# Privacy-Preserving Mining of Web Service Conversations

by

Faisal A. Binzagr

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science
(Computer and Information Science)
in the University of Michigan-Dearborn
2017

## Acknowledgements

First of all, I would like to thank my advisor Dr. Brahim Medjahed for his support from the moment I met him. He never hesitated to help and guide me to complete my thesis. I learned a lot from his great research experience. For instance, he assigned many papers that were beneficial for the thesis and for me. I didn't understand some of them in the beginning, because they were intense. However, after several meetings with him I could understand. Moreover, he made me feel confident to go deeper in research. I really appreciate what he has done for me so far.

I would also like to thank my friend Hemza Labbaci who was involved in the implementation and validation of the experiment for this thesis. Without his participation, the validation could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and all my brothers for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Faisal Binzagr

# Table of Contents

# Lists of Tables

## Lists of Figures

# ABSTRACT

Organizations and businesses are exporting their applications as Web services seeking more collaboration opportunities. These services are generally not used in silos. Indeed, the invocation of a service is often conditioned by the invocation of other services. We refer to the precedence relationships between service invocations as conversations or choreographies. As clients interact with Web services, they exchange an important quantity of sensitive data, hence raising the challenge to keep the privacy of various interactions. In addition to the data exchanged with Web services, users may consider the information about service usage as sensitive and would like to hide that information from third parties. However, conversation relationships may complicate the task of keeping such information secret.

In this Thesis, we extend the traditional concept of k-anonymity introduced for databases to Web service conversations. The goal is to determine the extent to which the invocation of a service can be inferred from downstream invocations. We first use the FP-Growth algorithm for mining service invocation logs. The mining process returns the probabilities of service conversations. We then define a probabilistic k-anonymity technique for Web service conversations based on the results of the mining process. The proposed approach assists users in selecting Web services that best satisfy their anonymity requirements. We conducted extensive experiments using real-world Web services to prove the efficiency of the proposed approach.

# CHAPTER 1

# Introduction

## 1.1 Motivation

Organizations and businesses are exporting their applications as Web services (a.k.a. APIs) such as Google Maps API and Uber API seeking more collaboration opportunities [12]. Web services interact with each other regardless of their heterogeneities to provide value-added functionalities capable to achieve complex goals that extend the use of one simple API [23].

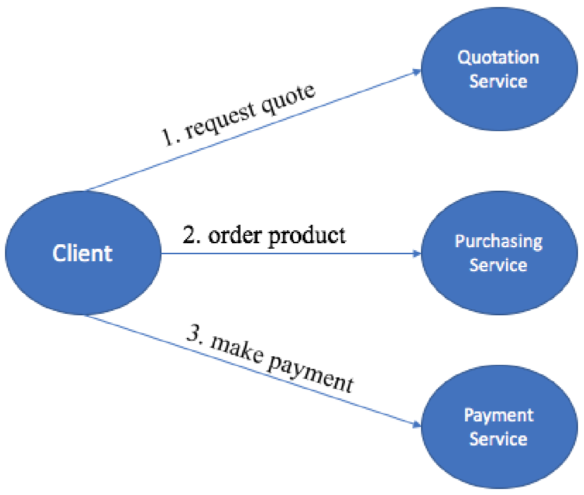


Fig. 1.1 Precedence Relationships between Web Services

Web services are generally not used in silos. Very often, the invocation of a service is conditioned by the invocation of other services. Fig. 1.1 shows an example where clients are first required to request quotes before ordering goods, then finally making payments for their purchase In this case there are precedence relationships, called *conversations* or *choreographies*, between the invocations of the three services [16].

As clients interact with Web services, they exchange an important quantity of sensitive data, hence raising the challenge to keep the privacy of various interactions [2]. In addition to the data exchanged with Web services, users may consider the information about service usage as sensitive and would like to hide that information from third parties. However, the precedence relationships between service invocation may complicate the task of keeping such information secret. In our example, if a third party knows that a client invoked the payment service, then they will infer that the same client invoked the quotation and purchase services as these two services are pre-requisites of the payment service.

## 1.2 Thesis Statement

Data anonymization techniques have been used in the literature to hide the identity of users. For example, in relational databases k-anonymity has been applied to prevent linking the data to specific records. In a previous work, [2] proposed an approach by integrating k-anonymity with privacy management framework to ensure the privacy requirements in service-oriented systems. [2] computes the k-anonymity of Web services using precedence relationships among Web services specified using the Web Services Conversation Language (WSCL) to determine the extent to which the invocation of the operation can be inferred from the downstream operation. In this research, we focus on computing more accurate anonymity scores based on the probability of invocation. In case of service selection, the information of previous interaction between the services can play a key role to determine the level of anonymity. Our proposed approach determines the anonymity of services using the probability of interactions: what is the probability that clients invoke a service given that they invoked another services. We take advantage of service logs to calculate the probabilities of invocation precedence and preserve the probabilistic anonymity of service invocations.

## 1.3 Related Work

Mining service logs has been subject to previous research in literature. [17][14] performed mining service logs with the aim to distinguish which invocations are part of the same composition relationship. [8][15] conducted mining software and service logs with the aim to prevent failures to happen. [8] used random indexing to represent the changes of states during a software lifetime. Vector support machine is then used to classify the changes of states as failure or non-failure. [15] identified the top k events in software and service execution that causes failures. [19] conducted mining the software components interaction history in logs to apply dynamic changes to a running software system (such as replacing a running component) without creating inconsistencies. The mining allows also the identification of potentially malicious (abnormal) behavior. [7] modeled dependencies among the events relating to software execution by using a Bayesian network. The Bayesian network depicts the causal relationships between the recorded log events. Such a network is used to predict the lateness probability of the process managed by the software.

Although existing work tried to leverage mining service logs to address various challenges related to Web services, they do not take advantage from mining service logs to accurately compute the anonymity among services. We conducted mining service logs to infer the choreography relationship among services. We also infer the probabilities of interactions of Web services, we use the obtained probabilities to compute in an accurate way the anonymity of Web services.

Computing the anonymity between the services has been subject to previous research in literature. [6] evaluated the measurement of anonymity in different system. [18] designed an approach to solve the anonymity in web transaction by grouping the users into set and they forward the request within a set. As a result, the provider server cannot know which client request the service. However, this solution does not recommend which services meet the developer requirement.

[2] proposed an approach that integrate k-anonymity with privacy management framework to guarantee the privacy of the services. This approach define the k-anonymity based on Web Services Conversation Language (WSCL) to determine the downstream invocation of particular service without consider the probability of invocation. [5] introduced an approach to calculate the top k-attack with maximum probabilities for each node in the system. The goal of that approach is to find the vulnerability or trust relationship. Moreover, we use the same definition of multiple-step attack to define the probability of the possible path probability which is equal to the product of all single-step probabilities.

Although existing work tried to solve the anonymity for various challenges related to Web services, they do not take advantage from mining service logs to accurately compute the anonymity

among services to provide the optimal service to select. We conducted an experiments rely on mining service logs to obtain probabilities of interaction, to compute in an accurate anonymity of Web services. We also compare our approach with existing approach(k-anonymity).
Using the probability for measuring the anonymity have been proposed but not in Web Services. Reiter and Rubin [18] define the anonymity as 1-p , where p is the probability of specific user assigned by attacker in the system. We believe that this information useful to compute the missing information needed by an attacker. However, by mining service logs, our approach can generate an accurate probability missing for the attacker. Moreover, Beryhold et al [4] define the anonymity as $\log_2(N)$, where N is the number of participants in the system. This only depends on the number of participants and does not consider the probability of interaction of each participant. [6] defines the maximum anonymity achieved when the attacker sees all participants as equiprobable. However, in practical the maximum anonymity achieved when the attacker missing exact probability of each participant. Therefore, in our model we use the probability of interaction, which reflect the accurate dependency of each services to provide the developers the critical information may need it for selecting a service.

## 1.4 Thesis Contribution

We summarize the main contribution of our thesis as follows. First, we propose a technique for mining service logs in order to infer precedence relationships between services and the probability of interaction of services. Such a probability can be used to improve the traditional way of computing the anonymity between services introduced in [2]. Second, we introduce the concept of probabilistic k-anonymity of Web service conversations. The proposed technique extends the approach proposed in [2]. Accurate computation of the degree of anonymity between services provides support to developers in their task of selecting the services that meet their anonymity requirements. Third, we conduct experiments to evaluate the proposed techniques.

## 1.5 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 describes the technique for mining service logs in order to infer precedence relationships between services. In Chapter 3, we present our approach for computing probabilistic k-anonymity. Chapter 4 concludes our Thesis.

# CHAPTER 2

# Mining Service Logs for Web Service Conversations

## 2.1 Introduction

Web service choreography is a model of service composition that invokes a large number of services. As Web services interact, the history of their interactions is recorded in service logs. Mining Web service logs allows finding which services are part of the same choreography. It also allows inferring the probabilities of interactions of Web services. This probability can be used to accurately compute the degree of anonymity among Web services. It supports developers in choosing services that guarantee a high level of anonymity.

Mining service logs allows inferring the relationships between services (such as the choreography) and the probability of interaction of services. Such a probability can be used to improve the traditional way of computing the anonymity between services [2]. Accurate computation of the degree of anonymity between services provides support to developers in their task of selecting the services that meet their anonymity requirements.

In this chapter, we discuss how do we perform mining service logs [10]. We also discuss how mining service logs can be used to achieve a better anonymity between services.

Existing works [17][14][8][15] perform mining service logs for several purposes. [17][14] perform mining service logs to discover which invocations of Web services are correlated. [8][15] perform mining service logs to prevent software and services failures. However, existing works do not take advantage from mining service logs to infer probabilities of interactions of services with the purpose to compute the anonymity among services.

In this section, we rely on our previous work [10] for mining service logs and inferring the probabilities of interactions among services. We use the obtained probability to compute the anonymity of Web services.

First, we clean and prepare service logs. Then, we use the parallel FP-growth algorithm [13] to discover which invocations are part of the same choreography relationship. We derive the choreography relationship among services, we infer the probability of interaction of Web services. We use the obtained probability to compute the anonymity among the Web services.

The rest of the chapter is organized as follows: Section 2.2 describes background techniques. Section 2.3 describes the proposed approach for learning the choreography relationship among services and inferring their probabilities of interaction. Section 2.4 describes the experimental setup and discusses the obtained results.

## 2.2 Background

In this section, we describe some existing techniques used in this thesis, namely the WSCL (Web Service Conversation Language) for the specification of service conversations and the FP-Growth data mining algorithm.

### 2.2.1 WSCL (Web Services Conversation Language)

The Web Services Conversation Language can be defines as a system upon which the business' operation regarding conversation or public interaction on a web platform such as the XML as defined by Douglas K. Barry [3]. WSCL control the sequence of Web services document exchange. In other words, Web services provide different operations which might depend on the sequence of invocation to accomplished specific task.

**WSCL Definition 1**

```
<ConversationTransitions>
      <Transition>
            <SourceInteraction herf="getTask1"/>
            <DestinationInteraction herf="getTask3"/>
      </Transition>
      <Transition>
            <SourceInteraction herf="getTask2"/>
            <DestinationInteraction herf="getTask3"/>
      </Transition>
      <Transition>
            <SourceInteraction herf="getTask3"/>
            <DestinationInteraction herf="getTask4"/>
      </Transition>
      <Transition>
            <SourceInteraction herf="getTask5"/>
            <DestinationInteraction herf="getTask4"/>
      </Transition>
</ConversationTransitions>
```

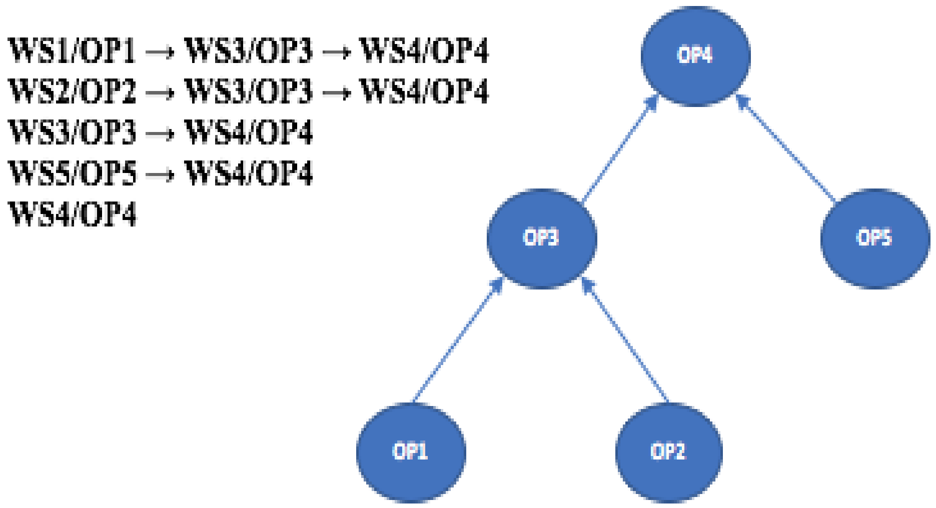


Fig. 2.1 A choreography of 5-operation invocation

Fig. 2.1 depicts an abstract choreography of the conversation defined in WSCL Definition 1. The defined conversation in WSCL Definition 1. Provide all possible routes for each possible operation. For instance, for the route **2 → 4**, **Op**$_2$ has one downstream which is **Op**$_3$. As a result , there are four possible operation leading to the downstream **Op**$_4$.

### 2.2.2 FP-Growth

FP-Growth has been proposed by Han [9], the algorithm is a useful datamining algorithm for discovering frequently co-occurrent items. In our case, FP-Growth is used to discover frequent co-occurrent Web service invocations. If a given set of Web service have been invoked frequently enough (over the support value, i.e., frequency threshold) by one same party, it means that invoked services are part of the same choreography.

FP-Growth allows inferring the association rules between the frequent items. FP-Growth works as following:

1- Calculate the **minimum support**. For example, minimum support = 30%, and number of Web services in the database of transactions = **8**. Then, Minimum support count = (30*8)/100 = **2.4**. We can round the minimum support to **3.**
2- Compute the frequency of occurrence of each Web service in the table of transactions. For example, The Web service with the ID:E appears 4 times in the database of transactions.
3- Sort Web services in a list according to their frequency of appearance. The most frequent Web services are placed in the top of the list. Eliminate the Web services with the frequency of appearance less than the minimum support. To illustrate further, Table 2.1 represents five services (A,B,C,D and E) which satisfied the minimum support.

| Item | Frequency | Priority |
|---|---|---|
| **A** | **5** | **3** |
| **B** | **6** | **1** |
| **C** | **3** | **5** |
| **D** | **6** | **2** |
| **E** | **4** | **4** |

Table 2.1 FP-Growth frequency table

4- Reorder the transactions from the service with the highest frequency to the service with the lowest frequency. Table 2.2 shows the transaction order.

| Interaction | Services | Ordered Services |
|---|---|---|
| **1** | **E,A,D,B** | **B,D,A,E** |
| **2** | **D,A,C,E,B** | **B,D,A,EC** |

| | | |
|---|---|---|
| **3** | **C,A,B,E** | **B,A,E,C** |
| **4** | **B,A,D** | **B,D,A** |
| **5** | **D** | **D** |
| **6** | **D,B** | **B,D** |
| **7** | **A,D,E** | **D,A,E** |
| **8** | **B,C** | **B,C** |

Table 2.2 FP-Growth transactions order

1- Draw the FP-Tree, the root node of the tree is the null node, as it shows in Fig 2.2 .

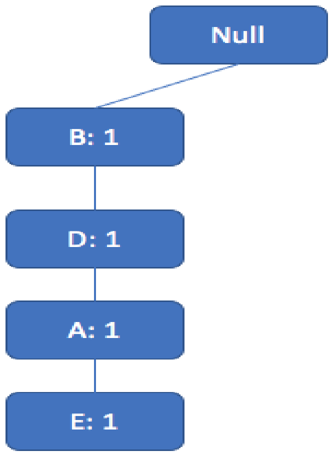


Fig. 2.2 FP-Tree

2- Update the previous tree by adding the services of each interaction,

3- Update the frequency of each service in the tree. For example, Fig. 2.3 shows frequent services for first two rows.

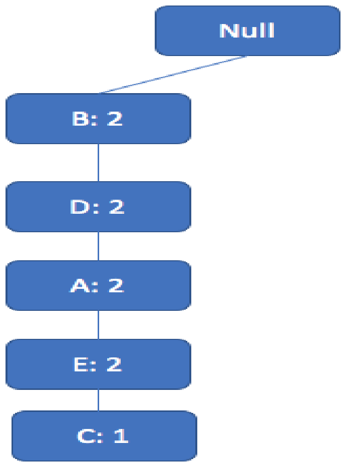


Fig. 2.3 FP-Tree for Row 1 and 2

4- Update the frequency of each service in the tree as it shows in Fig. 2.4.

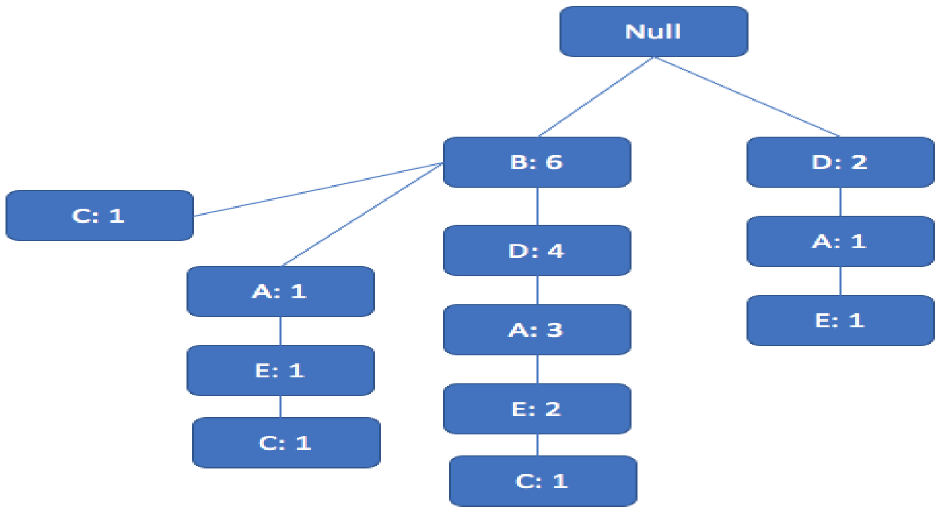
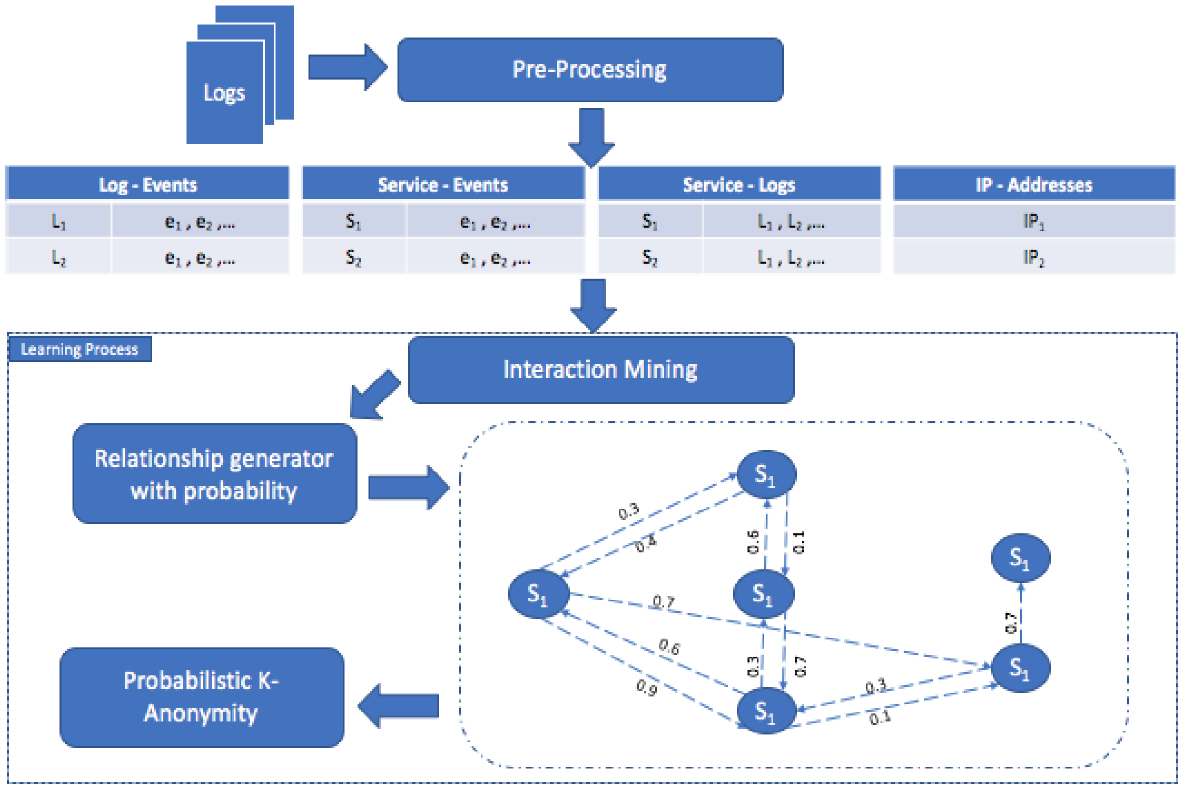


Fig. 2.4 Final FP-Tree

Use the FP Tree to generate the association rules with probability.

## 2.3 The Proposed Approach

This section describes the proposed technic for mining service logs and inferring the probabilities of interactions of Web services. Fig. 2.5 illustrates the main steps of the proposed approach. First, we clean the log file from irrelevant data. We focus on the following pieces of information of the log file (1) IP address of the client service who makes the invocation (2) The invocation date and time and (3) The status of the invocation (success or failure).



| Log - Events | |
|---|---|
| $L_1$ | $e_1, e_2, \ldots$ |
| $L_2$ | $e_1, e_2, \ldots$ |

| Service - Events | |
|---|---|
| $S_1$ | $e_1, e_2, \ldots$ |
| $S_2$ | $e_1, e_2, \ldots$ |

| Service - Logs | |
|---|---|
| $S_1$ | $L_1, L_2, \ldots$ |
| $S_2$ | $L_1, L_2, \ldots$ |

| IP - Addresses |
|---|
| $IP_1$ |
| $IP_2$ |

Fig. 2.5 Mining Service Logs for Web Service Choreographies

First, we identify the invocations that have been raised by the same client service, under a tiny time frame, and which status is success. For instance, service $S_i$ invokes $S_a$, $S_b$ and $S_c$ during timeframe $\delta t$, if this invocation is observed frequently enough (i.e., higher than the support value of the FP-Growth algorithm). Then, we assume that $S_i$ orchestrates the invocation of $S_a$, $S_b$ and $S_c$, and $S_i$, $S_a$, $S_b$ and $S_c$ are part of the same choreography. We also infer the choreography relationship among $S_a$, $S_b$, $S_c$, and Si, and we infer the probabilities of interactions of the previous services. We only keep the relationships with a probability of interaction higher than the confidence value of the FP-Growth algorithm.

Algorithm 1 summarizes the learning process. It returns the interaction windows (serviceIW) for each service and the probability of interaction as P.

**Algorithm 1** Learning Algorithm

```
Input: log-event, service-events, service-log, IPAdresses, confidence, min-support, $\delta t$,
Output: ServiceIWs, Probability
1: itemsetAllIW s ← serviceIW ← null
2: for each $IP_i \in$ IP Addresses do
3:     events ← service – events[$IP_i$]
4:     logs ← service – logs[$IP_i$]
5:     IW ← null
6:     used ← null
7:     for each $e_k \in$ events & $e_k \notin$ used do
8:         IW.append($L_k$)
9:         used.append($e_k$)
10:        for each $L_j \in$ logs do
11:            if(f $\in$ log-events[$L_j$] & f $\notin$ used & CorrelationCheck($e_k$, f,$\delta t$)) then
12:              IW.append($L_j$)
13:              used.append(f)
14             end if
15:        end for
16:        serviceIW[$IP_i$].append(IW)
17:        itemsetAllIW s.append(IW)
18:    end for
19: end for
20: return Probability ← FP-Growth(itemsetAllIW s, confidence, min-support)
```

If two events e and f recorded in $L_k$ and $L_j$, respectively have been raised by the same client service (i) under a given time frame $\delta t$ (i.e., e.dateTime – f.dateTime <= $\delta t$ ). Algorithm 1 puts i, e, and f in the same interaction window (IW). If this interaction window is observed frequently enough. FP-Growth algorithm generates the association rules between the previous services as following: rule1 i→ e with the probability **0.7** and rule2 i→f with the probability **0.8**. The interpretation of rule1 is: If the service e is invoked, the likelihood that such invocation has been raised by i is **0.7**. Using the obtained rules and probabilities, we create the probability table of Web service interactions.

We notice in the table two kind of invocations: Direct Invocations and Indirect Invocations. A direct invocation i→e with a probability p means the invocation of the service i is p percent of time followed by the invocation of the service e. An indirect invocation i→k with a probability p means the invocation of the service i leads to the invocation of a given set of services intermed-set, the invocation of intermed-set leads to the invocation of the service k. The probability that:

1- The invocation of i will be followed by the invocation of a service h in intermed-set, and
2- The invocation of h will be followed by the invocation of the service k is p.

To be able to use the previous probabilities to compute the anonymity among Web services, the probabilities of all paths leading to one same service need to sum-up to one. In other words, the probabilities of all the paths leading to the same service need to form a probability distribution. In practice, the probabilities of the different paths leading to the same service do not sum up to one. Thus, we normalize the different probabilities. The normalized probabilities will be used to compute the anonymity among Web services (Chapter 3)

| IP$_{client}$ | IP$_{service}$ | Probability |
|---|---|---|
| 1:22.0.1 | 1:22:0.3 | 0.6 |
| 1.22.0.4 | 1.22.0.5 | 0.3 |
| 1.22.0.3 | 1.22.0.5 | 0.9 |
| 1.22.0.5 | 1.22.0.3 | 0.3 |
| 1.22.0.1 | 1.22.0.2 | 0.4 |

Table 2.3 Probability Interaction Table

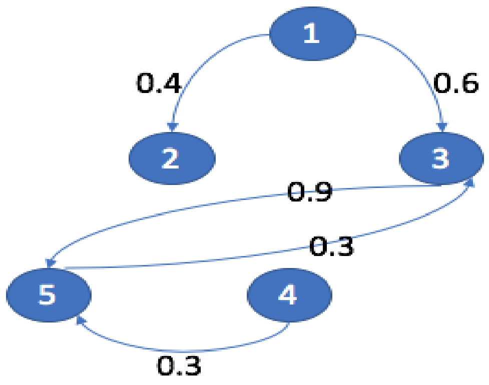


Fig. 2.6 Graph From Probability Table

For example, from Table. 2.3 and Fig. 2.6 the paths leading to **service 5** are:

- **<u>Path1: 1-3-5</u>**. The probability of Path1 is the probability that the invocation of service 1 will be followed by the invocation of the service 3, and the invocation of the service 3 will be followed by the invocation of the service 5. Probability = **0.6*0.9 = 0.54**
- **<u>Path2: 3-5</u>**. The probability of Path2 is the probability that the invocation of service 3 will be followed by the invocation of the service 5. Probability = **0.9**
- **<u>Path3: 4-5</u>**. The probability of Path3 is the probability that the invocation of service 4 will be followed by the invocation of the service 5. Probability = **0.3**

## 2.4 Implementation and Experiments

### 2.4.1 Implementation

We implement the proposed approach in Java environment and mainly consist of two steps. First, we need to find the interaction between the services. Therefore, we use Apache Spark and the parallel FP-Growth algorithm from the machine learning library MLIB for mining the service logs. The output from the mining would be the interaction of the services with their probability of that interaction. Second, from the first step we are able to generate a graph of that interaction as it shows in Fig. 2.7 , which will provide a useful information to compute the anonymity. We use  Jung and Jgrapht libraries for drawing and extracting the graphical information which will be use in the anonymity computation.

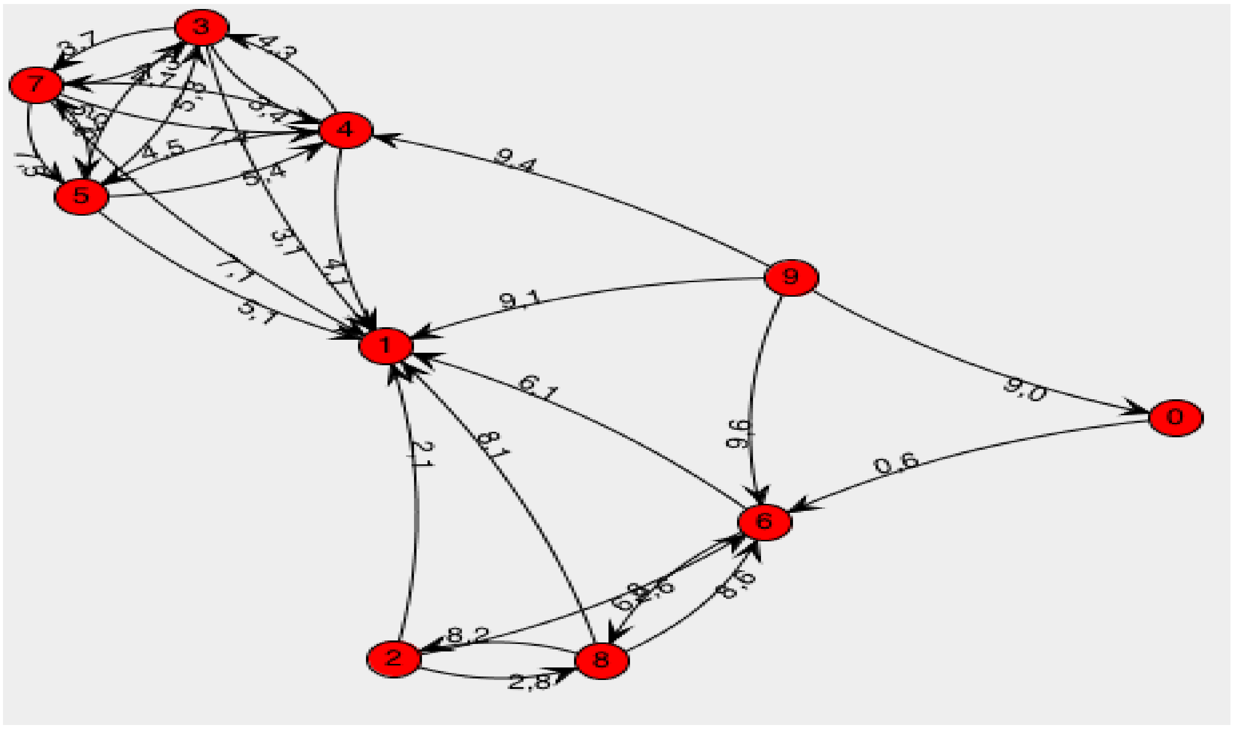


Fig. 2.7 Graph of services interaction

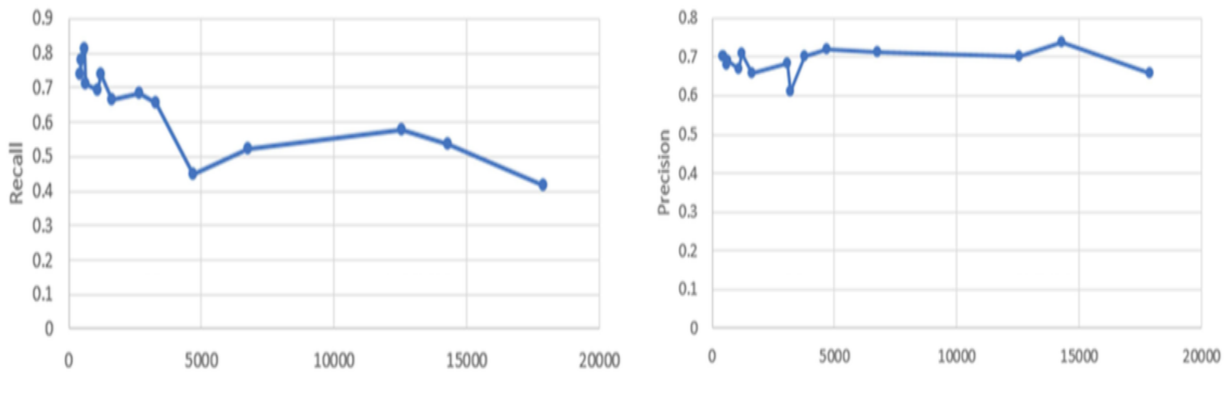
### 2.4.2 Experimental set-up

We ran our experiments on a macOS Sierra 10.12.5 environment, in a machine equipped with an intel i7 and 16 GB RAM. We used Java as a platform for processing log files. We use Apache Spark and the parallel FP-Growth algorithm from the machine learning library MLIB. It is more convenient for processing big amount of data. It leverages parallel computation algorithms and splits the computation task among several processors. Each service has been assigned a log file and an IP address. We simulated the invocations among a set of Web services. We recorded the details of the simulated interactions as events in logs in a such way that each service $S_i$'s invocation of service $S_j$ is recorded in $S_j$'s log file. The goal of the experiments is to assess the ability of the approach to learn which services are part of the same choreography relationship, and to learn the probabilities of interactions of the Web services.

### 2.4.3 Results

In this section, we vary the number of Web services from 100 to 1000, we simulate several invocations among services. The details of invocations are stored as events in log files. We assess the ability of the approach at correctly learning which invocations are part of the same choreography relationship by running the previously described approach.

We compute the precision and the recall of the learning process. Fig. 2.8 (a) and (b) show that **80%** of the choreography relationships have been learned with **70%** precision. Fig. 2.8 (a) shows that the percentage of the learned interaction windows (i.e., choreography relationships) decreases as the number of simulated interactions windows gets very high. The justification is that as we increase the number of interactions among services, the probability to observe frequent invocation patterns that satisfy the support of the FP-Growth algorithm decreases, indeed an invocation pattern needs to be very frequent such that it satisfies the support otherwise the choreography relationship won't be inferred among the services of that invocation pattern. On the contrary, Fig. 2.8 (b) shows that the precision of the learning process is not affected by the number of simulated interactions windows among services. The justification is that the percentage of the correctly learned choreography relationships increases as we increase the number of simulated interaction windows.



(a) (b)

Fig. 2.8 Initial Interaction Windows

# CHAPTER 3

# Probabilistic K-Anonymity for Web Services

## 3.1 Introduction

To improve computation of accurate anonymity between the services, it is important to calculate the probability of previous interaction between services. Since the interaction occurs in web services choreography, between abundant clients and providers with different dependency. Moreover, nowadays the number of services increase while some services provide the same functionality. However, the developers cannot determine which service meet their privacy requirements.

The existing solution of calculating the level of privacy between the services is k-anonymity. [17] The k-anonymity is used to determine the extent to which invocation of an operation can be inferred if downstream operation was invoked. In practice, as a web service interact [8] they begin to knit creating groups of partners with highly dense relationships which tend to have different probability between each other. [14] The maximum degree of anonymity has been achieved when the all services seen as equally probable of being the originator of the invocation.

In this chapter, we discuss how do we perform the calculation of the anonymity based on the k-anonymity and probabilistic k-anonymity. We also discuss how the developer can select the best services that achieve a better anonymity rely on the mining service log.

On one hand, we calculate the k-anonymity between the services, we rely on the chorography graph and we assume the probability of each participant as equiprobable since the k-anonymity approach does not provide information related to the probability of downstream invocation. On the other hand, we use the obtained probability to compute the probabilistic k-anonymity among the Web services

The rest of the chapter is organized as follows: Section 3.2 describes our knowledge about k-anonymity. Section 3.3 explains briefly the Web service privacy model. Section 3.4 the proposed approach for calculating the probabilistic k-anonymity based on the previous interaction, and how the developer can find the best services to meet the privacy requirement. Section 3.5 describes the experimental setup and discusses the obtained results.

## 3.2 Background

K-anonymity has been proposed by Samarati and Sweeny [20] to enhance the privacy requirements by using generalization and suppression techniques in relational database. The aim of the approach is to prevent linking individual to a specific record in data table by defining quasi-identifier QID. In order to have k-anonymous table, for one record must have qid value, at least k-1 other record have qid value. For instance, Table. 3.1 has 2-anonymous with QID= Race, Birth, Sex , ZIP.

| **Race** | **Birth** | **Gender** | **ZIP** | **Disease** |
|---|---|---|---|---|
| White | 1964 | m | 0214* | Obesity |
| White | 1964 | m | 0214* | chest pain |

|  |  |  |  |  |
|---|---|---|---|---|
| Black | 1966 | m | 0215* | Depression |
| Black | 1966 | m | 0215* | Cancer |
| Black | 1966 | m | 0215* | short breath |

Table 3.1 K-Anonymity For K=2

From our previous work [8], we leverage the notion of k-anonymity in operation level to guarantee the privacy between the services by using privacy management framework. We defined the k-anonymity to determine the extent to which invocation of an operation can be inferred if downstream operation was invoked. The computation of k-anonymity rely on Web Services Conversation Language (**WSCL**) definitions. However, in this thesis we extend the previous work , we proceed to the analysis and mining of services interaction in logs to compute an accurate probabilistic k-anonymity. Probabilistic k-anonymity take advantage of the previous interaction which provide an accurate value of anonymity.

## 3.3 Web Service Privacy Model

In our previous work, we defined a model that deals with the privacy of Web service selection. The aim of the model is to protect the privacy of the information at both data and operation usage levels. The model supported by a protocol which handle the matching and negotiation issues in case of incompatibility between client requirements and provider's policies. We briefly describe the Web service privacy model and refer the reader to [21][22] for more information.

**Web Service Privacy Model**. In our model, the first definition is the privacy policy $PP^{WS}$ for each WS provider. Second, since for each provider WS may have a client C which defines the defines privacy requirements $PR^{C/WS}$.The privacy requirements $PR^{C/WS}$ consider the information regarding input or output data and operations. C may need to check the compatibility types between $PP^{WS}$ and $PR^{C/WS}$ full or partial compatibility with certain threshold. In case of incompatibility between $PP^{WS}$ and $PR^{C/WS}$ , C and WS have two options: first, discontinue the interaction. Second, start a negotiation process to reconcile privacy policies and requirements. The model consists of the following six concepts:

- Resource: refer to the input or output data or invoked operation may consider as a private.
- Privacy level: the model need to check the privacy at data and operation levels. The data level considers the privacy of the input and output data of the operation which is consider as a data resource. For instance, for an operation that returns car insurance quotes, we use the input date resource (e.g., driverLinces#.) and output data resource (e.g. quotes_results). The operation level consider the privacy of usage particular operation.
- Privacy Rule is used to define the sensitivity of the resource. The rule $R_i$ consist of four parts and represented as a tuple $\langle T_i, L_i, D_i, S_i \rangle$. First, the topic $T_i$ which is gives the privacy facet of the rule and it takes the values {Purpose, Retention, Recipient, KAnon}. Second, the privacy level $L_i$ which is represent the privacy level of all resource and it takes the values {data, operation}. Third, the domain $D_i$ which can represent the possible values that can be used by a topic {noretention, indefinitely, statedpurpose , public, government, research, federaltax, same otherservices}. Finally, the scope $S_i$ which is the defines the granularity of the resource that is subject to privacy constraints in both operation and data, and may assign by different values {total, partial, GTE1, GTE2, GTE3, GTE4,INFINITE}.

In case of using K-Anonymity, the GTEx are assigned by K-Anonymity value where it should be at least as large as this value (INFINITE represents that the operation does not invoke by any downstream operations). Moreover, if the operation is assign by total scope, that indicate the whole entry of operation is private. The partial scope may assign only for operation because the data resources consider as atomic. Hence, the only scope value allowed is total.

- Privacy Assertion $A(R_i,rs)$ is the application of a rule $R_i=\langle T_i,L_i ,D_i,S_i\rangle$ on a resource rs and consist of two pair $\langle pf , g\rangle$. The pf is a propositional formula and defined as $pf = d_i \wedge \cdots \wedge d_j$ $d_i,...,d_j \in D_i$. The g represent the granularity of rs that is subject to privacy and defined as $g \in S_i$. For instance, the privacy assertion may state that a resource can be shared with car insurance company and government record (driver license record ). The propositional formula will be (car_insurance_company $\wedge$ government_record) to specify such statement.
- Privacy Policy $PP_{WS}$ which need to define for each $WS_i$ as a set of assertions that the service specifies on the resources to WS clients
- Privacy Requirement $PR^{C/WS}$ for each client C defines C's assertions about WS resources. $PR^{C/WS}$ assertions required two requirements. First, identifies C's expectation of the privacy of resources from the provider WS (noted as $A(Ri,rs^E)$). Second, C's practices regarding the returned value of WS (noted as $A(Ri,rs^P)$). Clients may assigning a value weight $w_j$ to each A(Ri, rs) in $PR^{C/WS}$. Moreover, higher the weight, reflect more important corresponding assertion. The weight represented by a decimal number from 0 to 1. The total of weights assigned to all assertions is equal to 1. The clients may control their privacy

requirements by assign the attribute $m_j$ to each assertion $(A_j(R_i,rs_k),w_j,m_j)$ in $PR^{C/WS}$.

**Privacy Preserving Web Service Selection**. In service oriented environment the composition service rely on other services invocation. The service composition plan CP, defined as any service WS rely on another service WS′. In other words, WS considered as a consumer of the data which is provided by WS′. Therefore, WS represented as a client and WS′ represented as a provider. In order to check the privacy compatibility, we used Privacy Compatibility Matching PCM algorithm to ensure the compatibility between the PR of WS and PP of WS′. The PCM algorithm considers a CP as privacy compatible in case if the all dependencies fully satisfied. If any dependency in CP dose not satisfied the assertion, then CP violates privacy which lead to discarded from CP.

**Privacy Compatibility Matching (PCM) Protocol**. The aim of the algorithm is to ensure that the assertions in both $PR^{WS}$ and $PP^{WS}$ are fully compatible in case of full compatibility. However, in case of partial matching the PCM combine the notion of matching degree which can be estimated as the ratio of $PR^{C/W\,S}$ assertions that are matched to $PP_W$ S assertions. We refer to $M \subset PR^{C/W\,S}$ as the set of all such $PR^{C/WS}$ assertions. Then, we can compute the degree by adding the weights of all assertions in M:

$$\text{Degree}(PR^{C/WS}, PP\ ) = w_j \,\forall\, (A_j\,(\,R_i,\,rs\,),w_j\,,\,m_j) \in M$$

The PCM and threshold $\tau$ is used to find the compatibility. The threshold $\tau$ provided by the client with minimum value allowed for matching degree. The PCM has two cases to determines that $PR^{C/W\,S}$ and $PP_W$ S are compatible:

- The privacy matching degree is above the threshold set by C: $\text{Degree}(PR^{C/WS},PP_{WS}) \geq \tau$.

- Every non-matched $PR^{C/W\ S}$ assertion is optional: $\forall ( A_j (R_i, rs_k ), w_j, m_j) \in (PR^{C/WS} - M) : m_j =$ “False”.

## 3.4 The Proposed Approach

The proposed approach consists of three stages. Create the graph from the probability table, calculate the probability of possible path and normalizing the probability and calculate the Anonymity based on the probability

### A. Create the graph from the probability of the interaction

The choreography graph rely on the probability table from the interaction between the services as we describe in Chapter 2. We represent each node and edge as a service and invocation of that service respectively. For example, in case of five services interaction. In the graph, we represent the IP address 1.22.0.1 as 1 and 1.22.0.2 as 2. Fig. 3.1 reflects the information of Table 3.2 as below.

| $IP_{source}$ | $IP_{destination}$ | Probability |
|---|---|---|
| 1:22.0.1 | 1:22:0.2 | 0.4 |
| 1.22.0.1 | 1.22.0.3 | 0.6 |
| 1.22.0.1 | 1.22.0.4 | 0.2 |
| 1.22.0.2 | 1.22.0.1 | 0.5 |
| 1.22.0.3 | 1.22.0.5 | 0.9 |
| 1.22.0.4 | 1.22.05 | 0.3 |
| 1.22.0.5 | 1.22.0.3 | 0.9 |

Table 3.2 Probability table of a 5 service

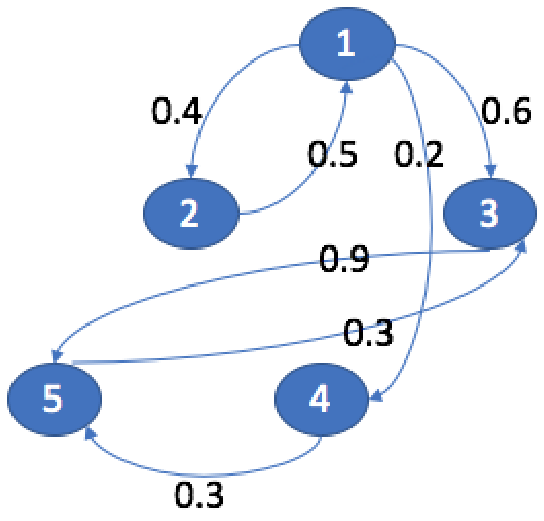


Fig. 3.1 Choreography of a 5 service

## B. Find the all possible path of interaction

We take advantage of the graph to find all possible path between each node and other nodes in the choreography as show an Algorithm 2. For instance, in case of service 1.22.0.2 compute all the possible path with all other nods and we get a list of indirect interaction of 1.22.0.3, 1.22.0.4 and 1.22.0.5. Moreover, the Algorithm 2 compute the probability of possible path by **multiplying** the probability of each path leading to the destination. Table 3.3 it shows all possible path in Fig. 3.1. Let P is the probability of indirect path and N is number of path leading to the destination node, as it shows below:

$$\prod_{i=1}^{N} Pi$$

| $IP_{source}$ | $IP_{destination}$ | **Probability** |
|---|---|---|
| 1.22.0.1 | 1.22:0.5 | 0.54 |
| 1.22.0.2 | 1.22.0.3 | 0.3 |
| 1.22.0.2 | 1.22.0.4 | 0.1 |
| 1.22.0.2 | 1.22.0.5 | 0.03 |
| 1.22.0.4 | 1.22.0.3 | 0.9 |

Table 3.3 Possible interaction of indirect paths

**Algorithm 2:** Calculate possible path

**Input:** probabilityArray, number-service
**Output:** possibleArray

```
 1: List possiblePathList←null, possibleProbability←1, $S_{source}$ , $S_{destination}$ ← 0
 2:  for $S_{source}$ ← 0 to  number-service do
 3:      for $S_{destination}$ ← 0 to number-service do
 4:             If(getPath($S_{source}$ , $S_{destination}$)!=0)
 5:                     possiblePathList← getPath($S_{source}$ , $S_{destination}$)
 6:                     for each path ∈ possiblePathList do
 7:                             possibleProbability= possibleProbability * possiblePathList.split()
 8:                             possibleArray [$S_{source}$] [$S_{destination}$] ← possibleProbability
 9:                     end for
10:             end if
11:      end for
12: end for
13: return possibleArray
```

### C. Normalizing the probability and calculate the k-Anonymity

First, we have all probabilities of possible interaction(direct and indirect interaction). We need to normalize the probability of all invocations. We assume that the **sum of all** probabilities for invoking particular service **equal one**. For example, 1.22.0.4 may invoked by 1.22.0.1 and 1.22.0.2 with different probability 0.2 and 0.1 respectively. After the normalization will be 0.67 and 0.33. Let $Pn$ be the probability after normalization, $P$ the probability before normalization and the $Pi$ is the sum of all probabilities of invoking specific service.

$$Pn = P \div \sum_{i=1}^{N} Pi$$

Algorithm 3 takes as input the possible Array which include all direct and indirect interaction and the number of service and return normalized Array for all possible paths.

**Algorithm 3**: Normalizing the Probability

```
Input: possibleArray, probabilityArray, number-service
Output: normalizedArray
 1: value,  total, $S_{source}$ , $S_{destination}$ ← 0
 2: for $S_{destination}$ ← 0 to  number-service do
 3:     for $S_{source}$ ← 0 to number-service do
 4:         value←probabilityArray[$S_{source}$][$S_{destination}$]
 5:         total←total+value
 6:     end for
 7:     for $S_{source}$ ← 0 to number-service do
 8:         normalizedArray[$S_{source}$][ $S_{destination}$] ← probabilityArray/total
 9:     end for
10: end for
11: return normalizedArray
```

Second, we need to calculating the k-anonymity. The k-anonymity is the number of routes leading to each downstream. From the previous example, the k-anonymity of 1.22.0.4 will be 2 if the source is 1.22.0.2 ,because there are two possible invocations can appear, it could be from 1.22.0.2 or 1.22.0.1. In that case the **k-anonymity** of  1.22.0.4 is **0.5** for both 1.22.0.2 and 1.22.0.1. Algorithm 4 shows the computation of k-anonymity

**Algorithm 4**: Calculate k-anonymity

```
Input: normalizedArray , number-Service
Output: k-anonymity
1: $S_{source}$ , $S_{destination}$ ← 0
2: for $S_{source}$ ← 0 to  number-service do
3:     for $S_{destination}$ ← 0 to  number-service do
4:         k-anonymity ← numOfService ($S_{source}$ , $S_{destination}$)
5:         k-anonymity ← 1/ k-anonymity
6:     end for
7: end for
7: return k-anonymity
```

By comparing the probability of k-anonymity which is equiparable with the probability of our approach, we can find the most optimal selection of services which meet the developer privacy requirements in case if multiple services offer same functionality. The approach take two inputs to find the optimal privacy. The list of services which have same functionality and the privacy requirement which is a number between 0 and 1. For example, from Table 3.4 if two APIs A and B have the same k-anonymity that imply they are equiparable based on k-anonymity, but they have different probabilistic k-anonymity of invocation from previous interaction. In case if the privacy requirement of the developer is 0.4 , the approach will search for the **lowest probability** of interaction that meet the user requirement. Therefore, the approach will return API A as an optimal API to integrate with.

| **API** | **K-Anonymity** | **Probabilistic K-Anonymity** |
|---|---|---|
| A | 0.5 | 0.3 |
| B | 0.5 | 0.7 |

Table 3.4 APIs with same k-anonymity

## 3.5 Experiments

First, we overview our experimental set-up. Then, we evaluate the k-anonymity approach with our approach for selecting services

### 3.5.1 Experimental Set-up

We ran our experiments on a macOS Sierra 10.12.5 environment, in a machine equipped with an intel i7 and 16 GB RAM. We used Java as a platform for building the graph and visualizing it.

We use the library Jung that allow building and visualizing simple and complex graphs. Jung also includes pre-built algorithms that allow finding the shortest paths between nodes invocation of service $S_j$ is recorded in $S_j$'s log file. The goal of the experiments is to find an accurate anonymity of Web services by comparing the k-anonymity approach with probabilistic k-anonymity approach.

### 3.5.2 Results and Discussion

In this section, we compare accuracy of each the of the **probabilistic k-anonymity approach** and the **k-anonymity approach** while used for Web service selection. The **probabilistic k-anonymity approach** takes into consideration the past history of interaction of Web service to compute the anonymity among Web services. It relies on mining service logs to determine the probability of interaction of each two services. The previous probability is then used to compute the anonymity among each two services. The computed anonymity allows the developer to accurately select the services that satisfy his anonymity requirements.

The **k-anonymity approach** relies on the network topology (i.e., the number of incoming edges to each service) to compute the anonymity and does not take advantage from service interaction history. All interactions are assigned equiprobable values.

Developers willing to build a mashup are facing the challenge to select the best services that satisfy their functionality and privacy requirements among a large space of services. A mashup is a composite service that orchestrates the invocation of a set of specific APIs to achieve a specific functionality. In the following, we describe some mashup developer real functional and privacy requirements:

**Scenario 1:**

Mashup1 needs to provide a composite service for car rental. Mashup1's aims first to provide to the customer a Maps service for geographic location selection. Once the customer selects his geographic location, the mashup needs to recommend to the customer a list of car rental services that are available near to the location he selected. To help the customer at choosing the car that meets his expectations, Mashup1 needs to invoke a service that provides updated information about each car and customers reviews. Mashup1 needs also to invoke a cloud storage service that saves the details of customers choices. This scenario consider one API have many candidates which is the mapping API. Fig. 3.2 illustrates the choreography which include Mashup1.

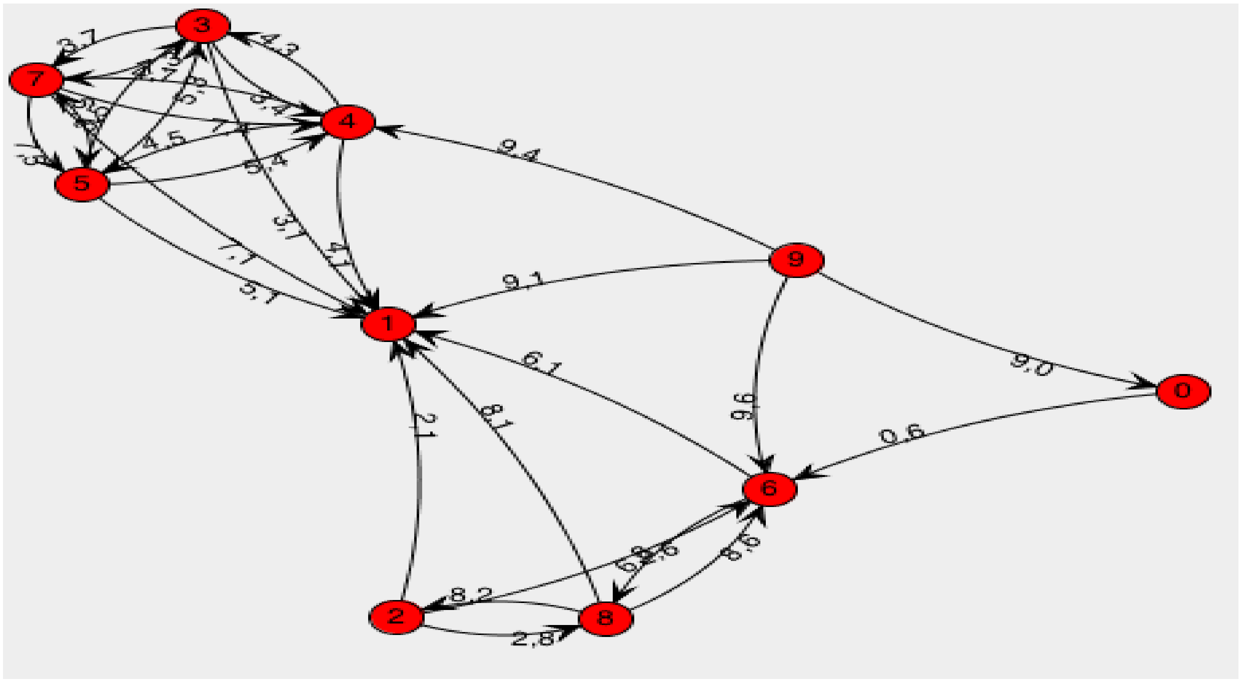


Fig. 3.2 Chorography interaction

**API 1:**

Mapping service which can provide the same functionality

**Candidates Service for API 1**

1. Google Maps API ( service 6)
2. Bing Maps API  ( service 7)
3. OpenStreetMap API ( service 8)

**API 2:**

**Rental Car Manager API** the service provides a search, reservation, and booking platform for car rentals. It allows travelers to shop for rental cars and complete reservations. Agencies can track car availability and rental activity at their locations, while also reporting on sales and analyzing trends. API methods support retrieval of vehicle categories, types, and availability for specified pick-up and drop-off locations and times. Methods support selection of an option presented and completion of the reservation booking process. The API also supports management of location operating information, such as staff members, open hours, and available inventory.

**API 3:**

The **Yelp Fusion APIs** are RESTful APIs and users can retrieve business review and rating, information for a particular geographic region or location. display review information for a particular business, determine accurate neighborhood name information for a particular location, track recent reviews for a particular business, display pictures of highly rated local businesses and of the top reviewers for that business, determine a particular business' review and rating information based on the phone number for that business.

**API 4:**

**Google Drive API** is a cloud based storage platform that lets users access their data, including files of any format, from any device or application that connects to the internet. The Google Drive SDK includes an HTTP API that lets developers integrate the files stored in a user's drive with their own third party applications. This gives users the ability to use multiple cloud apps to interact with their files that are stored in a single location in the cloud.

If the developer assign the **Required Anonymity 0.18** for the Mashup and the mapping APIs have candidate APIs that support the same functionality as below:

**API 1**: Google Maps

**API 2**: Bing Maps

**API 3:** OpenStreetMap

Mashup developers sift through a large space of APIs to select the ones that satisfy their anonymity requirements. The challenge is that many APIs provide comparable functionalities while the anonymity between the different APIs is different. In the following, we show how large is the space of choice for the different APIs. Then, we evaluate the accuracy of two types of developers Dev1 and Dev2 at selecting Web services that satisfy their anonymity requirements. Dev1 relies on the k-anonymity approach for Web service selection, Dev2 relies on the probabilistic k-anonymity. Each entry of Table denotes a source service and a downstream service, their k-anonymity, and their probabilistic k-anonymity. Table 3.5 shows that services S6, S7 and S8 have **similar k-anonymity** , however their **probabilistic k-anonymity is different.**

| Source | Downstream | Route | K-Anonymity | Probabilistic k-anonymity |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 9 | 6 | 4 | 0.25 | 0.245 |
| 9 | 7 | 4 | 0.25 | 0.15 |
| 9 | 8 | 4 | 0.25 | 0.222 |

Table 3.5 Mapping services information

We identify the following sets of services that provide comparable functionalities and different probabilistic k-anonymity for S6, S7 and S8. Dev1 chooses **randomly** S6, S7 and S8. S7 satisfies the anonymity requirements (0.18>0.15) while S6 and S8 did not (0.18<0.245 or 0.18<0.222). Only one service among the three services have been selected successfully. The **maximum accuracy of selection is around 50%.**

Dev2 relies on **probabilistic k-anonymity** for selecting S7. Both services S6 and S8 does not satisfy the anonymity requirements. S7 have been selected successfully, **accuracy of selection = 100%** as it shows in Fig. 3.3 .

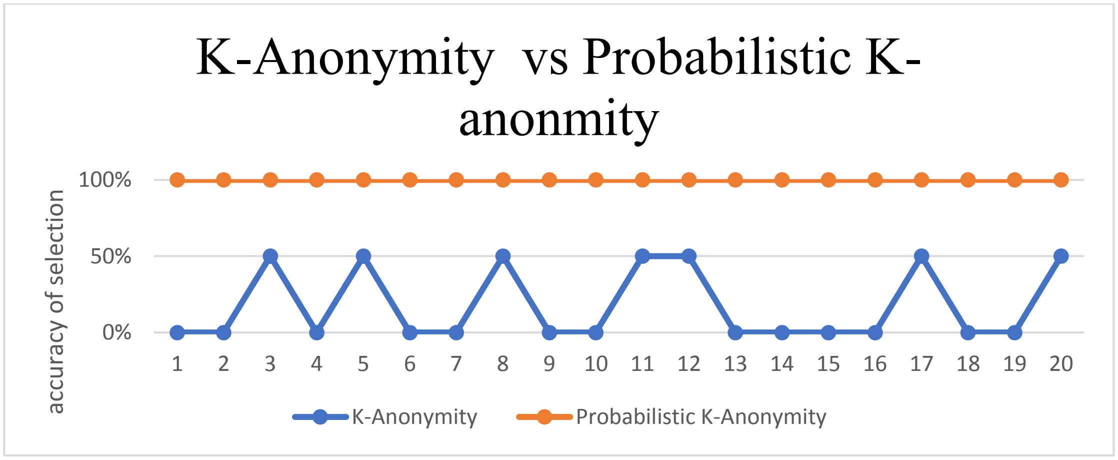


Fig. 3.3 K-Anonymity vs Probabilistic K-Anonymity

**Scenario 2:**

Mashup required functionality: Coined the **eHarmony** of home finding, Better Home integrates the 8 different API calls to match a user with their ideal home. Currently, the mashup is only specific to San Francisco, CA. This scenario consider many APIs have many candidates. Fig. 3.4 illustrate the choreography of the second scenario which include eHarmony mashup.

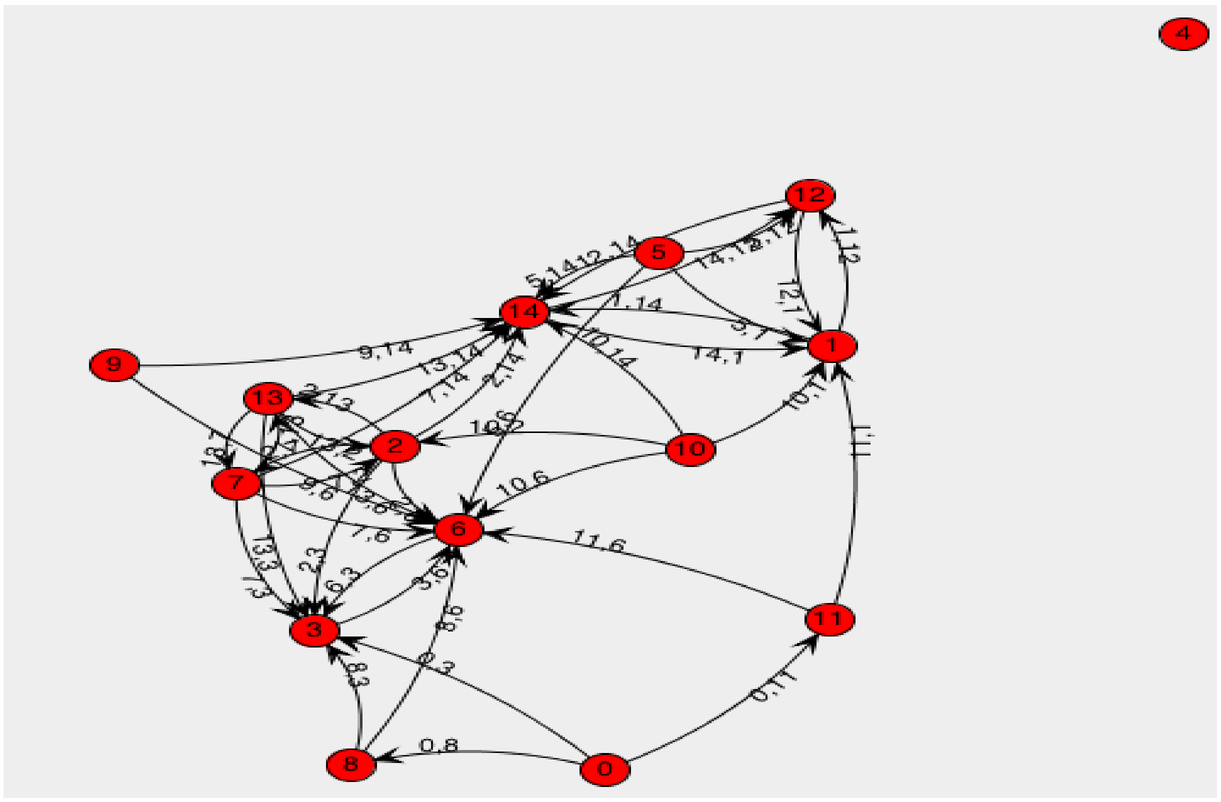


Fig. 3.4 Choreography include eHarmony Mashup

**API 1:**

The **Zillow API** is a network turns member sites into mini real estate portals by offering fresh and provocative real estate and mortgage content to keep people coming back. There are four categories of APIs. 1) Home Valuation API: Search results list, Zestimate™ home valuations, home valuation charts, comparable houses, and market trend charts. 2) Property Details API:

Property-level data, including historical sales price and year, taxes, beds/baths, etc. 3) City and neighborhood market statistics and demographic data 4) Mortgage rates and monthly payment estimates

**API 2:**

**Walk Score API** calculates the walkability of an address based on the distance from a house to nearby amenities. The Walk Score API uses a RESTful interface and returns the Walk Score for any latitude and longitude in the U.S. in XML or JSON format.

**API 3:**

The **Google Maps API** allow for the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript interface or a Flash interface. It is designed to work on both mobile devices as well as traditional desktop browser applications. The API includes language localization for over 50 languages, region localization and geocoding, and has mechanisms for enterprise developers who want to utilize the Google Maps API within an intranet. The API HTTP services can be accessed over a secure (HTTPS) connection by Google Maps API Premier customers.

**Candidates Service for API 3**

1. **Bing Maps API** is the re-branded name for the Microsoft Virtual Earth API and Virtual Earth SDK. It features an AJAX map control. Use Bing Maps to build maps which can include routes and traffic info. Gives developers the ability to code the controls, shapes, and layers of the maps, and can summon the birds-eye, 3D, and aerial imagery. For

commercial applications there is a SOAP-based Web Service also provides access to the Bing maps and geospatial features.

2. **OpenStreetMap API** is the free wiki world map, an open volunteer-driven initiative to collaboratively create a map of the world, and release the map data under a free and open license. There are actually many different APIs in and around the OpenStreetMap ecosystem. Many developers searching for an API, may actually be looking for an JavaScript web mapping library

**<u>API 4:</u>**

**Geocoder.us API** is a public service providing free geocoding of addresses and intersections in the United States. Using the service you can find the latitude & longitude of any US address and much more. Geocoder.us offers four different ways to access our web services: an XML-RPC interface, a SOAP interface, a REST interface that returns an RDF/XML document, and a REST interface that returns a plain text comma separated values result. The methods and return values are equivalent across all three interfaces.

**<u>API 5:</u>**

**Trulia API** is an online residential real estate search engine that lists information on properties for sale, real estate trends and neighborhood information. With the Trulia API, developers can add real estate data to their applications. Available data includes: neighborhoods in a city, cities and counties in a state, and geo-location information. Developers also have access to Trulia traffic statistics and sale listing data. The API uses RESTful calls with responses formatted in XML.

**<u>Candidates Service for API 5</u>**

1. **Real Estate MLS Cloud API** provides various services for real estate listings and Customer Relationship Management (CRM). Their API currently provides services for searching and retrieving real estate data. This is a REST-based API that returns data in JSON, and requires an API key for access

**API 6:**

**Factual API** is an open data platform for application developers that leverages large-scale data aggregation and community exchange. Our focus is on making data more accessible (i.e. cheaper, higher quality, less encumbered) for machines and developers, to drive and accelerate innovation in an unprecedented way. We take on the dirty work of data management and data curation, letting developers focus on higher value and more productive tasks. We provide clean, structured data with complete source transparency to developers via both download and API access on liberal terms.

**API 7:**

The **Yelp Fusion APIs** are RESTful APIs and users can retrieve business review and rating, information for a particular geographic region or location. display review information for a particular business, determine accurate neighborhood name information for a particular location, track recent reviews for a particular business, display pictures of highly rated local businesses and of the top reviewers for that business, determine a particular business' review and rating information based on the phone number for that business. The default output is JSON. This output format was chosen due to the availability of JSON parsers in many languages. The following Yelp

Fusion APIs are available: Search, Phone Number Search, Business Search, Transaction, Reviews, and Autocomplete - each API has a separate ProgrammableWeb entry.

<u>**Candidate Service for API 7**</u>

1. **ReatItAll API** use the RateItAll Consumer Rating API to add to your consumer reviews with some of the millions of consumer ratings in RateItAll's database, to collect and display ratings and reviews from your user base, and to distribute and promote your data to RateItAll.com's large user base. Also, monetization opportunities are available.

<u>**API 8:**</u>

**Socrata API** is an online community for producers, publishers, and consumers of data. Through a suite of innovative Web services, Socrata provides the world's most comprehensive platform for open data discovery. Socrata APIs are sets of REST resources you can use to manage Socrata entities and data. Resources are grouped by areas of related high-level functionality.
Service Candidates for API8:

If the developer assign the **Required Anonymity 0.15** for the Mashup and there are three APIs have candidate APIs that support the same functionality:

**API 1**: Google Maps, Bing Maps and OpenStreetMap

**API 2**: Real estate APIs and Trulia or MLS

**API 3:** Yelp Fusion and ReatItAll

From Tables 3.6 , 3.7 and 3.8 the optimal selection that satisfied the mashup required anonymity will be **S6** for <u>**API 1**</u> , **S12** for <u>**API 2**</u> and **S 13** for <u>**API 3**</u>

**API 1**

| Source | Downstream | Route | K-Anonymity | Probabilistic K-Anonymity |
|---|---|---|---|---|
| 0 | 1 | 10 | 0.1 | 0.108 |
| 0 | 3 | 10 | 0.1 | 0.153 |
| 0 | 6 | 10 | 0.1 | 0.105 |

Table 3.6 Candidate Services information API 1

**API 2**

| Source | Downstream | Route | K-Anonymity | Probabilistic K-Anonymity |
|---|---|---|---|---|
| 1 | 12 | 10 | 0.1 | 0.108 |
| 1 | 14 | 10 | 0.1 | 0.153 |

Table 3.7 Candidate Services information API 2

**API 3**

| Source | Downstream | Route | K-Anonymity | Probabilistic K-Anonymity |
|---|---|---|---|---|
| 2 | 7 | 3 | 0.333 | 0.343 |
| 2 | 13 | 3 | 0.333 | 0.323 |

Table 3.8 Candidate Services information API 3

By comparing the number of right selection that meet the privacy requirements to develop the eHarmony mashup with the two approaches. We can see that random selection based on the k-anonymity did not pick the right three services. While probabilistic k-anonymity provide the right services.

The **Accuracy** is the number of APIs that are successfully selected (that fit the privacy and functionality requirement of the developer) divided by the number of all APIs. It is the fraction of relevant APIs that are selected.

Fig. 3.5 compares the accuracy of the probabilistic k-anonymity approach and the k-anonymity approaches. It shows that up to 100% of APIs are accurately selected by DEV1 who adopts the probabilistic k-anonymity approach. However, this number decreases for DEV2 who adopts the k-anonymity based approach with 46%. The justification is that adopting probabilistic k-anonymity approach provides the developer with realistic information about the probability of the dependency between the different invocations among services. Hence, the developer can make the right choice of services with low dependency. On the other hand, the k-anonymity approach assigns equiprobable values for all interactions, hence the developer chooses **randomly** the services to select when the offer comparable functionalities.

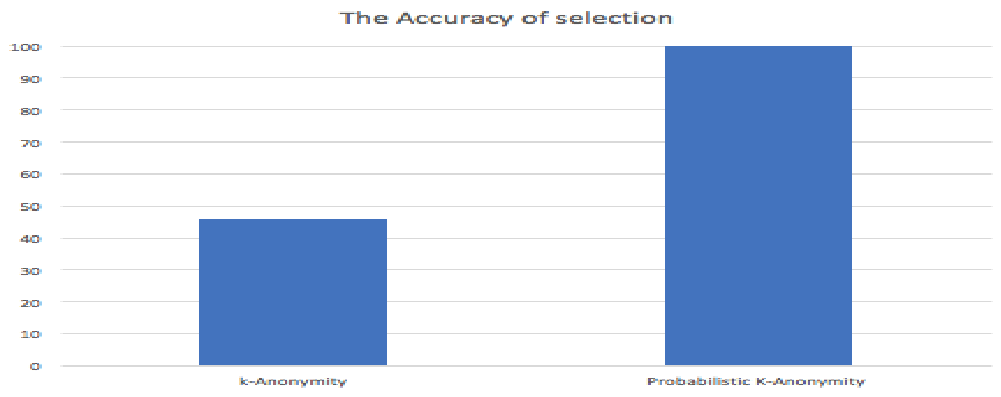


Fig. 3.5 The Accuracy of selection

# CHAPTER 4

# Conclusion

We proposed a novel approach for privacy-preserving mining of Web service conversations. The proposed approach supports developers in their task of selecting the services that best satisfy their privacy requirements. We used first the FP-Growth algorithm for mining service logs. The mining identified which services are part of the same choreography and inferred the probabilities of services future interactions. We used the probabilities of interactions to compute a new interaction-aware anonymity of Web service invocations. We conducted extensive experiments to prove the efficiency of the proposed approach. We used real APIs deployed on Programmableweb.com. We compared the accuracy of Web service selection of both the proposed probabilistic k-anonymity approach and the regular k-anonymity. We showed that the current approach by considering service interactions history achieves a much better accuracy than the regular k-anonymity, which assumes equiprobable service invocations. We plan to test the proposed approach over real-world service logs. We also plan to integrate our approach is real Web service engines.

# Bibliography

[1] Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In ACM SIGMOD Record (Vol. 22, No. 2, pp. 207-216). ACM.

[2] Ammar, N., Malik, Z., Medjahed, B. and Alodib, M., 2015, June. K-anonymity based approach for privacy-preserving web service selection. In Web Services (ICWS), 2015 IEEE International Conference on (pp. 281-288). IEEE.

[3] Banerji, A., Bartolini, C., Beringer, D., Chopella, V., Govindarajan, K., Karp, A., ... & Williams, S. (2002). Web services conversation language (wscl) 1.0. W3C Note, 14.

[4] Berthold, O., Pfitzmann, A., & Standtke, R. (2001). The disadvantages of free MIX routes and how to overcome them. In Designing Privacy Enhancing Technologies (pp.30-45). Springer Berlin Heidelb

[5] Bi, K., Han, D., & Wang, J. (2016). K maximum probability attack paths dynamic generation algorithm. Computer Science and Information Systems, 13(2), 677-689.

[6] Diaz, C., Seys, S., Claessens, J., & Preneel, B. (2002, April). Towards measuring anonymity. In International Workshop on Privacy Enhancing Technologies (pp. 54-68). Springer Berlin Heidelberg.

[7] Esfahani, N., Yuan, E., Canavera, K.R. and Malek, S., 2016. Inferring software component interaction dependencies for adaptation support. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 10(4), p.26.

[8] Fronza, I., Sillitti, A., Succi, G., Terho, M. and Vlasenko, J., 2013. Failure prediction based on log files using random indexing and support vector machines. Journal of Systems and Software, 86(1), pp.2-11.

[9] Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. In ACM SIGMOD Record (Vol. 29, No. 2, pp. 1-12). ACM.

[10] Labbaci, H, Medjahed, B, Aklouf, Y. Learning Interactions from Web Service Logs. DEXA 2017.

[11] Labbaci, H., Medjahed, B., Aklouf, Y., & Malik, Z. (2016, October). Follow the Leader: A Social Network Approach for Service Communities. In International Conference on Service-Oriented Computing (pp. 705-712). Springer International Publishing.

[12] Lemos, A.L., Daniel & F., Benatallah, B. (2016). Web service composition: A survey of techniques and tools. ACM Comput. Surv. 48(3), 33:1{33:41 (2016).

[13] Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.Y.: Pfp: parallel fp-growth for query recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008. pp. 107{114 (2008)

[14] Motahari-Nezhad, H.R., Saint-Paul, R., Casati, F. and Benatallah, B., 2011. Event correlation for process discovery from web service interaction logs. The VLDB Journal—The International Journal on Very Large Data Bases, 20(3), pp.417-444.

[15] Nie, X., Zhao, Y., Sui, K., Pei, D., Chen, Y. and Qu, X., 2016, December. Mining causality graph for automatic web-based service diagnosis. In Performance Computing and Communications Conference (IPCCC), 2016 IEEE 35th International (pp. 1-8). IEEE.

[16] Papazoglou M. & Van den Heuvel W. (2007). Service-oriented architectures: approaches, technologies and research issues. The VLDB Journal, 16(3):389{415.

[17] Reguieg, H., Benatallah, B., Nezhad, H.R.M. and Toumani, F., 2015. Event correlation analytics: Scaling process mining using MapReduce-aware event correlation discovery techniques. IEEE Transactions on Services Computing, 8(6), pp.847-860.

[18] Reiter, M. K., & Rubin, A. D. (1998). Crowds: Anonymity for web transactions. ACM transactions on information and system security (TISSEC), 1(1), 66-92.

[19] Sutrisnowati, R.A., Bae, H. and Song, M., 2015. Bayesian network construction from event log for lateness analysis in port logistics. Computers & Industrial Engineering, 89, pp.53-66.

[20] Sweeney, L. (2002). Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05), 571-588.

[21] Tbahriti, S. E., Medjahed, B., Malik, Z., Ghedira, C., & Mrissa, M. (2011, August). Meerkat-A dynamic privacy framework for web services. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on (Vol. 1, pp. 418-421). IEEE.

[22] Tbahriti, S. E., Medjahed, B., Malik, Z., Ghedira, C., & Mrissa, M. (2012, March). How to preserve privacy in services interaction. In Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on (pp. 66-71). IEEE.

[23] Yu, Q., Liu, X., Bouguettaya, A. & Medjahed, B. (2008). Deploying and managing web services: Issues, solutions, and directions. The VLDB Journal, 17(3):537{572.