services requires appropriate privacy mechanisms to guarantee the confidentiality of all participants along the composition chain. Scientific workflows are different from business workflows in several ways. First, unlike business workflows, which tend to be controlflow oriented, scientific workflows are dataflow oriented [39]. Second, they are often based on service and domain ontologies to support semantic discovery of workflows (e.g., MyGrid tools and BioWeb [86]). In fact, the Linked Open Data (LOD) initiative have focused on transforming health science data into ontology-based repositories using RDF as a universal exchange language to support semantic discovery of data. In such repositories an ontology is defined in OWL format including *classes* (e.g., Disease) and *data type properties* to link instances of those classes to their data (e.g., hasMedicine). The Bio2RDF project [22] provides a good example as it incorporates data from several domain ontologies, including PharmKGB, DrugBank, and GeneCDS. Therefore, services in such environments can be characterized by their computation logic, their data repositories, and

input and output data types. Several service composition (orchestration and choreography) languages have been proposed, including OASIS BPEL4WS (BPEL) [14] and W3C WS-CDL [53]. BPEL is a result of a large industry consensus and can be used for creating composite services in the form of business processes in a centralized service orchestration manner. While BPEL has been proposed as a standard for business workflows, a few efforts have been made to add language abstractions (e.g., hierarchical composition) to the BPEL syntax to make it more expressive as a *Scientific Process Execution Language* [105, 90, 108].

Table 5.1: Web services involved in our scenario.

Service	**Dinput**	**Doutput**	**Op**	**Description**
GenomicWS	hasGene	{hasGeneLocation, hasOMIM, hasGene, hasGender, hasAge}	getGeneInfo(g)	Returns detailed gene info of patients who have gene g
	hasRS	hasRS, hasSNPLocation, hasGene, hasGender, hasAge	getSNPInfo(rs)	Returns detailed SNP info of patients who have RS rs
	hasTrait	hasTrait, hasRS, hasGene, hasRSLocation, hasGender, hasAge	getGenPhenAssoc(d)	Returns Genotype Phenotype associations of patients who have trait (disease) d
DrugWS	hasDrug	hasDrug, hasGene, hasSNP, hasCoMedication, hasGenderhasAge	getDrugInfo(m)	Returns Drug info of Drug m
ClinicWS	hasTrait	hasHGVS, hasDisease, hasOMIM, hasDate, hasTreatment, hasTestResult, hasGender, hasAge	getDiagnosis(d)	returns clinical details of trait t
PharmaWS	hasDrug	Dosage, hasDrug, hasGeneDrugAsso, hasGenPhenAsso, hasGender, hasAge	getPharmInfo(m)	returns pharamy info about a drug m
ClinicWS	hasTrait, hasDrug	hasName, hasAddress, hasGender, hasAge	getClinicPersonInfo(d,m)	
DemogWS	hasAge, hasGender	hasName, hasAddress, hasGender, hasAge	getPersonalInfo(a,gd)	

A typical chain of operation invocations on those services is shown in Table 5.1. Phenotypes such as diseases and traits are associated to genes or variants (SNPs). Assume a service GenomicWS that provides both variation phenotype and gene phenotype lists which serve as sources of phenotype associations. Genetic disorders can be identified by linking genotypes to phenotypes, so an adversary can obtain a list of Gene information by invoking the $getGeneInfo()$ operation which returns the data type properties $GeneLocation, OMIM,$ etc. of the class

$Gene$. To further analyze the results, he invokes the $getSNPInfo()$ operation which returns ($rsno.$, $Location$, $Gene$) of the SNP class. He can then invoke the $getGenPhenAssoc()$ operation which searches a *Phenotype Genotype Association* dataset by a phenotype *Trait* and returns ($Trait$, $rsno.$, $Gene$, $Location$) of the $PhenGenoAssoc$ class. The adversary can conduct the above analysis in any order. For example, he can search the *PhenGenAssoc* dataset by a phenotype *Trait*. Similarly, he can search the $Gene$ data set by Gene IDs retrieved from the association results. $OMIM$ provides a catalog for human genes and genetic disorders. Thus, he can link genes or SNPs to diseases contained in the OMIM disease dataset through the *OMIM* number. He can also search an *SNP* dataset by SNP rs numbers retrieved from the association results. Later, he analyzes the data using $DrugWS$ which queries a data set that combines a drug with drug target (e.g. a gene sequence) information. A sample data set has the properties ($DrugLabel$, $InteractingGene$, SNP, $CoMedication$). For example, Citalopram can be used for the treatment of major depression and is associated with Gene symbol GRIK2 and SNP rs2518224. The adversary can then consult $PharmWS$ to query a pharmacogenomics knowledge base that encompasses clinical information including dosing guidelines and drug labels, potentially clinically actionable gene-drug associations and genotype-phenotype relationships. Finally, he can confirm the desired victim names by cross-referencing the possible last names with public records of people of similar ages and locations. He consults $DemographicWS$ for that and he indicates the age and $gener$ to identify people who participated in Genomic studies and detect that those people has been diagnosed for a certain disease.

Listing 5.1: RDF instance from our scenario

```
<rdf:RDF xmlns:mc="http://www.michcare.com/michcare.owl#" >
<mc:Patient rdf:about="Alice">
<mc:hasDisease>depression</mc:hasDisease>
<mc:hasPrivacyLevel>H</mc:hasPrivacyLevel>
<mc:hasDisease>flu</mc:hasDisease>
<mc:hasPrivacyLevel>L</mc:hasPrivacyLevel>
</rdf>
```

5.1 Privacy Policies

Privacy policies govern the usage of data owner's information and are often deployed by a privacy management engine implemented by the hosting service. Using those engines, services hosting the RDF-based data repositories can tie each data type property of each data instance to privacy policy rules based on the data owner's privacy preferences. For instance a patient Alice may indicate that the fact that she has been diagnosed with a disease (hasDisease data type property) is *highly* private (hasPrivacyLevel, H) for the depression disease but not as such (hasPrivacyLevel, L) for flu (Listing 5.1). However, in a composite system, process executions rely on service interactions via operation invocations by providing data as input parameters and retrieving output data which is supplied to the following operation as input and so on. Privacy can be achieved on input data, output data, and operation invocation; and to guarantee privacy at operation level, it is essential to know if there is compatibility between the invoking service's *privacy requirements* and the *privacy policies* of the invoked service.

5.2 Composition Logic As Program Execution

Processes are based on constructs that are equivalent to declarations in most programming languages (Fig. 5.1). Similar to a class definition in object-oriented languages, a process provides generic service interfaces (e.g.,DrugWS) to be called (classes) and the variables to be used (fields). Actual instantiation of a service is similar to an object of a class. Each service can be viewed as a control structure describing it's behavior according to an interface to be able to communicate with other services. However, it is unpredictable how a composite system will behave at run-time. An instance of a service (e.g., S1) can be identified by an instance of a process definition (e.g., P1) providing a service located at S1. Mapping service interfaces to actual services is achieved through *Endpoint* references. In a BPEL-based workflow, for instance, an implemented GenotypePhenotypeAssociation process would have an instance for each actual query or analysis task being processed. Each instance has its own state which is captured in BPEL variables. Since multiple service operations can be invoked by different clients, several instances of the same process can run simultaneously. Analogous to Object-Oriented programming, in which several program instances may produce different execution traces, a service composition may produce several outputs even if given the same input and initial state. Thus, we can analyze a composite system as a

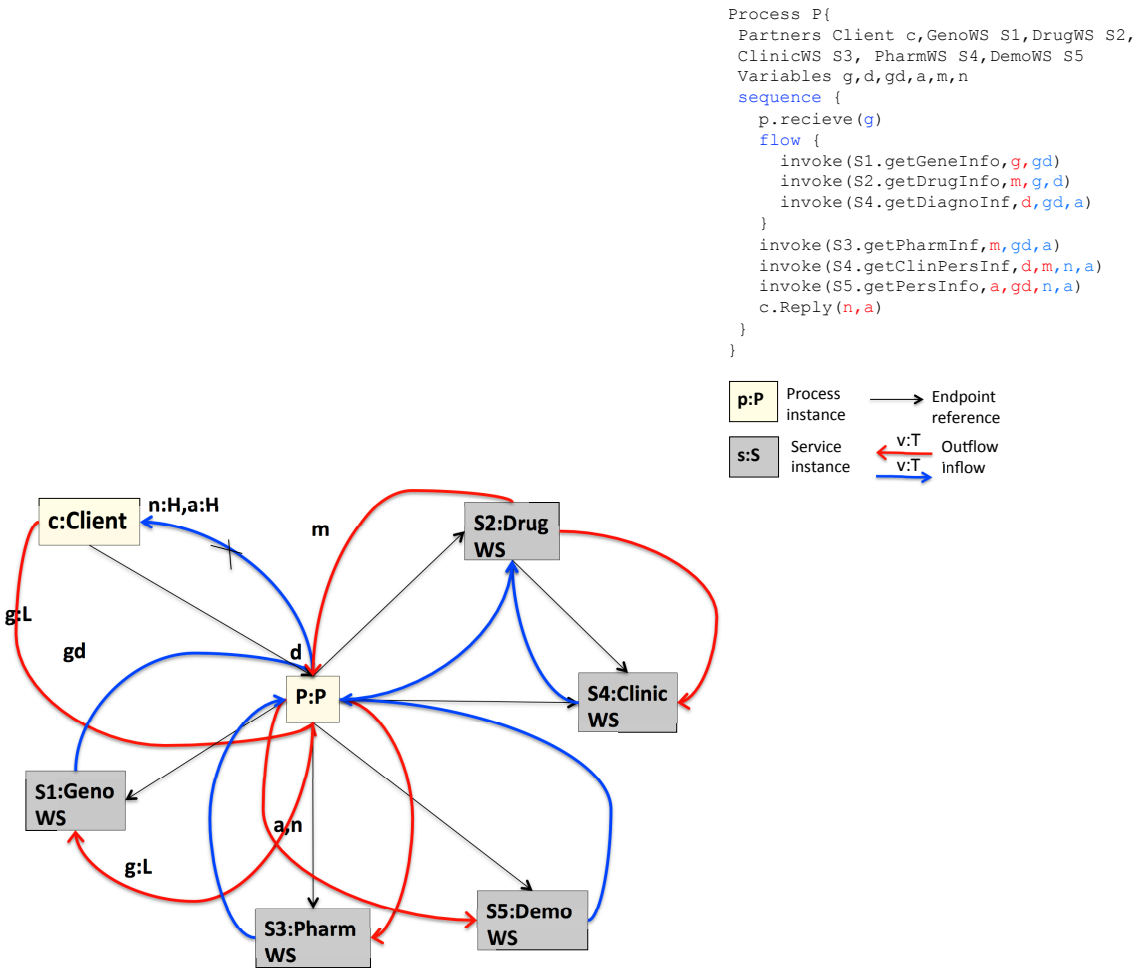


Figure 5.1: Composition as a program execution with dataflow communication links corresponding to variable read (recieve), variable write (assign) and operation invocation (invoke) expressions. The label g:L indicates a flow of variable g (due to operation invocation) with *low* privacy level. A prohibited flow link to the requesting client is due to the fact that the process is returning a variable with *high* privacy level.

program.

One form of program analysis is static analysis, which covers all possible execution scenarios of a program. Moreover, applying the analysis to different program instances (e.g. Main class in Java) may produce completely different results. Thus, we can use static analysis to analyze all possible executions of a *process*, and analyze a *composite system* in the context of each process instance. Since each process instance concerns particular set of data instances, and since those data instances can be tied to privacy preferences, we can extract privacy preferences from those policies and use them to supply the analysis (Fig. 5.2). For example, g:L in Fig. 5.1 indicates a flow of variable g with *low* privacy level. A prohibited flow link to the requesting client is due to the fact that the process is returning a variable with *high* privacy level. To this end, we propose

a static privacy flow analysis at the level of the composition engine, that uses data flow concepts from programming languages. The underlying framework is based on the Pi-Calculus for process algebra [75], which models a core for most service composition languages.

```
CompositeSystem {
 Client c, GenPhenAssoc GPA,
 GenoWS S1,DrugWS S2, ClinicWS S3,
 PharmWS S4, DemoWS S5
Process Client{
 S1.invoke(S1.getGenePhenAssocInfo,g:MH)
 receive(returnFinalResult,resultTuple)
  }
Process GenoPhenAssoc {
  Variables g:MH,d:MH,gd:<MH,MH>,a:MH,
            m:MH,n:MH
 sequence {
   recieve(S1.getGeneInfo,g:MH,gd:MH)
   flow {
    invoke(S1.getGeneInfo,g:MH,gd:H)
    assign(gd:H,gd:MH)
    invoke(S2.getDrugInfo,m,g,d)
    invoke(S4.getDiagnoInf,d,gd,a)
   }
   invoke(S3.getPharmInf,m,gd,a)
   invoke(S4.getClinPersInf,d,m,n,a)
   invoke(S5.getPersInfo,a,gd,n,a)
   c.Reply(n,a)
 }
}
```

```
<rdf:RDF mc=
http://www.michcare.com/genows.owl#>
<mc:Patient rdf:about="patient1">
 <mc:hasGene>GRIK2</mc:hasGene>
 <mc:privLevel>H</mc:privaLevel>
</rdf>
```

```
<rdf:RDF mc=
http://www.michcare.com/drugws.owl#>
<mc:Patient rdf:about="patient1">
 <mc:hasDrug>Citalopram</mc:hasDrug>
 <mc:privLevel>H</mc:privLevel>
 <mc:hasDrug>Actic</mc:hasDrug>
 <mc:privLevel>H</mc:privLevel>
  <mc:hasDrug>Cold drug</mc:hasDrug>
 <mc:privLevel>L</mc:privLevel>
</rdf>
```

```
<rdf:RDF mc=
http://www.michcare.com/clinicws.owl#>
<mc:Patient rdf:about="patient1">
 <mc:hasDisease>Cancer</mc:hasDisease>
 <mc:privLevel>H</mc:privLevel>
<mc:hasDisease>Cold</mc:hasDisease>
 <mc:privLevel>L</mc:privLevel>
<mc:hasDisease>Depression</mc:hasDisease>
 <mc:privLevel>H</mc:privLevel>
</rdf>
```

Figure 5.2: Composition as a program execution with privacy levels derived from policy files.

5.3 Contributions

This chapter's contribution is three-fold. First, we introduce Pri-calculus, as an extension of the Pi-calculus with privacy level types; Second, we define a data flow analysis for private service composition, including the lattice model, the abstract syntax, and dynamic semantics of our extended calculus; Third, we present a prototype implementation of our analysis as a privacy level annotations-based type checker and incorporate it into a composition engine;

The rest of this chapter is organized as follows: we provide background about the BPEL and WSDL standards used to model services and their composition. The proposed approach is then described in terms of the analysis formalization (Section 5.5). We present the prototype implementation in Section 5.6.

5.4 Business Process Execution Language

Business Process Execution Language (BPEL) describes business logic in terms of activities and interaction scenarios to be executed for a service to complete it's task. A service task could

be a reaction to another service invocation. Thus, a service can result from orchestrating other available services. BPEL enables inter-service communication via PartnerLinks, which provide a process with the ability to provide operations or invoke operations provided by other services. According to a BPEL describing the scenario above (Listing B.1), five participating services may be involved: GenomWS, DrugWS, and PharmaWS, ClinicWS, and DemoWS. Each service is expressed through a Web Services Description Language (WSDL) interface.

WSDL is a W3C standard that express the services functionality in terms of operation signatures as well as data variables used as parameters or return values by those operations. The roles within collaborations are specified at a PortType level via PartnerLink constructs. Port types are collections of web service operations. Listing B.2 shows excerpts from a WSDL definition. Variables in BPEL can be either entire messages, whose type is declared as a WSDL message in the WSDL description of the service using that message or a primitive XML type (e.g., string). Processes perform structured *activities* and primitive activities. Primitive activities include receive, invoke, and reply while structured activities perform primitive activities in *sequence*s, in *parallel*, or based on a *conditional*. Invoke and receive activities specify three arguments: a partner link identifying the partner service, the invoked or provided operation and a tuple of variables for storing the sent or received variable values. A process instance is created by triggering one activity at a time using the createInstance="yes" attribute. For instance, our example process gets instantiated through the receive activity.

BPEL performs service composition either *statically* or *dynamically* depending on the service selection style. Static composition involves *static binding* in which the service URL is hardcode as part of the composite service specification. *Dynamic binding*, on the other hand, can be done in two ways: either by reference by determining the URI of the service to be invoked from the value of a specified variable (e.g., assigned to a variable by a previously executed operation or passed by the client) or by lookup (e.g., definition of a query whose result will be used to determine the service to be invoked). PartnerLinkTypes are defined at the WSDL level, where the actual partner service may be dynamically determined. Since multiple service operations can be invoked by different clients, several instances of the same process can run simultaneously. Thus, responses to operation invocations need to be sent back to the corresponding service partner and to the corresponding

process instance provided by that partner. Therefore, WSDL interfaces can be directly compared to interfaces in Object-Oriented Programming languages, concrete services that result from dynamic binding can be compared to concrete classes implementing those interfaces, and an instance of a BPEL process can be used to resemble a main program.

The abstract syntax of BPEL (Fig. 5.3) expresses the composition of two or more activities using sequence, parallel, conditional, and alternative constructs. BPEL process syntax relies on sets of *ports* ranged over by x, y, z. An *activity* A receives a tuple $\tilde{u}$ and handles an action related to a *port* x. A port is a service identifier identifying a specific operation (end point) of that service. Thus, a port on a service S is expressed as x_S. A port receives operation invocation ($\mathsf{invoke}(\mathsf{x_S}, \tilde{\mathsf{i}}, \tilde{\mathsf{o}})$), input ($receive(x_S, i)$), or output ($reply(x_S, o)$) actions.

$$
\begin{array}{lll}
\mathsf{A} & ::=\mathsf{empty} & (empty) \\
 & |\mathsf{invoke}(\mathsf{x_S}, \tilde{\mathsf{i}}, \tilde{\mathsf{o}}) & (invoke) \\
 & |\mathsf{receive}(\mathsf{x_S}, \tilde{\mathsf{i}}) & (receive) \\
 & |\mathsf{reply}(\mathsf{x_S}, \tilde{\mathsf{o}}) & (reply) \\
 & |\mathsf{sequence}(\mathsf{A}, \mathsf{A}) & (sequence) \\
 & |\mathsf{flow}(\mathsf{A}, \mathsf{A}) & (parallel) \\
 & |\mathsf{switch}(\mathsf{x_S} = \mathsf{y'_S})\mathsf{A}; \mathsf{A} & (conditional) \\
 & |\mathsf{pick}(\mathsf{x}, \tilde{\mathsf{i}_1}, \mathsf{A}), (\mathsf{x}, \tilde{\mathsf{i}_2}, \mathsf{A}) & (alternative)
\end{array}
$$

Figure 5.3: Abstract Syntax of the BPEL calculus.

5.5 Formalization of the Analysis

In this section we describe the abstract syntax and the dynamic semantics of Pi-calculus (Sec. 5.5.1). The type system (Sec. 5.5.2) of our extended calculus is then provided. Finally, we describe our data flow analysis formalization (Sec. 5.5.3) illustrated through a series of examples from our scenario.

5.5.1 Pri-calculus Abstract Syntax

We formally define *Pri-Calculus* using the π-Calculus as a core of the BPEL language syntax. In Pi-calculus (Fig. 5.4), $P, Q, R, \ldots$ range over process names, processes interact over channels ranged over by $x, y, \ldots$, and exchange tuples ranged over by $\tilde{u}, \tilde{v}, \ldots$. Processes perform actions

$\beta_1, \beta_2, \ldots$. The syntax of processes and actions is given by:

$$P ::= (\upsilon x)P \mid P|P \mid \sum_{i \in I} \beta_i.Pi| \ A$$
$$\beta ::= 0| \ [x = y] \mid x(y) \mid \bar{x}[\tilde{u}]$$

Figure 5.4: Abstract Syntax of the Pi-calculus.

$(\upsilon x)P$ introduces the new name x with scope P, I is a finite index set, and β takes one of four forms: a null action, a match of two names, an output action, or an input action. $x(\bar{u})$ denotes receiving tuple $\bar{u}$ on channel x, $\bar{x}[\bar{u}]$ denotes sending tuple $\bar{u}$ on channel x. Summation states that only one element is selected from a set of pending communications while the others are aborted. Several processes can execute in parallel and communicate via compatible channels. Parallel process composition is denoted as $P|Q$. For example, $\bar{x}\ [u].P|x(v).Q$ contains two parallel processes. The first is ready to send u over the channel x; after this it continues executing P. The second, $x(v).Q$, is ready to receive v over the same channel x, then continue with Q. The name v is the formal argument and it is local to Q. Matching allows P to proceed only if channels x and y are the same channel. $!P$ denotes parallel composition of an infinite number of copies of P. Using Pi-calculus, we can describe our composition as follows:

$$(\upsilon\ \mathsf{c}, \mathsf{S1}, \mathsf{S2}, \mathsf{S3}, \mathsf{S4}, \mathsf{S5})(\mathsf{S1(g)}.\bar{\mathsf{S1}}[\mathsf{gd}]|\mathsf{S2(m)}.\bar{\mathsf{S2}}[\mathsf{g, d}]|$$
$$\mathsf{S4(d)}.\bar{\mathsf{S4}}[\mathsf{gd, a}].\mathsf{S3(m)}.\bar{\mathsf{S3}}[\mathsf{gd, a}].\mathsf{S4(d)}.\bar{\mathsf{S4}}[\mathsf{m, n, a}].$$
$$\mathsf{S5(a)}.\bar{\mathsf{S5}}[\mathsf{gd, n, a}].\mathsf{c(n, a)}$$

where the dotted notation specifies an action sequence in which the process receives the variable g on the channel c then sends it to channel S1 whose components are bound to the variable g, and so on.

The behavior of BPEL processes can be modeled using Pi-calculus dynamic semantics (Fig. 5.5). The first two rules state that we can reduce under both parallel composition and restriction. The communication rule R-COMM takes two processes, which communicate over a

channel x, and binds $v_1, ..., v_n$ (a.k.a., free names $f(n)$) to the variables $u_1, ..., u_n$ (a.k.a., bound names $b(n)$). Processes P and Q get discarded since at most one process in a summation is allowed to execute.

$$\frac{Q \mapsto R}{P \mid Q \mapsto P \mid R} \text{ [R-PAR]} \qquad \frac{P \mapsto Q}{(\upsilon x)P \mapsto (\upsilon x)Q} \text{ [R-REST]}$$

$$\frac{\tilde{u} = u_1, ..., u_n, \tilde{v} = v_1, ..., v_m, n = m}{(P + \bar{x}[\tilde{u}].Q)|(\bar{x}[\tilde{v}].R + Q) \mapsto \{\tilde{v}/\tilde{u}\}Q|R} \text{ [R-COMM]}$$

$$\bar{x}[\tilde{v}]|!x(\tilde{u}).P \mapsto P\{\tilde{v}/\tilde{u}\}|!x(\tilde{u}).P \text{ [R-REPL]}$$

$$\frac{x = y}{if\ (x == y)\ then\ P\ else\ Q\ \mapsto P} \text{ [R-BRANCH-T]}$$

$$\frac{x \neq y}{if\ (x == y)\ then\ P\ else\ Q\ \mapsto Q} \text{ [R-BRANCH-F]}$$

Figure 5.5: Dynamic semantics through reduction rules.

We introduce *Pri*-Calculus, which extends the *Pi*-Calculus abstract syntax (Fig. 5.4) with privacy level types. The metavariables $T_1, ..., T_n$ range over types, e ranges over expressions, and β ranges over actions. Types can be assigned to channel names (e.g., $x : T_i$) or variables in a tuple (e.g., $u_i : T_i$). The constituent parts of a process (i.e., variables, channels, operations, etc.) may have different privacy types at different locations in a process instance. Thus, we introduce the meta variable ℓ to range over locations. ℓ represents a result of computation, which eventually reduces to a value. Γ is a store typing that maps names to types. A store M maps locations ℓ to their contents: the defining service of the operation and the values stored in its variables. $M[\ell]$ denotes the store entry for ℓ. The store also holds the actual privacy level parameters for each location.

Privacy level parameters. Unlike java programs where classes are the basic unit of execution, in service composition there are two levels of definitions, the service definition and the process definition. We look at both processes and partners as services. Each service communicates through a port type and the communication link is identified by the partner link type. Also, each service provides an internal implementation of a process then that process gets instantiated and executed. Thus, there is a recursive process execution since composite services are themselves services and

P	$::= P \mid P$	(parallel composition)
	$\mid (\upsilon\ x : T_i)\ P$	(restriction)
	$\mid \sum \beta_i.P_i$	(alternative composition)
β	$::= \mathbf{0}$	(nill)
	$\mid [x == y]$	(Conditional)
	$\mid x(u_1 : T_1, ..., u_n : T_n)$	(input)
	$\mid \bar{x}[u_1, ..., u_n]$	(output)
e	$::= \beta \mid \ell$	(expression)
ℓ	$\in locations$	(store location)
T	$::= (T_1, ..., T_n)$	(privacy types)
$\mathbf{\Gamma}$	$::= \phi$	$(empty)$
	$\mid \mathbf{\Gamma}, x : T_i$	(typed name)
$\mathbf{M}$	$::= \ell \mapsto S < \bar{\alpha} > (\bar{v})$	(store Map)
$\mathbf{\Gamma}$	$::= \ell \mapsto T_i$	(store typing)
η	$::= x \mapsto T_i$	(type environment)

Figure 5.6: Abstract Syntax of Pri-Calculus (Pi-calculus with privacy types)

can be used as participants in other compositions at higher levels of abstraction. Thus, we parameterize both the process definition (BPEL process) and the service definition (WSDL interface) by a list of privacy level parameters $\bar{\alpha}$ that can be bound to actual values at run-time.

5.5.2 Privacy Levels Type System

The aim of the privacy levels type system is to formally encode privacy levels as types, and check whether a composite system is well-typed after binding abstract services in the orchestration to the corresponding concrete services in the service registry.

Typing and Subtyping Rules. A series of operations cause the value of a variable u to be derived from a value of another variable i. Thus derivation causes a flow from i to u.Typing rules guarantee private *explicit* (e.g., assign) and *implicit* (e.g., receive) flows. All services that would like to use users private data should possess the corresponding types before data transfer. We define type inference rules for making judgments to ensure private flow of data between services in a composition. We use those rules to formally reason on data privacy types and to propagate those types along the execution plan to newly computed data. Processes are well-formed under a set of privacy constraints on their bound $b(n)$ and free names $f(n)$. By adding types we restrict

a channel to certain types of input or output in a given context. The appropriate behavior of each participating process in a composite system can be ensured by statically type checking it in an environment η where the type of x allows only inputs of type T_1 that match it's type T.

The privacy levels type system defines rules that enable proving both typing judgments (i.e., $\eta \vdash e : T$), which indicate that an expression e has type T under environment η, and subtyping judgments (i.e., $T_1 <: T_2$). Data can flow only if both channel names or variables agree on their privacy levels or if the privacy level for one is a subtype of the other. We can define a subtyping relation $<:$ on types which states that an output action $\bar{x}[y]$ is well-typed only if $T_y <: T_x$, where $y : T_y$ and $x : T_x$. Similarly and input action $x(y : T)$ is well typed if $T_x <: T$ where $x : T_x$. The subtyping rule encodes the partial ordering relation $\sqsubseteq$ between the privacy levels lattice elements and the annotation semantics as we shall explain in Sections 5.5.3 and 5.5.3.

The typing rules (Fig 5.7) follow directly from the reduction rules, and express how they manipulate the privacy levels type system. All the typing rules are of the form $\eta, \Gamma \vdash e : Ok$, which means "in environment η and store typing Γ expression e is well-typed (is Ok). Parallel composition is well-typed if each participating process is well-typed in isolation. The summation behavior under typing is the same as Parallel composition. A replication is well typed if a single copy of the process is well-typed. An input expression $x(u_1 : T_1, ..., u_n : T_n).P$ is well typed if the type of x in the current context guarantees that a tuple of values read from x will have types or $<: T1, ..., Tn$. It is also necessary to check that the body of P is well-typed assuming that the channels $u1, ..., un$ behave consistently with the types $T1, ..., Tn$. Similarly, for an output expression $\bar{x}[u1, ..., un].P$ to be well-typed we check whether the process P is well-typed and that the type of x in η allows x to be used for outputting the tuple $\tilde{u}$. This implies that the type of x should be a subtype of the tuple of types $\Gamma(u1), ..., \Gamma(u2)$.

5.5.3 Private Data Flow Analysis

In this section, we define our data flow analysis. The lattice model and a more precise description of the core annotation system are provided by the formal semantics in Sec 5.5.3 and 5.5.3, respectively. Then, we describe the annotation-based transfer functions and the worklist algorithm.

The Privacy Levels Lattice Model. Our privacy levels lattice is a tuple $(L_{PA}, \sqsubseteq, \sqcup, \bot, \top)$, where

$$\frac{\eta, \Gamma \vdash P \ : \ Ok \ \ \eta, \Gamma \vdash Q \ : \ Ok}{\eta, \Gamma \vdash P \mid Q \ : \ Ok} \quad \text{[T-PAR]}$$

$$\frac{\eta, \Gamma \vdash P \ : \ Ok}{\eta, \Gamma \vdash !P; : \ Ok} \quad \text{[T-REPL]}$$

$$\frac{\eta, \Gamma, x : T \vdash P \ : \ Ok}{\eta, \Gamma \vdash (\upsilon x : T) \ P \ : \ Ok} \quad \text{[T-REST]}$$

$$\frac{\eta \vdash \Gamma(x) <: (T_1, ..., T_n) \qquad \Gamma, \ u_1 : T_1, ..., u_n : T_n \vdash P : Ok}{\eta, \Gamma \vdash x(u_1 : T1, ..., u_n : T_n).P : Ok} \quad \text{[T-INPUT]}$$

$$\frac{\eta \vdash \Gamma(x) <: (\Gamma(u_1), ..., \Gamma(u_n)) \ \ \Gamma \vdash P : Ok}{\eta, \Gamma \vdash \bar{x}[u_1, ..., u_n].P \ : \ Ok} \quad \text{[T-OUTPUT]}$$

Figure 5.7: Typing rules

L_{PA} is a set of abstract elements corresponding to privacy levels, $\sqsubseteq$ is a partial ordering relation between elements in L_{PA} and indicates an at least as precise as relation. $\sqcup$ is the least upper bound that should exist for every two elements in L_{PA}. $\sqcup$ merges two abstract values in L_{PA}, $\bot$ is the least element of L_{PA} and means part of the program (composition) that we have not yet analyzed. $\top$ is the greatest element of L_{PA} and means that we do not know anything about the program (composition) yet. Our lattice consists of the following:

$$L_{PL} = \{\bot, H, L, MH\}, \text{where } \top = MH$$

$$\bot \sqsubseteq H, \bot \sqsubseteq L, L \sqsubseteq MH, H \sqsubseteq MH$$

$$\bot \sqcup \mathsf{X} = \mathsf{X}, \top \sqcup \mathsf{X} = \top, \mathsf{X} \sqcup \mathsf{X} = \mathsf{X}, H \sqcup L = MH$$

Annotations for Private Data Flow. Annotations encode the privacy level types and have no semantic significance; they only serve as labels into the Pri-calculus language syntax to facilitate the analysis. A simplified annotation syntax is shown in Fig. 5.8. Annotations correspond to abstract elements in L_{PA}, so they may be H, L, MH, or a parameter α. Actual privacy level parameters on variables in the source process must be parameters of the enclosing service definition. During reduction, these parameters may be replaced with locations ℓ, indicating the variable that corresponds to that actual privacy level parameter. Annotations can be on channel names x as well as

tuples (variables) u. Examples of the concrete annotation language are shown in Sec. 5.6.

$$
\begin{array}{ll}
CS \in compositeSystem & ::=(\bar{P}, e) \\
P \in ProcessDefinition & ::= P < \bar{\alpha} > \{\bar{T}, \bar{S}, \bar{v}, \bar{\beta}\} \\
S \in ServiceDefinition & ::= S < \bar{\alpha} > \{\bar{x_S}\} \\
\beta \in ActionDeclaration & ::= 0 \mid [x_S 1 = y_S 2] \mid x_S(v) \mid \bar{x_S}[\tilde{u}] \\
T \in PrivLevelDecl & ::= \mathsf{privacyLevel}\ T_i \\
x_S \in ServicePortDecl & ::= x_S(\tilde{i}, \tilde{o}) \\
v, u, i, o \in Var & \alpha \in PrivacyLevelParam
\end{array}
$$

Figure 5.8: Simplified Annotation Syntax adapted from the formal system in Fig. 5.6.

Table 5.2 summarizes the constraints that the privacy type annotations place on value flow. The various annotations are listed along the left side and the top of the table. An X indicates that the data can flow from a location with the annotation on the left to a location with an annotation above. The table shows that `H` is a universal source (a variable with any privacy level value can be assigned a high privacy level value), and that `L` is a universal sink (variables with low privacy levels can be assigned a value with any type annotation). The other privacy level annotations must be kept separate from each other. The subtyping rule encodes the annotation semantics (Table 5.2), where H is a subtype of any other annotation, L is a supertype of any other annotation, and all other annotations must match exactly.

Table 5.2: Value flow between privacy levels

		To			
		H	**L**	α	**MH**
From	**H**	X	X	X	X
	L		X		
	α			X	
	MH		X		X

Abstract interpretation. A composition system CS is defined with a list of process definitions $\bar{P}$, a list of Web service interface definitions $\bar{S}$, and a main expression e (Fig. 5.8). A process definition gives the process a unique name P and defines it's constituent services as well as activities between those services and the variables that they exchange. A Web service definition gives

the service a unique name S and defines its operations and input and output variables and their types. The analysis starts with a root expression e_{root} that explicitly instantiates the root process P_{root} (the action associated with createInstance="yes"). The analysis starts by creating the process instance P_{root} from the main process definition then it abstractly interprets e_{root} in the context of P_{root}. First, the process creates all global channels that will be used by the participating processes. In our running example, a client initiates a process, which creates five channels with fresh names (υ GenomWS,PharmaWS, DrugWS, ClinicWS, DemoWS), sends the variable g along GenomWS, waits for response on GenomWS, then passes the output over to DrugWS, and so on.

The *abstraction function* maps each concrete expression e to an element in L_{PA}. A composition lattice in our case σ is a tuple lattice, and it is the set of all maps from variables to L_{PA}. $\sigma_1 \sqsubseteq_{PA} \sigma_2$ iff $\forall u \in Var\ \sigma_1(u) \sqsubseteq_{PA} \sigma_2(u)$. Also, $\sigma_1 \sqcup_{PA} \sigma_2 = \{u \mapsto \sigma_1(u) \sqcup_{PA} \sigma_2(u) | u \in Var\}$. For example, for the two tuple lattice values $[u \mapsto H, i \mapsto MH]$ and $[u \mapsto MH, i \mapsto L]$:

$$[u \mapsto H, i \mapsto MH] \not\sqsubseteq [u \mapsto MH, i \mapsto L]$$
$$[u \mapsto H, i \mapsto MH] \sqcup [u \mapsto MH, i \mapsto L] = [u \mapsto MH, i \mapsto MH]$$

In our case, the abstraction function derives abstract values from the privacy preferences attached to the data. *Flow (transfer) Functions* then compute dataflow information after executing an expression e from the dataflow information before executing that expression. Formally, we map a lattice element and a control flow graph (CFG) node corresponding to a process node to a new lattice element (i.e., $f_{PA}(\sigma, [e]) = \sigma'$), where σ is the old lattice, e is the expression that we are transferring over, and σ' is the new lattice. The intuition behind building a CFG is to connect nodes in order of operation defined by the language. For example, in a BPEL4WS conditional we evaluate the condition first.

We employ Killdal's *worklist algorithm* [55], which keeps track of nodes (expressions) to which we need to propagate dataflow information. The algorithm computes a results array, which keeps track of the dataflow value for each node i in the system. The algorithm initialize the lattice values for all nodes to the bottom element in L_{PA}. For the first expression to be executed in the process the results will be whatever the initial information is. We assume that initially any of the variables

Algorithm 9 Worklist Algorithm

```
1:  worklist = new Set();
2:  for all node indexes i do
3:      results[i] = ⊥_PA;
4:  end for
5:  results[entry] = ⊤_PA //MH;
6:  worklist.add(all nodes);
7:  while (!worklist.isEmpty()) do
8:      i = worklist.pop();
9:      before = ⊔_{k∈pred(i)} results[k];
10:     after = f_PA(before, node(i));
11:     if (!(after ⊑ results[i])) then
12:         results[i] = after;
13:         for all k ∈ succ(i) do
14:             worklist.add(k);
15:         end for
16:     end if
17: end while
```

could be highly private (i.e., may be, may be not). Next, the algorithm takes each node off the list and takes the join of the lattice values for all its predecessors. For instance, if it reaches the end of a conditional, the algorithm does a join on the lattice information obtained from the then branch and the else branch. So, if the privacy level type on a variable was H in one branch, and L in the other, the algorithm assumes that it is MH.

This provides the results for the lattice information obtained before executing a node. The algorithm then applies the transfer function f_{PA} to that before information, and the node that is at position i and computes the after lattice information based on that. Next, the algorithm determines how this information propagates in the system. If the results remain the same as they were before, it stops analyzing that particular node. However, if a change is indicated, it needs to add all the successors of this node to the worklist since they need to be analyzed with the new information. If the results got worse from before (i.e, they are not at least as precise as they were before), the algorithm updates the results array based on the newly computed after value, then it pushes all successors onto the worklist. The analysis terminates if all nodes are visited and the values do not change any more.

Table 5.3: Results of running the worklist algorithm (Listing 9) on the sample process execution in Listing 5.2

position	worklist	channels						tuples/variables					
		c	S1	S2	S3	S4	S5	g	d	gd	a	m	n
0	1	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH
1	2	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH
2	3	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH
3	4,5,9,10,11,12	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH	MH
4	5,9,10,11,12	MH	MH	MH	MH	MH	MH	**L**	MH	MH	MH	MH	MH
5	6,7,8,9,10,11,12	MH	MH	MH	MH	MH	MH	**L**	MH	MH	MH	MH	MH
6	7,8,9,10,11,12	MH	**L**	MH	MH	MH	MH	**L**	MH	**L**	MH	MH	MH
7	8,9,10,11,12	MH	**L**	**H**	MH	MH	MH	**MH**	**H**	**L**	MH	**H**	MH
8	9,10,11,12	MH	**L**	**H**	MH	**H**	MH	**MH**	**H**	**MH**	**H**	**H**	MH
9	10,11,12	MH	**L**	**H**	**H**	**H**	MH	**MH**	**H**	**MH**	**H**	**H**	MH
10	11,12	MH	**L**	**H**	**H**	**H**	MH	**MH**	**H**	**MH**	**H**	**H**	MH
11	12	MH	**L**	**H**	**H**	**H**	**H**	**MH**	**H**	**MH**	**H**	**H**	**H**
12		**H**	**L**	**H**	**H**	**H**	**H**	**MH**	**H**	**MH**	**H**	**H**	**H**

5.5.4 Working Example

We simulate the privacy flow analysis execution on the sample composition in Listing. 5.2. The labels 1,2,3,4,5,6,7,8,9,10,11,12 correspond to positions in the process execution. Table 5.3 shows the results of running the worklist algorithm (Listing 9) on a sample process execution (Fig. 5.1). First, we assume that all variables may have a high privacy level (MH). expressions 1 and 2 have no impact on the composition lattice since they are only declaration expressions. Expression 3 is a sequence activity, so it leaves the lattice as is. Next, the algorithm pushes all nested actions of that sequence on the worklist (i.e., expressions 4, 5, 9, 10, 11, and 12). Next, expression 4 initializes the gene variable g (assume that the client sent the value GRIK2 which has a low privacy level), so the algorithm changes it's lattice value to L. Expression 5 is a flow activity, so it leaves the lattice as is, and all it's nested expressions get pushed onto the worklist (i.e., 6,7,8). The next expression to be analyzed is the invoke expression at position 6, which takes g:L as input. Since the invoke is on S1, which is instance of GenomWS, the analysis propagates the privacy level type L to the subsequent operation invocations inside GenomWS (if any). Since S1 is the receiver of this invocation, it also gets assigned a privacy level type L. Also, the output variable gd gets assigned that value.

Listing 5.2: Example composition

```
[Partners Client c,GenoWS S1,DrugWS S2,ClinicWS S3,PharmWS S4, DemoWS S5]1
[Variables g,d,gd,a,m,n]2
```

```
sequence[
[ p.recieve(g)]4
flow[
[invoke(S1.getGeneInfo,g,gd)]6
[invoke(S2.getDrugInfo,m,g,d)]7
[invoke(S4.getDiagnoInf,d,gd,a)]8
]5
[invoke(S3.getPharmInf,m,gd,a)]9
[invoke(S4.getClinPersInf,d,m,n,a)]10
[invoke(S5.getPersInfo,a,gd,n,a)]11
[Reply(c,n,a)]12
]3
```

The main process continues with other executions to be instantiated, and the analysis continues analyzing each of those expressions based on the initial privacy level type on the received variable and propagating the privacy level types to the invoked services. Assume that the next invoke expression (expression 7) receives the value Citalopram for the medicine variable m with a high privacy level. The invoke expression takes m:H as input and propagates that to DrugWS. Since the operation returns the tuple g,d, and since g previously had an L lattice value, the algorithm takes the join of the two (i.e., L⊔ H=MH) and replaces the lattice value of g with this new value. For some of the following expressions the values flow from the previous results within the running process instance. For example, the disease input variable d in the invoke expression at position 10 is obtained from the output d of the invoke expression at position 7. Thus, it gets assigned the same privacy level type that d had. The analysis reaches a termination state whenever any further execution of the program does not change the composition lattice. The analysis may produce completely different results if the value of the supplied g variable was an indicator of a different disease (e.g.,flu).

5.6 Implementation

Fig. 5.9 depicts the architecture of the proposed system. In our scenario, a client initiates a business process then a service agent parses the abstract definitions of each of the collaborating services (S={GenomWS, PharmaWS, DrugWS, ClinicWS, DemogWS}) and looks up the

matching services to get the concrete BPEL composition (concrete processes). Each process instance communicates with the composition engine when it receives an action and when it finally replies. Upon receiving a query, a process is instantiated and the corresponding invoked services are looked up in the *service registry*. The run-time environment executes the service logic by invoking other services (through SOAP and HTTP modules). Upon receiving an operation invocation, each service looks up it's data repository for the matching instances (e.g. RDF files). Data owners specify minimum input by informing the hosting service about their privacy preferences. Thus, each data instance indicates a privacy level for each data type property.

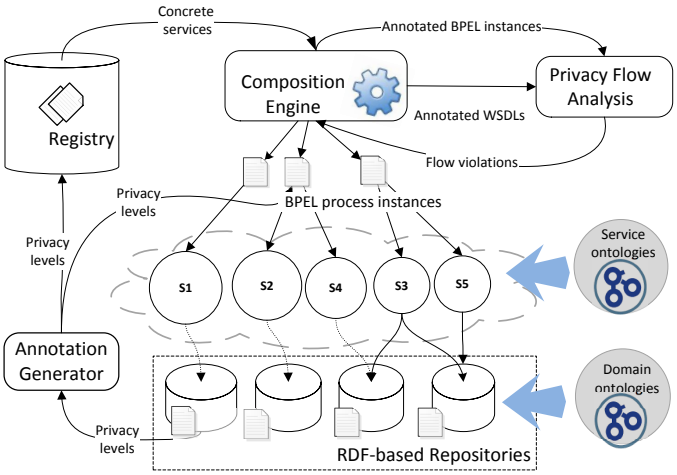


Figure 5.9: Composition with Private data flow

The *annotator* uses the initial privacy levels indicated at the data type properties of each of the data instances (required for the process execution) to generate the initial set of annotations. In our scenario, for instance, each of the variables $gene, disease, medicine, gender, age, name$ has an actual privacy level. Annotations are then added to both the variables involved in the process instance as well as service definitions in WSDL files. The privacy flow analysis uses the annotated process instance to analyze the actions in the context of that instance and infer privacy levels as they flow between services. Since data type properties flow as input or output variables throughout a process instance execution and throughout the entire composition, the analysis keeps track of the privacy levels needed for every process instance and uses those to propagate privacy level annotations to each subsequent activity in the process execution. The analysis uses an internal type checker that uses the annotations to enforce private flow of data between services. In the following, we explain how we implemented each component.

5.6.1 The Privacy Flow Analysis

We implemented our framework in Java using the Crystal static dataflow analysis framework [2]. We extended the Crystal framework to support our model. Crystal's dataflow anaylsis works on any control flow graph (CFG) and intermediate representation (Three Address Code TAC) of a language's Abstract Syntax Tree (AST), on which it performs AST-walking analyses. We refactored the core classes of the crystal framework to abstract away the concept of an `ASTNode` to work generically on any AST node, including both the BPEL4WS constructs and the WSDL constructs. We also extended the `ControlFlowGraph` interface to support the model generated from BPEL4WS and WSDL. We utilized the capability of the dataflow analysis infrastructure provided in Crystal to implement a *forward* analysis that is *context-sensitive* (it distinguishes between different invocation sites), *flow-sensitive* (the order of the execution affects the result of the analysis), and *branch sensitive* (to avoid loss of precision in handling conditional expressions). The *PrivacyFlowAnalysis* algorithm (Algorithm. 10) runs an instance of the worklist algorithm implemented in Crystal (Algorithm 9) on each process expression if it is not yet analyzed. The analysis core functionality lies in both the `PAASTVisitor` and the `Annotation-BasedPATransferFunction`.

Algorithm 10 AnnotatedPrivacyFlowAnalysis

```
1: Input P_root
2: output: result
3: worklist = createWorkList(P_root)
4: result = worklist.performAnalysis();
5: labeledResultsBefore=result.getLabeledResultsBefore();
6: labeledResultsAfter = result.getLabeledResultsAfter();
7: nodeMap = result.getNodeMap();
8: currentLattice = result.getLattice();
9: cfgStartNode = result.getCfgStartNode();
10: cfgEndNode = result.getCfgEndNode();
```

We implemented several privacy flow transfer functions that are aware of the privacy level annotations added to each `ASTNode` expression. Based on those function the `PAASTVisitor` checks each variable. For instance, the transfer function for an Invoke expression $invoke(x_S, \tilde{i}, \tilde{o})$ takes as arguments the invocation expression invoke and the tuple lattice, which maps a variable to it's abstract lattice value. The visitor checks whether parameters of the operation invocation are safe based by comparing their incoming actual privacy level annotations and the formal privacy

level requirements on that operation parameters. Given the operation binding the visitor obtains a summary of annotation info for that operation definition. It then looks up the annotation summary for the operation x_S, and for each parameter variable in i, it adds to the lattice the variable and it's corresponding lattice value.

5.6.2 The Privacy Analysis Type Checker

We implemented the type checker as a plugin to the Crystal framework. The type checker relies on an initially generated set of annotations added to a process instance based on initial set of privacy levels annotations. The *annotation generator* starts by adding the first round of annotations based on privacy preferences of the requested data in the concrete process instance that is being executed. It annotates every bound variable or service instance in the process instance with a privacy type. The annotation generation tool implements support for annotating process definitions as well as WSDL interface definitions of external services referenced in the BPEL process.

Annotations on BPEL processes are added to operation invocations, input, and output variables. Annotations on WSDL interfaces are added to operations input and output variables. The annotation generator rewrites BPEL and WSDL `ASTNode` expressions with the annotations. For BPEL process instances the annotation generator feeds the annotations as concrete privacy level types. Whereas for BPEL and WSDL definitions it defines formal privacy level parameters. For the WSDL definition in our scenario, the annotator adds the formal privacy level parameters `G,D` corresponding to the operation input and output variables, respectively. Upon receipt of the gene variable in the BPEL instance with actual privacy level H, the formal parameter G gets bound to to the actual value H.

```
<receive partner-Link="client" portType="GenoWSPT" Variable="gene"
   privacyLevel="H" createInstance="yes"/>

<wsdl:definitions>
 <privacylevelparams>
   <privacylevelparam name="G"/>
   <privacylevelparam name="D"/>
 </privacylevelparams>
<portType name="GenomWSPT">
      <operation name="getGenePhenAssoc">
```

```
            <input privacyLevel="G" message="gene"/>
            <output privacyLevel="D" message="disease"/>
        </operation>
</portType>
</wsdl:definitions>
```

The type checker then performs the analysis by feeding the process instance as an input to the analysis. Type checking is performed on annotations to guarantee that dataflow between services can only flow according to Table 5.2. The `PAASTVisitor` implements a method that is used by the type checker to report either a warning or an error based on the severity of the privacy violation. The errors and warnings get displayed in the Eclipse problems view.

5.6.3 Technology Implications

The proposed *privacy flow analysis* implementation can be incorporated into any service composition middleware. It could be integrated either in a composition development environment to assist developers during process design by performing compile-time process validation or in a run-time environment (*composition engine*) to perform run-time validation during process execution. For instance, it could be incorporated into the composition middleware provided by the WSO2 business process server (WSO2 BPS), which provides a comprehensive web-based console to manage, deploy, view and execute processes within a single server instance. WSO2 BPS implements an Apache ODE-powered BPEL engine and provides extensible RESTful management APIs. WSO2 BPS supports the BPWS4J [34] implementation of the IBM, Microsoft and BEA BPEL4WS specification. The BPWS4J platform is an Eclipse plug-in that consists of an engine and an editor. The BPWS4J engine takes the BPEL document for each process to be executed, a WSDL description of the interface that the process presents to the external clients or service partners (without binding information), and several WSDL documents of the partner services with which the process may interact.

CHAPTER 6 LITERATURE REVIEW

We discuss some of the existing approaches for privacy policy management organized by the following axes: context-awareness, dynamic rule evaluation, relative sensitivity, semantic-based, and implementation and evaluation.

6.1 Context Awareness

The literature has several works that have proposed context-aware privacy management systems [23, 40, 43, 12]. Some of these approaches dynamically handle a user request by applying techniques that regulate rather than prevent the data access such as HDB [58]. The dynamic trust adjustment model proposed in [23] also dynamically handles context, but they focus on access control, in terms of who has access to the information as opposed to what is being collected. Also, their approach relies on inferring context using sensed spatial and temporal information and they do not achieve dynamicity at rule level. Several technologies have been applied to achieve privacy policy enforcement by considering the requester's permission, the owner's consent, and the context [61, 43, 12]. Grandison [43] and Agrawal [12] leverage the Active Enforcement module of the Hippocratic Database technology (HDB) by transforming an original query to another query that is policy-compliant. Similar to our approach, those approaches do not rely on a third party for enforcement purposes. They also track the purpose of a query to determine if a query is suspicious or not, but do not keep track of usage context.

6.2 Dynamic Rule Evaluation.

Few researchers have started looking at dynamic policy rule evaluation as opposed to static policies [16, 51]. Among the relatively few researchers who took dynamicity of a context to a higher level by considering dynamicity of a rule is Pallapa et al. [80]. They proposed a context aware scheme for privacy preservation by maintaining a model of the user's environment, which is characterized by user's *activities* and *situations*. Their solution accounts for fine grained rules and they apply a dynamic rule generator. However, both the rule and the context types are still predetermined based on a set of activities and states in which the user could be. Also, these rules are not defined in semantic terms and do not govern what is potentially sensitive data. Our approach implicitly updates policy rules based on dynamically inferring a query's classification, what is considered *relatively* sensitive data, and diversity of queries.

6.3 Relative Sensitivity.

Some notable techniques that applied machine learning, data mining, or information theory for sensitive data detection include [10, 92, 70, 52]. Agrawal et al. [10] have done valuable work in that respect. In their work they defined conditional privacy using conditional entropy and information loss. We leverage similar techniques to partially define our context. Their notion of conditional privacy compares the distribution of the original data to that of the perturbed data to test if the original value can be guessed from the perturbed value. In contrast, our approach uses only the original data by comparing the data that appears in a newly submitted query to both the previously determined set of sensitive data and all the previously submitted data sets to dynamically identify potential breach of more sensitive data, and we incorporate the context into our privacy policy rule evaluation. Our approach for sensitive data detection is complementary to other approaches [69, 19]. Machanavajjhala et al. [69] have proposed the notion of perfect privacy using query containment mapping to ensure perfect privacy for relational data. Based on that, Barhamji et al. [19] developed a query rewriting approach for data mashup services and applied it to RDF views.

6.4 Semantic Policy Definitions

Few recent researchers have started using semantic concepts for defining privacy policies. For instance, Ferrini et al. [40] used XACML obligations to add axioms to an ontology using semantic functions to check for inconsistencies introduced in the ontology due to adding those axioms. We use a similar approach to dynamically add contextual information to an instance to make the rules that govern the data type properties of that instance smarter. However, our approach adds the inferred context to the instance and uses the inferred context to impose more strict rules in the corresponding policy. Thus, both the ontology instance and the policy definition of that instance stay in sync. Among the approaches that proposed solutions for defining policies on top of domain ontologies are [29, 85, 19, 101]. The work by Rahmouni et al. [85] stemmed from issues of diversity, complexity, and dynamicity of the rules governing privacy protection. They proposed a modeling approach to abstract rule complexities and facilitate the automation and enforcement of rules at the process level. The closest approach to ours is the one by Barhamji et al. [19], which ensures the dynamicity of a decision by query rewriting. Still, they rely on predefined user

preferences and do not incorporate dynamically inferred context. Tumer et al. [97] have proposed a semantic based privacy framework which takes into account hierarchies of nodes in an ontology and how they inherit policies from parent to child.

6.5 XACML Enhancements.

Several researchers have provided enhancements to the performance of XACML PEP and PDP components, such as efficiency and scalability [36, 77, 65] and adaptation [38, 56], but there exist very few works that have provided enhancements to the accuracy of the PEP by enhancing the context handler, which is the essence of our approach. Baily et al. and Laborde et al. [56] have recently implemented Self-Adaptive Authorization Frameworks based on XACML that improves the accuracy of a PEP by tracking malicious behaviors. Both works use obligations. Our work is different in that it does not dynamically update the original policy definitions, but implicitly incorporate context into rule evaluation. Brucker and Petritsch [25] enhanced the context handling protocol used in XACML, but they focus on the efficiency of attribute resolution strategies either via the PIP or the XACML context handler.

6.6 Dynamic Private Data Publishing.

Privacy preserving data publishing approaches can be classified into two major categories: data disclosure and anonymization. Our approach falls under the limited disclosure category. One representative approach in this regard is Hippocratic databases (HDB) technology, which enforces privacy at the database level rather than the application level. While they do dynamic privacy disclosure at the cell, column, and row levels, they do not incorporate previously inferred context into future query evaluation.

Anonymization techniques can be further classified into generalization and suppression techniques. One of the first approaches to anonymization is k-Anonymity [92]. Several researchers have provided practical implementations of these algorithms ranging from top-down vs. bottom-up to global vs. local to optimal vs. greedy to hierarchy-based vs. partition based. All these algorithms prevent uniquely identifying individuals through record linking, but do not prevent sensitive attribute disclosure. L-diversity [70] alleviates this problem by ensuring that sensitive attribute values in each equivalent class are diverse. However, it is possible to infer sensitive attributes when the distribution in a class is very different from the overall distribution of the same

attribute. T-closeness [62], on the other hand, considers the sensitive attribute distribution in each class, and its distance to the overall attribute distribution. The distance is measured using similarity scores for distributions.

LKC-Privacy [76] provides a generalization over the aforementioned approaches with more reasonable constraints on parameters. In essence, the main drawback to all generalization and suppression algorithms lie in the utility or information loss incurred. Since these approaches rely on frequency of an item, in some cases, certain co-occurrences of items are considered the source of utility especially when record linkage is performed.

6.7 XACML-based Privacy for Mobile Applications.

We are not the first to provide privacy-aware solutions for mobile applications. Anh et al. [15] implemented a middleware for building privacy-aware mobile applications. They built their solution on top of GAE and they used XACML obligations to define several functions to determine to what extent users can share their data with their friends in social networking environments based on similarity or filtering. De Cristofaro et al. [32] proposed a privacy-aware infrastructure for building participatory sensing applications to protect the data of both the data user and the provider. Their solution relies on a tagging mechanism that builds on the top of Identity-Based Encryption (IBE). Similar to these approaches, our approach defines fine-grained privacy policies rules at the level of each data item. Our approach further provides dynamicity at the rule rather than the decision level. also, our approach defines rules using semantic concepts and defines preferences partially at the instance level.

6.8 Anonymization Techniques

Anonymization techniques can be classified into generalization (replacing a value with a less specific but semantically consistent one) and suppression (not releasing a value at all) techniques.

6.8.1 K-Anonymity

One of the first approaches to anonymization is k-Anonymity [92]. Several researchers have provided practical implementations of these algorithms ranging from Top-down specialization vs. bottom-up generalization to Global (single dimensional) vs. local (multidimensional) to Complete (optimal) vs. greedy (approximate) to Hierarchy-based vs. partition based [61, 59, 20].

K-anonymity ensures that individuals cannot be uniquely identified by a record linking attack, but does not necessarily prevent sensitive attribute disclosure. The algorithm primarily provides a clustering of nodes into equivalence classes where each node is indistinguishable in its quasi-identifying attributes QID from some minimum number of other nodes. When there is not much diversity in the sensitive attributes inside an equivalence class, the sensitive attribute of everyone in the equivalence class becomes known with high certainty.

6.8.2 L-Diversity and T-Closeness

L-diversity [71] alleviates the problem of sensitive attribute disclosure inherent to k-anonymity by ensuring that sensitive attribute values in each equivalent class are diverse. A set of records in an equivalence class C is l-diverse if it contains at least l well-represented (measured by several ways including frequency counts and entropy). One drawback thus lies in the possibility to infer sensitive attributes when the sensitive distribution in a class is very different from the overall distribution of the same attribute. If the overall distribution is skewed, then the belief of someones value may change drastically in the anonymized data. Also, the possibility to detect equivalent classes which contain very similar sensitive attribute values. Also, l-diversity based approaches implicitly assume that each sensitive attribute takes values uniformly over its class, so when frequencies of sensitive attribute values are not similar there may be large utility loss on data. For example, consider a dataset that contains 1000 patients with some quasi-identifying (QID) attributes and a single sensitive attribute disease with two possible values cancer or flu. If there are only 5 patients with cancer in the table, to achieve 2-diversity, at least one patient with cancer is needed in each QID group. So at most 5 groups can be formed which causes information loss. T-closeness [63], on the other hand, considers the sensitive attribute distribution in each class, and its distance to the overall attribute distribution. The distance is measured using similarity scores for distributions.

All these algorithms prevent uniquely identifying individuals through record linking. The main drawback, on the other hand, to all generalization and suppression algorithms lie in the utility or information loss incurred, since they rely on frequency of an item, in some cases, certain co-occurrences of items are considered the source of utility especially when record linkage is performed. If an item occurs in no frequent itemset, then suppressing that item (attribute) incurs no information loss. However, if an item occurs in many frequent itemsets (tuples) then suppress-

ing the item incurs a large information loss, since all frequent itemsets containing that item are removed from the data.

6.8.3 K-Anonymity Applied to Relational Data

k-anonymity has been initially proposed by Samarati and Sweeny [92] to prevent linking an individual to a record in a relational data table through a quasi-identifier. Later, several researchers have applied the notion of K-Anonymity to privacy. However, most of the K-Anonymity protection models are at the relational data table level not at the Web service operation level. Therefore, the private resource in their case is the identity of the subject whose information is contained in the data. The K-Anonymity concept have been implemented in several real world systems, including DataFly, mArgus, and K-Similar. Recentrly, few researchers have studied probabilistic notions of k-Anonymity [49].

6.8.4 K-anonymity Applied to Graphs

The literature has a number of definitions derived from anonymity tailored to structural properties of network data. For example, k-degree anonymity [66], k-candidate anonymity [45], k-automorphism anonymity [110], k-neighborhood anonymity [109], and (k,l)-grouping [30]. *k-degree anonymity* indicates a graph as k degree anonymous if for every node v in V there exist at least k 1 other nodes that have the same degree as v. This anonymization technique can preserve privacy by preventing the re-identification of individual nodes by adversaries with a priori knowledge of the degree of certain nodes. However, the anonymized graph may not be useful. *k-neighborhood anonymity* has been first proposed by Wu et al. [109] and states that a graph is k-neighborhood anonymous if every node has a 1.5-hop neighborhood graph isomorphic to the 1.5-hop neighborhood graph of at least k-1 other nodes. *k-candidate anonymity* states that an anonymized graph satisifies k-candidate anonyity with respect to a structural query Q if there is a set of at least k nodes which match Q, and the likelihood of every candidate for a node in this set with respect to Q is less than or equal to 1/k. Zou et al. have proposed *k-automorphism anonymity*. A graph is k-automporphic if every node has the same subgraph signature as at least k-1 other graph nodes, and the likelihood of every candidate for that node is less than or equal to 1/k. Finally, *(k,l)-grouping* is a privacy mechanism that has been proposed by Cormode et al. to handle attacks in affiliation networks. It assumes that affiliation links can be predicted based on node attributes and

the structure of the network. They use a greedy algorithm that generalizes node attributes without modifying the network structure. They assume that each node is indistinguishable from at least k-1 other nodes in terms of node attributes.

6.9 Private Service outsourcing

Few researchers have studied privacy control in service outsourcing environments [54, 47, 107]. Hung et al [47] provided a private service outsourcing interaction protocol that enables sending only the desired parts of data for each outsourced invocation. Their approach focuses on data level privacy rather then the logic. Xiong et al. [107] have focused on outsourcing in data aggregation services by proposing several protocols for private data sharing between service-hosted databases. Recently, Jammalamadaka et al. [50] have developed a middleware for enforcing security constraints on outsourced data. They also defined an abstract secure service model. Ke et al. [54] have proposed an algorithm to enhance service trust by detecting conflicts due to privacy incompatibility in cloud service composition. They apply the Tablue algorithm to OWLS documents. Other researchers focused on outsourcing in other domains, including image reconstruction and databases. Cong et al. [103] studied outsourcing from image reconstruction perspective.

6.10 Operation Invocation Correlation

Other research have tackled similar questions about Web service related operation invocations but from different perspectives. Some of the questions that they tried to answer is given a service I plan to use, what other services are usually used together? and given several services I want to use together, can I find an operation invocation path to connect them, based on others past usage? and what do people who use these services also use? they have applied techniques such as association rule mining. Since we are tackling the problem from privacy perspective to anonymize invocations KAnonymity is a good solution. A few researchers have looked into services as networks, especially for analyzing scientific analysis reuse [93]. Tan et al. [93] studied service-service interactions and service-workflow interactions to identify possibilities for reuse. Their work incorporate metrics similar to ours, such as betweenness centrality, to measure services that occur the most between different source and target service or workflow nodes. In fact, our approach can be applied to that domain as well.

6.11 Model Checking of Services Based Interactions

Model checking of services-based interactions, in general, has received enough attention[102, 64, 98, 81, 82]. The goal of those analyses is to analyze whether composite Web services are well-formed from different perspectives, including concurrency, integration, failures, security, and privacy. Some of these works focused on privacy and security using language-based analyses [87]. Several researchers provided implementations of verification tools, including the WebJet checker by Gao et al. [42] and the verification tool by Abouzaid [8]. WebJet checker [42] is based on a type system and a mapping from BPEL to pi-calculus and performs three formal verification steps. Namely, open bisimulation, property checking and compatibility checking.

6.11.1 Process Calculi

Quite a few researchers extended the Pi-calculus by defining process calculi, including $\pi\zeta$-calculus proposed in [68], $Web\pi_\infty$ by Lucchi and Mazzara [67] and the recent WS-calculus by Lapadula et al. [57]. $\pi\zeta$-calculus models more expressive composition concepts (e.g., processes as objects and encapsulation). $Web\pi_\infty$ is a web-based extension of pi-calculus that takes into account the timing dimension. WS-calculus is a formalization for the constructs provided by both WS-BPEL and WSDL, with the purpose of verifying conformance between WS-BPEL programs and the associated WSDL definitions. Woodman et al. [106] proposed two XML languages based on the Pi-calculus. One language is an extension to the WSDL language to enable specifying the order in which operations should be invoked. The other is a composition language to define a composite service structure. The former verifies whether a given composite service is free from deadlocks and livelocks. Abouzaid [8], on the other hand, introduced the π-logic based the μ-logic associated with π-calculus in order to express properties to verify whether a composition is well-formed.

6.11.2 Type systems

Several researchers have defined type systems for Web service definitions, including Milner [75], Turner [99], Honda et al. [100], and Pierce et al. Milner proposed a type system based on channel names. Pierce and Sangiorgi [84] revised Milner's system by adding restrictions to read and write from channels. While Milner's typing is name-based, Pierce and Sangiorgi's language is structure-based and is more suitable to avoid misuse by processes that share a common resource

(e.g., a printer). Volpano et al. [102] proposed a type system for a procedural programming language to verify the noninterference security property. Their work was inspired by the works of Denning [37] and Bell and La Padula [21]. Bell and La Padula proposed a model for multi-level security based on which Denning proposed a lattice model. Hutter and Volkamer [48] leveraged Volpanao et al.'s type calculus to dynamic Web service composition plans. They used a semantic Web service composition language (OWLS-XPlan).

6.11.3 Dataflow Analyses an Lattice Models

Dataflow analysis for security purposes has attracted several researchers attention. Aziz et al. [17] used denotational semantics to statically analyze the security of mobile systems at the level of the communicated data. They assign security levels and apply a nonuniform dataflow analysis to detect two privacy breaches: information leakage and insecure communications. They also define a data flow framework in terms of a security levels lattice, abstraction and transfer functions. Their lattice has infinite width of security levels and their transfer function is in the form of a policy that localizes names depending on the level of the process to which those names are bound. They also use the notion of location to differentiate between the different copies of the same name (channel). Nielson et al. [78] developed a static data flow analysis to verify security properties that are different from ours. For example, they validate that messages are only received after a login action. They verify an action by checking whether there exists a path on the graph where an edge with label a precedes another edge with label b.

6.11.4 Privacy in Scientific Workflows

Privacy in Scientific workflow based data analysis have been explored in few research works [28, 35]. The majority of the solutions were to guarantee the privacy of the workflow logic rather than the privacy of data that flows between processes. None of the above works tackled the problem of privacy in Web service composition using data flow analyses that utilize annotation-based transfer functions in the manner described in this thesis. Thus, their analyses are not modular and do not propagate the analysis results to service definitions based on concrete process instances and concrete service endpoint references in those processes. Also, some of the above approaches provided theoretical frameworks to fix issues only at the design time while our approach works at run-time.

Table 6.1: An analytic evaluation of our approach compared to existing approaches.

Approach	**Context Awareness**	**Technique/ Technology**	**Dynamicity of Decisions**	**Dynamicity of Rules**	**Semantic Rules**
Bhatti et al. [23]	✓	Trust adjustment	✓		✓
Ferrini et al. [40]	✓	Semantic functions	✓		✓
Grandison et al. [43]	✓	HDB	✓		
Mont et al. [61]					
Pearson et al.[83]					
Agrawal et al. [12, 11]		HDB			
Barhamgi et al. [19]		Query rewriting	✓		✓
Tumer et al. [97]					
Chadwick et al. [26]					
Sweeney[92]		K-anonymity			
Chen et al.[27]	✓				
Backes et al.[18]		Policy refinement			
Hengartner et al. [46]	✓	Access rights graphs	✓	✓	
Wang et al. [104]		confidence bounding			
Machanavajjhala et al.[70]		l-diversity			
Scheffler et al.[88]					
Pallapa et al. [80]	✓	Privacy context graphs/ trees	✓	✓	
Mohammad et al. [76]		LKC-privacy			
Fung et al. [41]		LKC-privacy (data mashup)			
Our approach	✓	Probability and information entropy theory	✓	✓	✓

CHAPTER 7 CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis I defined possible solutions for dynamic privacy management in services based interactions. I defined and implemented three main components for dynamic, context-aware, semantic-based privacy policy management that build on top of existing Web service standards and technologies. I have also evaluated the feasibility of the implemented components in several case studies in comparison to similar existing works.

The evaluation results are promising and, in the future I would like to explore practical applications of the implemented systems. Future work also includes exploring other challenges that services based environments involve including privacy in service based social data sharing environments and services as collaborating networks. I would also like to explore some techniques including applied security and cryptography and sticky policies to enforce traveling constraints.

APPENDIX A K-ANONYMITY BASED PRIVATE WEB SERVICE SELECTION

A.1 GENERATED BPEL AND WSCL FILES

Table A.1: Results of 10 test runs of the KAnonymity algorithm on WSCL files of different sizes.

WSCL	Op	TR	TEERoutes	time PTOC	time PTONC	time OEOC
wscl01	5	3	4	2983	3169	2981
wscl02	6	3	5	2806	2812	2792
wscl03	12	3	4	2834	2805	2810
wscl04	7	4	8	2768	2808	2780
wscl05	5	4	8	2818	2642	2832
wscl06	5	3	5	2824	2797	2794
wscl07	8	5	8	2792	2866	2795
wscl08	12	6	8	2829	2956	2787
wscl09	5	3	4	2874	2813	2794
wscl10	9	4	5	2961	2840	2818
wscl11	12	5	8	2887	2727	2629
wscl12	5	2	2	2828	2823	2812
wscl13	5	3	5	2775	2775	2948
wscl14	4	4	8	2710	2825	2763
wscl15	10	4	6	2858	2651	2812
wscl16	12	7	10	2776	2906	2832
wscl17	11	3	3	2878	2862	2777
wscl18	10	6	11	2802	2664	2847
wscl19	5	2	4	2788	2866	2628
wscl20	6	4	8	2831	2648	2842
wscl21	5	5	10	2725	2765	2812
wscl22	5	4	7	2791	2815	2800
wscl23	7	4	7	2752	2827	2815
wscl24	12	6	9	2823	2883	2771
wscl25	4	3	5	2734	2835	2749
wscl26	6	3	4	2894	2979	2860
wscl27	8	4	7	2755	2929	2886
wscl28	4	3	3	2668	2911	2786
wscl29	13	4	5	2875	2830	2873
wscl30	4	2	3	2685	2839	2695

Table A.2: Results of 10 test runs of the KAnonymity algorithm on BPEL files of different sizes.

BPEL	Op	TR	TEERoutes	time PTOC	time PTONC	time OEOC
bpel01	7	40	80	526	546	570
bpel02	9	36	60	378	407	381
bpel03	9	42	90	409	415	421
bpel04	8	32	68	392	454	484
bpel05	7	46	72	409	465	415
bpel06	9	20	54	386	412	425
bpel07	7	35	76	408	412	427
bpel08	10	50	80	410	489	469
bpel09	8	34	57	358	403	384
bpel10	9	45	76	408	437	432
bpel11	9	38	76	412	410	425
bpel12	11	50	84	401	493	406
bpel13	9	41	80	412	446	444
bpel14	10	42	76	408	405	500
bpel15	5	40	72	383	537	389
bpel16	7	32	65	396	434	387
bpel17	7	29	54	370	399	385
bpel18	9	30	72	406	390	393
bpel19	8	52	95	448	441	447
bpel20	9	46	85	409	449	441
bpel21	10	42	74	386	402	458
bpel22	7	55	100	536	467	449
bpel23	10	31	80	446	500	573
bpel24	8	45	72	453	402	420
bpel25	11	41	80	432	468	422
bpel26	8	26	51	378	403	399
bpel27	8	33	68	368	400	449
bpel28	8	53	100	462	439	438
bpel29	6	31	76	457	421	500
bpel30	11	48	90	499	430	424

Table A.3: K-Anonymity scores derived from the different BPEL configurations in Table A.5

CS	BP	Op_i	OpType	Op_j	OpType	midmen	$\lvert T_{Op_{jRouts}} \rvert$	$\lvert T_{Op_{iRouts}} \rvert$	count	PTOC		PTONC		OEOC	
										k	Pr	k	Pr	k	Pr
1	1	1,1	inhouse	1,5	inhouse	0	1	0	1	1	100%	1	100%	1	100%
1	1	1,1	inhouse	OS1	outsourced	0	4	0	4	4	25%	4	25%	4	25%
1	1	1,1	inhouse	1,6	inhouse	0	4	0	4	4	25%	4	25%	4	25%
1	1	1,5	inhouse	OS2	outsourced	0	5	1	4	4	25%	4	25%	-	-
1	1	1,5	inhouse	1,6	inhouse	0	4	1	3	3	33%	3	33%	-	-
1	1	OS2	outsourced	1,6	inhouse	0	4	5							
1	1	1,5	inhouse	1,6	inhouse	0	4	1	3	3	33%	3	33%	-	-
1	2	2,1	inhouse	OS2	outsourced	0	5	0	5	5	20%	5	20%	5	20%
1	2	2,1	inhouse	2,3	inhouse	0	1	0	1	1	100%	1	100%	1	100%
1	2	2,1	inhouse	OS1	outsourced	0	4	0	4	4	25%	4	25%	4	25%
1	2	OS1	outsourced	2,4	inhouse	0	5	4	1	1	100%	1	100%	1	100%
1	2	OS2	outsourced	2,2	inhouse	0	1	5							
1	2	2,2	inhouse	2,4	inhouse	1	5	1	4	4	25%	3	33%	-	-
1	2	2,3	inhouse	OS1	outsourced	1	4	1	3	3	33%	2	50%	-	-
1	1	1,1	inhouse	OS2	outsourced	1	5	0	5	5	20%	4	25%	4	25%
1	1	1,1	inhouse	OS2	outsourced	2	5	0	5	5	20%	3	33%	3	33%
1	1	1,1	inhouse	OS2	outsourced	1	5	0	5	5	20%	4	25%	4	25%
1	1	1,1	inhouse	1,6	inhouse	1	6	0	6	6	16%	5	20%	5	20%
1	1	1,1	inhouse	1,6	inhouse	2	6	0	6	6	16%	4	25%	-	-
1	2	2,1	inhouse	2,4	inhouse	1	5	0	5	5	20%	4	25%	4	25%
1	2	2,1	inhouse	2,4	inhouse	2	5	0	5	5	20%	3	33%	3	33%
1	2	2,1	inhouse	2,4	inhouse	2	5	0	5	5	20%	3	33%	3	33%
1	2	2,1	inhouse	OS1	inhouse	1	4	0	4	4	25%	3	33%	3	33%

Table A.4: WSCL file properties

wscl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
#operations	5	6	12	7	5	5	8	12	5	9	12	5	5	4	10	12	11	10	5	6	5	5	7	12	4	6	8	4	13	4
#invocations	3	3	3	4	4	3	5	6	3	4	5	2	3	4	4	7	3	6	2	4	5	4	4	6	3	3	4	3	4	2

Table A.5: Parameters of BPEL documents.

bpel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Variables	32	62	42	52	32	42	42	42	52	32	52	42	42	42	52	52	52	52	72	72	42	42	52	42	42	42	72	42	42	42
services	4	3	5	4	4	3	5	3	3	4	4	4	4	3	3	4	4	4	4	4	4	3	4	3	4	4	2	4	2	3
Inhouse Invokes	5	5	5	5	4	5	4	7	4	5	5	7	6	6	3	3	4	5	4	6	5	4	7	4	7	6	4	4	4	7
Outsourced Invokes	2	4	4	3	3	4	3	3	4	4	4	4	3	4	2	4	3	4	4	3	5	3	3	4	4	2	4	4	2	4
Total Invokes	7	9	9	8	7	9	7	10	8	9	9	11	9	10	5	7	7	9	8	9	10	7	10	8	11	8	8	8	6	11

APPENDIX B PRIVACY FLOW ANALYSIS SCENARIO EXAMPLES

B.1 BPEL AND WSDL EXAMPLES

Listing B.1: Excerpts from the BPEL definition in our scenario

```
<process name="GeneticVariationDetection">
<partnerLinks>
<partnerLink name="client"/>
<partnerLink name="GenoWS"/>
<partnerLink name="DrugWS"/>
<partnerLink name="PharmWS"/>
<partnerLink name="ClinicWS"/>
<partnerLink name="DemoWS"/>
</partnerLinks>
<variables>
<variable name="g"messageType="integer"/>
<variable name="m" messageType="Drug"/>
<variable name="d" messageType="Disease"/>
<variable name="dm" messageType="ClinicPersInfo"/>
<variable name="agd" messageType="DemoInfo"/>
</variables>
<sequence>
<receive partner-Link="client" port-Type="SearchWSPT" Variable="g"
   createInstance="yes"/>
<sequence>
<flow>
<invoke partner-Link="GenoWS" port-Type="GenoWSPT" operation="getGenInfo"
   inputVariable="g" outputVariable="gd"/>
<invoke partner-Link="DrugWS" port-Type="DrugWSPT" operation="getDrugInfo"
   inputVariable="m" outputVariable="gd"/>
</flow>
<invoke partner-Link="ClinicWS" port-Type="ClinicWSPT" operation="
   getDiagnosisInfo" inputVariable="m" outputVariable="a"/>
<invoke partner-Link="PharmWS" port-Type="PharmWSPT" operation="getPharmInfo"
   inputVariable="d" outputVariable="gd"/>
<invoke partner-Link="ClinicWS" port-Type="ClinicWSPT" operation="
```

```
   getClinicPersonInfo" inputVariable="dm" outputVariable="agd"/>
<invoke partner-Link="DemoWS" port-Type="DemoWSPT" operation="getDemoInfo"
   inputVariable="agd" outputVariable="na"/>
</sequence>
</process>
```

Listing B.2: Excerpts from the WSDL definition in our scenario

```
<wsdl:definitions>
<message name="Disease">
  <part name="disease" type="mc:Disease"/>
</message>
<message name="GenInfo">
    <part name="gene" type="mc:Gene"/>
    <part name="drug" type="mc:Drug"/>
 </message>
<message name="DemoInfo">
    <part name="address" type="mc:Address"/>
    <part name="gender" type="mc:Gender"/>
 </message>
 <message name="ClinicPersInfo">
    <part name="disease" type="mc:Disease"/>
    <part name="drug" type="mc:Drug"/>
</message>
<portType name="GenomWSPT">
      <operation name="getGenePhenAssoc">
            <input message="GeneInfo"/>
            <output message="result"/>
      </operation>
</portType>
<portType name="ClinicWSPT">
     <operation name="getDiagnoInfo">
       <input message="getDiagnoInfo" />
     </operation>
</portType>
<partnerLinkType name="ClinicWSLT">
```

```
  <role name="PharmaWS" portType="PharmaWSPT" />
  <role name="ClinicWS" portType="ClinicWSPT" />
</partnerLinkType>
<partnerLinkType name="GenomWSLT">
      <role name="GenomWS" portType="GenomWSPT"/>
      <role name="client" />
</partnerLinkType>
</wsdl:definitions>
```

REFERENCES

[1] Axiomatics Language for Authorization (ALFA). http://www.axiomatics.com.

[2] Crystal: A Static Analysis Framework for Education and Research: yypetua hehttps://code.google.com/p/crystalsaf/.

[3] Sun's xacml implementation. http://sunxacml.sourceforge.net/, 2003.

[4] PatientsLikeMe. http://www.patientslikeme.com, 2005.

[5] CureTogether. http://curetogether.com, 2008.

[6] E-health information platforms (e-hip). https://distrinet.cs.kuleuven.be/research/projects/showProject.do?projectID=E-HIP, 2013.

[7] WSO2 Balana Implementation. https://github.com/wso2/balana, 2013.

[8] ABOUZAID, F. Toward a pi-calculus based verification tool for web services orchestrations. In *Computer Supported Acitivity Coordination* (2006), pp. 23–34.

[9] AGGARWAL, C. C., AND REDDY, C. K. *Data clustering: algorithms and applications*. CRC Press, 2013.

[10] AGRAWAL, D., AND AGGARWAL, C. C. On the design and quantification of privacy preserving data mining algorithms. In *SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2001).

[11] AGRAWAL, R., BIRD, P., GRANDISON, T., KIERNAN, J., LOGAN, S., AND RJAIBI, W. Extending relational database systems to automatically enforce privacy policies. In *International Conference on Data Engineering, ICDE* (2005).

[12] AGRAWAL, R., JOHNSON, C., ET AL. Securing electronic health records without impeding the flow of information. *International journal of medical informatics* (2007).

[13] AL-MASRI, E., AND MAHMOUD, Q. H. The qws dataset. *EB/OL], http://www. datatang. com/data/42831* (2008).

[14] ANDREWS, T., CURBERA, F., DHOLAKIA, H., GOLAND, Y., KLEIN, J., LEYMANN, F., LIU, K., ROLLER, D., SMITH, D., THATTE, S., ET AL. Business process execution language for web services (bpel4ws). *Microsoft, IBM, Siebel Systems, BEA, and SAP, version 1* (2003).

[15] ANH, D. T. T., GANJOO, M., BRAGHIN, S., AND DATTA, A. Mosco: a privacy-aware middleware for mobile social computing. *Journal of Systems and Software* (2014).

[16] ANTONOPOULOS, C. P., KAPSALIS, V., AND HADELLIS, L. Optimal scheduling of smart homes' appliances for the minimization of energy cost under dynamic pricing. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation, ETFA 2012, Krakow, Poland, September 17-21, 2012* (2012).

[17] AZIZ, B., AND HAMILTON, G. W. A privacy analysis for the π-calculus: The denotational approach. In *Roskilde University* (2002).

[18] BACKES, M., PFITZMANN, B., AND SCHUNTER, M. A toolkit for managing enterprise privacy policies. In *Computer Security, ESORICS*. 2003.

[19] BARHAMGI, M., BENSLIMANE, D., GHEDIRA, C., AND GANCARSKI, A. L. Privacy-preserving data mashup. In *International Conference on Advanced Information Networking and Applications (AINA)* (2011).

[20] BAYARDO, R. J., AND AGRAWAL, R. Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on* (2005), IEEE, pp. 217–228.

[21] BELL, E. D., AND PADULA, J. L. L. Secure computer system: Unified exposition and multics interpretation, 1976.

[22] BELLEAU, F., NOLIN, M.-A., TOURIGNY, N., RIGAULT, P., AND MORISSETTE, J. Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics 41*, 5 (2008), 706–716.

[23] BHATTI, R., BERTINO, E., AND GHAFOOR, A. A trust-based context-aware access control model for web-services. In *International Conference on Web Services* (2004).

[24] BROWN, G., POCOCK, A., ZHAO, M.-J., AND LUJÁN, M. Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *The Journal of Machine Learning Research* (2012).

[25] BRUCKER, A. D., AND PETRITSCH, H. Idea: efficient evaluation of access control constraints. In *Engineering Secure Software and Systems*. Springer, 2010, pp. 157–165.

[26] CHADWICK, D. W., AND LIEVENS, S. F. Enforcing sticky security policies throughout a

distributed application. In *Workshop on Middleware security* (2008).

[27] CHEN, H., FININ, T., AND JOSHI, A. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review* (2003).

[28] CHEUNG, W., AND GIL, Y. Towards privacy aware data analysis workflows for e-science. In *AAAI Workshop on Semantic e-Science* (2007), pp. 22–26.

[29] CHRISTOPOULOU, E., GOUMOPOULOS, C., ZAHARAKIS, I., AND KAMEAS, A. An ontology-based conceptual model for composing context-aware applications. *Research Academic Computer Technology Institute* (2004).

[30] CORMODE, G., SRIVASTAVA, D., YU, T., AND ZHANG, Q. Anonymizing bipartite graph data using safe groupings. *Proceedings of the VLDB Endowment* (2008).

[31] COVER, T. M., AND THOMAS, J. A. *Elements of information theory*. John Wiley & Sons, 2012.

[32] CRISTOFARO, E. D., AND SORIENTE, C. Participatory Privacy: Enabling Privacy in Participatory Sensing. *Network, IEEE* (2013).

[33] CUFF, D., HANSEN, M., AND KANG, J. Urban sensing: out of the woods. *Communications of the ACM* (2008).

[34] CURBERA, F., DUFTLER, M., KHALAF, R., MUKHI, N., NAGY, W., AND WEERAWARANA, S. Bpws4j. *Published online by IBM at http://www. alphaworks. ibm. com/tech/bpws4j* (2002).

[35] DAVIDSON, S. B., KHANNA, S., ROY, S., AND BOULAKIA, S. C. Privacy issues in scientific workflow provenance. In *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science* (2010), ACM, p. 3.

[36] DECAT, M., LAGAISSE, B., AND JOOSEN, W. Middleware for efficient and confidentiality-aware federation of access control policies. *Journal of Internet Services and Applications* (2014).

[37] DENNING, D. E. A lattice model of secure information flow. *Communications of the ACM 19*, 5 (1976), 236–243.

[38] ERRADI, A., MAHESHWARI, P., AND TOSIC, V. Policy-driven middleware for self-adaptation of web services compositions. In *Middleware 2006* (2006).

[39] FEI, X., AND LU, S. A dataflow-based scientific workflow composition framework. *Services Computing, IEEE Transactions on 5*, 1 (2012), 45–58.

[40] FERRINI, R., AND BERTINO, E. Supporting rbac with xacml+ owl. In *Symposium on Access control models and technologies* (2009).

[41] FUNG, B., TROJER, T., HUNG, P. C., XIONG, L., AL-HUSSAENI, K., AND DSSOULI, R. Service-oriented architecture for high-dimensional private data mashup. *Services Computing, IEEE Transactions on 5*, 3 (2012), 373–386.

[42] GAO, C., LIU, R., SONG, Y., AND CHEN, H. A model checking tool embedded into services composition environment. In *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference* (2006), IEEE, pp. 355–362.

[43] GRANDISON, T., GANTA, S. R., BRAUN, U., KAUFMAN, J., ET AL. Protecting privacy while sharing medical data between regional healthcare entities. *Studies in health technology and informatics* (2007).

[44] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: An update, 2009.

[45] HAY, M., MIKLAU, G., JENSEN, D., WEIS, P., AND SRIVASTAVA, S. Anonymizing social networks. *Computer Science Department Faculty Publication Series* (2007).

[46] HENGARTNER, U., AND STEENKISTE, P. Avoiding privacy violations caused by context-sensitive services. In *International Conference On Pervasive Computing And Communications, PERCOM* (2006).

[47] HUNG, P. C., CHIU, D. K., FUNG, W., CHEUNG, W. K., WONG, R., CHOI, S. P., KAFEZA, E., KWOK, J., PUN, J. C., AND CHENG, V. S. End-to-end privacy control in service outsourcing of human intensive processes: A multi-layered web service integration approach. *Information Systems Frontiers 9*, 1 (2007), 85–101.

[48] HUTTER, D., KLUSCH, M., AND VOLKAMER, M. Information flow analysis based security checking of health service composition plans. In *ECEH* (2006), pp. 59–70.

[49] IKARASHI, D., KIKUCHI, R., CHIDA, K., AND TAKAHASHI, K. k-anonymous microdata release via post randomisation method. *arXiv preprint arXiv:1504.05353* (2015).

[50] JAMMALAMADAKA, R. C., GAMBONI, R., MEHROTRA, S., SEAMONS, K., AND

VENKATASUBRAMANIAN, N. A middleware approach for outsourcing data securely. *computers & security 32* (2013), 252–266.

[51] KABBANI, B., LABORDE, R., BARRÈRE, F., AND BENZEKRI, A. Specification and enforcement of dynamic authorization policies oriented by situations. In *6th International Conference on New Technologies, Mobility and Security, NTMS 2014, Dubai, United Arab Emirates, March 30 - April 2, 2014* (2014).

[52] KASTHURI, S., AND MEYYAPPAN, T. Detection of sensitive items in market basket database using association rule mining for privacy preserving. In *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME)* (2013).

[53] KAVANTZAS, N. Web services choreography description language (ws-cdf) version 1.0. *http://www. w3. org/TR/ws-cdl-10/* (2004).

[54] KE, C., HUANG, Z., LI, W., SUN, Y., AND XIAO, F. Service outsourcing character oriented privacy conflict detection method in cloud computing. *Journal of Applied Mathematics 2014* (2014).

[55] KILDALL, G. A. A unified approach to global program optimization. In *Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (New York, NY, USA, 1973), POPL '73, ACM, pp. 194–206.

[56] LABORDE, R., KABBANI, B., BARRÈRE, F., AND BENZEKRI, A. An adaptive xacmlv3 policy enforcement point. In *COMPSAC Workshop* (2014).

[57] LAPADULA, A., PUGLIESE, R., AND TIEZZI, F. A wsdl-based type system for asynchronous ws-bpel processes. *Formal Methods in System Design 38*, 2 (2011), 119–157.

[58] LEFEVRE, K., AGRAWAL, R., ERCEGOVAC, V., RAMAKRISHNAN, R., XU, Y., AND DEWITT, D. Limiting disclosure in hippocratic databases. In *VLDB* (2004).

[59] LEFEVRE, K., DEWITT, D. J., AND RAMAKRISHNAN, R. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (2005), ACM, pp. 49–60.

[60] LEFEVRE, K., DEWITT, D. J., AND RAMAKRISHNAN, R. Mondrian multidimensional k-anonymity. In *ICDE* (2006).

[61] LEFEVRE, K., DEWITT, D. J., AND RAMAKRISHNAN, R. Mondrian multidimensional

k-anonymity. In *ICDE* (2006).

[62] LI, N., LI, T., AND VENKATASUBRAMANIAN, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE* (2007).

[63] LI, N., LI, T., AND VENKATASUBRAMANIAN, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (2007), IEEE, pp. 106–115.

[64] LI, Y. H., YOUNG PAIK, H., AND BENATALLAH, B. Formal consistency verification between bpel process and privacy policy. In *In Privacy Security and Trust* (2006), pp. 212–223.

[65] LIU, A. X., CHEN, F., HWANG, J., AND XIE, T. Designing fast and scalable xacml policy evaluation engines. *Computers, IEEE Transactions on* (2011).

[66] LIU, K., AND TERZI, E. Towards identity anonymization on graphs. In *ACM SIGMOD international conference on Management of data* (2008).

[67] LUCCHI, R., AND MAZZARA, M. A pi-calculus based semantics for ws-bpel. *The Journal of Logic and Algebraic Programming 70*, 1 (2007), 96–118.

[68] LUMPE, M., ACHERMANN, F., AND NIERSTRASZ, O. A formal language for composition. *Foundations of Component Based Systems* (2000), 69–90.

[69] MACHANAVAJJHALA, A., AND GEHRKE, J. On the efficiency of checking perfect privacy. In *SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2006).

[70] MACHANAVAJJHALA, A., KIFER, D., GEHRKE, J., AND VENKITASUBRAMANIAM, M. l-diversity: Privacy beyond k-anonymity. *Transactions on Knowledge Discovery from Data (TKDD)* (2007).

[71] MACHANAVAJJHALA, A., KIFER, D., GEHRKE, J., AND VENKITASUBRAMANIAM, M. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 3.

[72] MAYER, P., AND LÜBKE, D. Bpelunit-the open source unit testing framework for bpel. *URL http://www. bpelunit. net 45*, 50.

[73] MEDJAHED, B., AND ATIF, Y. Context-based matching for web service composition. *Distributed and Parallel Databases* (2007).

[74] MICHAILIDIS, M. Kazanova software for analytics. http://www.kazanovaforanalytics.com/, 2012.

[75] MILNER, R. *Communicating and Mobile Systems: The* Π *calculus*. Cambridge University Press, 1999.

[76] MOHAMMED, N., FUNG, B., HUNG, P. C., AND LEE, C.-K. Anonymizing healthcare data: a case study on the blood transfusion service. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), ACM, pp. 1285–1294.

[77] MOURAD, A., AND JEBBAOUI, H. Towards efficient evaluation of xacml policies. In *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on* (2014).

[78] NIELSON, H., AND NIELSON, F. A flow-sensitive analysis of privacy properties. In *Computer Security Foundations Symposium, 2007. CSF '07. 20th IEEE* (July 2007), pp. 249–264.

[79] OF HEALTH, U. D., AND SERVICES, H. Breaches Affecting 500 or More Individuals, 2001.

[80] PALLAPA, G., DI FRANCESCOY, M., AND DAS, S. K. Adaptive and context-aware privacy preservation schemes exploiting user interactions in pervasive environments. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (2012).

[81] PAPAPANAGIOTOU, P., AND FLEURIOT, J. Formal verification of web services composition using linear logic and the pi-calculus. In *Web Services (ECOWS), 2011 Ninth IEEE European Conference on* (Sept 2011), pp. 31–38.

[82] PAVLICH-MARISCAL, J., FRANKY, M., AND LOPEZ, A. Towards security assurance in round-trip engineering: A type-based approach. *Electron. Notes Theor. Comput. Sci. 292* (Mar. 2013), 83–94.

[83] PEARSON, S., MONT, M. C., AND KOUNGA, G. Enhancing accountability in the cloud via sticky policies. In *Secure and Trust Computing, Data Management, and Applications*. 2011.

[84] PIERCE, B., AND SANGIORGI, D. Typing and subtyping for mobile processes. In *Logic*

in Computer Science, 1993. LICS'93., Proceedings of Eighth Annual IEEE Symposium on (1993), IEEE, pp. 376–385.

[85] RAHMOUNI, H. B., SOLOMONIDES, T., MONT, M. C., AND SHIU, S. Privacy compliance in european healthgrid domains: An ontology-based approach. In *International Symposium on Computer-Based Medical Systems, CBMS* (2009).

[86] ROMANO, P., BARTOCCI, E., BERTOLINI, G., DE PAOLI, F., MARRA, D., MAURI, G., MERELLI, E., AND MILANESI, L. Biowep: a workflow enactment portal for bioinformatics applications. *BMC bioinformatics 8*, Suppl 1 (2007), S19.

[87] SABELFELD, A., AND MYERS, A. C. Language-based information-flow security. *Selected Areas in Communications, IEEE Journal on 21*, 1 (2003), 5–19.

[88] SCHEFFLER, T., GEISS, S., AND SCHNOR, B. An implementation of a privacy enforcement scheme based on the java security framework using xacml policies. In *International Information Security Conference* (2008).

[89] SHANNON, C. E. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* (2001).

[90] SLOMINSKI, A. Adapting bpel to scientific workflows. In *Workflows for e-Science*. Springer, 2007, pp. 208–226.

[91] STANDARD, O. extensible access control markup language (xacml) version 3.0., 2013.

[92] SWEENEY, L. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* (2002).

[93] TAN, W., ZHANG, J., AND FOSTER, I. Network analysis of scientific workflows: a gateway to reuse. *IEEE Computer* (2010), 54.

[94] TBAHRITI, S.-E., MEDJAHED, B., MALIK, Z., GHEDIRA, C., AND MRISSA, M. Meerkat A Dynamic Privacy Framework for Web Services. In *The IEEE/WIC/ACM International Conference on Web Intelligence* (2011).

[95] TBAHRITI, S.-E., MEDJAHED, B., MALIK, Z., GHEDIRA, C., AND MRISSA, M. How to Preserve Privacy in Services Interaction. In *International Conference on Advanced Information Networking and Applications Workshops* (2012).

[96] TRUONG, H.-L., AND DUSTDAR, S. A Survey On Context Aware Web Service Systems, 2009.

[97] TUMER, A., DOGAC, A., AND TOROSLU, I. H. A semantic-based user privacy protection framework for web services. In *Intelligent Techniques for Web Personalization*. 2005.

[98] TURKMEN, F., JUNG, E., AND CRISPO, B. Towards run-time verification in access control. In *Policies for Distributed Systems and Networks (POLICY), 2011 IEEE International Symposium on* (2011).

[99] TURNER, D. The polymorphic pi-calculus: Theory and implementation.

[100] VASCONCELOS, V. T., AND HONDA, K. Principal typing schemes in a polyadic π-calculus. In *CONCUR'93* (1993), Springer, pp. 524–538.

[101] VILLATA, S., COSTABELLO, L., DELAFORGE, N., AND GANDON, F. A social semantic web access control model.

[102] VOLPANO, D., AND SMITH, G. A type-based approach to program security. In *Proceedings of the 7th International Joint Conference CAAP/FASE on Theory and Practice of Software Development* (London, UK, UK, 1997), TAPSOFT '97, Springer-Verlag, pp. 607–621.

[103] WANG, C., ZHANG, B., REN, K., AND ROVEDA, J. M. Privacy-assured outsourcing of image reconstruction service in cloud. *Emerging Topics in Computing, IEEE Transactions on 1*, 1 (2013), 166–177.

[104] WANG, K., FUNG, B. C., AND PHILIP, S. Y. Handicapping attacker's confidence: an alternative to k-anonymization. *Knowledge and Information Systems 11*, 3 (2007), 345–368.

[105] WASSERMANN, B., EMMERICH, W., BUTCHART, B., CAMERON, N., CHEN, L., AND PATEL, J. Sedna: A bpel-based environment for visual scientific workflow modeling. In *Workflows for e-Science*. Springer, 2007, pp. 428–449.

[106] WOODMAN, S. J., PALMER, D. J., SHRIVASTAVA, S. K., AND WHEATER, S. M. Notations for the specification and verification of composite web services. In *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International* (2004), IEEE, pp. 35–46.

[107] XIONG, L., CHITTI, S., AND LIU, L. Preserving data privacy in outsourcing data aggrega-

tion services. *ACM Transactions on Internet Technology (TOIT) 7*, 3 (2007), 17.

[108] ZHAO, Z., ZHANG, R., LIN, J., CHEN, Y., ZHANG, H., AND LI, L. An improved visual bpel-based environment for scientific workflow. In *Seventh International Conference on Grid and Cooperative Computing, GCC 2008, Shenzhen, China, October 24-26, 2008* (2008), pp. 435–441.

[109] ZHOU, B., AND PEI, J. Preserving privacy in social networks against neighborhood attacks. In *ICDE* (2008).

[110] ZOU, L., CHEN, L., AND ÖZSU, M. T. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment* (2009).

ABSTRACT

DYNAMIC PRIVACY MANAGEMENT IN SERVICES BASED INTERACTIONS

by

NARIMAN AMMAR

May 2016

Advisor: Dr. Zaki Malik

Major: Computer Science

Degree: Doctor of Philosophy

Technology advancements have enabled the distribution and sharing of users personal data over several data sources. Each data source is potentially managed by a different organization, which may expose its data as a Web service. Using such Web services, dynamic composition of atomic data items coupled with the context in which the data is accessed may breach sensitive data that may not comply with the users preference at the time of data collection. Thus, providing uniform access policies to such data can lead to privacy problems. Some fairly recent research has focused on providing solutions for dynamic privacy management. This thesis advances these techniques, and fills some gaps in the existing works. In particular, dynamically incorporating user access context into the privacy policy decision, and its enforcement.

AUTOBIOGRAPHICAL STATEMENT

NARIMAN AMMAR

EDUCATION

- M.Sc. Computer Science, Wayne State University, Detroit, MI, USA, August 2011
- B.S. Computer Engineering, BirZeit University, Ramallah, Palestine, July 2006

PUBLICATIONS

1. N. Ammar, Z. Malik, B. Mejahed, A. Rezgui. "Data Flow Analysis for Privacy-Preserving Web Service Composition", IEEE TSC 2016.
2. N. Ammar, Z. Malik, E. Bertino, A. Rezgui *"XACML Policy Evaluation With Dynamic Context Handling.*. lCDE, 2016.
3. N. Ammar, Z. Malik, B. Mejahed, M. Alodib *"Private Web Service Outsourcing: K-Anonymity Beyond Data Tables."*. IEEE TSC 2016.
4. N. Ammar, Z. Malik, B. Mejahed, M. Alodib *"K-Anonymity Based Private Web Service Selection"* (Best paper award). 22nd IEEE ICWS, 2015.
5. N. Ammar, Z. Malik, E. Bertino, A. Rezgui *"XACML Policy Evaluation With Dynamic Context Handling.* IEEE Transactions on Knowledge and Data Engineering (TKDE), 2015.
6. N. Ammar, Z. Malik, M. Alodib, A. Rezgui *"MobiDyC: Private Mobile-Based Health Data Sharing Through Dynamic Context Handling.* International Conference on Mobile Systems and Pervasive Computing (MobiSPC), 2014.
7. N. Ammar, Z. Malik, E. Bertino, A. Rezgui *"Dynamic Privacy Policy Management in Services-Based Interactions.* Database and Expert Systems Applications, 2014.
8. K. Hashmi, E. Najmi, N. Ammar, Z. Malik, B. Mejahed *"Sentiment Analysis for Intelligent Ratings Management."* ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), 2014.